I’m currently studying to sit the AWS Solutions Architect Associate certification. To do this I’m (a) going through the exam blueprint, (b) writing blogs on my AWS free-tier lab, (c) watching the excellent [http://acloud.guru](http://acloud.guru/) videos. I bought the “Associate Bundle” so I plan on taking all 3 associate level courses eventually. I’ve decided to consolidate the past 5 articles into 1 large article for ease of searching (and so that I’m not updating 5 separate articles while I continue to study):

Main site for all certs: <https://aws.amazon.com/certification/>

AWS-CSA: <https://aws.amazon.com/certification/certified-solutions-architect-associate/>

AWS Solutions Architect Associate exam blueprint: <https://d0.awsstatic.com/training-and-certification/docs-sa-assoc/AWS_certified_solutions_architect_associate_blueprint.pdf>

AWS Quick start reference architectures: <https://aws.amazon.com/architecture/>

AWS Whitepapers for AWS-CSA Exam Prep:

* <http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf>
* <http://media.amazonwebservices.com/AWS_Security_Best_Practices.pdf>
* <http://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf>
* <http://media.amazonwebservices.com/AWS_Storage_Options.pdf>
* <http://media.amazonwebservices.com/AWS_Cloud_Architectures.pdf>
* <http://media.amazonwebservices.com/AWS_Development_Test_Environments.pdf>
* <https://d0.awsstatic.com/whitepapers/Backup_Archive_and_Restore_Approaches_Using_AWS.pdf>
* <http://media.amazonwebservices.com/AWS_Amazon_VPC_Connectivity_Options.pdf>
* <http://d0.awsstatic.com/whitepapers/aws_pricing_overview.pdf>

**AWS – Concepts and Components**

* AWS Global Infrastructure
  + 12 Regions & 33 AZs, 5 more Regions & 11 more AZs coming throughout the next year
  + Region = 2 or more AZs
  + AZ = DataCenter
  + Edge Location = CDN End Points for CloudFront
* Networking
  + VPC = Virtual Private Cloud
  + Direct Connect = connecting to AWS w/out using Internet connection
  + Route53 = DNS service (port 53… duh)
* Compute
  + EC2 = virtual server
  + EC2 Container Service = EC2 with Docker
  + Elastic Beanstalk = Service for deploying web applications and services. “AWS for beginners”
  + Lambda = “Most powerful/revolutionary service”. Run code w/out servers. Pay for execution time, only charged when code is executed.
* Storage
  + S3 = Object based storage, a place to store flat files in the cloud
  + CloudFront = CDN (content delivery network), local caching of content
  + Glacier = long term backup, 3-5 hours to retrieve data
  + EFS = NAS in the cloud, block level storage (in preview)
  + Snowball = Import/Export service. For moving large amounts of data in/out of AWS. They ship you a physical suitcase of disks 
  + Storage Gateway = VM that you run locally that replicates data from local datacenter to AWS
* Databases
  + RDS = SQL, Aurora, Oracle, PostgreSQL, MySQL, MariaDB
  + DynamoDB = NoSQL
  + Elasticache = Caching DB services in cloud to relieve stress on RDS for high I/O environments
  + Redshift = Data warehousing service. Great performance
  + DMS = Database Migration Services. How to migrate/convert local DBs into AWS
* Analytics
  + EMR = Elastic Map Reduce. A way of processing big data
    - Managed web service Hadoop clusters
  + Data Pipeline = moving data from one service to another
  + Elastic Search = Managed service to deploy/operate a search engine in cloud
  + Kinesis = managed service platform for real time streaming of big data.
    - Web apps, mobile devices, wearables generate huge amounts of streaming data.
    - Use kinesis to digest big data
  + Machine Learning = for use by developers to work with machine learning…. (not in test)
  + Quick Sight = Business Intelligence service (not in test)
* Security & Identity
  + IAM = control users, roles, groups, policies
  + Directory Services
  + Inspector = install agents on EC2 instances & check for vulnerabilities (not in test)
  + WAF = Web Application Firewall condition sets:
    - IP Match
    - String Match
    - SQL Injection Match
    - Size Constraint
    - Cross-site Scripting Match
  + Cloud HSM = Hardware Security Module
  + Certificate Manager
* Management Tools
  + CloudWatch = Monitor
  + CloudFormation = Use templates to create infrastructure stacks
    - Use “CloudFormer” to create a template of your existing infrastructure to capture and redeploy applications you already have running
  + CloudTrail = track user & API activity
    - By default, log files are stored indefinitely
  + Config = Track resources & inventory changes (not in test)
  + OpsWorks = automation
    - Orchestration service that uses Chef
    - Chef consists of recipes to maintain a consistent state
    - Look for “chef”, “recipes”, “cookbook” in exam & think Opsworks
  + Service Catalog = not in test
  + Trusted Advisor = scans environment for ways to save money & increase security
* Application Svcs
  + API Gateway = not in test
  + AppStream = AWS version of XenApp
  + CloudSearch = Managed search solution
  + Elastic Transcoder = Media transcoding service, change media files from source format to destination format
  + SES = Simple Email Service = send/receive emails
  + SQS = Simple Queue Service, a way of decoupling infrastructure
  + SWF = Simple WorkFlow Service
* Dev Tools (not in test)
  + CodeCommit = “Github”
  + CodeDeploy = automates code deployment
  + CodePipeline = build, test, deploy code
* Mobile Svcs (not in test, except for SNS)
  + Mobile Hub = test mobile apps
  + Cognito = save mobile user data in AWS cloud
  + Device Farm = test against real smartphones & tablets in AWS cloud
  + Mobile Analytics =
  + SNS = big topic in exam, Simple Notification Service. Way to send notifications from cloud
* Enterprise Applications
  + WorkSpaces = VDI
    - Replaces Windows PC in the cloud (PCoIP)
    - Runs Windows 7, provided by Windows Server 2008 R2
    - Are persistent (EBS)
    - All data on D drive backed up every 12 hours
    - Do not need an AWS account to login to workspaces
    - Don’t need an existing AD domain, can use free client app
    - Can integrate with existing AD domain
    - By default:
      * Users can personalize their WorkSpaces with wallpaper, icons, shortcuts, etc..
      * Users have local admin access to install apps
  + WorkDocs = DropBox for enterprise
  + WorkMail = Exchange
* IoT
  + Internet of Things = not in test

