AWSCCP

Exam Domains

Domain 1: Cloud Concepts 28%: Second weighted
1.1 Define The AWS Cloud and it's value proposition
1.2 Identify Aspects of AWS Cloud Economics
1.3 List the Different AWS Cloud Architecture Design Principles

Domain 2: Security 24%: Third weighted
2.1 Define the AWS shared responsibility model
2.2 Define AWS Cloud Security & Compliance Concepts
2.3 Identify AWS Access Management Capabilities
2.4 Identify Resources for Security Support

Domain 3: Technology 36%: First weighted
3.1 Define Methods of Deploying & Operating in the AWS Cloud
3.2 Define the AWS Global Infrastructure
3.3 Identify the Core AWS Services
3.4 Identify resources for Technology & Support

Domain 4: Billing & Pricing 12%: Fourth weighted
4.1 Compare & Contrast the various pricing models for AWS
4.2 Recognize the various account structures in relation to AWS billing & pricing
4.3 Identify Resources Available for billing and support

Domains listed by order of weight:
1. Technology
2. Cloud Concepts
3. Security
4. Billing & Pricing

Chapter 1: The Cloud

Exam Domains:
Domain 1: Cloud Concepts
1.1 Define the AWS Cloud and its value proposition
1.2 Identify aspects of AWS Cloud economics
1.3 List the different cloud architecture design principles

Chapter Summary

- Sheer size and quality of AWS
- - Higher quality securiy - Better than local/onsite
- - Higher quality availability - Better than local/onsite
- - Higher quality reliability - Better than local/onsite

- Customers responsible for applications run in Cloud
- - Don't need to worry about underlying physical infrastructure

- Only pay for services you use as you use them
- - Virtually no CAPEX
- - Manage only OPEX

- Server Virtualization
- - More Densely pack software operations on physical hardware
- - Drive down costs
- - Improving time to deployment of compute workloads
- - Serverless computing

- Cloud-optimized platforms
- - Take advantage of scalability & elasticity of cloud platforms

Exam Essentials

Understand how a large and geographically dispersed infrastructure imporves service quality

- Sheer scale & geographic redundancy of physical compute & networking
- Guarantees level of reliability & availability; hard to reproduce in any other environment

Understand How metered, pay-per-use options makes for flexible compute options

- Experiment
- Sandbox
- Reassess & Update

Understand that cloud services come in a wide range of forms
-IAAS
-- Near full control over virtualized hardware resources
-- Closely emulates administering actual physical servers
-PAAS
-- Abstracts Underlying Server
-- Simplifies interface for adding application code
-SAAS
-- Provides services over a public network Directly to end users

Understand how serverless computing can be cheap & efficient
-AWS Lambda
- Access compute power for up to 15 minutes for a single function
- Operate code in response to real-time event triggers

Understand how scalability allows applications to grow to meet need
- Cloud-optimized application:
-- Allows for automated provisioning of server instances designed from scratch to perform needed compute function within appropriate framework

Understand how elasticity matches compute power to both rising and falling demand
- Scaling services of cloud provider configured to force compliance with budget and application needs
- set upper/lower limits - server handles startups and shutdowns

Introduction

To make smart choices about AWS Cloud:
- first need to properly understand
- - figure out just what cloud is
- - technologies built on
- - cost savings
- - operational advantages
- - how cloud-based applications work differently than traditional cousins

What is Cloud Computing?

Using a public cloud is about using other people's servers to run your digital workloads

- no significant difference between running applications in-house vs. cloud
- both cases: sufficient compute, memory, network, storage
- both cases: fast deployments, avoiding over-provisioning
- big difference: sheer size of AWS -> offers service, cost, reliability performance not able to reproduce on own

Highly available & scalable resources

Multiple Layers of Redudancy
Connect Resources in Geographicallly remote locations
Provide Access to as much compute power and delver that power-on-demand
scale efficiency of platform enable price drop

Professionally Secured Infrastructure
- IT security team -> not better informed, equipped, trained than counterparts
- Shared Responsibility Model
- - Resources run ON cloud customer responsibility
- - Cloud ITSELF Amazon's responsibility

Metered Payment Model
- cloud computing platform automatically allocates resources -> meet client requests
- Billed only for what you use, for how long
- Compare costs against on-premises deployments
- CAPEX vs. OPEX

Server Virtualization: The Basics

Virtualization 'secret sauce'
- gives customers on-demand compute resources
- Wide range of configurations
- request Virtual Machine: processor, speed, memory capacity, storage size
- AWS carves resources from larger devices

Virtualization Model offers two compelling benefits:

Virtualization Model Benefits
Speed: Process of configuring physical server takes long time; requesting cloud-based VM/logging in/getting to work happens fast; Restarting VM happens very fast
Efficiency: Resources quickly scalable based on immediate need; not possible with physical server setups

-Amazon formidable scale and logistical abilities
-- Often able leverage benefits of virtualizations
-- provides both superior performance and pricing

SCALABLE:
-Automatically changes capacity based on increased or decreased demand
-self-monitors, makes adjustments whenever pre-set performance metrics might to unmet

ELASTIC:
- Services built to be easily and automatically resized

Set maximum and minimum and the AWS services you're using will automatically add or remove resources to meet changing usage demands

Cloud Platform Models

Infrastructure as a service (IAAS)

- Generally simulates look & feel from managing physical resources
- Direct access to a providers compute, storage & networking assets
- customer responsible for consequences of any bad configurations; get to configure every layer of operating stack
- Elastic Cloud Compute (EC2): virtual machine instances
- Elastic Block Store (EBS): storage volumes
- Elastic Load Balancing (ELB): load balancing

Platform as a service (PAAS)

- Hides complexity of infrastructure running applications
- interface for defining behavior & environments for running application including code for running application
- Elastic Beanstalk
- Elastic Container Service

Software as a service (SAAS)

- Services meant to be accessed by end users
- Simple Email Service
- Amazon WorkSpaces

Serverless Workloads

- Allows for individual developers to run their code for seconds or minutes at a time on someone else's cloud services
- Makes it possible to design code that reacts to external events
- AWS Lambda

Chapter 2: Understanding Your AWS Account

Exam Domains:
Domain 4: Billing & Pricing
4.1 Compare and Contrast The Various Pricing Models For AWS
4.2 Recognize The Various Account Strucures In Relation to AWS Billing & Pricing
4.3 Identify Resources Available for Billing Support

Chapter Summary

Free Tier:
- Full year of free access to light configurations of most AWS services
- Track useage of free tier allowance from billing dashboard
- add '/pricing' to url of any AWS service to access pricing page
- understand billing rates & metrics used to measure them

AWS simple monthly calculator/AWS total cost of ownership calculator
- - Anticipates real-world usage costs for AWS deployments
- - Compares your spending with on-premises deployments

Resource Requests
- Sometimes refused if request pushes consumption past service limits

AWS Billing Dashboard
- Hub for accessing account administration , payment, tax status management, cost monitoring, cost control

Exam Essentials

Understand the Value of 12-month Free Tier

- Can run light services
- Goal to get you comfortable with AWS Environment
- examples: t2.micro ECS instance type; 30 GB SSH EBS volume

Understand the value of permanent Free Tier services

- Goal to give opportunity for launching proof of concept deployments
- examples: 10 GB of stored objects from Glacier; 62,000 outbound emails through Amazon SES

Know how to access Amazon's resource pricing online documentation

- Each service resources is billed by metrics unique to its characeristics

Use the AWS Simple Monthly Calculator to accurately model multitiered application stack pricing

- Pricing for variations of core AWS services prebuilt into calculator

Use the AWS total cost of ownership calculator to compare on-premises with AWS deployment costs

- compare on-premises vs. cloud-based for opex -> know whether cloud really suitable for workload

Understand how your use of AWS services is limited by default

- access to all resources restricted by default limits

Understand the value of cost management tools for avoiding costly cloud overspends

- AWS budgets can be configured to send alerts for resource consumption approaching/bypassing limits
- Cost Explorer provides visualizations to more easily montor historical and current costs
- send indepth and ongoing CSV-formatted data to Redshift or Quicksight for analysis
- Asset and Cost tags
- AWS organizations

Introduction

Two account-level things to be clear about:

What's it all going to cost you?

- cloud services scalable
- virtually no limit to how much can be purchased
- virtually no limit to how much can cost you

Who's running the show?

- who has authority to make big decisions (firing up/shutting down resources)
- financial implications
- define organization's public presence

Overall Concerns:

- Taking advantage of the AWS free tier
- understanding how AWS services are priced
- How to model pricing for complicated combinations of resources
- understanind the usage limits AWS places on some services and how to raise those limits
- Keeping on top of your actual account costs

The Free Tier

- Can freely experiment with light versions of most AWS services without being billed

How does the free tier work?

- allotment of specific amount of time for light versions
- exceed allotment, get billed
- examples:
- - t2.micro Elastic Cloud Compute (Windows/Linux) for up to 750 hours/month

Tracking Free Tier Usage

- Two ways:

- - waiting for messages to arrive in inbox:

- - - turn off from preferences page in AWS mgmt. console dashboard

- - - choose different email account

- - tracking tool appears toward bottom of billing dashboard for any free-tier eligible account

- - - View All button to view all free tier

What's available under the free tier?

- two kinds of free services:
- - Light implementations available through first twelve months of new AWS account
- - Light usage of certain AWS services that are always available even after initial months are up

The 12-Month Free Tier

- 30 GB magnetic/ssd drives from Elastic Block Store (EBS)
- EBS volumes primary drives for EC2 instances
- 500 mb/mo free storage -> Amazon Elastic Container Registry (ECR): service for storing and managing Docker images
- 50 GB outbound data transfers
- 2 million HTTP/S requests for Amazon Cloudfront distros: Cloudfront = Amazon's cloud delivery network
- 1 million API calls/mo on API Gateway: API Gateway = tool for managing program's API

Permanently Free Services

- some AWS services useful even when consumes at levels so low they don't even need to be billed
- 10 custom metrics and alarms on Cloudwatch: Cloudwatch -> track resource performance and costs (use directly or in combination with other tools)
- 10 GB data retrievals on Amazon Glacier/mo: Glacier -> low-cost storage service ideal for long-term storage (not likely require quick access)
- 62,000 outbound emails using Simple Email Service: SES -> enterprise-scale email service -> geared to organizations that regularly send large volumes of emails
- 1,000,000 requests and 3,200,000 million seconds of compute time for AWS Lamdba functions: Lambda -> Lets you run code in relation to Network Based Triggers
- AWS free tier benefits do not apply to resources launched in GovCloud (US) regions or through an AWS Organizations member account

Product Pricing

Finding AWS Pricing Documentation

- Each AWS service has own page dedicated to pricing
- Add '/pricing' to url for each service
- pricing can vary between AWS regions

Working with online calculators

- simple monthly calculator
- - understand what running any combination of running and AWS resources will cost you

- Total cost of ownership calculator
- - closely compare what you're currently paying for on-prem servers with what same workloads cost on AWS

Service Limits

- AWS imposes limits on scope of resources you can use
- Amazon's desire to ensure that all classes of resources should be reliably available to meet new demand - not all held by a few large customers
- most service limits are soft
- - can manually request and increase limits
- up-to-date list of service limits:
https://docs.aws.amazon.com/general/latest/gr/aws_service_limits.html

Billing and Cost Management

Billing Dashboard

- accessible through account drop-down menu at top of AWS management console
- main billing and cost management dashboard contains helpful visual spend summary
- includes links to pages for set up payment methods and enter organizations tax info

AWS Budgets

- tool for tracking specified set of events to trigger alert when preset threshold met

AWS Budge types
COST- montor current costs
USAGE - track particular categories of resource consumption
RESERVATION - active EC2, RDS, Redshift, or Elasticache reserve instances

Budget Setup Process

- setup terms of budget
- define how and when want alerts sent

Monitoring costs

- watch financial events from multiple perspectives

Cost Explorer

- lets build graphs for visualizing accounts historical and current cost
- can open heavily customizable view to filter by date, region, availability zone, instance type, platform
- CSV-formatted output

Cost and Usage Reports

- accessed from reports link on billing dashboard
- configured reports to include full range of activity on account
- enable support for Redshift and/or Amazon Quicksight

Cost Allocation Tags

- tag-metadata id element representing a resource and its actions
- **two types**
- - **Resource tags**
- - - used in busy accounts
- - - quickly id purpose and owner of a particular running resource
- - **cost allocation tags**
- - - only meant to interact with billing tools

AWS Organizations

- Formerly known as Consolidated Billing
- Allows to centralize administration of multiple AWS accounts

Chapter 3: Getting support on AWS

Exam Domains:
Domain 2: Security
2.4 Identify resources for security support
Domain 3: Technology
3.4 Identify resources for technology support
Domain 4: Billing and Pricing
4.3 Identify resources available for billing support

Chapter Summary

Four Support plans
- Basic: Free
- Developer: $29
- Business: $100
- Enterprise: $1500

Non-free plans billed as % of total monthly resource usage

Higher support level:
- More responsible and personalice the technical troubleshooting

Developer support
- Aimed at organizations still testing and planning deployments

Business support
- For small operations running relatively light production infrastructure

Enterprise support
- Ideal for large production w/global footprint that cannot tolerate downtime

Exhausting and updated documentation
- Multiple styles
- Number of web platforms
-- user guide documentation pages
-- knowledge center
-- resource specific to security
-- discussion forums

AWS trusted advisor
- Alerts users to vest practice compliance of running account resources

Limited alerts
- Basic support
- Developer support

Unlimited alerts
- Business support
- Enterprise support

Exam Essentials

Know how to choose a support plan that reflects your operational needs
- Whatever it takes to design a lean and security-hardened application is a justifiable expense

Understand benefirs of enterprise support plan's technical acount manager
- Ongoing personalized attention

Understand how to find AWS resource usage guidance through official AWS documentation
- user guides available in multiple formats (HTML, PDF, KINDLE, Markdown-Github
- Knowledge center
- Understand how to use trusted advisor for alerts to common systm misconfigurations
-- Trusted advisor five categories:
1. cost optimization
2. performance
3. security
4. fault tolerance
5. service limits
- set up admin routine

Introduction

Preparing and securing effective on-premises server workloads can be complicated and stressful
- AWS makes it easy to provision and launch app environments
- Breaking through experience barrier can be difficult
- AWS provides layers of support
- Four AWS support plans
- professional services
- online documentation
- online forums
- Amazon's Trusted Advisor

Support Plans

Four AWS support plans
- Basic
- Developer
- Business
- Enterprise

Amazon paid support plans

Developer	Business	Enterprise
Greater of $29 or...	Greater of $100 or...	Greater of $15,000 or...
3% of monthly account usage	7% of usage up to $80,000	10% of usage for the first $0-$150,000
	5% of usage up to $250,000	7% of usage from $150,000-$500,000
	3% of usage over $250,000	5% of usage from $500,000-$1,000,000
		3% of usage over $1,000,000

