|  |  |
| --- | --- |
| 1. **CLOUD CONCEPTS & TECH** | |
| Cloud computing: on-demand delivery of compute, database storage, applications, and other IT resources through a cloud services platform via the internet with pay-as-you-go pricing. (Think of it as renting someone else's computer!) | |
| 6 advantages of cloud computing | 1. Trade capital expense for **variable expense**  * not invested in data centers and servers before you know how you're going to use them * only paying for actual consumption of resources  1. Benefit from massive **economies** of scale  * Amazon has a larger purchasing power (they build their own servers!)  1. Not guessing about **capacity**  * cloud can scale with business needs without a long-term contract  1. Increase speed and **agility**  * it scales infinitely with demand * no longer have to buy servers and rent data center space  1. Not spending money running and maintaining data centers 2. Go **global** in minutes  * lower latency and better experience for customers |
| 3 Types of Cloud Computing | 1. Infrastructure As A Service (**IaaS**)   You manage a physical or virtual server; the data center provider will have no access to your server. E.g. **EC2**   1. Platform As A Service (**PaaS**)   Someone else manages the underlying infrastructure (usually hardware and operating systems); you do not have to worry about security patching, updates, maintenance, etc. You only focus on the **deployment and management** of your **applications**.  E.g. **Elastic Beanstalk**   1. Software As A Service (**SaaS**)   Typically referring to end-user applications. You do not have to think about how the service is maintained or how the underlying infrastructure is managed (someone else takes care of the data centers, servers, networks, storage, maintenance, patching, etc.); you only need to think about how you will use that particular piece software.  E.g. A web-based email **Gmail**, where you can send and receive email without having to manage feature additions to the email product or maintaining the servers and operating systems that the email program is running on. |
| 3 types of Cloud Computing Deployment Models | 1. **Public Cloud**: fully deployed in the cloud and all parts of the application run in the cloud (AWS, Azure, GCP) 2. **Hybrid**: connects infrastructure and applications between cloud-based resources and existing resources that are not located in the cloud 3. **On-Premises** (or "Private Cloud"): you manage it in your data center (Openstack, VMware) |
| Core Services: | Compute  EC2 (virtual machines in the cloud)  Lightsail (e.g of PaaS)  ECR  ECS  EKS  Lambda (is Function-as-a-Service (FaaS), don’t have to worry about VM, only your code)  Batch  Elastic Beanstalk  Serverless Application Repository |
| Storage  S3 (Simple Storage Service)  EFS (Elastic File System)  S3 Glacier  Storage Gateway (**act as a file gateway as a file system mount on S3**) |
| Database  RDS (Relational Database Service)  DynamoDB (Non-relational Databases) |
| Security, Identity, & Compliance  IAM |
| AWS Cost Management  Networking & Content Delivery  VPC (enables customers to provision a logically isolated section of the AWS Cloud to launch their resources)  CloudFront  Route53 (DNS service)  API Gateway  Direct Connect |
| AWS Global Infrastructure  ([Regions and Availability Zones]((Regions%20and%20Availability%20Zones))) | * Region: a geographical area (physical location) in the world designed to provide **high availability** to a specific geography, which consists of **2 or more Availability Zones** for fault tolerance.   + High Availability relates to the ability of your application to **withstand failures** in Cloud Infrastructure. * Availability Zone (AZ): one or more data centers, each with redundant power, networking and connectivity, housed in separate facilities. * Edge location: **endpoints** for AWS used for **caching** **content** (typically this consists of CloudFront, Amazon's Content Delivery Network (CDN)) to reduce latency.   Number of Edge Locations > Number of Availability Zones > Number of Regions  You have a mission-critical application which must be globally available at all times. You should use a **Multi-Region** deployment strategy cuz it’ll best ensure global availability. |
| Choosing the right AWS region | * Data Sovereignty Laws * Latency to end users * AWS services |
| Support Plans | 1. Basic  * Access to community forums  1. Developer  * Technical support (email, business hours) * Response time: general < 24, system imparied < 12 * Starts at $29/month  1. Business  * Technical support (phone + email + chat, 24/7) * AWS Trusted Advisor * Response time: production system down < 1   Starts at $100/month   1. Enterprise  * Technical support (phone + email + chat, 24/7) * Technical Account Manager (TAM), monitors environment and assists with optimization * Response time: business-critical system down < 15 minutes * Starts at $15,000/month |
| Identity Access Management (IAM)  Global AWS services   * IAM * Route53 * CloudFront * DNS * SNS (simple notif service) * SES (simple email service) * Tag editor * Organisations | Security, Identity & Compliance   * Global service (you do not specify a region) * When you create a user or group, it is created globally * Policies are applied to user/role/group to grant permissions   Access/Interact AWS platform in 3 ways   * via the Console (console.aws.amazon.com) * programmatically (using the command line) * using the Software Developers Kit (SDK)   purpose of IAM: to control security within your AWS account by  **Create individual IAM users & use groups to assign permissions**   * **Root account**: the email address you used to set up your AWS account   + has full administrator access   + create a user for each individual within your organization   + secure the root account using **multi-factor authentication**   + Root account should never be used for actual work * **Users** – end users * **Group**: a place to store your users   + users inherit all permissions that group (HR, finance) uses   + to set the permissions in a group, you must apply a policy to the group * **Policy**: consist of JSON (key value pairs), applied to user/role/group to grant permissions * **Role**: defines a set of **permissions** for making AWS service requests. IAM roles are not associated with a specific user or group. Instead, trusted entities assume roles, such as IAM users, applications, or AWS services such as EC2.   + IAM roles allow you to delegate access with defined permissions to trusted entities without having to share long-term access keys. You can use IAM roles to delegate access to IAM users managed **within** **your** account, to IAM users under a **different** AWS account, or **to an AWS service** such as EC2. |
| Simple Storage Service (S3)  Services that give global views but are regional   * S3   (you can see all your buckets, but they’re located in diff regions) | Storage   * A place to put your flat files (static media. U cant store a database file or operating system on S3 cuz its always changing) * **Object-based** storage; allows you to upload files * Files **0 Bytes to 5 TB** * There is unlimited storage * Files are stored in **buckets** (can create up to 100 S3 buckets per AWS account. However, you can request for additional buckets by submitting a service limit increase.) * S3 is a **universal namespace** (that is, names must be unique globally)   + Example bucket name: https://s3-eu-west-1.amazonaws.com/acloudguru * Receive a HTTP 200 response code if upload was successful * To get **detailed records for the requests** made to the bucket, when setting up the properties of an S3, select **Server access logging**   S3 Object (think of Objects as files)  Objects consist of:   * **Key**: the name of the object * **Value**: the data, made up of a sequence of bytes * Version ID * Metadata (data about the data you are storing) * Subresources * Access Control Lists * Torrent   **Data Consistency model for S3**   1. Read after write consistency for PUTS of new objects  * If you write a new file (PUT a new file in S3) and read it immediately afterwards, you will be able to view the data immediately  1. Eventual consistency for overwrite PUTS (update an existing) and DELETES (or delete a file)  * When you read the file, you may get the older version * Changes to existing objects can take some time to propagate   What is the default policy on newly created buckets?   * By default all newly created buckets are PRIVATE   Features   1. Tiered Storage Available 2. Lifecycle Management 3. Versioning 4. Encryption 5. **Restrict bucket access** (i.e. how to setup access control to buckets)  * Bucket Policies (applies across the whole bucket) * Access Control Lists/Object policies (applies to indiv files) * IAM policies to users & groups (e.g. dw finance team to view HR related info)   6 types of Storage Classes  S3 Standard   * 99.9% availability, 99.999999999% durability (11 9) stored redundantly across multiple devices in multiple facilities (designed to sustain the loss of 2 facilities concurrently)   S3 IA (Infrequently Accessed)   * data requires rapid access, for infrequent use; lower fee than S3, but you are charged a retrieval fee   S3 One Zone IA   * lower-cost option for infrequently accessed data, but do not require the multiple AZ data resilience   S3 Intelligent Tiering   * use ML to optimize costs by automatically moving data to the most cost-effective access tier, without performance impact or operational overhead   S3 Glacier   * secure, durable, and low-cost storage class for data archiving; retrieval times configurable from minutes to hours   S3 Glacier Deep Archive   * lowest-cost storage class, where a retrieval time of 12 hours is acceptable   S3 Pricing Based On...   1. Storage 2. Requests 3. Storage Management Pricing 4. Data Transfer Pricing 5. Transfer Acceleration  * enables transfers of files over long distances between end users and an S3 bucket * takes advantage of Amazon CloudFront’s globally distributed edge locations. Requests for content in S3 buckets are automatically routed to the nearest edge location where content is cached. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized path, data transfer is accelerated 🡪 improves **performance**  1. Cross Region Replication Pricing  * replicates file to a secondary bucket so you have disaster recovery   Tips for Creating an S3 bucket   1. Bucket names share a common name space; you cannot have the same bucket name as someone else 2. When you view your buckets, you view them globally (similar to IAM and Route 53), but you can have buckets in individual regions 3. You can replicate the contents of one bucket to another bucket automatically using Cross Region Replication 4. You can change the storage class and encryption on the fly  * S3 Standard * S3 IA * S3 One Zone IA * S3 Intelligent Tiering * S3 Glacier * S3 Glacier Deep Archive  1. Transfer Acceleration  * Upload file to an Edge location, then Amazon uses those backbone networks to upload the file to its location   Create an S3 Website   * You can use bucket policies to make entire S3 buckets public (‘edit public access settings’ tab) * You can use S3 to host static websites (such as .html; no database connections – these are dynamic websites) * S3 scales to meet your demand; useful for a large number of requests (e.g movie preview) |
| CloudFront | A Content Delivery Network (CDN) is a **system of distributed servers** that deliver **webpages** and other web content to a user based on the geographic locations of the user, the origin of the webpage, and a content delivery server.  Terminology   * **Edge Location**: the location where **content** will be **cached** (there are more of these than there are AZs or regions). Purpose of Edge Locations: * You can READ and WRITE to them (i.e. put an object onto them, so u can use S3 transfer acceleration) * Objects are cached for the life of the TTL * You can clear cached objects but you will be charged. * **Origin**: the origin of all the files that the CDN will distribute. This can be an S3 Bucket, an EC2 Instance, an Elastic Load Balancer, or Route53. * **Distribution**: this is the name given the CDN which consists of a collection of Edge Locations.   + Web Distribution (typically used for websites)   + RTMP (used for media streaming) * **TTL** (Time To Live): Time (in seconds) in which the Objects are cached for.   A picture containing flying, toy, kite, air  Description automatically generated  E.g. We have our origin in London (this is our S3 bucket containing our files). The users query the file (do you [Edge Location] have a copy of this file?). The Edge Location will not have the file upon this first query, and there will be added latency for this user. The **Edge Location will connect to the origin and download the file, then stream it to the user**. When the second user queries the same file, that file is already cached at the Edge Location, so the second user doesn't have to download it from the origin. They can get it from an Edge Location nearest them. |
| EC2 (Elastic Compute Cloud) | A **virtual server** (or servers) in the cloud.   * Reduces the time to obtain and boot new server instances to minutes, allowing you to quickly scale capacity (up and down) as your computing requirements change.   Pricing Models   1. **On Demand**  * Allows you to pay a fixed rate by the hour (or by the second) with no commitment   Useful for   * Users who want low cost and flexibility of EC2 without any upfront payment or long term commitment (e.g students) * Applications with short-term, spiky or unpredictable workloads that can’t be interrupted * Applications being developed or tested for the first time  1. **Reserved**  * Provides a capacity reservation, and offers a significant discount on the hourly charge for an instance * most economical option for **long-term workloads** with **predictable** **usage** patterns. * Contract terms are 1 or 3 year terms * available with **all** upfront, **partial** upfront, or **no upfront** (AURI, PURI, and NURI) pricing  1. **Spot**  * Enables you to **bid** whatever price you want for instance capacity (spot price > bid price, then u lose ur instance) * Allows greater savings if   + your applications have **flexible start/end times** (e.g. when u do batch processing,   + apps that are only flexible at v low compute prices (run apps in the middle of the night when prices are lower. Stop running apps on Mon morn)   + users with urgent computing needs for large amt of additional capacity (provided spot price < bid price; if not use on-demand) * If the Spot instance is terminated by EC2, you wont be charged for a partial hour of usage. * But if you terminate the instance yourself, you’ll be charged for any hour in which the instance ran  1. **Dedicated Hosts**  * Physical EC2 server dedicated for your use (when you’ve regulatory or licensing requirements that don’t support multi-tenant virtualisation or cloud deployments e.g. Microsoft, Oracle, VMware licensing ) * Can be purchased On-Demand hourly or as a reservation for up to 70% off the On-Demand price. * Helps reduce costs by allowing you to use your existing server-bound software licenses   EC2 Instance Types  Fight Dr. McPxz  F - FPGA  I - IOPS  G - Graphics  H - High Disk Throughput  T - Cheap general purpose (think T2 Micro)  D - Density  R - RAM  M - Main choice for general purpose apps  C - Compute  P - Graphics (think Pictures)  X - Extreme Memory  Z - Extreme Memory and CPU    EBS (Elastic Block Store)   * A **virtual disk** in the cloud that the virtual servers run off of. * Allows you to **create storage volumes** and attach them to **EC2** instances. Once attached, you can create a file system on top of those volumes, run a database, or use them in any other way you would use a block device. * Placed in an Availability Zone (same one as the EC2 instance), where they are automatically replicated for protection from the failure of a single component.   