**Identity Access Management (IAM)**

* Central control of AWS account
* Share access
* Granular permissions of accounts/groups/roles/policies
* Identity Federation (AD, Facebook, LinkedIn, etc…)
* MFA = Multi Factor Authentication
* Temp access for users/devices/services
* Pwd rotation policy highly customizable
* Policies = JSON key/value pairs
* IAM is universal, applies to all regions consistently
* New Users have no permissions when 1st created
* New Users are assigned an access key ID & secret access key when first created, only viewable once so download it & secure!
* Always setup MFA on root
* Integrated with AWS marketplace

**S3**

* Secure, durable, highly scalable object storage. “Unlimited storage”. A hard drive in the cloud.
* Object based NOT block based storage (no OS or DBs -> that’s Elastic Block Storage (EBS)). i.e. allows you to upload files
* 0 byte to 5Tb file size
* Files are stored in buckets
* S3 is a universal namespace, each one must be unique:
  + http://<bucket>.s3-aws-region.amazonaws.com
  + [http://s3-aws-region.amazonaws.com/<bucket](http://s3-aws-region.amazonaws.com/%3Cbucket)>
* EXAM Tips
  + Read after Write consistency for PUTS of new Objects
  + Eventual consistency for overwrite PUTS and DELETES as it can take time to propagate
* S3 = Object based. Objects consist of the following:
  + Key = name of the object
  + Value = the data
  + Version ID (for versioning)
  + Metadata (tags)
  + Subresources
  + Access Control Lists (ACLs)
* 99.99% availability
* 99.999999999% durability
* Tiered storage
* Lifecycle mgmt.
  + Can be used in conjunction with versioning
  + Can be applied to both current & previous versions
  + Actions:
    - Transition to S3-IA (128Kb & 30 days after creation)
    - Archive to Glacier (30 days after S3-IA, if relevant)
* Encryption, ACLs & Bucket Policies
* Storage Tiers
  + S3
    - 99.99% availability
    - 99.999999999% durability
    - Redundant, designed to sustain loss of 2 facilities concurrently
  + S3-IA (infrequently accessed)
    - 99.9% availability
    - 99.999999999% durability
    - Lower fee than S3, but charged a retrieval fee
  + S3-RRS (Reduced Redundancy Storage)
    - 99.99% availability
    - 99.99% durability
  + Glacier
    - Very cheap (as little as $0.01 GB/mo.)
    - Used for archive only
    - Takes 3-5 hours to restore from Glacier
* Versioning
  + Stores all versions of an object (including all writes and deletes)
  + Great backup tool
  + Cannot disable versioning once enabled, but you can suspend
  + Integrates with lifecycle rules
  + Can use MFA delete capability, so that you can’t delete without MFA
  + Cross Region Replication requires versioning – only applies to files manipulated \*after\* CRR is turned on
  + Can take up a LOT of space on files that change a lot (because it stores each changed version)

**S3 – Security & Encryption**

* By default, all new buckets are PRIVATE
* 2 types of access control for buckets
  + Bucket policies
  + ACLs
* Buckets can be configured to log all requests
  + Can be done to another bucket or to another AWS account
* Encryption – 4 methods
  + In transit – information to/from bucket
    - Uses SSL/TLS
  + At rest:
    - Server Side Encryption (SSE)
      * S3 Managed keys – **SSE-S3**
      * AWS Key Management Service, Managed Keys – **SSE-KMS**
        + Provides usage audit trail
      * SSE w/ Customer Provided Keys – **SSE-C**
  + Client Side Encryption – the customer encrypts data prior to uploading to bucket

**CloudFront – CDN (Content Delivery Network)**

* Edge Location – Where the content will be cached (different from Region or AZ)
  + Not just read only, can write to them too.
  + Objects are cached for the life of the TTL (default 24 hours)
  + Can clear cached objects, but you will be charged
* Origin – Where the original server content is located (S3 Bucket, EC2 instance, Route53, or ELB for AWS)
* Not faster for the 1st user, but faster for every other subsequent user
* Can be used for static, dynamic, streaming & interactive content
* Requests are automagically routed to nearest Edge Location
* Optimized to work well with other AWS services (duh)
* Also works with non-AWS origin servers (the “definitive version”)
* 2 types of Distributions:
  + Web Distribution – Used for websites
  + RTMP Distribution – used for media streaming
* CloudFront options
  + Restrict Viewer Access – restrict using signed URLs or signed cookies

**Storage Gateway**

* Connects on-prem software appliance with AWS storage to provide seamless & secure between an org’s on-prem IT environment & AWS storage infrastructure.
* Asynch replication backed up to S3 as EBS snapshots
* Data is stored within a single region (user specified)
* Software appliance is supported on VMware or Hyper-V
* 3 types of storage gateways:
  + **Gateway Stored Volumes (cloud is backup)**
    - Keep entire data set on-prem & asynch backed up to S3
    - Create storage volumes up to 16TB in size & mount them as iSCSI devices
    - Used for offsite backups
    - Constantly replicating changes up to S3 in the form of Amazon EBS snapshots
  + **Gateway Cached Volumes (cloud is primary)**
    - Only most frequently accessed data is stored on-prem, entire data set is stored in S3
    - Using S3 as your SAN array
    - Create storage volumes up to 32TBs in size & mount them as iSCSI devices
    - If you lose internet access, you lose access to all your data
  + **Gateway Virtual Tape Library (VTL)**
    - Limitless collection of virtual tapes
    - Up to 10 virtual tape drives per gateway
    - Exposes iSCSI interface so populat backup application (Netbackup , Backup Exec, Veeam, ect..) can point directly to VTL
* Pricing:
  + Only pay for what you use, 4 pricing components:
    - Gateway usage (per gateway per month)
    - Snapshot storage usage (per GB per month)
    - Volume storage usage (per GB per month)
    - Data xfer out (per GB per month)