Key benefits of the AWS support plans

Basic	Developer	Business	Enterprise
7 Trusted Advisor Checks	7 TA Checks	All TA Checks	All TA Checks
	8 a.m. to 6 p.m. (local time) email access to associate	24/7 email, chat, phone access to engineer	24/7 email, chat, phone access to engineer
	General Guidance within 24 business hours	General guidance within 24 hours	General guidance wtihin 24 hours
	System impaired help within 12 business hours	System impaired help within 12 hours	System impaired help within 12 hours
		Production system down help within 1 hour	Production system down help within 1 hour
			Business-critical system down help within 15 minutes

Basic Support Plan
- Not paying beyond regular costs of consuming AWS resources
- Only publicly available documentation: white papers, tutorials and support forums
- Limited access to trusted advisor tool
- Receive alerts concerning interruptions to any AWS services: personal health dashboard

Developer support plan
- For organizations running non-production workloads
- Non-production: Still in development stages; not handling critical transactions or heavy traffic
- Access to cloud support associates is limited
- Can only ask for information on general use cases
- Cannot discuss specific problems

The Business support plan
- Can meet the needs of many organziations by offering relatively fast and detailed answers
- Guarantees an answer from a cloud support engineer via email, chat or phone w/i one hour
- Less severe issues up to 24 hours
- Help troubleshooting interoperability between AWS resources and 3rd party software and OS
- Additional fee = in-depth guidance in design stage

Enterprise Support Plan
- Starts at $15,000
- Appropriate only for large organizations (Global scope, no downtime)
- TAM- Technical Account Manager
- TAM closely involved in deployment, guidance team through planning launches and productive reviews
- 24/7 access to senior cloud engineers

AWS Professional Services
- APS team provides ‘offerings': detailed guides to achieving specific outcomes

Documentation and Online Help

No support plan help:
- self-help style resources

Documentation
- Tons available
- Learn how to navigate to needed pages
- Look for ‘latest’ in url for specific page to ensure latest documentation

Knowledge Center
- Very enhanced FAQ

Security resources

- Dedicated url with links to security resources

Discussion Forums
- Self-explanatory (for more specific, unique questions)

Trusted Advisor

Use to visually confirm whether account resource configurations are sound

Organizes compliance alerts across five categories:
- Cost optimization: Identifies any resources that are running and costing you money but are either underutilized or inactive
- Performance: Identifies configuration settings that might be blocking performance improvements
- Security: Identify any failures to use security-best practice configurations
- Fault Tolerance: ID's any running resources that, through poor configurations, are unnecessarily vulnerable to service disruption
- Service Limits: Identifies resource usage that's approaching AWS region or service limits

Status of Advisor Check: Icon next to a particular item

Full range of trusted advisor alerts available only to business or enterprise plans

Chapter 4: Understanding the AWS Environment

Exam Domains:
Domain 2: Security
2.1 Define the AWS Shared Responsibility Model
Domain 3: Technology
3.2 Define the AWS Global Infrastructure

Chapter Summary

AWS region
- Connects at least two availability zones located within a single geographic area into low-latency network
- Building secure, access-controlled regional environmanes eminently possible

Availability Zone
- One or more independent, fault-protected data centers located w/i a single geographic region
- Be careful of region selected
- Executed operations will launch in context of region
- Design structure of regions allows to build infrastructure to provide best customer service experience while meeting security/regulatory needs
- Some global resources are geographically unrestricted (IAM, Cloudfront, S3)

Can connect to AWS services using their endpoint addresses - generally incorporates host region's designation

EC2 Virtual machine instances
- Launched with IP addresses issued from network subnet
- Associated w/single availability zone

High Availability
- Makes infrastructure more resilient and reliable
- launches parallel redundant instances in multiple availability zones

AWS edge locations
- Globablly distributed Data Serviers
- Can store cached copies of AWS-based data
- Can be efficiently served to end users

AWS shared responsibility
- Defines elements of AWS platform you are expected to secure and maintain

Exam Essentials

Understand the imortance of resource isolation for cloud deployments
- Priority: placing resources w/i right availability zones
- carefully setting appropriate access controls
- improves application security and performance

Understnad role of autoscaling in highly available deployment
- Scalability - automate process of increasing or decreasing scale based on need
- Automate application recovery after a crash

Understand role of load balancing in highly scalable deployments
- Ability to automatically redirect incoming requests away from non-functioning instance to backup

Understand the principle of AWS shared responsibility model
- AWS: Security and Administration for underlying physical infrastructure and full stack of managed service
- Customer: Responsible for all else

Understand principles of AWS Acceptable use policy
- Keep it legal, keep it white hat

Introduction

Learn to configure efficiently and effectively understand how AWS infrastructure works

General Topology:
- Hundreds of globally distributed data centers
- Divided into regions
- Divided into availability zones

Globally distributed locations extend network reach of applications

Cloudfront: Content Delivery Service

AWS Shared Responsibility Model

AWS Acceptable Use Policy

AWS Global Infrastructure: AWS Regions

Hundreds of thousands of servers maintained within physical data centers

Widely distributed set of geographic regions

As global footprint of AWS grows, number of regions grows

Some regions not accessible from regular AWS account

Deploymen into U.S. Gov Govcloud requires special permission

Regionally Based Services
- Request instance of AWS service; underlying hardware of instance carved out of server
- EC2
- EBS
- S3
- Sometimes run parallel resources
- Sometimes ‘edge-case’ scenarios
- must be aware of region active
- easy to accidentally launch new resource into wrong region
- can make it impossible for mutually dependent app components to find each other
- check current region

Dividing resources among resources
- Locate your infrastructure geographically closer to your users to allow access with the lowest possible latency
- Locate your infrastructure within national borders to meet regulatory compliance with legal and banking rules
- Isolate groups of resources from each other and from larger networks to allow the greatest possible security

Globally based services
- Some resources not visibly tied to any one region
- AWS IAM
- Amazon Cloudfront
- Amazon S3 buckets

Service Endpoints
- Work with or access resourced running w/i AWS regions; know how resources identified
- Connect with resources through application code or shell scripts
- Often authenticate by referring to endpoint

AWS Global Infrastructure: Availability Zones

AWS region (except Osaka-Local) encompasses at least two distinct availability zones

Single AZ = 1 fully independent data center

Built on hardware and use resources used by not other AZ

If one AZ loses power or suffers some kind of catastrophic outage, chances of spread in region are limited

No two AZ's will ever share resources from a single physical data center

Availability Zone Designations
- Before launching some instance. specify network subnet associated with A.Z.
- Subnet/A.Z. combo will be instance's host environment
- A.Z.'s not listd in order in subnet drop-down menu
- Intentional so as to avoid 1st choice always being only selection
- Prevents overloading

Availability Zone Networking
- Organize AWS resources into segments (subnets) for proper resource utilization
- Maybe isolate different servers to prevent leakage
- Distributing production workloads among subnets improves availability and fault tolerance
- Subnet = single block of I.P. addresses
- Compute device needs connectivity:
- - Identified by IP unique to network servers/other networked devices assigned I.P. w/i one subnet
- - Communication outside subnet may be restricted by firewall rules

Available private IPv4 address ranges

From	To	Total
192.168.0.0	192.168.255.255	16,535
172.16.0.0	172.31.255.255	1,048,576

- Up to 200 subnets per A.Z.

Availability Zones and High Availability
- Hardware will fail
- Single point of failure: resource running w/o backup
- Use redundancy: provision two or more instances of what is required
- Don't use parallel - distribute

- Generally cheaper and easier to create redundancy in cloud
- AWS AZ's geographically distributed
- AWS uses auto-scaling and load balancing
- - Autoscaling - Replaces or Replicates resources to ensure predefined service level maintained
- - Load Balancing - Orchestrates use of multiple parallel resources

AWS Global Infrastructure: Edge Locations

Edge Location
- Deploys physical server infrastructure to provide non-latency user access
- More focused on narrower set of offerings vs. full-fledged data center
- Narrower range of hardware
- Front-line resource for directing traffic most benefitting from speed
- Cloudfront = Amazon edge location CDN: Content Delivery Network

Edge Location Services
- Amazon Route 53
- AWS Shield
- AWS Web app firewall
- Lambda Edge

Regional Edge Cache Locations
- Used for handling less popular content
- Not as fast as edge cache
- Compromise between regional edge location and A.Z.

The AWS Shared Responsibility Model

Security and Reliability of cloud
- Amazon's responsibility

Security and Reliability of what's in the cloud
- Customer's responsibility

Cloud = Physical buildings, servers, networking hardware
AWS = locations, secure, reliably powered, properly maintained
AWS = proper encryption, patching, maintaining OS

Customer	AWS
Customer Data	Hardware and Network Maintenance
User Apps, Access Config	AWS Global Infrastructure
OS, Network, Access Config	Managed Resources
Data Encryption	

Managed Resources
- Will ‘hide’ some or all of underlying configuration and administration work needed to keep things running

Unmanaged resources
- Responsible for app and infrastructure (user)

Service Health Status
- Regularly updates, region-by-region
- Reports on status of physical servers
- Publicly available
- Check service health dashboard first always

Acceptable Use Policy
- Don't do anything illegal, harmful, offensive
- Even pentesting against self = need permission

Chapter 5: Securing Your AWS Resources

Exam Domains:
Domain 2: Security
2.2 Define AWS Cloud Security and Compliance Concepts
2.3 Identify AWS Access Management Capabilities

Chapter Summary

Enforce Use of Strong Passwords
- Create a Password Policy in IAM
- Require MFA

Programmatic & Command Line Access to Resources
- Use Security Credentials
- Access Key ID
- Secret Access Key
- SSH Access to EC2 Linux Instances authenticated using AWS generated key pair

Efficiently control resource access for large numbers of users through IAM management:
- Use Principle of Least Privelege

IAM Role
- Set of permissions
- Permits access to beneficiary process
- Defined set of resources
- Best to securely enable functionality between parts of AWS infrastructure
- Review important information on user sec status from credit report in IAM and read about AWS compliance in AWS Artifact

Exam Essentials

Know How to Lock Down Accounts Root User to Reduce Exposure to Risk
- Make sure root user has strong password
- MFA-enabled
- Never use for day-to-day administration tasks

Know how to enforce use of strong passwords for all users
- Set IAM password policy
- Lower, upper/lower, numbers, non-standard characters

Understand how AWS manages access credentials for EC2 key pairs, secret access keys, and encryption keys
- Make use of AWS encryption services
- Secure terminal connections to EC2 servers, API access, or privacy of data

Know how to provide federated access to AWS resources based on third-party authorizations systems like Google
- Use standards such as SAML 2.0 and A.D.
- Incorporate external authorization into AWS infrastructure

Be aware that AWS Key Management Service manages encryption keys
- KMS-managed keys used across wide range of AWS services- EBS, RDS, DynamoDB, S3

Be aware that AWS Artifact is a compliance information resource
- AWS Artifact: provides access to official documentation on compliance of AWS infrastructure

Introduction

IT application infrastructure security

Critically important elements

Ensure assets properly protected

Learn to use Identity and Access Management (IAM)
- Control which people and processes get past wall guard
- IAM smart use of users, groups, roles, federated id's

Applying encryption to cloud data
- Managing encryption keys
- AWS compliant with wide range of regulatory frameworks

AWS Identity and Access management

IAM
- Management dashboard connects to all admin tools to manage accounts

Protecting the Root User
- Root user has admin permissions to perform any task
- - Launching any resources in a region
- - Authorizing any resources

Activating Root over long-term is a major security risk

Protecting root
- Created complex password
- Implement MFA

For most administrative activities, use IAM users instead

Authentication
- proving who you are
- user ID and password
- for programmatic/cli access - set of access keys needed

Passwords
- Root User: loging via email & password
- use randomly generated strings for passwords
- use upper/lower letters, symbols, numbers
- avoid passwords related to your life
- don't reuse passwords

Password policy
- configured from accounts settings sections of IAM dashboard
- MFA:
- - something you know
- - something you have
- - something you are
- associates physical device with account

Access Keys
- Programmitic/cli access: authenticated by access keys (w/o MFA)

Security Credentials
- Allows for creating new set of keys while logged onto AWS management console

Create New Access Key button

- option to download key to computer
- show actual access id key and secret asset key in dialog
- - copy & store somewhere safe

Security Credentials Page
- List all keys
- Allows to activate or deactivate or delete
- logged in here as root - delete all keys associated w/root

Secure Shell Key Pairs
- SSH protocol - industry standard for safely encrypting remote login sessions

Encryption
- converst plain-text packets into what looks like gibberish
- require public/private key pair to handle encryption/decryption

Users, groups and roles
- Don't use root for regular admin duties
- Principle of least privelege

IAM users/groups
- Use groups to administrate process of assigning permissions

IAM roles
- defines limits for what can be done w/i AWS account
- Role - used by apps and services

Trusted Entity
- What is trusted by the specific role

Providing Federated Access
- Federation: expands available tools for managing authentication beyond simple IAM options
- Can integrate 3rd party standards:
- - SAML (Security Assertion Markup Language)
- - A.D.
- Lets you add existing login = SSO
- Amazon SSO
- AWS Microsoft A.D.