Types (of virtual disks in the cloud): [‘volume type’]   1. SSD  * General Purpose SSD (GP2) * Provisioned IOPS SSD (IO1) - Highest performance  1. Magnetic  * Throughput Optimized HDD (ST1) – designed for freq accessed * Cold HDD (SC1) – less freq accessed workloads (file servers) * Magnetic - Previous Generation   Provision an EC2 Instance   * Build web servers in the cloud. * It's not serverless, it's an actual server in the cloud.   Common Ports (How Computers Communicate)   * Linux = SSH (Port 22) * Microsoft = Remote Desktop Protocol (RDP) (Port 3389) * HTTP = Port 80; HTTPS = Port 443   Key Pair  Use a **private key** to connect to EC2 instances.  You can have many copies of the padlock, and a key to unlock it.   1. Public Keys  * many keys can unlock the padlock  1. Private Keys  * only one key can unlock the padlock   Security Group  i.e. a virtual firewalls in the cloud. You need to open ports in order to use them.   * To let everything in: 0.0.0.0/0 * To let just one IP in: X.X.X.X/32   Launch Instance   * Download your private key. * From the Terminal, navigate to Downloads and change the permissions for your private key: chmod 400 MyPrivateKey.pem * Then SSH into your instance: ssh ec2-user@3.81.122.72   Note: 3.81.122.72 will be the IP address generated after you have launched your instance. You can find the key IPv4 Public IP in the Description of your instance, then copy the IP address to your clipboard.   * To become the root user: sudo su # root gives us admin acess * Update your EC2 instance with the latest security patches. yum update -y   Design for Failure  Have one instance in each availability zone (at least 2). If one is down, the other will serve the traffic  Interact with AWS using Command Line  # SSH into your EC2 instance  $ ssh ec2-user@3.81.122.72 -i MyPrivateKey.pem  $ sudo su  aws configure to configure credentials: AWS Access Key ID: (from the credentials.csv file) Aws Secret Access Key: (from the credentials.csv file) Note: Someone can obtain control of your AWS account with this information.  # aws [service] [make bucket] [bucket name]  aws s3 mb s3://carissaallen  # aws configure  # cd .aws   * Contains config and credentials files.   Roles to further secure AWS environment [‘IAM’]   * More secure than using access key IDs and secret access keys * Policies are effective immediately * Can apply roles to EC2 instances anytime. When you do this, the change takes place immediately * Roles are universal (no need to specify what region they are in, similar to users)   Build a Web Server (Turn EC2 into web server)  Install Apache  # yum install httpd -y  Start the Service  # service httpd start  Navigate to:  # cd /var/www/html/  Create an index.html file to create your website.  # nano index.html  To view the web server, paste your IP address into the web browser.  **types of Load Balancers**   1. **Application** Load Balancers  * they’re **Layer 7 Aware** (Make Intelligent routing decisions) * E.g. you’ve a group of servers that is just for your payment gateway, i.e. your most impt group. Your app load balancers can detect that when you’re making a payment, it’ll send traffic to that group. If you’re just browsing the website, it might go to another group of EC2 instances * uses Listeners, Targets, and Target Groups, and is often deployed to load balance HTTP-based traffic.  1. **Network** Load Balancers: Extreme Performance/Static IP Addresses 2. **Classic** Load Balancers: Test & Dev, Keep Costs Low   *Configure Instance Details*   |  |  | | --- | --- | | #!/bin/bash  # yum update -y  # yum install httpd -y  # service httpd start # chkconfig on  cd/var/www/html  echo \* <html> <body> <h1> Hello, this is webserver </h1></body></html> index.html | – install apache, started Apache service  we told EC2 that if our EC2 reboots, start our Apache service automatically | |
| Relational Databases (RDS) | RDS (SQL/OLTP): think of a traditional spreadsheet! (**Aurora** is Amazon's cloud native RDS)   * pay for Aurora using **On-Demand** or **Reserved** instance pricing.   Features   1. Multiple Availability Zones **for disaster recovery** 2. Read replicas (copies of your production database) A picture containing screenshot     Description automatically generated   How It Connects   * Your EC2 instance points to a DNS string (connection string: e.g., myexampledb.alb2c3d4wxyz.us-west-2.rds.amazonaws.com) which points to your primary database in AZ 1 and a secondary database in AZ 2. * You can set it up so your EC2 instances do their writes to your primary database, and all their reads from the read replica (up to 5 copies!) * This increases perf, cuz EC2 do all their reads from Read replicas, and writes to primary database   Non Relational Databases (**DyanmoDB**)   * use when your application primarily indexes and queries data with no need for joins or complex transactions * Benefits: Provides flexibility and horizontal scalability (can add columns without impacting all rows) * Collection = Table * Document = Row * Key Value Pairs:   E.g. {"\_id":#1235353",  "name":"Samson",  "nickname":"Sammy",  "age":"6",  "address":[  {"street":"21 Jump Street",  "suburb":"Pearl"}  ]  }  Data Warehouse (OLAP) (**Redshift**)  A specialized type of relational database, which is optimized for **analysis and reporting** of large amounts of data. Redshift is a managed data warehouse service that is designed to operate at less than a tenth the cost of traditional solutions.   |  |  | | --- | --- | | Online Transaction Processing (OLTP) | Online Analytics Processing (OLAP) | | Differs in terms of the **types of queries** you will run  E.g. Query Order No. 2120121 Returns the row of data: Name, Date, Delivery Address, Delivery Status. (Inserts or retrieves a row in the database.) | E.g. Query net profit for EMEA and Pacific for the Digital Radio Product. (Pulls in large numbers of records.) |   ElastiCache  **In memory caching engine in the cloud** **for your most common queries**; will take a big load off your production databases because they are querying ElastiCache instead of your production databases, so it returns results to end users a lot faster.  Benefits:   * Improves **performance of web applications** by allowing you to **retrieve** information from fast, managed, in-memory caches, instead of relying entirely on slower disk-based databases. * It reduces the administrative burden that comes with **launching** and **managing** a **distributed cache**. * It makes it easier to **set up, operate, and scale** a distributed cache in the cloud.   You don't want web servers querying the same information over and over again. So it queries ElastiCache (holding most common queries) to improve performance.  Supports two open source in-memory caching engines: - Memcached – Redis |
| Provision an RDS Instance | * Provision an RDS Instance * Open MySQL Port to Web-DMZ SG * Create an EC2 Instance * Install Word Press using Bootstrap script * Register the EC2 Instance to the Target Group (i.e. put it behind ALB) * Updated Wordpress to the DNS name of ALB * Take a Snapshot to create an **autoscaling** group (this **scales out automatically based on load and other requirements** you set, and allow u to have a fault-tolerant website)   Bootstrap script  #!/bin/bash  yum install httpd php php-mysql -y  cd /var/www/html  echo "healthy" > healthy.html  wget https://wordpress.org/latest.tar.gz  tar -xzf latest.tar.gz  cp -r wordpress/\* /var/www/html/  rm -rf wordpress  rm -rf latest.tar.gz  chmod -R 755 wp-content  chown -R apache:apache wp-content  service httpd start  chkconfig httpd on  1: Shebang provides path to our interpreter; interprets the commands and runs them at root level. 2: Installing Apache (turns our EC2 Instance into a web server), PHP, and PHP MySQL. 3: Change our directory to where our web server will be stored. 4: Create a file called healthy.html. -: Installing Word Press (unzips, deletes, changes permissions on Apache so we can view)  WordPress site: http://my-alb-1487840289.us-east-1.elb.amazonaws.com/index.php/2019/03/18/welcome/ (xfVPC name of the Application Load Balancer (ALB))  Snapshot of web server: You can go to your EC2 Instance to create an image under Actions. Takes a photograph of the server (hard driver) and stores that image in S3 so you can provision exact copies of that web server. You can do that behind an Auto Scaling Group (ASG). You can view this image in AMIs.  AMI = Amazon Machine Image  Web server is behind an Application Load Balancer, and we have 2 EC2 instances in multiple availability zones. We have a fault tolerant website in the cloud. |