**Snowball (Import/Export)**2 Types:

* Import/Export Disk
  + You ship your disks to AWS site of your choice
  + Import into S3, Glacier, or EBS
  + Export from S3
* Import/Export Snowball
  + Available in US, EU(Ireland) & APAC(Sydney)
  + 50TB or 80TB models available
  + 256-bit encryption
  + TPM ensures chain-of-custody
  + Import into S3 only
  + Export from S3

**S3 Transfer Acceleration (probably not in exam yet)**

* Use Edge Network to accelerate uploads to your S3 bucket
* Better performance the further you are away from your bucket
* Incurs an additional fee

**EC2 (Elastic Compute Cloud) – “**A web service that provides resizable compute capacity in the cloud. Reduces time required to obtain & boot new server instances to minutes allowing the ability to quickly scale capacity both up and down.”

Pricing models:

* On Demand – pay fixed rate by the hour with no commitment
  + Best for burst need servers & unpredictable workloads that cannot be interrupted
  + For users that want flexibility of EC2 w/out up-front payments or long-term commitment
  + Test/Dev for apps running on EC2 for the 1st time.
  + Supplement reserved instance servers (for extra temporary server load)
* Reserved – 1 or 3 year term. Discount compared to On Demand, the longer your contract, the more you save.
  + Best for “steady state” systems that you’ll always have running
  + Apps that need reserved capacity, steady state or predictable usage
    - Domain Controllers
    - 1st web server
* Spot – Allows you to bid for whatever price you want to pay for instance capacity (by hour).
  + When your bid = spot price, you get a server
  + When spot price exceeds your bid, you lose server with 1 hour warning
  + Best used for grid computing where instances are disposable & applications have flexible start/stop times
  + If spot instance is terminated by EC2, you don’t get charged for partial hour of usage. If \*you\* terminate, you’ll get charged for the full hour.

EC2 Instance Types:

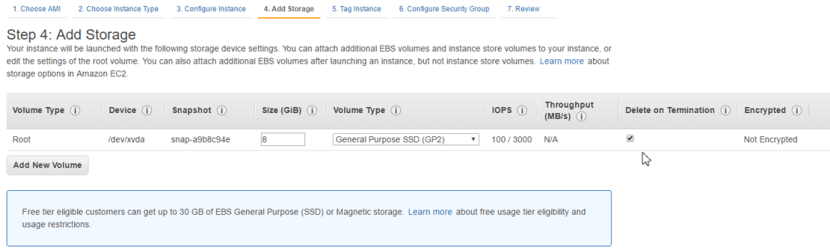
|  |  |  |
| --- | --- | --- |
| **Family** | **Speciality** | **Use Case** |
| T2 | Lowest Cost, Gen Purpose | Web Svr, small DB |
| M4 | Gen Purpose (**M**ain) | App |
| M3 | Gen Purpose (**M**ain) | App |
| C4 | **C**ompute Optimized | High CPU App/DB |
| C3 | **C**ompute Optimized | High CPU App/DB |
| R3 | Mem Optimized (**R**AM) | High Mem App/DB |
| G2 | **G**raphics | Vid Encoding, 3D Apps, Streaming |
| I2 | High Speed Storage (**I**OPS) | NoSQL DBs, Data Warehousing |
| D2 | **D**ense Storage | File srv, Hadoop |

**EBS (Elastic Block Storage) –**Storage volumes that are attached to EC2 instances (think VMDKs)

* Can’t attach 1 EBS instance to 2 EC2 instances (use EFS for that)
* Can attach multiple EBS instances to 1 EC2 instance
  + How to “grow” an EBS volume:
    - Detach the original Amazon EBS volume.
    - Create a snapshot of the original Amazon EBS volume’s data in Amazon S3.
    - Create a new Amazon EBS volume from the snapshot, but specify a larger size than the original volume.
    - Attach the new, larger volume to your Amazon EC2 instance in place of the original. (In many cases, an OS-level utility must also be used to expand the file system.)
    - Delete the original Amazon EBS volume.
* Placed in specific AZs & automatically replicated
* EBS 3 Volume Types
  + General Purpose SSD (GP2)
    - 99.999% availability
    - Ratio of 3 IOPs per GB & ability to burst up to 3k IOPS for short periods for volumes under 1Gb.
    - Use if you need up to 10k IOPS
  + Provisioned IOPS SSD (I01)
    - For I/O intensive apps (large DBs).
    - Use if you need more than 10k IOPS
  + Magnetic (standard)
    - Cheapest
    - Good for infrequently accessed data (fileservers)

***Know how to create a VPC from memory for exam!***

* When creating an AMI, on Step 4(Add storage) “Delete on Termination” is checked and not encrypted by default (i.e. Termination protection is turned off by default):



* On an EBS-backed instance, the default action is for the root EBS vol to be deleted when the instance is terminated.
* Root volumes cannot be encrypted by default, you’ll need a 3rd party tool (bit locker, etc) to encrypt root vols.