Credentials Report
- Accessed from IAM dashboard
- Allows to ‘download report’
- CSV-file:
- - state of account security
- - all current IAM users
- - When last logged in
- - MFA enabled

- - Active Access Keys
- - Keys last rotated

Encryption

Encryption
Data protection goes beyond remote login sessions

Encrypt data wherever stored or consumed

AWS provides enterprise-strength encryption tools

AWS KMS (Key Management Service)
- Custom Master Key (CMK)- used to apply encryption

KMS or encryption keys page
- Used to manage keys
- Most anything can be encrypted

Client-side encryption more complicated
- Needs KMS-managed customer master key
- Client-side master key

Regulatory Compliance

AWS Artifact - set of links to compliance docs
- Including:
- - FedRAMP
- - GC Partner Package
- - APRA “Management of Security Risk in Information and Information Technology” Handbook
- - PCI DSS Attestation of Compliance
- - Responsibility summary for handling credit card transaction data

Service Organization Controls
- SOC 1,2,3 audits of Infrastructure

Chapter 6: Working With Your AWS Resources

Exam Domains:
Domain 2: Security
2.2 Define AWS Cloud Security and Compliance Concepts
2.4 Identify resources for security support
Domain 3: Technology
3.1 Define Methods of Deploying and Operating in AWS Cloud
3.3 Identify Core AWS Services
Domain 4: Billing and Pricing
4.4 Identify Resources Available for Billing Support

Chapter Summary

Most of time using management console in beginning
- changes frequently; will be notified of the change

CLI will become more useful as time goes on
- must for scripting AWS tasks
- must for collecting AWS resources in bulk

Cloudwatch
- Collects metrics for AWS services
- Creates alarms
- Receives and stores logs from AWS and non-AWS services
- extracts metrics using metric filters

Cloudtrail
- Records Events that occur agains AWS account
- History log captures last 90 days by default
- - Want more - create trail to store in S3 bucket
- - Configure trail to stream logs to Cloudwatch logs for storage, viewing, and searching

Exam Essentials

Understand when to use AWS management Console vs. CLI management console
- required for point-n-click operations
- For visual elements like Cloudwatch graphs or Cost Explorer Graphs
- Log in w/email address and password
- - If IAM user, need acct. alias or number, IAM username and password
- - Prompted for MFA

CLI
- Used to manage resources manually from CLI or using scripts
- Good for repetitive or bulk tasks
- Need access key ID and secret keys

Know how to use resource tags and resource groups
- Resource tags = keys associated with AWS resources (can contain value)
- Can label resources however you like
- Resource groups - According to tags or Cloudformation stacks

Be able to identify use cases for Cloudwatch
- Can collect metrics from AWS and non-AWS services
- Can create alarms to send notifications using SNS
- Can take action using autoscaling action
- Graph metrics to visual view
- Aggregate and search log files
- VPC and Rte 53 - configure to stream vended logs to Cloudwatch logs
- - extract metrics using metrics filters

Cloudwatch - take action according to events that happen with AWS resources and acts in response to specific API ops

Know options for developing applications that integrate with AWS
- SDKs for variety of programming languages and platforms
- SDKs for server, webbased, or mobile apps
- SDKs handle request authorization, serialization, connection management
- Don't need to learn HTTPS based details

Understand what Cloudtrail is and how it differs from and integrates with CloudWatch
- Cloudtrail
- - Logs management and data operations on your account
- - By default - Logs 90 dyas of management events per session
- - For more - create trail and log in S3 bucket
- Optional
- - Stream Cloudtrail logs to Cloudwatch for storage searching, and analysis

Introduction

AWS offers two common tools to interact w/all services

1. AWS Management Console
- - Web based, user friendly point and click interaction
- - AWS console mobile app
- - - View or manage some services from Android or iOS

2. AWS CLI
- Perform Config and view info from CLI of favorite OS
- Allows for working w/services at API level

SDKs
- Help devs write apps to seamlessly integrate w/AWS services

Cloudwatch
- monitor resources, create alarms, collect, view and search logs

Cloudtrail
- Keeps Details Logs, Key security app

Cost explorer
- What AWS services are costing you, bill and use trends, recommendations to save money

The AWS Management Console

AWS management console, AWS console, AWS web console
- web interface you can use to manage all AWS cloud resources using browser
- used for point and click interface
- each AWS service has own console
- some services (Cloudwatch, AWS billing) offer visual reports to view and download

Most AWS resources managed via console

Some advance options needs AWS CLI

Compatbile with following browsers:
- Apple Safari
- Google Chrome
- Microsoft Edge
- Microsoft Internet Explorer

- Mozilla Firefox

Accessing the AWS management console
- https://console.aws.amazon.com
- sign in as root user or IAM user

Logging in as root
- email and password associated with AWS account

Logging in as IAM user
- Enter account ID or account alias and click next
- Will be brought to different page to enter IAM username and password

Opening a service console
- Each AWS service has own console to manage services
- choose services link lets you search for service by name
- can browse by type, compute, storage, or data base, etc...
- New Service Launch = always available w/i 180 days; usually immediately available

Working with shortcuts
- Can pin icons with pushpin icon

Select a region
- Most AWS services are region specifig
- Some global = IAM, Route 53, S3

Account name menu
- Displays IAM user and account alias/ID
- root name associated with AWS account

My account - takes you to billing service console page that displays your account info
My organization - Takes you to AWS organizations service console
My billing dashboard - Takes to billing and cost management dashboards
My security credentials - takes you to the my password page in IAM services console where you can change passwords
Switdh Role - Lets you assume an IAM role where you can perform tasks as a different principal with different permissions
Sign Out - Signs you out of the AWS management console

Resource Groups
- Already available from AWS management console
- Gives options for creating most resource groups and view existing ones
- Lets you view, manage, and automate tasks on multiple resources
- Ideal for grouping AWS resources compose a particular application - add resources to resource group
-

Resource Group
- Collection of AWS resources in same region
- Query based on tags/from Cloudformation stack - create resource group

Resource Tags
- Optional metadata assigned to AWS resources
- contains ‘key’ and maybe ‘value’

Tag keys/values
- letters, numbers, spaces, +-=._:/@
- Up to 50 tags assigned; case-sensitive

AWS Cloudformation Stacks
- Programatically Deploy and manage multiple AWS resources as single unit called stack
- Can create resource groups containing some or all resources in Cloudformation stack

Tag Editor
- Create resource based on tags
- - need to tag resources to include in group

Choose resource groups link in navigation bar of the AWS Management Console, then choose tag editor

1. Create query to find resources to tag
2. select one or more region and resource type
3. can list resources that have existing tag or have no tag
4. no specified tag key - name tag key used

Tagging strategies
- Technical, Automation, Business, Security
- Use multiple
- Make own types

Technical
- Tag according to technical properties of a resources
- Name to individual resource
- tags don't have to be unique; be careful not to give the same tag to multiple resources
- Environmental - Production, test, development, infrastructure
- Application - classify type of role a resource performs

Automation
- Define resources part of automation process
- updating security patches
- Backups
- Deleting old snapshots
- Turning off development servers after hours
- specify date or time tags to specify when automation tasks occur

Business
- group resources into categories for business (billing, management, analysis)

Tag examples:
- Owner to identify the person or group responsible for the resources
- Business Unit or Cost Center to indicate who's responsible for paying for the resource
- Project to identify the name of the program or project the resource is a part of
- Customer to identify resources that are dedicated to a particular owner

Security
- IAM policy conditions
- Look at resource tages to determine whether to allow a particular action

Strict security requirements
- tag resource to confidentiality level of data or compliance

Confidentiality
- use resource tags to designate level of confidentiality

Compliance
- tag resources according in compliance requirements

The AWS Console Mobile Application
- smartphone application
- - manage AWS account and resources on the go
- features dashboard showing info:
- - service health
- - Cloudwatch alarms
- - billing
- Use app to make limited changes to AWS resources
- Cloudwatch alarms, ECS security groups, EC2 instances, Cloudformation stacks

Supported Services
- AWS billing and cost management
- AWS cloudformation
- Amazon cloudwatch
- Amazon dynamo db
- Amazon EC2
- AWS Elastic Beanstalk
- Elastic Load Balancing
- AWS Opsworks
- AWS personal health dashboards
- Amazon relational database service (Amazon RDS)
- Amazon Route 53
- Amazon Simple Storage Service (Amazon S3)
- Amazon Virtual Private Console (Amazon VPC)

Requirements
- AWS console mobile app
- supports: Apple iOS 7.0+ and Android 4.0+
- Amazon app store
- Google play
- iTunes
- Launch -> authenticate to existing AWS account by adding id
- Add multiple id's
- - root credentials
- - IAM user name and passwords
- - Access key id

The AWS Command Line Interface

Unified CLI to manage AWS resources

Gives acces to public API's for all AWS services w/i 180 days

If you can do it in the management console, you can do it in the CLI

useful for repetitive tasks

enter commands manually or incorporate into scripts

Requirements
- Windows
- Apple
- Linux
- Network should allow outbound access to internet on TCP port 443
- IAM access key ID and secret key to authenticate to AWS to CLI

Installation
- Preinstalled on Amazon Linux AMIs
- Otherwise install by OS type

Installation using Python and PIP
- Python version 2.6.5+ or Python 3.3+
- PIP
- Internet access during installation

Installation using standalone Installer
- Doesn't require internet access
- One for LInux/MAC and another for Windows

- Linux/Mac = bundled installer (AWS CLI, dependencies, install script)

Windows
- MSI installer (64/32 bit)

Software Development Kits

Simplify use of AWS services in custom apps

App devs use SDK to integrate app services

save app devs from having to write low-level code

SDKs for following languages:
- Java
- .net
- node.js
- php
- python
- ruby
- javascript
- go
- c++

Mobile Software Development Kits
- AWS Mobile SDK for android
- AWS mobile SDK for iOS
- AWS mobile SDK for Unity
- AWS mobile SDK for .net and xamarin
- AWS Amplify
- - Opens source framework to build mobile and web apps on AWS
- - Build app backened on AWS and integrate w/Android, iOS, web apps
- Uses AWS to offer cloud-based
- - Authentication
- - notification
- - offilne data synchronization
- - analytics
- - more
- integrate mobile apps w/o writing lots of custom code
- javascript library supports react native

IOT SDK integrate with:
- Sensors
- Microcontrollers
- Smart APIs

- Smart Lightbulbs
- AWS IOT Lightbulbs
- Collection services allows IOT devices interact w/AWS services, other apps
- Centrally onboard, manage, monitor IOT devices
- AWS IOT SDKs -> IOT platform integration
- - S3
- - DynamoDB
- - Kinesis
- - Lambda
- - SNS
- - SQS

Allow devs optimize memory, network, power usage

Reduce size of apps

Ensure secure fast communication with AWS

IOT Device SDKs Available:
- Embedded C
- Javascript
- Arduino Yun
- Python
- Java
- C++

Cloudwatch

Helps plan, monitor, fine-tune AWS infrastructure and apps

Lets collect, search, visualize data from apps and aws resources in logs, metrics, events

Common use cases:

Infrastructure monitoring and troubleshooting

Resource optimization

Application monitoring

Log analytics

Cloudwatch metrics
- Collects numeric performance metrics (AWS and non-AWS)
- metric: varialbe contains time-ordered set of data points, time stamp, unit of measure

- All AWS send metrics to Cloudwatch
- Store metric for up to 15 months

Cloudwatch Alarms
- Watches over value of single metric
- If crosses set threshold and stays, alarm takes action

Protocols:
- SNS
- Http(s)
- sqs
- lambda
- mobile push notification
- email
- email-json
- sms texts

Autoscaling

EC2 Action

Cloudwatch Dashboards
- one stop shop for eyeing all importan metrics
- create multiple dashboards
- - metric graphs
- - latest value for metrics
- - cloudwatch alarms
- save for future
- share w/ others
- visualize metrics from multiple AWS regions

Cloudwatch Logs
- collects and stores log files from AWS and non-AWS

Log evenits, streams, and groups
- configure apps and AWS services send
- Log events -> Cloudwatch logs

Log event - time stamp and event message
- vended log
- - route 53 dns query log
- - vpc flow logs
- - cloudtrail logs
- - custom logs from applications
- stores log event from same source in single log stream
- log group
- stores log events indefinitely by default

Metric Filters
- extracts data from log events in a log group and stores that data in a custom cloudwatch metric
- use metric filters to track number of times particular string occurs

Cloudwatch Events
- Let's you continuously monitor for specific events that represent a change in AWS resources
- especially write-only operations
- start by creating rule to define events to monitor
- Action to take in response to those events
- Define action by selecting target:
- - Lambda functions
- - EC2 instances
- - SQS queues
- - SNS topics
- - ECS tasks

Cloudwatch responds to events as they occur in real time

Event trigger immediately

Create schedule auto perform actions at regular intervals

Cloudtrail

Keeps detailed log events of every action

Includes following parameters
- service
- name of API action
- region
- response elements
- principal that made request
- date and time of request
- ip address of requestor

API and non-API events
- API actions (nothing to do with how action performed)
- - launching instance,
- - creating S3 bucket
- - creating new IAM user
- - taking EBS snapshot

Non-API
- Everything else

Management and Data Events

- Two other classification dimensions
- management events (control plane operations)
- principal attempts to execute against AWS resource
- write-only events
- API operations and resources
- read only events

Data Eventa
- S3 object-level activity
- Lambda function executions

Event History
- stored 90 days of management events
- can view, search or downloads
- doesn't record data events
- maintains separate event history logs containing events fro each region
- global services included in events history for every region

Trails
- Directs cloudtrail to record specified events in log files and deliver on S3 bucket
- can log events from single or all regions
- log management, data, both
- log as read only, write only, or both
- cloudtrail not search trail logs; written in json
- can download log files from S3
- configures cloudtrail -> send trail log to cloudwatch logs

Log File Integrity Validation
- Optional feature
- Provides assurance -> no cloudtrail log files modded or deleted
- cloudtrail writes log files to trail's S3 bucket
- calculates and store cryptographic hash, unique value based on contents of log file itself
- file modded -> hash changes
- log files encrypted = server-side w/S3
- server side encryption: S3-managed encryption keys (SSE-S3)
- server side encryption: AWS KMS-Managed keys (SSE-KMS)

Cost Explorer

Feature of AWS Billing and cost management

Offers configuration reporst and graphs to help understand how AWS services impacts monthly bill

Three categories of reports:
- Cost and useage reports
- reservation reports
- reserved instance recommendations

Cost and usage
- can generate cost and usage reports
- show forecast: current month, 3 months, 12 months

Filter by parameters:
- services
- availability zone
- region
- instance type
- usage type
- tag
- api ops
- charge type
- platform

Five default cost and usage reports:
- monthly costs by service
- monthly costs by linked account
- monthly EC2
- daily cost
- AWS marketplace

Reservation Reports
- Two instance reservations
- save money by prepaying for compute instance
- EC2, Elastigraph, Elasticache, RDS, Redshift

Reserved Instances Utilization
- shows percentage of resource instances
- how much saved or overspent
- net savings

Reserved Instances Covers
- how many instances running
- instance hours covered by instance reservations
- spent for on-demand instances
- purchasing reserved instances

Reserved instance recommendations
- Cost Explorer
- - provide reserved instances recommendations to help reduce costs

cost explorere analyzes on-demand instance over past 2, 20, or 60 days

searches for all available reserved instances that would cover that usage

selects most cost-effective reserved instances

makes recommneds separately for each

ignores any uasge already covered by reservation

updates report data at least once every 24 hours

Chapter 7: The Core Compute Services

Exam Domains:
Domain 3: Technology
3.1 Define Methods of Deploying and Operaing The AWS Cloud
3.3 Identify the Core AWS Services
Domain 4: Billing and Pricing
4.1 Compare and Contrast the Various Pricins Models for AWS
4.2 Recognize the Various Account Structures In Relation to AWS Billing and Pricing

Chapter Summary

Configuring EC2 Instances
- Mirrors process of provisioning and launching on-premises servers
- Instances defined by:
- - AMI
- - Instance Type
- - Storage Volume
- - Pricing Model

AMI Four Categories
- Quick start
- Custom
- AWS Market Place
- Community
- Can creat from snapshot: based on EBS volume of EC2 instance

EC2 instance types:
- Designed to fit specific applications demands
- Individualized optimizations available in varying sizes

EBS storage volumes
- Encryptable
- More like physical hard drives in flexibility of usage
- Instance store volumes located on same physical server hosting instance -> faster performance

EC2 on-demand pricing
- Best for short term workloads that can't be interrupted
- Longer term workloads cheaper when purchased as reserved instances
- Spot instances for compute-intensive data operations that can survive shutdowns

Lightsail, Elastic Beanstalk, Elastic Container Service, Elastic Container Service for Kubernetes,

Lambda
- Designed to provide abstracted compute services
- Simplify, Automate and reduce cost of compute operations

Exam Essentials

Understand the elements required to provision an EC2 instance
- Requires base OS (AMI)
- App stack
- Instance layer for hardware profile
- EBS or instance volume for storage

Understand the sources, pricing, and availability of EC2 AMIs
- Quickstart AMI - Supported by Amazon or recognized 3rd party vendor
- Marketplace AMI - Support by Amazon or recognized 3rd party vendor
- Community collection - Maybe not supported by Amazon or 3rd party vendor
- Confirm - using AMI incur extra charge?