| Domain Name System (DNS) | Works like a phonebook: computers use it to resolve domain names to IP Addresses.  Route 53 (This is Amazon's DNS Service)   * Named after Route66, the first interstate highway across the United States. DNS works on Port 53. * You can use it to direct traffic around the world, and to register a domain name. * Route53 is global (similar to IAM and S3, i.e. don’t have to specify a region). * 2 policies that allow you to route data to a 2nd resource if the 1st is unhealthy, and route data to resources that have better performance   + Failover Routing   + Latency-based Routing * location based routing (Geoproximity and Geolocation) only consider where the client or resources are located before routing the data. They do not take into account whether a resource is online or slow. * Simple Routing can also be discounted as it does not take into account the state of the resources |
| Elastic Beanstalk | Allows you to provision your EC2 Instance, Security Groups, Application Load Balancers, etc. at the click of a button.   * Quickly deploy and manage applications in the AWS Cloud. * Do not have to worry about the infrastructure for those applications. * Simply upload your application, and Elastic Beanstalk will automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring. * **Grows out your infrastructure based on the code you provide it**. |
| Cloud Formation | Helps you model and set up your Amazon Web Services resources.   * Spend less time managing AWS resources, more time focusing on your applications that run in AWS. * **Create a template that describes all the AWS resources that you want** (e.g EC2 instances or RDS DB instances) and CloudFormation **provisions and configures those resources** for you. * Do not need to individually create and configure AWS resources (or worry about what is dependent on what).   9 valid sections allowed   1. Parameters 2. Resources 3. Outputs 4. Transform 5. Conditions 6. Metadata 7. Description 8. Mappings 9. AWSTemplateFormatVersion   Cloudformation supports **JSON** and **YAML** for authoring CloudFormation **templates**  Elastic Beanstalk vs. CloudFormation   * Free services; however, the services they provision are not free. * Elastic Beanstalk is limited in what it can provision and is not programmable. * CloudFormation can provision almost any AWS service and is completely programmable. |
| Architecting for the Cloud: Best Practices | [*Read the whitepaper*](https://d1.awsstatic.com/whitepapers/architecture/AWS_Well-Architected_Framework.pdf)*.*  Traditional Computing vs. Cloud Computing   |  |  | | --- | --- | | Get a purchase order, purchase physical servers, 3-5 year contract, the servers would need to be racked, connected to the networking gear, must install the operating systems, etc. | Faster   * IT Assets are available as provisioned resources   CloudFormation allows us to have templates (using JSON) to create EC2 Instances, S3 Buckets, almost anything inside the AWS ecosystem | |  | Global, Available, and Scalable Capacity | |  | Higher Level Managed Services (for Machine Learning) | |  | Built-in Security (IAM, etc.) | |  | Architecting for Cost (can architect your environment to be cost-efficient) | |  | Operations in AWS |   • Stop guessing your capacity needs  AWS Well-Architected Framework is based on 5 pillars   1. Operational excellence   3 best practice areas   1. Prepare 2. Operate 3. Evolve 4. Security 5. Reliability 6. Performance efficiency 7. Cost optimization   **Design Principles**   1. Scalability   2 types   1. Scale up  * Increasing RAM or amount of CPU inside a virtual machine.  1. Scale out  * E.g. Add **multiple virtual machines** behind an application load balancer * **Stateless Applications** using Lambda. E.g. Alexa – forgets what happened after it returns the result to you * **Distribute Load to Multiple Nodes** (e.g., multiple EC2 servers, and database replicas) * **Stateless Components**   + Do not need to remember the information (e.g store user’s login info, account details in the cookie, and is valid for a period of time. Once the cookie is invalid, users have to sign back in again) * **Stateful Components**   + Do not want to lose information; store in database or something stateful (keep users’ sign in cookie in his browser, but u want to know what they buying/doing/looking in online store, so you store that info in RDS) * Implement **Session Affinity**   + Sticky session: put a cookie in a user's browser so every time they visit that website the Application Load Balancer will detect that cookie and send them back to that same EC2 Instance. You're "stuck" to a particular EC2 Instance. * **Distributed Processing** * Implement Distributed Processing   + E.g. Elastic map reduce; it allows you to have a whole bunch of different EC2 Instances, and they process large, complex data. You have thousands of instances to **reduce the time to process that data**.  1. **Disposable resources instead of fixed servers**  * Opt for EC2 instead of physical assets e.g. renting servers for 3-5 years, cuz more flexibility (i.e. terminate when u want instead of locked down into any contract) * **Instantiating Compute Resources**   + Bootstrapping (you do not want to manually configure your EC2 Instances; we can use a bootstrap script to install updates, or Word Press, for example)   + Golden Images (set up autoscaling; took an image of our configured EC2 Instance for reuse)   + Containers   + Hybrid (containers and EC2 Instances)  1. Automation  * **Infrastructure As Code** (e.g. use CloudFormation to provision RDS, EC2 etc – allows u to quickly and easily deploy and manage your environment without reliance on humans to complete all the tasks) * **Serverless Management and Deployment** * Should go serverless so u don’t have to worry about your infrastructure. Lambda, S3 takes care of itself * Only have to worry about deployment - Using code pipeline, code deploy, etc. * **Infrastructure Management and Deployment** * AWS Elastic Beanstalk * Amazon EC2 auto recovery * AWS System Manager * Auto Scaling * **Alarms and Events** * Amazon CloudWatch alarms * Amazon CloudWatch events * A way of having your environment proactively respond to a change in the environment (e.g. detect someone upload an image to S3, it'll trigger lambda function and generate a lack of watermark on that image). * AWS Lambda scheduled events * AWS WAF security automations * WAF: Web Application Firewall * Can automatically respond to someone doing something to your site (e.g., SQL Injection)  1. Loose Coupling  * **Well Defined Interfaces**   + Amazon API Gateway     - Allows you to create your own APIs and expose them to the internet * **Service Discovery**   + Implement Service Discovery     - If you have an EC2 Instance that needs to connect to an RDS instance using its DNS name with multiple AZ turned on, if the RDS instance fails, AWS will switch it to the other availability zone. Allows one component of AWS automatically discover another component of AWS. * **Asynchronous Integration** * **Distributed Systems Best Practices**   + Graceful Failure in Practice     - E.g. If you have an S3 website and a page doesn't exist, you have an error.hmtl page to tell the users there has been a failure. Additionally, you have a mechanism to report this back to your system administrators.  