**Security Group Basics:**

* All inbound traffic is blocked by default (except for ssh for listros and rdp for windows)
* All outbound traffic is allowed by default
* Can edit security groups on the fly. **Edits take effect immediately**.
* To install Apache on AWS AMI:
  + *yum install httpd –y*
  + *service httpd status*
  + *service httpd start*
  + *chkconfig httpd on*
* Can’t add a rule to deny a specific protocol inbound or outbound
* Security groups are stateful:
  + If you allow a protocol inbound, automatically it’s added to outbound
* Can have any # of instances in a security group

**Volumes vs Snapshots**

* Volume
  + A volume is a virtual hard disk (think VMDK)
  + Volumes exist on EBS
  + If you take a snapshot of a volume, this will store that volume on S3
* Snapshot
  + Point in time copy of a volume
  + Exists on S3
  + Are incremental, only the blocks that have changed since the last snap are moved to S3
  + 1st snap takes some time to create
  + Can use snap to create a new volume & change the disk type (magnetic -> GP2 or IO1 or any other combination)
  + If you want to snap a root volume, you should stop the instance before taking snap
    - If you don’t, AWS will stop it prior to taking snap.

Go into EC2 -> Volumes -> create volume (make sure it’s in the same AZ as your server!) -> Actions -> attach to server.

Use *lsblk*to view disks to confirm new volume attached.

Use *file –s /dev/xvdf*to make sure it’s clean

Use *mkfs –t ext4 /dev/xvdf*to make file system, then *mkdir /fileserver*to create directory*, & mount /dev/xvdf/fileserver*to mount

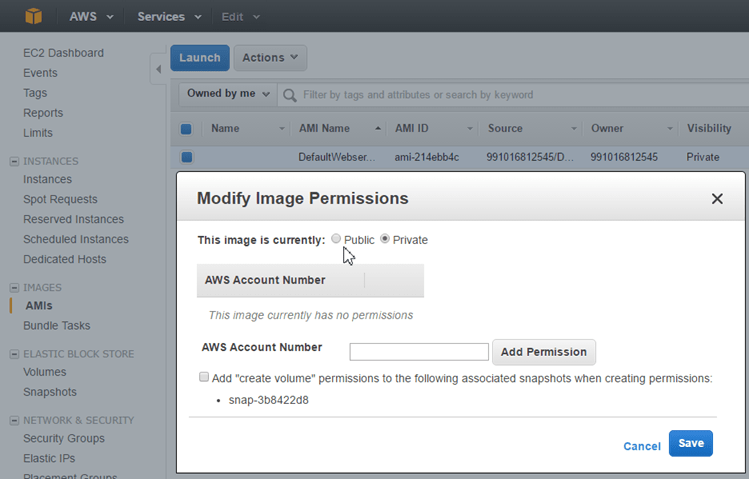
**Volumes vs Snapshots – Security**

* Snapshots of encrypted vols are encrypted automatically
* Vols restored from encrypted snaps are also automatically encrypted
* You can share snaps, but only if they are unencrypted
  + They can be shared to other AWS accounts or made public

**RAID, Volumes & Snapshots**

* RAID = Redundant Array of Independent Disks
  + RAID 0 – Striped, no redundancy, good performance
  + RAID 1 – mirrored, redundancy
  + RAID 5 – good for reads, bad for writes, **AWS does not recommend ever putting RAID 5’s on EBS**
  + RAID 10 – Striped & Mirrored, good redundancy, good performance
* Why create a RAID in AWS?
  + Not getting Disk I/O that you require from GP2 or IO1 on a single volume.
* How do you snap a RAID array?
  + Stop the app from writing to disk… how?
  + Take application consistent snap using one of these 3 methods:
    - Freeze file system
    - Unmount RAID array
    - Shut down EC2 instance

**Create an AMI (Amazon Machine Image)**

* AMI = template VM
* Are regional. You can only launch an AMI from the region where it’s stored. You CAN copy AMI’s to other regions using the command line/console/API.
* Contains:
  + Template for root volume for the instance (OS, application servers, apps, etc)
  + Launch permissions that control with AWS accounts can use the AMI to launch instances
  + Block device mapping that specifies which volumes to attach when launching instance
* By default, any AMI you create is private. You can modify image permission to make it public.
* 
* **Read these articles on how to harden & clean up an AMI before making public!**
  + <https://aws.amazon.com/articles/9001172542712674>
  + <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/building-shared-amis.html>

**AMI Types (EBS vs Instance Store)**

* You can select your AMI based on:
  + Region
  + OS
  + Architecture (32 or 64 bit)
  + Launch Permissions
  + Storage for the Root Device (root vol), 2 types:
    - Instance Store (ephemeral storage)
      * Can’t “stop” an instance of this type, only reboot or terminate. If the underlying host fails, you will lose data.
      * You can reboot without losing data, if you stop the instance, the data will be wiped.
      * “Ephemeral storage” means exactly that, not persistent
      * The root device for an instance launched from the AMI is an instance store volume created from a template stored in S3
      * Cannot be detached and reattached to other EC2 instances
    - EBS backed volumes
      * Are persistent
      * The root device for an instance launched from the AMI is an EBS volume created from an EBS snapshot
      * Can be stopped, you will not lose data if the underlying host fails.
      * Can be detached and reattached to other EC2 instances
      * By default, both root vols will be deleted on termination, but you can choose to keep an EBS vol on termination, not for ephemeral.

**Elastic Load Balancers (ELB)**

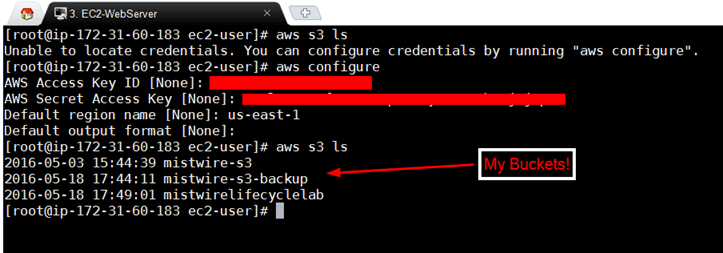
* ELB is never given a static IP address, just DNS name.
* ELBs can be “In Service” or “Out of Service”
* Thresholds
  + Unhealthy Threshold = how many intervals with no response before flagging as Out of Service
  + Healthy Threshold = how many intervals with response before flagging as In Service
* Support the following X-Forwarder headers:
  + X-Forwarded-For
  + X-Forwarded-Proto
  + X-Forwarded-Port