Understand how EC2 instance types determine computer power of instance
- Instance types divided into families
- - focuses on functional niche: purpose, compute optimized, memory optimized, accelertaed, storage)
- App needs and budget: Determine instance type

Understand differences between EBS and instance store volumes
- EBS volumes are versatile
- - convertible to AMI
- - Survive instance shutdown

- Instance store volumes
- - faster read/write
- - more secure for some purposes
- Storage types -> instance type chosen

Understand Difference Between EC2 Pricing Models
- On demand:
- - Most expensive
- - Flexible
- - Reliable
- - Control Start/Stop

- Reserved:
- - Good for persistence

- Spot instance:

- - Least expensive
- - Shut down with only two minute warning

Be familiar with Amazon's managed deployment services
- Lightsail:
- - Provides blueprints for simplified flat-rate deployments -> uses EC2 under the hood

- Elastic beanstalk:
- - manages underlying infrastructure for apps
- - auto scales according to demand

Understand How container and serverless models work in cloud
- Containers:
- - Share OS kernel and device drives with host
- - Share common software layers w/each other
- - Produce fast and lightweight apps

ECS & EKS
- Focused on simplifying docker orchestration within EC2 framework

Lambda Functions
- Designed to respond to event triggers to launch short-lived operations

Introduction

EC2:
- Introduced in 2006
- Mirrors functionality of traditional on premises data centers:
- - provision/launch virtual servers (instances)
- - run some application workloads similar to legacy servers
- - more resilient, scalable, cheaper than on premises

Other Compute Tools:
- Elastic Beanstalk
- Lightsail
- Docker (VIA Kubernetes Orchestrator)
- Lambda

Deploying Elastic Compute Cloud Servers

Get VM instance running
- First define elements one at a time
- Select Amazon Machine Image
- - Instance type matching needs
- No need to choose cpu, memory, adapters and adding to motherboard
- Define virtual storage volumes available through Elastic Block Store (EBS)

Amazon Machine Images
- Image
- - Software bundle
- - built from template definition
- - made available within AWS region
- - copied to freshly created storage volume
- - once extractable -> bootable drive
- - turns attached VM into fully operational server
- - available in many flavors:
- - - RHEL
- - - Ubuntu
- - - Windows Server
- - - Amazon Linux
- - - Preloaded with hundreds of software stacks: OpenVPN, TensorFlow, Juniper firewall
- AMIs organize in four collections:
- - Quick start set
- - Custom
- - AWS marketplace
- - Community

Using quick start AMIs
- 3 dozen or so most popular AMIs
- Readily available through quick start tab
- - chooses and Amazon machine image page
- - First page choose launch instance button in EC2 dashboard
- mostly free-tier eligible

Available Linux Distros
- LTS releases mostly
- Eligible for security and functional updates for at least five years
- FOSS OS always free
- others (Windows server, RHEL) -> licensing charges
- - charges billed through AWS account

- Image specific instances
- - Deep learning
- - Container Hosting
- - .net core
- - Microsoft SQL server

Using AWS marketplace community AMIs
- AWS marketplace
- - software store managed by Amazon
- - Vendors make products available to Amazon customers as ready-to-run AMIs
- - Also Cloudformation stacks and containers

SAP, ORACLE, NVIDIA
- Package software solutions to AMIs

Select Marketplace AMIs
- Show total billing cost
- Separate software license
- EC2 infrastructure
- Limit to free trial
- Reduced annual subscription rates
- informal - end user responsible for security and reliability

Create your own AMIs
- Convert EC2 instance into AMI
- Create snapshot from EBS volume used w/instance
- create image from snapshot
- resulting image available as AMI:
- - AMI menu on EC2 dashboard
- - AMIs tab in ‘choose and Amazon Machine Image’

Understanding EC2 instance types
- description of intial hardware resources your instance will be using
- use instance type definitions to figure out best match for your application
- vCPU - ‘mysterious and arbitrary’ metric
- - Describes compute power from a given instance
- - Notoriously difficult to map a value of single vCPU against any on real world device
- Not always true that ‘more vCPUs and memory, the better’
- EC2 -> instance type families optimized for very different tasks
- T2, T3, M5: included in general purpose family types

Family	Instance Types
General Purpose	A1, T3, T2, M5, M5a, M4, T3
Compute Optimized	C5, C5n, C4
Memory Optimized	R5, R5a, R4, X1e, X1, High Memory, z1d
Accelerated Computing	P3, P2, G3, F1
Storage Optimized	H1, I3, D2

- Choosing correct instance may allow for fewer cpus and memory = lower cost
- Dedicated instance/host -> good for physically isolated instances

Dedicate instance/host
- Dedicated Instance
- - Regular EC2 instance
- - runs on isolate host
- - set aside exclusively for account
- Dedicated Host
- - Instance Isolation
- - Allows higher level of control
- - - how instances will be placed and run within host environment
- - - simplify and reduce costs of running licensed software
- both levels
- - extra per-hour pricing
- - available using:
- - on-demand ; reserved ; spot

Server storage: Elastic Block Store and instance store volumes
- Storage volumes holding OS and data will be virtual
- some instance types:
- - only volumes from Elastic Block Store (EBS)
- - instance store volumew
- - both

Amazon Elastic Block Store
- Physical driver where EBS volume exists may be far away from physical server giving instance life
- EBS volumes speak with instance over super-low lateny network connections
- EBS - EC2 responses not as good as EC2 instance store volumes

EBS offsets
- data stored on EBS volumes survive shutdowns and system crashes
- EBS volumes can be encrypted
- EBS volumes can be moved around

EC2 Instance store volumes
- enjoy benefits of having data on physical drives attached directly to physical instance server
- downsides of instance store volumes:

- - ephemeral data
- - no encryption
- - lack of flexibility
- - offset by faster data reads and writes

Understanding EC2 pricing models
- on-demand instances
- - most expensive tier
- - pay for every hour instance running regardless of actual uses
- - great for workloads that run for a limited time with no interruption

- reserved instances
- - better for run uninterrupted for more than a month at a time
- - unlikely to find reservations less than twelve months
- - 2 -3 year contract norm
- - actually paying for right to run instance at specified cost during reservation term
- payment types
- - all upfront
- - partial upfront
- - no upfront
- - pay more upfront, cost less overall

- Spot instances
- - Don't need to run contantly
- - surviving unexpected shutdowns
- - unused compute capacity available at steep discounts
- - capacity can be reclaimed with two minute notice
- - can be automated
- - - defining capacity -> using spot fleet
- - - pricing needs -> using spot fleet
- can be integrated into sophisticated multi-tier operations deeply integrated with other automation deploymet tools

Simplified Deployments Through Managed Services

Managed services simplify deployment by handling underlying infrastructure

More expensive

Relational Database Service (RDS)
- Lets youset basic configuration parameters for database engine
- Gives endpoint address through which applications connect to database

- Takes care of details invisibly
- End user only makes top-level configurations

Managed Deployment
- one step beyond managed service
- whoe stack taken care of behind the scenes
- only need application code or content

Managed Deployments
- Amazon Lightsail/ Elastic Beanstalk

Lightsail
- Four low stress ways to enter cloud world
- - Blueprints
- - - automatically provisions all compute, storage, database and network resources
- pick pricing level and add startup script
- Lightsail uses all aforementioned - AMIs, etc - to convert plans to reality
- Move stack to EC2? -> all covered

Blueprint:
- OS: Amazon, Linux, Ubuntu, Debian, FreeBSD, OpenBSD, Windows Server
- Applications: Wordpress, Magento, Drupal, Joomla, Redmine, Plesk
- Stacks: node.js, gitlab, LAMP, MEAN, NGINX
- Flat monthly rate

AWS Elastic Beanstalk
- Even simpler than Lightsail
- Define application platform
- Upload code
- only two steps required of end user

Preconfigured environments
- Go
- .net
- java node.js
- docker containers -> code define and specifically formatted wtih dockerrun.aws.json files
- generates costs according to how resources consumed
- not get to choose how many vCPUs or memory consumed
- App scales resource consumption according to demand
- variations in demand determine billing

Deploying Container and Serverless Workloads

Virtualized servers tend to be resource hungry

Containers
- avoid overhead by allowing individual containers to share Linux kernel with physical hostso
- share layers with other containers

Amazong Elastic Container Service
Amazon Elastic Container Service for Kubernetes
- used to orchestrate swarms of Docker containers on AWS using EC2 resources

Serverless Functions
- Uses smaller resource footprint than containers
- not require OS kernel
- spring into action, perform some task, die within minutes
- AWS Lambda
- - Amazon serverless service
- - Looks like Elastic Beanstalk
- - Define function by setting runtime environment
- - - node.js
- - - .net
- - - python
- - - upload code
- Lambda only runs when triggered by preset event
- time out after 15 minutes
- scales to meet demand

Chapter 8: The Core Storage Services

Exam Domains
Domain 2: Security
2.2 Define AWS cloud security and compliance concepts
2.3 Identify AWS access management capabilities
Domain 3: Technology
3.1 Definte methos of deploying and operating the AWS cloud
3.2 Identify the Core AWS services

Chapter Summary

S3 - primary storage service in AWS
- integrates w/ all other AWS services
- Works especially well with S3 Glacier, EC2 Lambda
- Great for durable, highly available cloud storage
- - modify with bucket policies
- - for long term storage or infrequently accessed data
- - - S3 Glacier
- Local storage: AWS storage gateway
- use V.M. that syncs w/S3
- Snowball best for getting large amounts of data to and from S3
- - rugged, tamper-resistant device
- Snowball edge:
- - Same functionality as snowball
- - Also functions as local file server
- - User NFSv3 and v4 protocols
- - run EC2 instances
- - run Lambda function

Exam Essentials

Understand the difference between durability and availability in S3
- Durability: object not lost over course of year
- Availability: Percentage of time object accessible during year

Be able to select the best S3 storage class given, compliance, and availability requirements
- Six storage classes
- Standard highest available (99.99%)
- Replicates objects across three zones
- Most expensive -> monthly storage costs / gig
- Onezone_1A
- - Lowest available @ 99.5%
- - Stores object in one zone
- - Monthly per-gig storage less than half of the STANDARD storage class

Know the Different Options for getting data into and out of S3
- Upload/Download objects using S3 service console
- - Using AWS CLI
- - Directly accessing object's url
- AWS storage gateway
- - Lets on-prem servers use industry-standard services
- - iSCSI; NFS; SMB
- Snowball and Snowball edge
- - Allow secure physical transport of data to and from S3

Understand when to use bucket policies, and access control lists in S3
- Bucket policies/ACL's
- - Grant anonymous access to objects (webpages, images, etc...)
- - User policies to grant specific IAM policies

Be able to explain different between S3 and Glacier
- S3
- - Highly-Available, real-time retrieval of objects
- Glacier
- - Two step onthly for retrieval
- - request archive using Standard, Expediter or Bulk retrieval options
- - Download archive once retrieval complete

Know how to use encryption, versioning, and object life cycle configuration in S#
- Server-side/ Client-sid encryption
- protect object at rest
- versioning: protects against object server overwrites and deletions
- Object life cycle configuration: lets you delete or move objects after they reach a certain age

Understand the three virtual machine types offered by AWS Storage Gateway
- File gateway: offers access to S3 via NFS and SMB storage protocols
- Volume and tape gateways: offers access via iSCSI Block storage protocol
- Tape gateways: Designed to work with common backup applications

Introduction

Data often have data split onthly cloud and on-premises

Understand different options for storing data included

Transferring between cloud/on-premises

Doing both cloud and on-premises

cost-effective

Performance-effectiveness

Simple Storage Services (S3)
- Flagship cloud storage service
- store/retrieve unlimited amounts of data
- part of ecosystem
- Available to other AWS services

S3 Glacier
- Long-term archiving Like backups

AWS Storage gateway
- Virtual appliance
- Seamlessly moves data back and forth between on-premises servers and S3
- Industry standard protocols

AWS Snowball
- Hardware storage appliance
- Designed to move massive amounts of data between onsite and S3

Simple Storage Service

Store and retrieve unlimited amounts of data from anywhere in world

Store any kind of file

Use access controls and encryption:
- restrict access to files to specific individual and IP address

Many uses outside and inside AWS

Many services:
- Use S3: Store logs and retrieve data
- host static websites

Objects and Buckets
- S3 is not EBS
- S3 stores files: “objects on disks in AWS data centers
- Any file: text, image, video, database, etc…
- can be up to 5 tb
- filename = ‘key’
- - up to 1,024 bytes
- - consists of onthly lity including !-_ . *()
- Stores objects in container called ‘bucket’
- - bucket = ‘flat file system’
- Bucket stores unlimited objects
- Bucket needs name = 3 – 63 characters, name globally unique
- within bucket:
- - object key unique
- - make easier to access objects using S3 endpoint url
- buckets only exist in one region
- - security, cost, compliance requirements
- cross – region replication
- - copy contents to other buckets
- - two distinct copies
- - AWS never moves objects between regions
- - organizes objects into file system with forward slash as part of name
- not charged for upload
- maybe charged for download
- monthly fee for storage size and class

S3 Storage Classes
- Some data require more availability
- Some data require more durability

Durability and reliability
- Durability: percentage likelihood data not lost over cost of year; greater durability = less likely data will be lost
- Availability: percentage of time object will be available for retrieval
- Level of Durability and availability depends on storage class

Storage Class	Durability	Availability	Availability Zones	Storage Pricing in the US East Region (pre GB/month)
STANDARD	99.999999999%	99.99%	>2	$0.023
STANDARD_IA	99.999999999%	99.9%	>2	$0.0125
INTELLIGENT TIERING	99.999999999%	99.9%	>2	Frequent Access Tier: $0.023 Infrequent Access Tier: $0.0125
ONEZONE_IA	99.999999999%	99.5%	1	$0.01
GLACIER	99.999999999%	Varies	>2	$0.004
REDUCED REDUNDANCY (RRS)	99.99%	99.99%	>2	$0.024

Generally want highest degree of durability Available
Availabilty is the next considertation (how frequently you want to access files)

Three Access Patterns for Files
- Frequently accessed objects
- Infrequently accessed objects
- Mix of Frequent and Infrequent access

Storage classes for frequently accessed objects
- Frequently access with minimal latency
- - Standard: default: highest levels of durability/availability; objects replicated across at least three availability zones
- - Reduced Redundancy: Meant for data that is easily replaced; lowest durability of all classes; same availability of standard. AWS recommends against using this storage class