1. Services Not Servers   *Use serverless services as much as possible so that servers do not have to be managed*   1. Databases   Relational Databases (Aurora)   * Scalability * Will always have 6 copies of your data across 3+ availability zones * Anti-patterns (i.e. situations where you won’t use a particular tech): would not use Aurora if you do not have a need for joins or complex transactions; use No-SQL instead   Non-Relational Databases (DynamoDB)   * Scalability * High availability, multiple AZ * Anti-patterns requires joins or complex transactions, use RDS instead. or you have large binary files (audio, image, video), store them in S3   Data Warehouse (Redshift)   * Scalability * High availability, multiple AZ * Anti-patterns: not meant for Online Transaction Processing (OLTP)   Graph Databases (Amazon Neptune)   * Scalability * High Availability  1. Managing Increasing Volumes of Data: **Data Lake**  * An architectural approach that allows you to store massive amounts of data in a **central** location (like S3!) so it is readily available to be categorized, processed, analyzed, and consumed by diverse groups within your organization. * Since **data** can be **stored as-is**, you do not have to convert it to a predefined schema, and you no longer need to know what questions to ask about your data beforehand.  1. **Removing Single Points of Failure**  * Introducing Redundancy * Detect Failure * Durable Data Storage (if you’ve impt data, u dw it on S3 1 zone availability in terms of storage class, u want a normal S3) * Automated Multi-Data Centre Resilience (if 1 AZ is down, it’s automatically failed over to another AZ. Should do it on a region basis too) * Fault Isolation and Traditional Horizontal Scaling (i.e. scaling out) * Sharding (E.g. Elastic map reduce, where u split data to process them faster)  1. **Optimize for Cost**  * Right Sizing * Elasticity   + Your application will expand (e.g. black Friday sale) or contract depending on usage * Take advantage of the variety of purchasing options/pricing models (spot, reserved, etc.)  1. **Caching**  * Application Caching (Using ElastiCache) * Edge Caching (use CDN e.g CloudFront)  1. **Security**  * Use AWS Features for Defense in Depth * Share Security Responsibility with AWS   + You and AWS are each responsible for certain things * Reduce Privileged Access (give developers *enough* access to do their job) * **Security as Code** (e.g create a golden environment by using Hardened EC2 instances i.e. instances with security patches on them and you referenced them using CloudFormation template, and deploy these golden environments to different regions using the same template) * Real-Time Auditing   + AWS Inspector and other security services |
| AWS service that can be used/deployed on premise | 1. Snowball 2. Snowball Edge 3. Snowball Gateway 4. CodeDeploy\* 5. Opsworks\* (similar to Elastic Beanstalk) 6. IoT Greengrass   \*Services that can be used to deploy apps on premise |
| Cloudwatch  (personal trainer in gym) | A monitoring service to monitor performance (e.g. your AWS resources and the applications you run on AWS e.g # of online traffic)  Can monitor things like:  Compute  - EC2  - Autoscaling groups  - Elastic Load Balances  - Route53 health checks  Storage & Content Delivery  - EBS Volumes  - Storage gateways  - CloudFront  Billing  notified when you have crossed a billing threshold  Physical servers e.g. EC2 (monitor events every 5 min by default, but can turn on detailed monitoring and have 1 min intervals)  EC2 sit on hosts such as physical servers. Hosts reports back to Cloudwatch and u can get things like CPU utilisation, disk utilisation, network utilisation, Status checks of your instances (4 host level metrics)   * Can also create custom metrics by writing a script that sends info back to Cloudwatch (e.g. EBS volume, RAM utilisation) * Can create cloudwatch alarms that trigger notifications   AWS Config provides a detailed view of configuration of resources in your account.   * This includes how the **resources** are related to one another and how they were configured in the past so that you can see how the configurations and relationships **change** over time (e.g security group changing)   AWS Trusted Advisor inspects customer environments and identifies opportunities to save money, close security gaps, and improve system reliability and performance.  Difference between CloudTrail, CloudWatch, AWS inspector, AWS Trusted Advisor, AWS config. Scenario based qns |
| AWS System Manager | Allows u to **manage fleets** of EC2 instances & virtual machines at scale   * A piece of software is installed on each VM * Can be both inside AWS and on premise * Run 1 command (e.g. update, install, patch, uninstall software) and it can be deployed to all instances * Integrates with cloudwatch to give u a dashboard of our entire estate |
| **2. BILLING & PRICING** | |
| Billing and Pricing  [Whitepaper](https://d0.awsstatic.com/whitepapers/aws_pricing_overview.pdf) | CAPEX vs. OPEX   * Capital Expenditure (pay upfront. Fixed, sunk cost) * Operational Expenditure (pay for what you use. Good to scale your biz at early onset e.g utility bill)   Types of Pricing:   1. Pay as you go 2. Pay less when you **reserve** (longer contract term, save more) 3. Pay even less per unit by using more 4. Pay even less as AWS **grows** 5. Custom pricing   **Pricing policies**   1. Understand the **fundamentals** of pricing 2. Start early with cost **optimisation** 3. Maximise the power of **flexibility** 4. Use the right **pricing** model for the job    1. Understand the fundamentals of pricing   **3 drivers of cost with AWS**   * Compute * Storage * Data Outbound Transfer   1. Start early with cost optimisation * Adopting cloud service isn’t just a technical evolution. It also requires changes to how organizations operate. As you move from IT being treated as a capital investment that happens periodically to a world where **pricing is closely tied to efficient use of resource**s, it pays to understand what drives cloud pricing so you can build a strategy for optimizing it. * When it comes to understanding pricing and optimizing your costs, it’s never too early to start. It’s easiest to put **cost visibility and control mechanisms** in place before the **environment grows** **large and complex**. Managing cost-effectively from the start ensures that managing cloud investments doesn’t become an obstruction as you grow and scale.  1. Maximize the power of flexibility  * AWS services are priced **independently and transparently**, so you can choose and pay for exactly what you need and no more. No minimum commitments or long-term contracts are required unless you choose to save money through a reservation model. By paying for services on an **as-needed** basis, you can redirect your focus to innovation and invention, reducing procurement complexity and enabling your business to be fully elastic. * One of the key advantages of cloud-based resources is that you **don’t pay for them when they’re not running**. By turning off instances you don’t use, you can reduce costs by 70 percent or more compared to using them 24/7. This enables you to be cost-efficient and, have all the power you need when workloads are active.  1. Use the right **pricing model** for the job   AWS offers several pricing models depending on product. These include:   * **On Demand** means you pay for compute or database capacity with no long-term commitments or upfront payments. * **Dedicated Instances** (available with EC2 run in a virtual private cloud (VPC) on hardware that’s dedicated to a single customer. * **Spot** Instances are an Amazon EC2 pricing mechanism that lets you purchase spare computing capacity with no upfront commitment at discounted hourly rates. * **Reservations** provide you with the ability to receive a greater discount, up to 75 percent, by paying for capacity ahead of time. C.f. on demand   ***What services are included in the Free Tier?