**CloudWatch – Performance Monitoring Service**

* Standard monitoring = 5 minutes
  + Turned on by default
* Detailed monitoring = 1 minute
* Monitors the hypervisor, NOT the guest OS
  + Does not monitor memory
* Dashboards – create/configure widgets to monitor your environment
* Alarms – notify when a given threshold is hit
* Events – automatically respond to state changes in your AWS resources
* Logs – aggregate, monitor & store logs. Agent installed onto EC2 instances

**AWS Command Line –**[**http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html**](http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html)

* You can now assign a role to an EC2 instance on the fly!
* AWS command line preinstalled on the AWS AMI
* Commands:
  + *Aws configure*
    - Input access key, Secret Access key, default region name (in doc above) & output format (I just hit enter)
  + *Aws s3 help*
    - Make Bucket = mb
    - Remove Bucket = rb



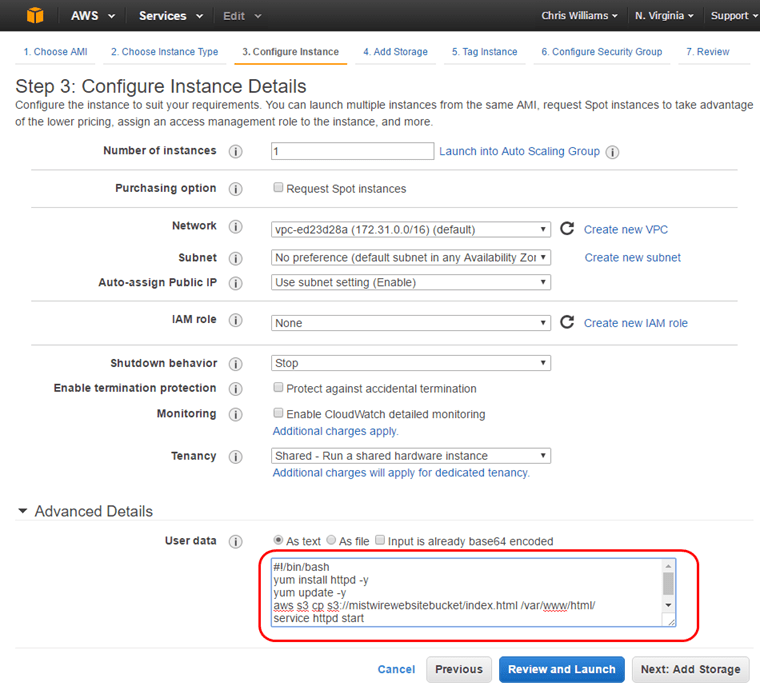
* If you use roles, you don’t have to store your credentials on your EC2 instance (which is a security risk)

**IAM – Roles**

* Roles can only be assigned to an EC2 instance when you are launching it.
* Roles are more secure than storing access keys on individual EC2 instances
* Roles are easier to manage
* They are universal, can be used in any region/AZ
* Useful for:
  + Federated (non-AWS) user access
    - Microsoft AD, LDAP, Kerberos
    - Can create trust if org supports SAML 2.0
  + Cross-Account Access
    - Multiple AWS accounts
  + Applications running on EC2 instances that need access to other AWS resources
    - EC2 instance hitting an S3 bucket or DynamoDB table

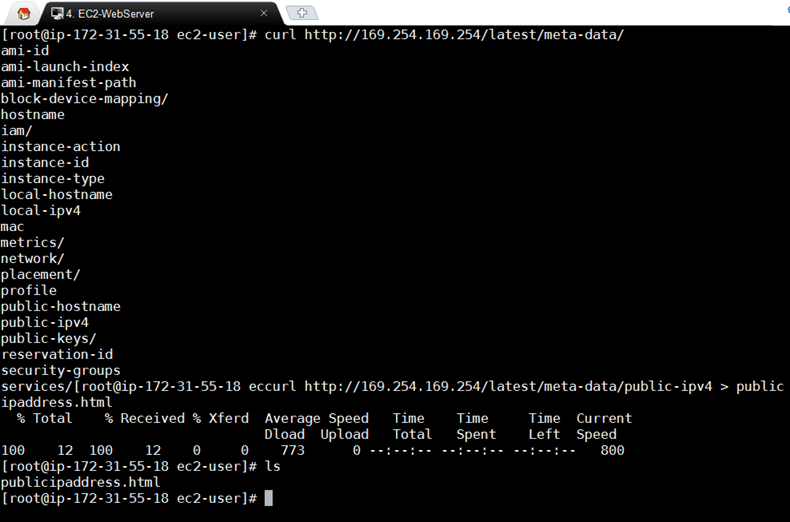
**Bash Scripting**

* Write a script that EC2 instance will run when 1st being provisioned
  + Install apache
  + Run updates
  + Move file from S3 to apache dir to create website
* How to write the bash script
  + *#!/bin/bash*
  + *Yum install httpd –y*
  + *Yum update –y*
  + *Aws s3 cp s3://<BUCKETNAME>t/index.html /var/www/html*
  + *Service httpd start*
  + *Chkconfig httpd on*
* Provision an AWS AMI instance per usual, but in the advanced section put in the above script



**Instance Metadata –**

* How to access instance metadata from within an EC2 instance. From CLI:
  + Sudo su
  + Curl <http://169.254.169.254/latest/meta-data>
    - This could be triggered from a bash script & returns a bunch of different variables, which can then be used to perform various functions:
      * Write data to an html page
      * Trigger a lambda function to update DNS
      * Whatever else you can think of 



**Auto scaling Groups**

* Have to have a launch configuration to have an auto scaling group
* Can create rules to spin-up and/or shut down instances based on monitor triggers
* Deleting an auto scaling group will automatically delete any instances it created

**EC2 Placement Groups**

* A logical grouping of instances within a single AZ.
  + Can’t span AZs (duh)
* Enables applications to participate in low-latency, 10 GBps network
* Recommended for apps that benefit from low latency networks, high network throughput, or both
  + Grid computing
  + Hadoop clusters
* Name must be unique within your AWS account
* Only certain types of instances can be launched in a placement group
  + Compute Optimized
  + GPU
  + Memory Optimized
  + Storage Optimized
* AWS recommends homogenous instances within a placement group (size & family)
* Can’t merge placement groups
* Can’t move an existing instances into a placement group. You \*can\* create an AMI from your existing instance THEN launch a new instance from that AMI into a placement group… if you really wanted to.