Storage classes for infrequently accessed objects
- IA: infrequent access
- millisecond-latency
- high durability
- lowest availability
- for objects at least 128 kb in size

STANDARD_IA: for important data that can't be recreated; objects sotred in multiple onthly lity zones; onthly lity of 99.9%
ONEZONE_IA: One onthly lity zone, lowest availability; 99.5%; outage of one availability zone could affect availabity of objects stored inzone; used for data that can be re-created
GLACIER: Long-term archiving of objects that rarely need to be retrieved; can't retrieve objects in real time; initiate a restore request and wait until the restore is complete
- - Restore time depends on retrieval option: 1 min to 12 hours
- - Availability of data stored in Glacier varies

Storage classes for frequent and infrequent access
INTELLIGENT_TIERING: auto-moves objects; most cost-effective storage tier based on past onthl patterns
- - if objects are not accessed for 30 days they are moved to the lower-cost infrequent access tier
- - If objects are re-accessed: moved back to frequent access tier
- - objects greater than 128 kb always charged at higher cost frequent access tier rate
- - charged onthly monitoring/automation fee

Access Permissions
- Bucket policies
- User policies
- Bucket and object access control lists

Bucket Policies
- resource-based policies applied to buckets
- - grant access to all objects
- - specific objects
- - Control which principals and accounts can read, write, delete
- - grant anonymous read access to make an object available everywhere on web

User Policies
- Grant IAM principals access to S3 objects
- User policies can only be applied to IAM principal
- Can't grant public anonymous access to an object

Bucket and Object Access Control Lists
- Legacy access control methods superseded by bucket/user policies
- use bucket/object acl's to grant other AWS accounts and anonymous users access to S3 resources
- Grant access to IAM principals
- Use bucket and user policies instead of ACLs
- Use any combination of bucket policies, user policies and ACLs

Encryption
- S3 does not change contents of object (stored unenctrypted
- S3 can encrypt objects before storing them in S3 (encryption at rest)
- Serverside encryption
- - S3 encrypts object and saves only content
- - easiest to implement
- - Doesn't require keep track of encryption keys
- - Amazon manages keys and has access to your objects
- Clientside encryption
- - Encrypts prior to uploading to S3
- - Decrypts when retrieved
- - More complicated
- - User responsible for encryption and decryption
- - Lose keys: not able to decrypt
- - This method prevents Amazon from reading encryption

Versioning
- creates new copies of saved files if changed
- Original stays intact
- If object is deleted, contents of object are not deleted -> S3 adds a delete marker to object and hides it from the S3 service console view
- Disbaled by default when bucket created
- Must be explicitly enabled on bucket
- no limit to number of versions
- Delete manually or automate using lifecycle configuration
- deleting bucket equals deleting objects

Object lifecycle configurations
- Help control costs:
- - auto move object to different storage class / delete after a certain time
- applied to bucket -> one or both types of actions
- Transition actions: move objects to different storage after reaching certain age
- Expiration actions: auto deletes after object reaches a certain age; with versioning enabled: delete version after certain age
- Can also use them together

S3 Glacier

Long-term archiving of infrequently accessed data at low cost
- 99.999999999% durability over one year
- Samed as GLACIER storage class in S3

Archives and Vaults
- store files in archives = block of information
- usually not a single file: make .zip/.tar and upload as archive: 1 byte to 40 TB
- Glacier stores archives in a vault

Vault
- region specific container
- names unique within region
- does not need to be globally unique
- create and delete with Glacier service console
- need AWS CLI for upload and download objects
- - can write own code with SDK
- - 3rd party programs
- - - Cloudberry Backup
- - - FastGlacier
- - - Arq Backup

Retrieval options

- designed for long-term archiving
- no real-time access to archives
- downloading is a two-step process
1- initiate job retrieval
2- download data once complete
3- Length of time for job completion depends on retrievel opiont
- - expedited: usually complete in 1 – 5 minutes; archives greater than 250 MB take longer; US EAST: $0.003/gig
- - Standard: default option; 3 – 5 hours; US EAST: $0.01/gig
- - Bulk: lowest-cost option; 5 – 12 hours; US EAST: $0.0025/gig

AWS Storage Gateway

Makes it easyto connect on-premises storage to AWS cloud

Uses industry-standard protocols

Provision AWS Storage Gateway V.M. on-premises and connect servers to iit

Storage Gateway handles data transfer

V.M. can run on VMWare ESXi or Microsoft Hyper-V hypervisor

3 VM Types:
- file gateways, volume gateways, tape gateways

File gateways
- can use NFS and SMB protocols
- - store data in S3
- - data caches locally; low-latency access
- - can function as normal on-premises server
- - data stored in S3 – can take advantage of all S3 features: versioning, bucket policies, life cycle management, encryption, cross-region replication

Volume gateways
- S3-backed storage volumes
- on-premises access via internet Small Computer System Interface protocol
- two configurations
- - stored – stores all data locally; asynchronously backups to S3 as Elastic Block Store snapshot
- - good option if you need uninterrupted access to data
- - 1 GB to 16 TB in size

Cached Volumes
- Stores data in S3
- Frequently used subset of data that is cached locally

- 1 GB to 32 TB

Both configuration
- take manual or scheduled EBS snapshots of volumes

Tape Gateways
- Mimics traitional tape backup infrastructure
- works with Veritas Backup Exec, NetBackup, Commvault, Microsoft Systems Center Data Protection Manager
- Connects via iSCSI
- create virtual tapes between 100 GB and 2 TB
- Stores virtual tape library backed by S3
- Backup software writes data to tape
- Tape Gateways asynchronously uploads data to S3
- data kept in cache storage until it is complete
- if need to recover, tape gateway downloads from S3-backed VTB and stores in buckets
- for cost effective long-term storage
- - move out of VTL and into virtual tape shelf backed by Glacier
- - initiate retrieval request; takes up to 3 hours
- - complete: virtual tape available in s3-backed VTL; available to transfer to tape gateway

AWS Snowball

Hardware APP device

Moves massive amounts of data

Examples
- migrate Data from office to cloud
- quick transger large amounts of dat to and from S3 Back Up
- Distributing large volumes of content to customers and partners

Idea of Snowaball
- Quicker to physically shipe larges amounts of data then transfer over network
- nominal fee for device
- transfer files and ship ack
- AWS receives and transfers to 1 or more S3 buckets

Hardware specs
- Largest: 80 TB; $250; store up to 72 TB
- 50 TB; $200; store up to 42 TB
- RJ45 and SFP+ network interfaces/ 10 GBPS
- keep up to 10 days
- - after this, $15/day
- - can keep up to 90 days

- - use snowball to export data from S3

Security
- Snowball contained in rugges, tamper-resistant enclosure
- TPM chip
- two layers of encryption
- - to or from transfer: SSL
- - Encrypted at rest
- - enforces encryption by only allowsing transfer via Snowball client/SDK adapter
- - uses AES 256-bit encryption
- - AWS erases data before sending it to another customer
- - uses NIST media sanitization standards: special publication 800-88

Snowball Edge
- Like Snowball
- wider variety of features
- adds QSPF+ port to network connectivity: faster network speeds than Snowball
- only transfers between local environments and S3
- same functionality as Snowball plus:
- - Local storage for S3 buckets
- - Compute power for EC2 instances and Lambda functions daily
- - File server functionality usint the Network File System version 3 and 4 protocols

Three different device options
- Storage optimal
- - 80 TB usable storage
- - 24 vCPUs
- - 32 GB memory
- - QSPF+ up to 40 GBPS

- Compute Optimized
- - Most Compute power
- - 52 vCPUs
- - 208 GB memory
- - 39.5 TB usable storage
- - 7.68 TB dedicated to compute instances

- Coompute optimized with GPU
- - Identical to compute optimized
- - NVIDIA V100 tensor core gpu
- - machine learning and high-performance compute APIs
- - QSPF+ network interface capable of up to 100 GBPS

- can cluster 5-10 Snowball Edge devices together
- - build highly available compute or storage cluster
- - not support virtual private clouds
- - not place EC2 instance running snowball edge in vpc

- - - assign instance IP address on local network

Comparison of Snowball and Snowball Edge

Feature	Snowball	Snowball Edge
Transfer data to and from S3	Yes	Yes
Local EC2 instances	No	Yes
Local compute with Lambda	No	Yes
File server functionality using NFS	No	Yes
Local S3 buckets	No	Yes

Chapter 9: The Core Database Services

Exam Domains
Domain 3: Technology
3.1 Define methods of deploying and operating in AWS cloud
3.3 Identify the core AWS services

Chapter Summary

Using a relational or non-relational database: in most cases, the choice has already been made for you
- If migrating a database-backed application from your data center to AWS, most likely application using SQL
- If so:
- - Use REDS
- - Build and maintain SQL server on one or more EC2 instance

- If developing a new database-backed app:
- - Relational vs. non-relational database: not an easy decision, not clear-cut

Relational	Nonrelational
Designated for complex or arbitrary queries	Designed for a few well-defined queries
Requires structured data	Can store structured or unstructured data
Ideal for reporting and analysis	Ideal for highly transactional applications

- Trade-off for storing unstructured data: non-relational databases are more limited in queries
- Structured database stores equals greater search flexibility
- - can search for any query imaginable, including attributes and ranges
- Non-relational databases (DynamoDB):

- - scale horizontally
- - spreads data across more partitions
- - thousands of reads/writes per second
- Relational Database (RDS)
- - Can scale horizontally, but not feasible to do so
- - Better to scale vertically by upgrading to more powerful instance/class

Exam Essentials

Understand the major differences between rational and nonrelational databases
- Relational
- - Designed for structured data
- - Contains a specific number of attributes/record
- - Complex queries again a variety of dimensions
- - Ideal for reporting and analytics
- Non-relational
- - Designed for data that follows a predictable structure
- - Each item in nonrelational database has a key
- - Queries are based on keys

Know Vertical and Horizontal scaling options for RDS
- Can scale RDS instance vertically by upgrading to a larger instance calss
- more processing power, memory, disk, network, select provisioned IOPS SSD storage
- For horizontal scaling of reads, must use read replicas

Be able to describe the components of RDS
- deployment consists of at least one instance
- select instance class: vCPU and memory
- select D.B. engine
- storage
- - general purpose or provisioned IOPS SSD
- - magnetic storage: legacy option not available on new deployments
- - can also read replicas to scale horizontally to improve read performance
- Multi-AZ deployment: can add additional secondary instances that primary synchronously replicates data to

Know the backup and recovery options for RDS
- Can schedule automated snapshots for RDS
- Can occur daily during 30-minute backup window of choice
- Backups retained between 1 and 35 days
- Automatic backup enables point-in-time recovery
- - allows to restore database up to 5 minutes before fail
- - restore a snapshot = create a new instance
- - Can also take a manual snapshot at any time

Understand how DynamoDB stores data
- Stores data as items in tables
- Each item has a primary key
- - Values are unique within table : how DynamoDB uniquely identifies an item
- Primary key name and data made when table created
- When an item is created, other attributes and primary key also created
- Primary key distributes items across different partitions
- Number of partitions/table depends on number of WCU and RCU configured

Be able to identify scenarios for using Redshift
- Data warehousing service for storing and analyzing structure data from multiple sources
- Includes relational databases and S3
- Can store much more than RDS up to 2 PB

Introduction

Use AWS CLI to provision DB
AWS handles installation and maintenance of Database software
AWS handles Data backup and replication

Three different Database services
- Relational database service (RDS)
- DynamoDB
- Redshift

Database Models

Relational model

Non-relational model

- model chosen depends on how application needs to store, organize, retrieve data
- - Determines model chosen
- - Needs of application determines database model

Relational Database
- Analogous to Spreadsheet: columns/rows
- Been around a long time
- columns = ‘attributes’
- rows = ‘records’
- table = stores attributes and records
- database contains multiple tables

- - Defined in columns
- Each row must be unique in database table
- Primary key ensures uniqueness of rows
- predefine types of data that can be stored in columns
- good for data: predictable, well-defined format
- quickly searchable
- - specific values
- - retrieve data from different tables
- - combining into many forms
- - ideal for complex analytics
- - good for making reports agains large data sets
- the larger the database = longer retrieval times

Structured query language
- used by databases
- - create statements using SQL statements
- - read/write data
- - can perform tuning and maintenance tasks
- select statement
- - reads data from database
- - controls formatting
- - combining from different databases: add join clause to select statement
- - must execute one select statement when retrieving data
- Insert statement
- - writes data to table
- - app writes to database: use insert

Non-relational (NOSQL) DB
- good for tens of thousands of searches per second
- good for storing weird data structures
- - unstructured data
- - schemaless
- also store info in tables
- - sometimes called ‘collections’
- - each row or record called an ‘item’
- don't require to specify in advance all types of data
- only need i.d. of primary key
- queries are more limited
- greater flexibility
- Designed to query based on primery key
- best suited for apps that only need to perform just a few well-defined

Amazon Relational Database Service

Amazon's managed relational database service
- lets you provision popular RDBMs- relational database management systems
- - Microsoft SQL server, Oracle, MySQL, PostGRE SQL
- can install and configure database server on EC2 instance
- RDS offers advantages
- - Amazon sets up multiple compute instances
- - Takes care of installing and configuring RDBMs of choice
- - compute instances not EC2 that can secure SSH into
- - - connected to vpc of choice
- - - allows application to take full advantage of RDS-hosted database
- - - RDS use Elastic Block Service (EBS) volumes for storage
- - - can choose multi-availability zone for multiple deployments
- - - manual or automatic EBS snapshots: easily restore to new RDS instances
- - - will handle patching and upgrades

Database Engines
- Created RDS instance, choose database engine
- Only one DB per instance
- six database engines:
- MYSQL, MariaDB, Oracle, PostGRESQL, Microsoft SQL Server, Amazon Aurora
- - All except Aurora: open source or commercially available
- - Aurora is proprietary to Amazon, compatible with MY/POSTGRE - SQL

Licensing
- Two licensing options:
- 1 License included
- 2 Bring your own license

1 - included in pricing for each RDS instance
- Microsoft SQL and Oracle Database offer this license type
- MariaDB, MYSQL, POSTGRESQL use this license type exclusively

2 - Must provide own license
- License costs not built in
- Oracle only

Instance Class
- Implementing relational database
- - capacity planning
- - needed availability and performance
- number of vCPUs
- amount of memory
- Max network and storage throughput

- there are three types of vCPUs: Standard, Memory Optimized, Burstable Performance
- Standard
- - meets requirements for most apps
- - Between 2 and 96 vCPUs
- - 8 - 384 GB of memory
- Memory Optimized
- - for the most demanding database requirements
- - most disk throughput
- - most network bandwidth
- - between 4 and 128 vCPU
- - 122 - 3,904 GB of memory
- Database instances use EBS storage
- EBS-optimized
- dedicated bandwidth transfers to and from EBS
- Burstable Performance
- - non-production D.B.
- - Minimum performance requirements
- - between 2 & 8 vCPUs
- - 1 - 32 GB memory