***   * Amazon VPC (VPC = a virtual data center in the cloud) * Elastic Beanstalk\* * CloudFormation\* * IAM * Auto Scaling\* * Opsworks (similar to Elastic Beanstalk)\* * Consolidated Billing   \*But you need to pay for the resources they provision (e.g EC2)  *What determines price of EC2?*   * **Clock** Hours of Server Time * **Instance** Type (t2.micro, etc.) * Pricing Model (spot, on-demand, etc.) * Number of Instances * **Load Balancing** (An Elastic Load Balancer can be used to distribute traffic among Amazon EC2 Instances. The number of hours the ELB runs and the amount of data it processes contribute to the monthly cost.) * Detailed Monitoring (monitor every 1 min interval instead of 5) * Auto Scaling * Elastic IP Addresses * Operating Systems (windows more ex than open source OS like Linux) and Software Packages   *What determines price for Lambda? (alexa uses Lambda)*   * Request Pricing (# of requests)   + Free Tier: 1 million requests per month   + $0.20 per 1 million requests thereafter * Duration Pricing * Additional Charges (if your Lambda function uses other services, for example)   *What determines price for EBS? (Virtual hard disk in the cloud attached to EC2)*   * Volumes (per GB) * Snapshots (per GB) * Data Transfer   *What determines price for S3?*   * Storage Class (Standard, IA, 1 AZ IA, etc.) * Storage (how much data are you storing) * Request (GET, PUT, COPY) * Data Transfer   *What determines price for Glacier?*   * Storage * Data retrieval times   **Snowball** A PB-scale *data transport solution* that uses secure applications to transfer large amounts of data into and out of the AWS cloud.  Think of it as a gigantic disk to move your data into AWS.  *What determines price for Snowball?*   * Service fee per job * Daily charge * Data Transfer (data in, free; data out, costs)   *What determines price for RDS?*   * Clock hours of server time * Database characteristics – depends on db engine, size, memory class * Database purchase type - When you use **On-Demand** DB Instances, you pay for compute capacity for each hour your DB Instance runs, with no required minimum commitments. With **Reserved** DB Instances, you can make a low, one-time, upfront payment for each DB Instance you wish to reserve for a 1- or 3-year term * Number of database instances – can provision multiple DB instances to handle peak loads * Provisioned **storage** * Additional storage * Requests - number of input and output requests to the DB * Deployment Types - depending on the number of * Availability Zones you deploy to * Data Transfers   *What determines price for DynamoDB?*   * Provisioned throughput (write) * Provisioned throughput (read) * Indexed data storage * Data transfer - There is no additional charge for data transferred between DynamoDB and other AWS within the same Region. Data transferred across Regions (e.g., between Amazon DynamoDB in the US East (Northern Virginia) Region and Amazon EC2 in the EU (Ireland) Region) will be charged on both sides of the transfer   *What determines price for CloudFront?*   * Traffic Distribution - pricing is based on the edge location through which your content is served. * Requests - number and type of requests (HTTP or HTTPS) made and the geographic region in which the requests are made. * Data Transfer Out |
| AWS Budgets vs. Cost Explorer | AWS Budgets   * gives you the ability to set custom budgets that **alert** you when your costs or usage exceed (or are forecasted to exceed) your budgeted amount * Used to budget costs BEFORE they’ve been incurred   Cost Explorer   * Has an interface that lets u visualise, understand and manage costs and usage over time * Explore costs AFTER they’ve been incurred |
| Support Plans | * Basic: Free Developer: $29/mo (12 hours) * Business: $100/mo (1 hours) * Enterprise: $15,000/mo (15 mins)   A close up of a scoreboard  Description automatically generated  A screenshot of a cell phone  Description automatically generated   1. Basic  * Access to community forums  1. Developer  * Technical support (email, business hours) * Response time: general < 24, system imparied < 12 * Starts at $29/month  1. Business  * Technical support (phone + email + chat, 24/7) * AWS Trusted Advisor * Response time: production system down < 1 * Starts at $100/month  1. Enterprise  * Technical support (phone + email + chat, 24/7) * Technical Account Manager (TAM), monitors environment and assists with optimization * AWS Trusted Advisor * Response time: business-critical system down < 15 minutes * Starts at $15,000/month   Note: Response time based on system impaired or production down. |
| Tagging & Resource Groups | Tags: key value pairs attached to AWS resources   * Contains metadata (data about data) * Tags can sometimes be inherited (Eg. All the resources you created using CloudFormation will be tagged with the tags that you set when you first deployed the 1st template)   Resource groups is a collection of resources that share one or more tags (or portions of tags.)   * make it easy to group your resources using tags that are assigned to them (e.g region, name, employee ID, Dept) * is useful for describing and reporting on resources for grouping identification and internal cost recovery * Contain specific info   + EC2: Public & Private IP Addresses   + ELB: Port Configurations   + RDS: Database Engine, etc. * Using resource groups, u can apply automation to resources tagged with specific tags. E.g. we stopped all EC2 instances in Stockholm region * Resource groups tgt with AWS System manager allow u to control and execute automation against entire fleets of EC2 instances, all w 1 button   Tag Editor (global service) finds and filters tags associated with resources, and add extra tags to resources. Newer regions may take some time to be compatible with tag editor |
| Organizations & Consolidated Billing  **Cloudwatch** monitors **perf**, typically used in utilisation rate of EC2 and autoscaling.  E.g Cloudwatch monitors EC2 instances, and as soon as sth triggers that (e.g. it goes above 80% for 5min) autoscaling will provision multiple EC2 instances and scale out) | AWS Organizations: an **account management service** that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage. (i.e. a **Management and Governance** tool)  Available in 2 feature sets:   * AWS Organizations with **full access**/All features * With organisation turned on, you can get full access to your organisations * We create OU (org units e.g. finance, HR) and put AWS account behind those OU, apply policies to OU or AWS accounts, get consolidated billing (the more you use, the less u pay) * Hierarchical based control over groups of IAM users and roles, within multiple Accounts * AWS Organizations with **Consolidated billing** only   Full access  A close up of a sign  Description automatically generated  A screenshot of a cell phone  Description automatically generated  Soft limit of 20 linked accounts, can add more  Unused reserved EC2 instances are applied across the group   * E.g. Your Development team uses 4 on-demand EC2 instances and your QA team has 5 reserved instances, only 3 of which are being used. Assuming all AWS accounts are under a single AWS Organization, the Dev team will be billed for 2 instances at on-demand prices and 2 instances at the reserved instance price.   Billing Alerts:   * when monitoring is enabled on the paying account, the billing data for all **linked** accounts is included. * U can still create billing alerts per **indiv** **account**   Advantages of consolidated billing:   * **1 bill** per AWS account * Easy to **track** charges and **allocate** costs * **Volume pricing discount**   Best practices with AWS Organisations   * Always enable multi-factor authentication on root account (i.e use Google authenticator) * Strong and complex password on root account * Paying account should be used for billing purposes only. Don’t deploy resources (e.g EC2 instances or S3 buckets) into paying account   **CloudTrail (CCTV)** monitors **API calls** in AWS, i.e. when u provision new EC2 instance, create new S3 bucket etc(For **auditing**).  How to use cloudtrail with new AWS organisations   * Per AWS account and is enabled per region. * Can **consolidate logs** (i.e API events) into a single **S3** bucket in JSON format belonging to the **paying account**:  1. Turn on CloudTrail in paying account. 2. Create a S3 bucket policy that allows cross-account access. 3. Turn on CloudTrail in the other accounts and use the bucket in the paying account.   (i.e. you’re pushing out your logs from all your other accounts, into an S3 bucket in the paying account. So you’ve all the logs of people’s activities (who delete an instance, create sth etc) in a centralised account. And you can only write logs, you can’t read or make any changes. Single source of truth) |