**Elastic File System (not in exam yet)**

* File storage for EC2 instances
* Elastic capacity
* Can mount multiple EC2 instances to 1 EFS “volume”
* Supports NFSv4 & thousands of connections
* Only pay for the storage you use (don’t need to pre-provision)
* Scales up to PBs
* Data is stored across multiple AZs within a region
* Read after write consistency
* File based storage

**Lambda concepts**

* Compute service that runs your code in response to events and it automatically manages the underlying compute resources for you
* Can automatically run code in response to events
  + Modifications to objects in S3 buckets
  + Messages arriving in Kinesis stream
  + Table updates in DynamoDB
  + API call logs created by CloudTrail
  + Etc…
* A new abstraction layer – run code without worrying about infrastructure at all
* Javascript is the supported programming language
* 99.99% availability for the service and the functions it operates
* 1st 1 million requests are free, $0.20 per 1 million requests afterwards

**Route53 (DNS)**

* IPv6 not fully supported yet.
* Alias records work like CNAME records
  + Used to map resource record sets in your hosted zone to ELB, CloudFront distributions, or S3 buckets that are configured as websites.
  + Difference – a CNAME can’t be used for naked domain names (i.e. w/out “www”), you can with A record or Alias.
  + Automatically recognizes changes in the record sets
* ELBs don’t have a pre-defined IPv4 address, resolved using DNS
  + This can be an issue because naked domain names need an IP address.
  + Hence the need for Alias records
* Given a choice, always choose an Alias record because you won’t incur additional charges (as you would with a CNAME)

**DNS Routing Policies:**

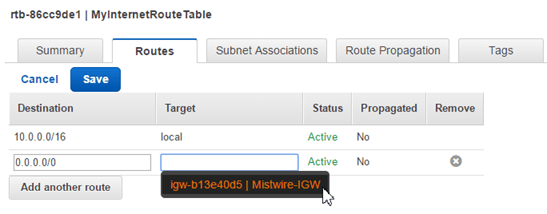
* Simple
  + Default when you create a new record set
  + Most commonly used when you have a single resource that performs a given function (i.e. 1 webserver)
  + No built-in intelligence
* Weighted
  + Split traffic based on weighted assignments (10% to X, 90% to Y)
  + Different regions, ELBs, AZs, etc.
  + Commonly used when testing a new website & you only want a small subset to see the new site
* Latency
  + Route traffic based on lowest network latency for your end user
  + Need to create a latency resource record set for the EC2 or ELB resource in each region you want participating.
  + Great for improving global page load times
* Failover
  + Used when you want to create an active/passive set up.
  + Route53 will monitor health of primary site using a health check (which monitors your end points)
* Geolocation
  + You choose were traffic will be sent based on location of users
  + Ex. All EU users get routed to servers w/ local language and prices in Euros