Scaling Vertically
- Changing way resources allocated to specific instance
- scale up to more power instance class
- - add memory
- - improve compute
- - improve networking
- - scale down = save costs

Storage
- types of storage affect performance
- new RDS instances use EBS volume store
- max throughput funcion of:
- - instance type
- - number of in/out operations per second (IOPS)
- - higher IOPS means faster read/write

Three types of storage
- General purpose SSD
- Provisioned IOPS SSD
- Magnetic

General-Purpose SSD
- good enough for most DataBases
- allocation volumes from 20 gb to 32 tb
- number of IOPS/volume allocated for storage
- more storage = better read/write
- using bursting to temporarily achieve high number of IOPS

- If there are heavy read/write spikes: bursting kicks in

Provisioned IOPS SSD
- Specify exact number IOPS (in thousands) per volume
- allocate up to 32 TB
- does not offer bursting
- can adjust IOPS amount later

Magnetics
- available for backwards compatibility with legacy RDS
- doesn't use EBS
- cannot change size after creation
- only 4 tb
- only 1,000 IOPS
- Increase size of EBS does not affect performance
- Decreaase size of EBS does affect performance
- migrate to other storage type; possible temporary performance suffers
- magnetic uses EBS
- possibly serveral days to finish
- - still usable, but may not be optimal

Scaling Horitontal with Read Replicas
- Read replica
- - additional instance of RDS
- - performs only reads to D.B. (only master performs writes)
- - improves performance of data-base backed applications
- RDB -
- - only master can write to d.b.
- read replica
- - removes burden of read-only queries
- - master instance can focus on writes exclusively
- - best benefit for read-intensive applications
- - good for compute heavy inquiries

High Availability with multi-AZ
- ensures multiple instances of master db called standby instances
- - runs in different a.z. than primary
- - primary replicates data instantly to second instance
- - every time data writes to main, replicates to second instance
- - if primary fails
- - - RDS fail over to secondary
- - - failover up to two minutes
- - - monthly availabilty 99.99%
- - - instance outage:
- - - - patching
- - - - instance upgrades

Aurora
- Additional benefits when using multi-AZ
- RDS instances are part of Aurora cluster
- All instances use shared storage volume
- - synchronously replicated across three different AZ's
- - Cluster volume can auto expand up to 64 TB

Backup and Recovery
- Whether use/not use multi AZ:
- - RDS can make manual/auto EBS snapshot
- - snapshots stored acros multiple AZ's
- Restore snapshot
- - RDS restores to a new instance
- - useful for backups
- - useful for making copies for testing or development purposes
- Can take a manual snapshot at any time
- Can configure automatic snapshots to occur during a 30-minute backup window
- RDS
- - auto snapshot retained for 1 - 35 days; default of 7 days
- - manual snapshots retained until deleted

Auto Snapshots
- enables point-in-time recovery
- saves database change logs every 5 minutes
- with auto snapshot can restore up to 5 minutes before failure

Determining your Recovery Point Objective
- Do you need snapshots and multi-AZs?
- - snapshots and multi-AZs serve different purposes

Snapshot
- restores entire DB instance
- recover data

Multi-AZ
- Keep DB up and running in event of failure
- Data Synchronized to secondary instances

Recover Point objective
- How much data loss sustained in event of failure
- Example: Can tolerate losing one hour of data? RPO would be one hour
- - automatic snapshots with point-in-time sufficient
- For RPO of less than 5 minutes:
- - use multi-AZ to synchronously replicate data to secondary instance

DynamoDB

Managed nonrelational DB service

Highly transactional: very high reads/writes

Items and Tables
- Item is basic unit of organization
- - analogous to row or record in RDB
- items stored in tables
- table stored across one or more partitions
- - partitions backed by SSDs
- - partitions replicated across multi-AZ/region
- 99.99% monthly availability
- Each item has unique value for primary key
- Attributes = items key/value pair
- each item can store up to 400 kb data
- item only needs a primary key
- every attribute must define data type
- - can be one of following:
- scalar: only one value; can be string, number, binary data, Boolean
- set: multi-scalar values; each value must be unique within set
- Document: List and map; lists are ordered; maps are not
- - store value of any type; good fro structured values
- - DynamoDB - can recognize and extract specific values nested within a document

Scaling Horizontally
- DynamoDB:
- - uses primary key to distribute items across mutliple partitions
- - enables low-latency reads/writes regardless of number of items/table
- Write capacity units (WCU)
- Read capacity units (RCU)
- - determines number of partitions
- higher transaction values = higher wcu/rcu
- higher values = higher distribution and lower latency
- change wcu/rcu if demand changes
- auto-scaling = auto-change wcu/rcu
- - ensures consistent performance

Queries and Scans
- nonrelational db:
- - quickly retrieve item from table based on value of primary key
- - search for value on attribute not primary key is possible but much slower

Amazon Redshift

Specialized type of RDB
- ‘data warehouse’

Data warehouse:
- stores large amounts of structured data
- allows complex queries/analysis

Create cluster:
- at least one compute node and up to 128 nodes

Dense compute nodes
- can store up to 326 TB data on magnetic disks

Dense storage nodes
- up to 2 PB data on SSD

Redshift spectrum:
- Analyze data stored in S3
- data must be structured
- data must be defined so Redshift understands it

Chapter 10: The Core Networking Services

Exam Domains
Domain 3: Technology
3.1 Define Methods of Deploying and Operating in the AWS Cloud
3.2 Define the AWS global infrastructure
3.3 Identify the core AWS services

Chapter Summary

Virtual Private Cloud
- Provides virtual network infrastructure for many resources
- Most notable: EC2

VPC can connect to other networks:
- the internet via an internet gateway
- external private networks via direct connection or a virtual private network
- other VPCs using VPC peering

Route 53:
- Two distinct but related DNS services
- Register for many top-level internet domain names
- Transfer/register domain you control
- Provides DNS hosting services
- Use route 53 with a public domain
- - create public hoste zone
- - use rte 53 for name resolution within VPC, create private hosted zone

Cloudfront
- Content Delivery Network
- Improves Delivery of data to end users by storing content in edge locations

Exam Essentials

Know the components of a VPC
- Key components of a VPC
- - at least one subnet

- - security groups
- - network access control lists
- - internet gateways

Understand the different options for connecting to resources in a VPC
- Connect to resources in a VPC
- - over internet
- - direct connect link
- - VPC peering connection
- - VPN

Understand the difference between a Route 53 public hosted zone and a private hosted zone
Public
- Allows anyone on internet to resolve records for associated domain name
Private
- Allows resolution only from resources within associated VPC

Be able to select the best Route 53 routing policy for a given scenario
- All routing policies except simple
- - use health checks to route around failures
- - to direct traffic to any available resource:
- - - Failover, Weighted, Multivariate Answer routing policies
- - - Latency policy - for performance
- - - Geolocation policy - for directing users based on specific location

Know how CloudFront improves the speed of content delivery
- Cloudfront
- - Caches objects in edge locations
- - Auto directs users to edge location that will give the best performance at any time

Be able to identify scenarios where Cloudfront where would be appropriate
- Cloudfront
- - Designed to give users fastest possible access to content regardless of physical location
- - Caches content in edge locations distributed around the world
- - Helps ensure content is alwasy close to users

Introduction

Networking:
- Tranporting data
- - Factors:
- - - data type
- - - speed data transport
- - - security requirements
- - - who/what accessing

Three Core Networking Services:
- Virtual Private Cloud
- Route 53
- Cloudfront

Virtual Private Cloud

Network Backbone for many AWS services

Virtual Private Cloud
- Virtual Network AWS Cloud
- Logically isolates from other networks

Most Well-known use
- Connecting EC2 instances together and to other AWS services/networks, and internet

Create AWS account
- Amazon auto creates VPC in region
- Default VPC
- - configured to allow instances within VPC access to network

Can Create non-default VPC
- Fully isolated from every other network and AWS resources
- includes VPCs
- Must be configured explicitly if you want to have access to other networks outside VPC

VPC CIDR blocks
- Each VPC requires CIDR block
- - defines range of IPv4 VPC resources can use
- - Default VPCs have CIDR: 172.31.0.0/16 (172.31.0.0 to 172.31.255.255)
- - must choose CIDR between /16 and /28
- - smaller CIDR: greater the number of IP addresses
- - CIDR block examples:
- - - 10.0.0.0/16 (10.0.0.0 - 10.0.255.255)
- - - 192.168.0.0/24 (192.168.0.0 - 192.168.0.255)
- - - 172.16.0.0/28 (172.16.0.0 - 172.16.0.15)
-
- At your request AWS can assign IPv6 CIDR block, which will be a global unicast IPv6 with /56

Subnet
- VPC requires further subdivision into one or more subnets
- Subnet
- - logical separation and isolation of resources within same VPC
- Must define CIDR for each subnet

- subnet CIDR subset pf VPC CIDR, /16 to /28
- Each subnet exists only within one availability zone
- Each EC2 instances exists within subnet
- - This is why within default VPC, Amazon creates default subnet in A.Z.
- - Enables launching of EC2 instance without ever having to configure VPC

Internet Access
- Internet Gateway
- - VPC resources
- - Allows EC2 instances obtain public IP address and access internet
- instances in subnet have internet access
- - subnet must contain default Route to internet gateway that's attached to the VPC

Public subnet
- subnet within default route to internet Gateway
- Each instance has a public IP address
- Launch instance: choose AWS to auto assign
- - can't reassign
- - when stop/terminate instance, ip is lost
- can allocate elastic ip
- - assignt to instance
- - can be reassigned to different instances and don't change until deallocated

Security Groups
- Firewall determines if traffic will pass/not pass
- each instance must have at least one security group attached
- consists of inbound/outbound rules

Inbound rules
- control what ip addresses can send traffic to instances

Outbound rules
- control what ip addresses and instance may send traffic to

By default:
- sec. groups do not contain inbound rules
- - ensures no unsolicited traffic
- - must create any inbound rule

By default:
- outbound rule allows access to any ip address
- note:
- - when instances sends traffic out, the security groups allows reply traffic to reach the instance, regardless of what inbound rules are configured
- Every VPC contains default security group: can modify

Network Access Control Lists

- Firewall operates at subnet level
- consists of inbound/outbound rules: by default allow all traffic
- Can't restric traffic between instances in same subnet
- can prevent traffic entering/leaving subnet
- Each VPC has default NACL: can be associated with one or more subnets

VPC Peering
- VPC peering connections
- - Private, point to point connection
- - only two VPCs
- - Allow resources from different VPCs communicate over private AWS network instead of internet
- - Allows instances in one VPC to access resources other VPCs
- - Fast, reliable secure
- - no need to use internet access for VPC peering
- - peered VPCs: same or different regions

Virtual Private Networks
- Allows VPC to connect to external network via secure connection
- Set up VPN:
- - create virtual private gateway
- - attach it to VPC
- - next configure customer gateway
- - - connect to VPC

Customer Gateway Types
- Cisco
- Juniper
- Palo Alto
- Checkpoint
- encrypted using AES-128/256
- Ip routing configured statically
- also use Border Gateway Protocol to share routers between VPC and external networks
- single VPC: up to 10 VPN connections

Direct Connect
- Provides private network connectivity to VPC and public services
- No need to have separate internal circuit
- can bypaass internet altogether to access AWS resources
- doesn't provide internet access; still need separate internet connection
- offered through AWS Partner Networks (APN) networks
- operates using dedicated link: 1 - 10 gbps
- not subject to high or unpredictable latency of broadband
- fast, consistent: most expensive
- if need less than 1 gbps, obtain hosted Direct Connect connection from APN partner
- 50, 100, 200, 300, 400, 500 Mbps

Route 53

Amazon's global domain name system service

Name resolution
- translatin domain name to ip addressess

Name resolution more than just mapping domain names to IP addresses
- store mappings for different data types
- - IPv6
- - Mail servers
- - Arbitrary text

Resource records
- for dns in domain
- several fields:
- - most important
- - name
- - type
- - value

Domain Name Resolution
- Publid Domain
- - Anyone on internet can resolve
- - must be registered with registrar (to prevent duplicate)
- - Under top-level domain: .com,.net,.org
- Registering domain:
- - Control for life of lease (1-10 year increments)
- - can renew in yearly increments
- - can transfer to Route 53
- - transferring a domain entails extending registration by at least one year

Registering/hosting:
- two different functions
- - registering: gives control of domain for life of lease (including right to specify hosting service)
- - registrar/hosting compan: sometimes same, sometimes not

Hosted Zones
- Route 53 hostss DNS for public domain
- - create public hosted zone
- - specify domain name
- - define resource records for domain
- - Route 53 auto creates public hosted zone
- - Route 53 provides name resolution for private domain names
- - DNS resolution for single domain name within multiple VPCs

- - - Useful for assigning user-friendly domain names to VPC resources to EC2 instances or application load balancers
- Define record in private hosted zone that points to database servers ip address
- Private IP not on internet: pick any name (no registrars)
- Name resolution for private hosted zones not available outside VPC

Routing Policies
- Case specific
- - Domain name resolved to particular ID
- - If want value of resource record change dramatically
- - Route 53 can accomplish this with a variety of policies

Routing policy types
- Simple
- - Default for new resource records
- - Map a domain name to single static value
- - Doesn't check if resource the record points to is available
- Weighted
- - Distributes traffic across multiple resources according to a ratio
- Latency
- - Sends user to closest AWS region
- Failover
- - Routes traffic to a primary resource unless it's unavailable
- - Will then route to secondary resource
- Geolocation
- - Routes user based on specific continent, country, or state
- Multivariate answer
- - Evenly distributes traffic across multiple resources
- - Returns all records sorted randomly

Health Checks
- All routing policies except simple use health checks to determine if they should route users to a given resource
- Check
- - Endpoint
- - Cloudwatch Alarm
- - Another Health Check
- - 10 or 20 second repeat

Endpoint
- Connects to endpoint monitored via HTTP, HTTPS, TCP
- Route 53 has health checkers in several regions
- Can choose which health checks are used
- Ensures endpoint reachable globally

Cloudwatch Alarm
- Route 53 can monitor health check alarm

- useful for monitoring potentially unhealth resource if experiencing latency
- useful for monitoring if endpoint has high number of connects

Another Health Check
- Monitors status of other health status of other health checks

Traffic Flow and Traffic Policies
- Use Route 53 Traffic Flow Visual Editor
- Create diagram represents a desired routine
- Diagram created
- - represents a traffic policy
- - - can save and associate with a domain name by creating a policy record
- Route 53 doesn't create individual resource record
- - Hides routing behind single policy record
- - $50/month per policy record
- - Can use same policies as normal resource records
- Can also use ‘Geoproximity’
- - Direct users to resource based on closeness to geographic location
- - Differs from Geolocation
- - not based on user's continent, country, or state