| Quick Start & Landing Zone | Quick Start: way of **deploying environments** quickly in one account using CloudFormation **templates** built by AWS Solutions Architects who are experts in that particular technology.  AWS Landing Zone: a solution that helps customers more quickly set up a secure, multi-account AWS environment based on AWS best practices.   * Currently has 4 diff accounts – Organisations, Shared Services, Log Archive, Security |
| AWS Calculators | 1. AWS Simple Monthly Calculator    * **Estimate** potential AWS cost    * <https://calculator.s3.amazonaws.com/index.html> (hosted on S3) 2. AWS Total Cost of Ownership Calculator    * Cost of On-Prem vs. AWS (**comparison tool**)    * Generate reports that you can give to your C-level execs to make a biz case to move to the cloud    * <https://aws.amazon.com/tco-calculator/>   The best way to estimate costs is to examine the fundamental characteristics for each AWS product, estimate your usage for each characteristic, and then map that usage to the prices posted on the  website |
| 1. **SECURITY & COMPLIANCE** | |
| Compliance & Artifact | [‘Security, Identity, & Compliance’]  AWS Artifact is a central resource for compliance-related information. provides on-demand access to AWS’ security and compliance reports and select online agreements. You can get copies of audit controls report and give to your auditors  (AWS Support's site provides general information about compliance, but does not hold the secure reports or certificates.)    **PCI DSS** Level 1 certification attests to the security of the AWS platform regarding **credit card transactions**.  A **HIPAA** certification attests to the fact that the AWS Platform has met the standard required for the secure storage of **medical** records in the US Further information: |
| [Shared Responsibility Model](https://aws.amazon.com/compliance/shared-responsibility-model/)  [Shared Responsibility Model (Song)](https://www.youtube.com/watch?v=tIb5PGW_t1o) | While **AWS** manages security **of** the cloud, security **in** the cloud is the responsibility of the **customer**.  Customers retain control of what security they choose to implement to protect their own content, platform, applications, systems and networks, no differently than they would in an on-site data center.  E.g: If you forgot to update the security patches on your EC2 Instances and you get hacked--this is your responsibility. Responsible for **IAM, customer data, operating systems on your EC2 instances, subnets, security groups, network access control**.  E.g. AWS is responsible for security of the **hardware**, **availability** **regions, any software on those stacks** (virtualization or application software, such as IDS or SQL Server, **VPC**).    Visualise what the qn is asking you  Can you do this in AWS console or in EC2?  If yes, you’re likely responsible. Security Groups, IAM Users, Patching EC2 Operating Systems, patching databases running on EC2  If no, AWS. Mgt of data centers, security camera, cabling, patching RDS operating systems  Encryption is a shared responsibility (combination of you and AWS)   * If you’ve a S3 bucket and Amazon is doing the encrypting for you, then AWS responsible for encryption and the key * But you’re responsible cuz u need to turn on encryption and make sure that when you send stuff to S3, it’s being encrypted in transit (e.g. use HTTPS), and if you’re providing your own key |
| WAF and shield | WAF = Web application firewall   * Helps protect your web applications from common web exploits that could affect application **availability**, compromise **security**, or consume excessive **resources**. * Operates at Layer 7 firewall (at the **application layer**), protects against **hackers**. * If a WAF is in front of your load balancer, it will not pass the attack (SQL injection, cross site scripting, etc.) through to your EC2 Instance.   AWS Shield  A managed Distributed Denial of Service (**DDoS**) protection service that safeguards web applications running on AWS.   * Provides **always-on detection** and **automatic inline mitigations** that minimize application downtime and latency, so there is no need to engage AWS Supp\*. 0 ort to benefit from DDoS Protection. * Designed to stop DDoS attacks.   + Standard Tier (comes automatically with all account)   + Advanced Tier (Only Advanced offers automated application (level 7) layer monitoring.) |
| Inspector vs. Trusted advisor vs. cloudtrail | AWS Inspector  *Inspects* ***EC2*** *instances for vulnerabilities.*   * An automated security assessment service that helps improve the security and compliance of applications deployed on AWS. * Automatically assesses applications for vulnerabilities or deviations from best practices. * After performing an assessment, it produces a detailed list of security findings prioritized by level of severity. * These findings can be reviewed directly or as part of detailed assessment reports, available via the Inspector console or API.   It's an agent that you install on your EC2 Instances; it will review the EC2 instances for common vulnerabilities (e.g., need to update security patches).  Trusted Advisor  *Inspects your AWS account as a* ***whole*** *(not just EC2). It does Security checks, Cost Optimization, Performance, and Fault Tolerance.*  An online resource to help you **reduce cost, increase performance, and improve security** by optimizing your AWS environment.  - Provides real time guidance to help you provision your resources following AWS best practices.  - Will advise you on Cost Optimization, Performance, Security, Fault Tolerance.  - (1) Core Checks and Recommendations - (2) Full Trusted Advisor (Business/Enterprise accounts only)  CloudTrail **(CCTV)** aka **Object-level logging**  *Increases visibility into your user/ and resource activity by recording AWS Management Console actions and API calls.*   * You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred. |
| Athena vs. Macie | Athena  Interactive **query service** so u can analyse and query data located in S3 using standard SQL  E.g. u want identify all network traffic in a particular VPC, across a particular security group   * serverless, so nothing to provision, pay per query/per TB scanned * No need to set up complex ETL process * Works directly with data stored in S3   Used for?   * Query log files stored in S3 e.g ELB logs, S3 access logs * Generate biz reports on data stored in S3 * Analyse AWS cost and usage reports * Run queries on click-stream data   Macie  **Security service** which uses ML and NLP to discover, classify and protect sensitive data stored in S3   * Uses AI to recognise if your S3 objects contain sensitive data (e.g **PII** personal identifiable info) * Dashboards, reports and alerts * Can also analyse CloudTrail logs * Great for PCI-DSS and preventing thefts * Works directly with data stored in S3 |

Exam simulator questions from A Cloud Guru

|  |  |
| --- | --- |
| EMR is a web service that makes it easy to process large amounts of data efficiently |  |
| AWS Database Migrations Service is the best choice for conventional data migrations |  |