**Databases**

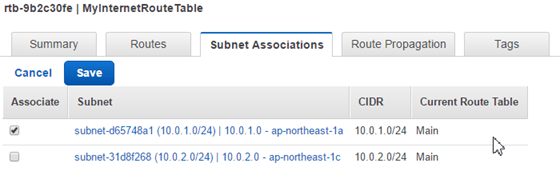
* RDS – Been around since the 70s. Database: tables, rows, fields (columns) -> think spreadsheet
  + Read this FAQ: <https://aws.amazon.com/rds/faqs/>
  + For OLTP
  + SQL Server
  + Oracle
  + MySQL
  + PostgreSQL
  + Aurora
  + MariaDB
* DynamoDB – non-relational databases (No SQL)
  + Database:
    - Collection     = Table
    - Document     = Row
    - Key/Value pairs    = Fields
* ElastiCache
  + web service that deploys, operates & scales an in-memory cache. Improves performance of web apps by retrieving info from RAM instead of disk.
  + Supports 2 open source in-mem caching engines
    - Memcached
    - Redis
  + Caches most consistently queried data
* Redshift (data warehousing)
  + OLAP
  + Used for BI. Cognos, Jaspersoft, SAP Netweaver
  + Used to pull in large & complex data sets. Usually used to do queries on data.
* DMS (database migration services)
  + Migrate your prod DB into AWS
  + AWS manages all the complexities of migration like data type transformation, compression & parallel xfer
  + Schema conversion tool:
    - Convert source DB to a different target DB (Oracle -> Aurora, etc…)
* Backups, Multi-AZ & Read Replicas
  + Backups (2 types):
    - Automated
      * Recover DB to any point in time within retention period (between 1 – 35 days)
      * Point in time recovery down to a second, up to the last 5 minutes
      * Enabled by default
      * Backup data is stored in S3
      * Free backup storage equal to size of DB
      * Backups are taken within a defined window, retention period up to 35 days
      * During backup, I/O suspended (typically a few minutes)
        + This can be avoided if you go Multi-AZ as the backup is taken of the standby
    - DB Snapshots
      * Done manually (user initiated), full backup
      * Stored even after you delete the original RDS instance, until you explicitly delete them
      * When you restore either automated or snap, the restored version will be a new RDS instance with a new endpoint
  + Encryption
    - At rest is supported for MySQL, Oracle, SQL, PostgreSQL & MariaDB
    - Done using AWS KMS
    - Once your RDS instance is encrypted at rest – underlying storage, backups, read replicas and snaps are also encrypted
    - Turning on encryption for an existing instance isn’t supported… create a new encrypted instance & migrate data to it
  + Multi-AZ
    - Primary RDS instance uses synchronous replication to an RDS in a diff AZ.
    - Automatic failover, same DNS point, AWS handles replication
    - Disaster Recovery only, not performance improvement
    - Only in:
      * SQL Server
      * Oracle
      * MySQL Server
      * PostgreSQL
      * MariaDB
  + Read Replica
    - Uses asynchronous replication to create up to 5 read-only DB copies
    - Used for performance improvement & Scaling, not DR:
      * Write to prod, read from read replicas
    - Must have automatic backups turned on
    - You can have read replicas OF read replicas, but watch out for latency if you do this.
    - Each read replica will have it’s own DNS end point.
    - Cannot have read replicas that have Multi-AZ but you CAN create read replicas of Multi-AZ source DBs
    - Can break replication & turn a read replica to it’s own source DB
    - Only in:
      * MySQL Server
      * PostgreSQL
      * MariaDB
  + DynamoDB vs RDS
    - DynamoDB offers “push button” scaling -> scale DB on the fly with no downtime
    - RDS isn’t as easy -> usually need to create bigger instance size manually or add a read replica
* DynamoDB
  + Fast, flexible NoSQL DB service.
  + Used for apps that need consistent, single-digit millisecond latency at any scale
  + Fully managed & supports document and key/value data models
  + Stored on SSD storage
  + Spread across 3 “geographically distinct” data centers
  + Multiple consistency models:
    - Eventually consistent reads (default)
      * Consistency usually reached within 1 second (best read performance)
    - Strongly consistent reads
      * Returns a result that reflects all writes that got a successful response prior to the read
      * Use this if your app needs data back immediately & in less than 1 second.
  + Pricing (not in exam):
    - Write throughput $0.0065 per hour every 10 units
    - Read throughput $0.0065 per hour every 50 units
    - Storage = $0.25 per GB per month
    - Expensive for writes, cheap for reads
* Redshift
  + Fast (10 times faster), fully managed petabyte-scale data warehouse service
  + Can start small for $0.25 per hour with no commitments & scale up to PB or more for $1,000 per TB per year.
  + OLAP transactions
  + Data warehousing DBs us diff type of architecture from both a DB perspective & infrastructure layer.
  + 2 Configurations:
    - Single node (160Gb)
    - Multi-node
      * Leader Node (manages client connections and receives queries)
      * Compute Node (store data & perform queries and computations). Up to 128 Compute Nodes
  + Columnar Data Storage – instead of rows, redshift organizes data by column
    - Only columns involved in the queries are processed
    - Columnar data is stored sequentially on the storage media
    - Block size of 1MB for columnar storage
    - Therefore requires far fewer I/Os, greatly improving performance
  + Advanced Compression
    - Columnar data can be compressed much better than row based data
    - Redshift automatically samples data & chooses the best compression scheme
  + Massively Parallel Processing (MPP):
    - Automatically distributes data & query load across all nodes & newly added nodes
  + Pricing:
    - Compute Node Hours
      * 1 unit per node per hour
    - Backup
    - Data Transfer
  + Security
    - Encrypted in transit using SSL
    - At rest using AES-256
    - By default RedShift does it’s own key mgmt.
      * Can manage keys through HSM (hardware security modules) or KMS if you want
  + Only available in 1 AZ
    - Can restore snaps to new AZs in the event of an outage
  + Good choice if mgmt. runs lots of OLAP transactions & it’s stressing the DB
  + Think Business Intelligence (BI)
* Elasticache
  + Caches things – if your app is constantly going to a DB to pull the same data over and over, you can cache it for faster performance
  + Used to improve latency and throughput for **read-heavy app** workloads (social networks, gaming, media sharing) or **compute heavy** workloads (recommendation engine)
  + Improves application performance by storing critical pieces of data in mem for low-latency access.
  + Types of elasticache
    - Memcached
      * Widely adopted mem object caching system.
    - Redis
      * Open source in-mem key/value store.
    - Supports master/slave replication & multi-AZ to achieve cross AZ redundancy
  + Good choice if your DB is read heavy & not prone to frequent changing
* Aurora
  + MySQL compatible RDS DB engine
  + Speed & availability of commercial DBs
  + Simplicity & cost-effectiveness of open source DBs
  + 5x better performance than MySQL @ 1/10th the price of commercial DB w/ similar performance & availability
  + Big challenge to Oracle
  + Scaling capabilities:
    - Start w/ 10Gb, scales in 10Gb increments up to 64Tb
    - Compute scales up to 32vCPUs & 244Gb of mem
    - 2 copies of DB in each AZ w/ a min of 3 AZs (6 copies of data)
    - Can handle loss of 2 copies w/out affecting write availability
    - Can handle loss of 3 copies w/out affecting read availability
    - Storage is self-healing. Blocks & disks are constantly scanned & repaired
  + Replica features:
    - Aurora Replicas (currently 15)
    - MySQL read replicas (currently 5)