Cloudfront

Content Delivery Network
- Delivers Static and Dynamic content
- Faster than serving out of AWS region
- Cacehs content in Edge locations
- More than 150 Edge locations on continents
- Sends users to Edge location for best performance
- - Generally closest locale
- - Copies of content stored in multiple locations
- More Edge locations
- - More redundancy
- - Better performance
- - Price increase with more edge locations
- - Can't select individual locations
- - - choices:
- - - US, Canada, Europe
- - - US, Canada, Europe, Asia, Africa
- - - All edge locations
- Must create distribution
- - Distribution defines content type to cache, as well as content's origin
- Distribution types:
- - Web accessed via HTTP HTTPS
- - Real Time Messaging Protocol

Web
- Most common type
- Static/Dynamic content
- Access: HTTP/HTTPS
- Must create a specified origin as authoritative source for content
- Origin
- - Web server or public S3 bucket
- - Cannot create non-public S3 bucket

RTMP
- Delivers Streaming Video/Audio content
- Must provide a media player and media files to stream
- Must be stored in S3 buckets

Chapter 11: Automating Your AWS Workloads

Exam Domains
Domain 1: Cloud Concepts
1.3 List the different cloud architecture design principles
Domain 3: Technology
3.1 Define methods of deploying and operating in the AWS cloud
3.3 Identify the core AWS services

Chapter Summary

AWS provides many different ways to automate tasks

Automation
- Defining a task as code that system carries out

Code for AWS can be written using and imperative or declarative approach

Imperative Approach
- AWS Systems manager
- AWS Codebuild
- Codedeploy
- User data scripts w/ EC2 Autoscaling

Declarative Approach
- Service providing automation translates declarative statements into step-by-step operations
- Cloud Formation
- Ops Works for Puppet Enterprise
- Ops Works Stacks

Imperative and Declarative approaches not oppsites
- Declarative approach abstracts away step-by-step details in favor of more results oriencted, user-friencly paradigm

Configuration Management
- Form of Automation
- Empahsizes Configuration Consistency and Compliance
- Many platforms like Puppet and Chef use Declarative Language
- AWS Systems Manager provides configuration management useing the Imperative approach

Exam Essentials

Know what specific tasks AWS services can automate
- Cloud Formation
- - Automatically deploy change, delete AWS resources
- - AWS dev tools: CodeCommit, CodeBuild, CodeDeploy, CodePipeline
- - - use to help automate some or all of: development, testing and deployment process

EC2 Autoscaling
- Auto scaling set number of EC2 instances
- Optionally scale in/out according to demand or schedule

Systems Manager Command Documents
- Automate tasks against instances Operatins System
- Patching
- Installing software
- Enforcing configuration settings
- Collecting inventory
- Automate many administrative AWS tasks that normally require management console or CLI

OpsWorks for Puppet Enterprise/OpsWorks for Chef Automate
- Lets you configure instances and deploy software
- Do so using Declarative language of Puppet modules or Chef recipes

Opsworks Stacks
- Automate build and deployment of application and supporting infrastructure

Understand the Benefits of Automation and infrastructure as Code
- Automation
- - Allows common, repetitive tasks to be executed faster than doing manually
- - Reduces risk of human error
- Automate infrastructure builds using code
- - the code becomes de facto documentation
- - the code can be place in version control (easily track changes/roll back)

Be able to explain the concepts of continuous integration and continuous delivery
- Continuous integration
- - Devs regularly checking code as they create or change it
- - Automated process performs builds and tests actions against it
- - immediate feedback loop: allows devs to fix problems quickly and early
- Continuous delivery
- - Expands upon continuous integration
- - Deploys application to production after manual approval
- - Enables push-button deployment of app

Introduction

Automation
- best practice for designing architectures in Cloud

Automation benefits
- rapid testing/ experimentation
- reducing expenses
- minimizing human error
- rolling back changes safely and completely

- Automation not replacement human element
- Requires tremendous amount of ongoing human interacion
- Increases capacity of people and work
- You can deploy automation at different levels of AWS environment, picking and choosing where to apply it

Two Different approaches
- Imperative
- Declarative

Imperative
- Scripting
- Bash
- Powershell
- AWS CLI
- Focuses more on specific step-by-step instructions

Declarative
- Declares desired result
- Requires 'Intelligent Software'
- - Figures out operations require to achieve desired result
- Cloud Formation
- - Takes Declarative Approach
- - Write code containing specifics of AWS resources
- - present that code to Cloud Formation
- - Cloud Formation builds and configures

Infrastructure as Code
- Using code to define infrastructure and configurations
- The fundamental element is automation in the cloud
- Relies on code being executed by machines which reduces risk of human error

AWS services which enable automation:
- Cloud Formation: Automate building/configuration of AWS resources

- AWS Dev Tools: CodeCommit, CodeBuild, CodeDeploy, CodePipeline: Automate testing, Building, and Deployment - ECS; on-premises
- EC2 Auto Scaling: Auto launches, Configures and terminates EC2 instances
- Systems Manager: Automates Common operational tasks
- OpsWorks: Collection of three different offerings to automate instance configuration and app development; uses Chef and Puppet configuration management platform

Cloud Formation

Automatically creates and configures your AWS infrastructure
- takes from code that define the resources wanted to create and how want resources configured

Templates
- Text files that store code that defines your resources
- uses proprietary cloud formation language
- written in json or yaml format
- contain description of resources cloud formation will create
- simultaneously functions as infrastructure documentation
- because templates are files, they can be stored in version control systems: S3 Buckets/Git repos
- use same template to build AWS infrastructure repeatedly
- use single template to build many similar services
- uses random id's as naming resources to ensure that resources don't conflict
- can write templates to optionally take parameter inputs
- allows to customize resources without modifying original template

Stacks
- provision resources from a template:
- Specify stack name that is unique within your account
- Stack = container that organizes resources described in template
- Purpose: collectively manage related resources
- If stack is deleted
- - Cloud Formation auto deletes all of resources in it
- - perfect for test/development environments that need to be provisioned as pristine and thrown away when not needed
- - Ensures all resources are deleted and not left lingering

- Cloud Formation:
- - enables the creation of test environments that mirror production environments

Stack Updates
- Cloudformation
- - Changes individual resources in Stack
- - modify template: delete, modify, add resources
- - instruct Cloudformation to perform stack updates using template
- - auto update

- - - Other resources dependent: Cloudformation lets you review and implement/not-implement
- Stack Policy
- - JSON document
- - Separate from template
- - specifies what resources may be updated
- - Used to prevent any/all updates
- - Can temporarily override if need to update

Tracking Stack Changes
- Each Stack
- - Timestamped record of events
- - Includes record of resources created, updated, deleted
- - makes it easy to see all changes
- - Resources in Cloudformation Stacks can be changed manually
- - Drift Detection: Feature which monitors stacks for changes and sends alerts when they occur

Cloud Formation Vs. AWS CLI
- AWS CLI:
- - Can use Script commands to create resources
- - Resources not kept in stack: lose advantage of resources in stacks
- - can't update resources as easily as those in CloudFormation
- Cloud Formation
- - Adjust template/Parameters to change parameters
- - Cloud Formation figures out intentions and performs changes
- AWS CLI:
- - You must understand change
- - Must ensure change made to one resource does not break another resource

AWS Developer Tools

Collection of tools; lets devs develop, build, test, deploy onto EC2 and on-prem instances

Facilitate/Automate tastks that get new app revision released to public

Use as part of IaC (Infrastructure as Code)
- Automate Deployment and Configuration of AWS infrastructure

Dev Tools:
- Code Commit
- Code Build
- Code Deploy
- Code Pipeline

CodeCommit
- Lets create private Git repos that integrate with other AWS services

- - versioning and version control
- useful
- - teams of people need to collaborate on same set of files
- - allows to checkout code: copying/cloning
- - make changes locally and check back to repo
- Git Differencing
- - identifies differences between different file versions
- - one of fundamental differences between version control system and file storage system

CodeBuild
- Fully manages build service
- Builds
- - Set of actions performed on source code

Build Actions
- build process runs tests against new coed
- automated testing key part of software development
- - continuous integration
- - - devs check new/modded code multiple times/day
- - code build runs test against new code
- - rapid testing/feedback: provides early detection of bugs
- Build
- - Also consists of compiling source code into a binary
- - use codebuild to create docker containers and AMIs
- - Define specific tasks and build specific files
- - - most include source code
- - - source code from CodeCommit, Github, Bitbucket, S3 Bucket
- Codebuild: performs multiple builds simultaneously
- - Outputs/Artifacts codebuild creates stored in S3 buckets accessible from rest of AWS environament

Build Environments
Codebuild
- performs builds in isolated environments that runs on compute instance
- created environment before testing; discards when build finishes
- isolation ensures consistent build process
- - not affected by leftovers from previous builds
- Build environment
- - OS and docker image
- - includes programming language runtime and tools
- Preconfigures build environments:
- - Java
- - Ruby
- - Python
- - Go
- - Node.JS
- - Android

- - .net core
- - PHP
- - Docker
- - Custom
- 3 compute types for build environment:
1. build.general1.small - 3 GB of memory and 2 vCPU
2. build.general1.medium - 7 GB of memory and 4 vCPU
3. build.general1.large - 15 GB of memory and 8 vCPU

CodeDeploy
- Autodeploy apps
- - EC2 instances
- - ECS
- - Lambda
- - On-premises
- - pulls source files from S3 bucket, Github or Bitbucket repo
- - must include application's source files and application-specific file that contains information about how to deploy application
- - Does not offer option to deploy from a CodeCommit repo
- - - CodePipeline makes this possible

Deploying to EC2 or On-Premises Instances
- CodeDeploy can deploy
- - Applications, scripts, Configuration files, any file to an EC2 ro on-premises instance
- - must install codedeploy agent
- - agent allows CodeDeploy service to copy file to instance and perform deployment tasks on it
- tested with current versions of
- - Amazon Linux, Ubuntu Server, Windows Server, RHEL
- - Gives many options to carry out app deployment
- - can deploy according to specific resource tags
- - - based on auto-scaling group or done manually
- - can control cadence of deployment
- - - one at a time, all at once, half at a time, anywhere in between
- Upgrade Deployments
- - CodeDeploy supports two types:
- - - In place: Deploys to existing instances; if app running, CodeDeploy can stop, perform cleanup, redeploy, restart
- - Blue/Green: Deploys to new set of instances created manually or with codedeploy by replicating existing auto scaling group
- - Codedeploy deploys application to new instances
- - Elastic Load Balancer: CodeDeploy auto redirects traffic to new instances

Deploying to ECS
- Almost the same as deploying to EC2
- Deploys Docker images to run applications in containers
- Deploy upgrade to app using application load balancer
- - CodeDeploy can shift traffic

- - - from old containers to new ones
- - - works with both EC2 and Fargate ECS

Deploying to Lambda
- serverless computing platform
- runs functions written in various languages
- no need to think about servers running Lambda
- - codedeploy simply creates new Lambda
- if need update existing funcion
- - CodeDeploy creates new version of function
- - choose how codedeploy handles switchover to new version
- codedeploy shifts traffic slowly from one version to another
- - can do immediate, full cutover to new version

Codepipeline
- helps orchestrate/automate every task
- - required to move software from development to production
- - works taking source code and processing through stages including deployment stage
- - - in between stages: can add other stages
- enables automation of certain tasks that respective services offer on their own
- example
- - codedeply not allow directory from codecommit repo
- - does allow deploying from S3 bucket
- codepipeline
- - authorize take application in codecommit repo
- - trigger codebuild to perform automated testing against it
- Once testing passes
- - codepipeline packages up application files
- - places in S3 bucket
- - can require manual approval
- - continuous delivery
- each stage consists of one or more actions
- - six types of actions: source, build, test, approval, deploy, invoke
- with exception of approval action, each action performed by provider
- - provider can be AWS or third-party service
- Source providers: S3, CodeCommit, GitHub, Elastic Container registry (ECR): stores docker containers for Elastic container service
- Build and test providers: CodeBuild, CloudBees, Jenins, TeamCity
- Deploy providers:
- - Cloud formation
- - - auto deploy AWS infrastructure
- - - allows devs create on dev infrastructure
- - codedeploy
- - - can only source application files from GitHub or S3
- - - configure Codepipeline pull files from Codecommit, package up, put in S3 bucket
- - ECS
- - - Deploy Docker containers directly to ECS

- - - Combine with ECR for source stage
- - - use ECR as a sourcer for images
- - - not have to keep in S3 bucket
- - S3
- - - Keep files in codecommit repo
- - - if want to update, make changes in repo; codepipeline detects change and copies updates to S3 buckets

- Other supported deploy providers:
- - elastic beanstalk
- - opsworks stacks
- - AWS service catalog
- - Alexa Skills kit

Invoke - invokes Lambda functions; works only with AWS Lambda
Approval - Insert manual approvals anywhere in pipeline after source stage

EC2 Auto Scaling

Automatically launches preconfigured EC2 instances

Ensures you have just enough resources to meet user dmemand without over-provisioning

Launch configuration and Launch Templates
- spawns instances
- - launch configurator
- - launch template
- - -defines instances characteristics
- - - AMI, Disk Configuration, Instance Type
- - - Install Software/make custom configuration; placing commands into user data script; automatically runs when auto scaling launches new instance

Differences: Launch Templates/ Launch Instances
- Launch templates: newer; can be used to launch EC2 instances manually, even without auto scaling; can modify after creating
- Launch instances: Only used with auto scaling; once created, cannot modify

Auto Scaling Groups
- Instances: made by Auto Scaling; organized into auto scaling group
- - can be auto registered with application load balancer target group
- application load balancer: distributes traffic to instances, spreading demand out evenly

Desired Capacity
- Configure auto scaling
- - define desired capacity

- - - number of instances want autoscaling to create
- - - number of instances created: autoscaling strives to maintain this capacity
- raise/lower capacity:
- - Autoscaling launches/terminates instances to match

Self-healing
- Failed instances self-heal
- - fails/terminates: auto-scaling creates replacement
- - always gets number of instances expected
- auto scaling
- - use EC2/ELB health checks to determin instance healthy
- EC2: Basic health of instance:
- - running?
- - network connectivity
- ELB
- - Health of application running on instance
- - unhealthy: treated as failed instance and terminated

Scaling actions
- Control when Auto Scaling launches/terminates instances
- Control number of instances launched/terminated by specifying minimum/maximum group size
- Auto scaling ensures number of instances never goes outside range
- - Desired scaling within maximum and minimum boundary

Dynamic Scaling
- Scale out
- - A.S. launches new instances in response to increased demand
- Scele in
- - A.S. terminates instances in response to decreased demand
- Scale in/out metric:
- - avg cpu utilization or number of concurrent application users

Scheduled Scaling
- Auto Scaling can scale in or out according to a schedule
- - Useful if demand follws predictable peaks and valleys

Predictive scaling
- Looks at historic usage patterns; predicts future peaks
- auto creates scheduled scaling action to match
- needs a minimumof one day's worth of traffic to create scalin schedule

EC2 Auto Scaling
- Fulfills one of AWS core principles of sound architectural design
- ‘Don’t guess capacity needs'
- - save money by reducing capacity when not needed
- - improve performance by increasing when needed