| Benefits of AWS RDS read replicas (read replicas are copies of your source database instance) | 1. enhance performance by reducing the instance’s workload 2. increase availability when the instance fails. 3. Designed for security because RDS sets up communication between them and the source DB instance using public key encryption.   (Automated backups are a feature of multi-AZ deployments) |
| need to run software functions to customize offerings for students in **various locations** around the globe based on parameters that each student enters. Which AWS service will provide this capability with the highest performance efficiency? | Lambda@**Edge** provides the capability to run Lambda functions at Edge Locations based on events generated by the CloudFront content delivery network, allowing customers to extend their web applications globally.  (Amazon Elastic Container Service and Amazon API Gateway would require implementations in each desired region) |
| AWS RDS | You’re responsible for   * optimization of your application using RDS   AWS responsible for   * Database software installation and patching * Operating system installation and patching * Scaling * Database backups |
| AWS service that provides **predictable monthly pricing**, and he expects the **instance to have the ability to burst above the baseline level of CPU performance when needed** | Lightsail  (EC2 is more suitable for consistently high CPU performance, and costs vary according to usage) |
| What comprises an AMI? i.e what info AMI needs to launch an EC2 instance? (Amazon machine image) | * 1. Amazon EBS snapshot or snapshots for backing up EBS volume data to S3, or a template for the root volume of the instance if the AMI is instance-store-backed.   2. launch permissions for controlling which AWS accounts can use the AMI to launch an instance.   3. block device mapping for specifying the volumes to attach to the instance when it’s launched.   (the EC2 instance is part of the AMI itself) |
| maximum amount of storage that Amazon Aurora (RDS) offers | 64TB   * With each database instance launched in AWS, Aurora can auto-scale up to 64 TB when needed. |
| True statements of EC2 | * It provides scalable computing capacity in the AWS cloud. * It supplies various configurations of CPU, memory, storage, and network capacity. * It provides virtual computing environments   (EC2 does not limit the max number of virtual servers) |
| What services can Amazon CloudWatch use as a target to deliver near real-time streams of system events that describe changes in AWS resources | * EC2 instances * AWS Lambda functions, * SQS queues |
| Kinesis Video Streams | is an AWS service you can use to store and watch video streams in real time as they are received in the cloud.   * you can process and analyze the streams by using third-party solutions that you can get at the AWS Marketplace * i.e. it’s a media service and an analytics tool. |
| Subnet | A virtual private cloud, or VPC, is the virtual network you create in your AWS account. When you create a VPC, you split it into smaller network segments by specifying a **range of IP addresses**. These segments are referred to as subnets, and this is where you launch your AWS resources. |
|  | **a Private Subnet** **communicate** with **Public** Subnets in the same VPC and Other **Private** **Subnets** in the **same VPC**  (In order to communicate to the internet, a NAT Gateway and Internet Gateway are required, and to enable communication between subnets in different VPCs the VPCs must first be peered.) |
| Easiest way to encrypt and archive some data into Glacier | Send the data to Glacier and do nothing more - all data in Glacier is encrypted by default  (Although you can encrypt your data before sending it to Glacier, unless you need the extra security this brings over using the inbuilt encryption this is not needed) |
| A mobile shopping list app needs to be able to add, delete, and update items on specific lists anytime a user desires. The backend for the app will run on Amazon EC2 instances with Auto Scaling to manage fluctuations in user demand. Many times, a user will perform maintenance on many list items in a single session. What design characteristic must be incorporated into the app for these requirements to be met? | In order for horizontal scaling to be effective, you'll want to make sure the **app doesn't store previous transaction or session information on specific EC2 instances.** That way, any EC2 instance provisioned by Auto Scaling can process the request.  (Leveraging load balancing is also a good practice, but doesn't address the need for a stateless app.  Session affinity goes the other direction, directing a load balancer to route transactions to a specific instance each time.  Bootstrapping runs scripts each time an EC2 instance is provisioned) |
| Shared Controls are elements of the Shared Responsibility Model where both AWS and the customer have shared responsibilities within their own contexts. | Awareness & Training is a Shared Control, since AWS trains AWS employees, but a customer must train their own employees.  (Datacentre Security is AWS’s responsibility.  Configuration of an Application within an EC2 instance, and IAM is customers’ responsibility) |
| create a cloud governance scheme. The company is organized into multiple autonomous departments which will all be using AWS resources. These departments each sponsor independent projects that are reviewed by regulatory boards for the approval of customer price increases. The code and infrastructure for each project has production, development, and testing environments. Which of the following account strategies will **maximize security and operational efficiency** for the company? | Create an Organization Unit structure in AWS Organizations with **separate** underlying **accounts** for **production, development, and testing environments**  (While there is no physical AWS account at the department level, Service Control Policies can be applied at the Organizational Unit level, and billing can be reported separately for each department.) |
|  | **Federated Access** is when you **use** an **external** **directory**, such as your coroporate one, to **grant users** in that directory access to AWS **resources**  (Role-based and User group are more used to definied which resources a user is able to access once they have access, not the method by which they gain that access.) |
| streamline access management for your AWS **administrators** by **assigning them a pre-defined set of permissions** based on their job role | Use IAM **Roles**  IAM **Groups** lets you create a list of pre-defined permissions that any **user** made a part of that group will be granted. |
|  | With WorkMail, you can send and receive email and manage your calendars in the AWS cloud. (WorkSpaces is for providing a cloud-based desktop user experience, WorkDocs is an enterprise-level storage and sharing service for improving employee productivity, and WorkLink is for providing secure mobile access to internal websites and mobile apps.) |
| ensure you have the **optimal** number of EC2 **instances** to handle your application's **load**, based on rules you specify. | **Auto Scaling**  Application Load Balancer, Elastic Load Balancer, Route53 can help distribute load amongst existing resources, but do not have the ability by themselves to create new resources. |
| scale vertically vs. horizontally | To scale vertically is to increase the specifications of a single AWS resource as needed. E.g. resizing EC2 or RDS instances for more CPU, memory, or network capabilities.  Adding more EC2 or RDS instances, is scaling horizontally because it involves the increase in the number of resources |
|  | With a **partition** placement group, **EC2** instances are placed into logical segments called partitions, each of which has its own set of racks to supply its network and power source. This level of isolation (i.e don’t share the same underlying hardware) is designed to **prevent the impact of hardware failure** within your application. |