Configuration Management

- Ensure accurate and consistent configuration of system
- - automation: concerned with carrying out tasks
- - configuration management: enforcing/monitorig internal configuration state to ensure instances are as expected
- Include
- - OS configuration
- - software
- - imperative/declarative approaches
- - two tools (both approaches)
- - systems manager
- - ops works

Systems Manager
- uses imperative approach
- - gets instances and AWS environment into wanted state

Command Documents
- scripts
- run once/periodically
- - get system into state wanted
- install software on instance
- install latest security patch
- take inventory of all software
- same bash or powershell commands
- - sys manager run periodically
- - trigger: new instance launches
- requires agent installed on instances you want it to mange

Configuration Compliance
- feautre of systems manager
- show whether instance is in compliance
- - desired configuration state
- - - cearting version of an application installed
- - - up-to-date on latest operating system security patches

Automation Documents
- Defines admin ops normally performed with AWS management console or AWS CLI
- performed with systems manager
- automatically create EBS snapshot
- launch/terminate instance
- create Cloud Formation instance
- Create AMI from existing EBS volume

Distributor
- Systems Manageer distributor
- - deploys installable software packages to instances
- create .zip archive
- - installable/executable software packages
- - o.s. recognizes
- - put archive in S3 bucket
- - tell distributor where to find
- Especially useful for deploying standard set of software packages

Opsworks
- set of three different services
- declarative approach to configuration management
- declarative approach
- - requires some intelligence to translate declarative code to imperative operations
- Opsworks
- - Chef
- - Puppet
- Puppet/Chef
- - popular configuration management platforms
- - configure os's
- - deploy applications
- - create databases
- - perform any configuration task - all using code
- - used for on-premises deployments
- - - opsworks brings their power to AWS

Opsworks three flavors
- AWS Oppsworks: Puppet Enterprise
- AWS Oppsworks: Chef Automate
- - Robust/Scalable: run managed puppet/chef servers on AWS
- - - Use configuration management across all instances
- AWS Oppsworks Stacks
- - provides simple/flexible approaches
- - using configuration management just for deploying application
- - use just for deploying/configuring apps
- oppsworks stacks takes care of setting up supporting infrastructure

AWS Opsworks for Puppet Enterprise and AWS Opsworks For Chef Automate
- High-level architectures:
- - AWS Opsworks puppet enterprise/chef automate = similar
- - - one puppet master server
- - - chef server
- - - - communicate with managed nodes
- - - - EC2 on-premises instances
- - - - - using installed agent
- Define configuration of instance

- - operating system configuration applications
- - - use puppet modules/chef recipes
- - - contain declarative code
- - - written in platform's domain specific language; specifies resources to provision
- code stored in central location: git repo; S3 bucket
- Opsworks manages servers
- - responsible for understanding: operating puppet/chef software
- - Need to know how they work and how to manage
- - both check and puppet have large ecosystems: lots of off-the-shelf code

AWS Opsworks stacks
- If like IaC, but not familiar with managing puppet/chef servers
- - can use opsworks stacks
- - lets build application infrastructure in stacks
Stack: collection of all resource an application needs
- - Example: database-backed application that consists of three layers:
- - - app layer containing EC2 instances
- - - database layer consisting of sefl-hosted or relational database servers
- - - application load balancer

Two basic type of layers opsworks uses
- Opsworks layers
- service layers
- - opswork stack not same as CloudFormation stack; doesn't use CloudFormation templates

Opsworks Layers
- Template for set of instances
- - specifies instance level settings
- - - security groups
- - - whether to use ip addresses
- auto-healing option
- - automatically re-creates instances if fail
- can perform load or time-based auto scaling
- Adds more EC2 instances as needed to meet demand
- can provision linux/windows EC2 instances
- add existing Linux/EC2 on-premises instances to a stack
- support amazon linux, ubuntu server, cento, rhel
- used declarative chef recipes as automate
- - doesn't provision Chef server
- opsworks stacks performst configuration managements tasks using chef solo client

Service Layers
- stack
- - includes service layers: extends functionality of stack to include other AWS services
- Service layer include:
- - Relational Database Service (RDS): can integrate application with existing RDS instance
- - Elastic Load Balancing (ELB): have muliple instance in stack; create app load balancer; distribute

traffic to them and provide high availability
- - Elastic Container Service (ECS): if prefer deploy app to containers instead of EC2 instances; create ECS cluster layer that connects opswork stack to existing ECS cluster

Chapter 12: Common Use-Case Scenarios

Exam Domains
Domain 1: Cloud Concepts
1.3 List the different cloud architecture Design Principles
Domain 3: Technology
3.1 Define Methods of Deploying and Operating in the AWS cloud

Chapter Summary

AWS
- Design, build, run almoast any application as you would in a traditional data center
- Allows to implement a given scenario in a variety of ways
- each way has advantages and disadvantages
- Recall shared responsibility model
- - You/AWS: Responsibility for different aspects of application running on AWS
- Adhere to princples laid out
- - five pillars 'Well architected framework'
- - - see where responsibility begins and ends for each AWS resource
- Use pillars to evaluate tradeoffs of different design decisions

Exam Essentials

Know Five Pillars of AWS Well-Architected Framework
1. Reliability
2. Performance efficiency
3. Security
4. Cost Optimizaion
5. Operational Excellence

Understand how AWS resources work together and separately to form the Five Pillars of Well-Architected Framework
- Be able to look at Solution Architect's Diagram
- Explain how resources achieve 5 Pillars

Be able to evaluate trade-offs between different design decisions

- given two different implementation options for given scenario:
- - identify advantages and disadvantages of each

Introduction

Running application
- AWS makes possible to achieve operational excellence
- provides set of guiding principles to achieve this

AWS Well-architected framework
1. Reliability
2. Performance Efficiency
3. Security
4. Cost Optimization
5. Operational Excellence

Learn how pillar form foundation of any sound cloud architecture

The Well Architected Framework

Set of principles
- AWS recommends as way of evaluating pros/cons of designing and implementing applications in cloud

Framework divides into Five Pillars
1. Reliability
2. Performance Efficiency
3. Security
4. Cost Optimization
5. Operational Excellence

Think of Pillars as goals

1. Reliability
- Avoiding complete failure of application
- Applications may depend on:
- - compute
- - storage
- - database
- - networking
- Avoiding complete failure: two things:
1. Strive to avoid failure of resources application depends on

2. If failure: Replace with healthy one

Fail possibilities:
- misconfigured
- overloaded
- shut down: accidental or intentional

Replacing failed resource
- removes need for coming up with different procedures to deal with failures

2. Performance Efficiency
- Getting performance without overprovisioning capacity
- not sacrificing reliability
- - want enough resources to provide redundancy
- - add resources as demand increases
- need assurance: right amount of resources; no more, no less
- performance/reliability:
- - inextricably related
- resource overloaded:
- - likely app performance is poor
- - may fail
- monitoring utilization/performance
- - adjusting based on those metrics: criticl aspects of performance reliability
- Performance
- - not just power/quantity of compute and storage
- - also performance of netowork between resources and users
- - significantly improve performance at minimal cost to achieve performance efficiency

3. Securiy
- CIA Triad
- - Least Access

Following Basic Principles
- Following principles of least privelege to create IAM user and resource policies
- grant delete/modify access only as needed
- Avoid data loss
- - Backups/Replications
- - Use EBS snapshots
- - - recovery points for EC2 instances
- - - S3 object versioning/replication: make possible to recover modified/destroyed data
- Enforce confidentiality using encryption: protect data at rest/in transit
- Track every activity that occurs on AWS resources
- - enable detailed logging
- - - help determine veracity of security procedures

4. Cost Optimization
- Meet needs at lowest possible costs

- - Analyze: where is money going
- AWS cost explorer
- - cost/usage reports
- - - spending: resources over time
- cost allocation tags:
- - owner
- - department
- - application
- pay only for resources needed at any given point in time
- purchase instance reserve
- use spot instances to save on-demand costs
- opex

5. Operational Excellence
- Automated Processes Required:
- - Achieve/Maintain:
- - - Reliability
- - - Performance Efficiency
- - - Cost Optimization
- - - Security
- Few Organizations Automate Everything
- Basic idea
- - incremental: improve/automate more activities; strengthen other pillars

Automation helps:
Reliability:
- Elastic Load Balancing health checks
- - Monitor Health App Run EC2 Instances
Performance Efficiency
- EC2 auto scaling: scale in/out auto
Security
- Codebuild: auto test new app code for security vulnerabilities
- Cloudformation: auto deploy fresh, secure infrastructure
Cost of optimization
- auto shut down/ decommission unused resources
- S3 lifecycle configuration
- - deleted uneeded objects
- - auto shutdown EC2 instances
- - - end of workday
- - - restart following day

Understanding how failures affect applications can help avoid such failures in the future and automate recovery when they occur

A Highly Available Web App

A Highly Available Application Using Autoscaling and Elastic Load Balancing
- High level look at how it fits together
- EC2 auto scaling
- - initially provisioning two EC2 instances
- - running in different availabilty for zones
- - - inside default VPC
- - - - Default subnet in each A.Z.
- - - - Ne need create subnets explicitly
- Application Load Balancer
- - Proxy connections from clients on internet
- distribute those connections to instances

Fours steps implement scenario
1. Create inbound server group rule in VPC's default security group
2. Create application load balancer
3. Create a launch template
4. create auto scaling group

Creating an inbound security group rule
- every VPC
- - comes w/ default security group
- - security groups block traffic
- - - not explicitly allowed by rule

Creating an application load balancer
- will route traffic to healthy instances
- uses various types of distribution algorithm
- can ignore busyness
- can configure to perform health check to monitor the health of the application on each instance

Creating a Launch Template
- Before creating auto scaling group
- create launch template
- - autoscaling used to launch instances and instantly configure apache web server software
- creating launch template is cumbersome
- - deploy cloudformation template

Create an autoscaling group
- EC2 autoscaling
- - responsible to provision certain number of healthy instances
- - provides reliability: automatically replaces instances that fail EC2 health check
- Achieve performance efficiency and cost-effectiveness
- - create dynamic scaling policy

- - - scale in/out
- - - - depends on aggregate cpu utilization of instances
- Target trafficking policy: ensuring proper instances for resources
- EC2 reports metrics to CloudWatch

Static Website Howsing Using S3

Static
- Site's assets: static files in S3 Bucket
- - can update
- - delivered to end user: same content stored in S3

Set up S3 to host static website three steps
- Create S3 bucket
- Configure for static website hosting
- uploading web assets that contain the content you want to server
- by default
- - files in S3 buckets not public
- - accessible only by account owner
- - any authorized users
- have custom domain name
- - can forward to static website
- - bucketname is same as domain name
- - static websites hosted on S3
- - - not use encrypted HTTPS
- Create Cloudfront: use HTTPS

Appendix B: Additional Services

1. Athena lets you use SQL queries to find data stored in S#. If you have data stored in CSV, JSON, ORC, Avro, or Parquet format, simply upload it to S3 and use Athena to query it. Athena is serverless, so there's no need to provision your own database or import your data into it. For more information, visit:
https://aws.amazon.com/athena/
2. AWS Backup lets you centrally configure backup policies and monitor backup activity for all of your data stored on AWS. It supports EBS volumes, RDS databases, DynamoDB tables, EFS file systems, and Storage Gateway volumes. For more information, visit:
https://aws.amazon.com/backup/

3. Data can live in a variety of places on AWS. AWS Glue can discover, clean, and bring this data together in one place for analysis using the Apache Spark big data framework. It can extract and analyze data from S3 objects and relational databases such as MySQL, Oracle, and Microsoft SQL Server. For more information, visit:
https://aws.amazon.com/glue/
4. Batch computing jobs are used for performing complex analysis on large data sets. Some examples of batch computing include financial risk modeling, graphics processing, simulations, and even analyzing genomes. Batch allows you to run thousands of batch computing jobs on AWS without having to build any infrastructure. Simply define your batch jobs as a Docker container and submit it, and AWS takes care of the rest. For more information, visit:
https://docs.aws.amazon.com/batch/
5. Cognito lets you add user access control to your application. Cognito integrates with many identity providers including Amazon, Google, Microsoft Active Directory, and Facebook. You can also use Cognito to provide your users acces to AWS resources without havin to give them their own IAM credentials. For more information, visit:
https://aws.amazon.com/cognito/
6. Database Migration Service (DMS) makes it easy to migrate data from one database to another, whether it's in the cloud or on-premises. DMS supports both relationsl databases such as Aurora, Oracle, Microsoft SQL Server, MariaDB, and PostgreSQL, as well as nonrelational databases including MongoDB, DocumentDB, and DynamoDB. DynamoDB. DMS also supports migrating data to S3, Elasticsearch, and Kinesis Data Streams. For more information, visit:
https://docs.aws.amazon.com/dms/
7. The Elastic File System (EFS) is a scalable file system for Linux instances. You can attach multiple instances to a single EFS volume so that they can all share the same files. EFS volumes are highly available, spanning multiple Availability Zones in a single VPC. EFS can scale up to petabytes in size without disruption and automatically scales down so you're not charged for space you're not using. Unused files are automatically moved to a cost-optimized storage class. For more information, visit:
https://aws.amazon.com/efs/
8. Elastic MapReduce (EMR) lets you analyze enormous amounts of data stored in the cloud. EMR supports the Apache Hadoop, Apache Spark, HBase, Presto, and Flink big data platforms. For more information, visit:
https://aws.amazon.com/emr/
9. Inspector analyzaes your EC2 instances for security vulnerabilities and common misconfigurations. For more information, visit:
https://aws.amazon.com/inspector/
10. Kinesis can ingest and process large amounts of data in real time. It's useful for analyzing larage amounts of streaming data including access logs, video, aueio, and telemetry. For more information, visit:
https://aws.amazon.com/kinesis/
11. Macie automatically finds and classifies sensitive data stored in AWS. It uses machine learning to recognize sensitive data such as personally identifiable information or trade secrets and shows you how that data is being used in AWS. For more information, visit:
https://awz.amazon.com/macie/
12. Neptune is a graph database that you can use to store and query highly connected data sets. It's useful for recommendation engines, social networks, fraud detection, and network security. For more information, visit:
https://aws.amazon.com/neptune/

13. Simple Queue Service (SQS) enables developers to create decoupled, distributed applications in the cloud. SQS is a message broker that different components of your application can use to send messages to each other. SQS scales automatically to accommodate any volume. For more information, visit:
https://aws.amazon.com/sqs/
14. WorkDocs is a secure content sharing and collaboration service. You can store any type of ile in WorkDocs, and it provides preview and commenting functionality for documents such as Microsoft Office files, PDFs, and text files. For more information, visit:
https://aws.amazon.com/workdocs/
15. Workspaces lets you provision Linux or Windows virtual desktops in the cloud. AWS manages the operating system, patching, and virtual desktop infrastructure. Users can connect to their virtual desktop from any PC and a variety of mobile devices. For more information, visit:
https://aws.amazon.com/workspaces/