**VPC (Virtual Private Cloud)**

* **For the exam know how to build a custom VPC from memory**
  + Create VPC
    - Define IP range (automatically creates default route table)
    - Create subnets (automatically creates route table & nACL)
      * Largest = /16, Smallest = /28
      * AWS reserves the 1st 4 and last 1 IP address of any subnet, so /28 = 11 useable IPs
    - Create IGW
      * By default it’s detached, need to manually attach it to VPC
    - Create custom route table & attach IGW to it
    - Associate public subnet(s) to use new route
    - Launch 1 instance per subnet
    - Provision EC2 NAT instance
      * Create security group for NAT instance
* VPC = Think of it as a Virtual Datacenter
  + By default you are allowed 5 VPCs per region
  + Logically isolated section of AWS where you can launch AWS resources in a virtual network of your own definition
  + You control the network environment: IP address range, subnets, routing tables, gateways, etc
* Default VPC vs Custom VPC
  + Default is user friendly, can deploy instances right away
  + All subnets in default VPC have an internet gateway attached
  + Each EC2 instance has both a public & private IP address
  + If you delete default VPC, you have to call AWS to get it back
* VPC Peering
  + Connect 1 VPC to another VPC via direct network route using private IP addresses
  + Instances behave as if they were on the same private network
  + You can peer VPC’s with other AWS accounts & with other VPC’s in the same account **within a single region**
  + AWS uses the existing infrastructure of a VPC to create a VPC peering connection.
  + It is not a gateway or a VPN connection.
  + It does not rely on a separate piece of hardware
  + No SPoF for communication or bandwidth bottleneck
  + Peering is done in a star configuration. VPC A  VPC B  VPC C = A cannot talk to C unless you connect directly (**no transitive peering**)
  + **Peers cannot have matching or overlapping CIDR blocks**
* By default when you create a VPC it will automatically create a route table
* If you choose dedicated tenancy for your VPC, any instances you create in that VPC will also be dedicated
* 1 subnet = 1 AZ, you cannot have subnets cross AZ
* Don’t forget to add internet gateway
  + 1 IGW per VPC
  + Need to attach IGW after you create it
* Need to create InternetRouteTable if you want VPC to communicate in/out



* + Once you’ve created your IGW, any subnet associations you make to it will be internet accessible:



* + A security group can stretch across multiple Regions/AZs where a subnet cannot
* VPC Flow Logs:
  + Enables you to capture information about the IP traffic going to and from network interfaces in your VPC
  + Data is stored using Amazon CloudWatch Logs, you can view and retrieve its data in Amazon CloudWatch Logs.
  + Help with a number of tasks:
    - Troubleshoot why specific traffic is not reaching an instance
    - Diagnose overly restrictive security group rules
    - Security tool to monitor the traffic that is reaching your instance.

**Network Address Translation (NAT)**

* Allows your instances that do not have internet access the ability to access the internet via a NAT server instance
* create security group
* allow inbound & outbound on HTTP and HTTPS
* provision NAT inside public subnet
* **On a NAT instance, you need to change source/destination check to disabled**
* Set up route on private subnet to route through NAT instance

**Access Control Lists (ACLs)**

* A numbered list of rules (in order, lowest applies first)
* Put down network access lists across the entire subnet
* Over rules security groups
* Acts as a basic firewall
* VPC automatically comes with an ACL
* When you create a new ACL, by default everything is DENY
* Only one ACL per subnet, but many subnets can have the same ACL

**Application Services**

**SQS – most important service going into exam**

* Read FAQ for SQS for exam: <https://aws.amazon.com/sqs/faqs/>
* A distributed message queueing service that sits between a “producer” and “consumer” to quickly and reliably cache that message.
* Allows you to decouple the components of an app so that they can run independently.
* Eases message management between components
* Any component can later retrieve the queued message using SQS API
* Queue resolves issues if:
  + The producer is producing work faster than consumer is processing
  + Producer or consumer are only intermittently connected to network
* Ensures delivery of each message at least once
* Supports multiple writers and readers on the same queue
* Can apply autoscaling to SQS
* A single queue can be used by many app components with no need for those components to coordinate amongst themselves to share the queue
* SQS does NOT guarantee first in, first out (FIFO) delivery of message
  + If you want this, you need to place sequencing information in each message so that you can reorder the messages after they come out of queue
* SQS is a pull based system
* 12 hour visibility time out by default
* Engineered to provide “at least once” delivery of mgs, but you should design your app so that processing a message more than once won’t create an error
* Messages can contain up to 256KB of text in any format
* Billed at 64KB “chunk” – a 256KB msg will be 4 x 64KB “chunks”
* 1st 1 million SQS requests per month are free
* $0.50 per 1 million requests per month thereafter
* A single request can have from 1 to 10 messages, up to a max total payload of 256KB
* Each 64KB ‘chunk’ of payload is billed as 1 request.
  + Ex: 1 API call with a 256KB payload is billed as 4 requests

**SWF – Simple Workflow Service**

* Makes it easy to coordinate work across distributed app components
* Enables apps to be designed as a coordination of tasks
* Tasks represent invocations of various processing steps in a app which can be performed by:
  + Executable code
  + Web service calls
  + Human actions
  + Scripts
* Amazon uses SWF to process orders on the amazon website to get you your stuffs
* SWF vs SQS
  + SQS has a retention period of 14 days, SWF up to 1 year for workflow executions
  + SWF presents task-oriented API, SQS = message-oriented API
  + SWF ensures a task is assigned only once and never duplicated, with SQS you need to handle the potential for duplicate messages
  + SWF keeps track of all the tasks & events in an application. With SQS you need to implement application-level tracking, especially if you have multiple queues.
* SWF Actors (3 types):
  + Workflow Starters – an app that can initiate a workflow (amazon.com front end when placing an order)
  + Deciders – control the flow of activity tasks (if cc declined – decide to send to alternative payments page)
  + Activity workers – carry out tasks (payment now successful, go pull widget off shelf & mail it)

**SNS – Simple Notification Service**

* Web service to setup, operate & send notifications from AWS.
* Scalable, flexible, cost-effective way to publish messages from an app & deliver them to subscribers or other apps
* Push notification to Apple, Google, Fire OS, Windows devices, etc..
* Can deliver via SMS text messages, email, SQS queues, any HTTP endpoint
* Can also trigger Lambda functions
* SNS Subscribers:
  + HTTP/S
  + Email/Email-JSON
  + SQS
  + Application
  + Lambda
* Allows you to group multiple recipients using topics
* One topic can support delivery to multiple endpoints types
  + “Autoscale change” to my phone, my email etc… all properly formatted for the endpoint
* All messages published to SNS are stored redundantly across multiple AZs
* Instantaneous, push-based deliver (no polling)
* Simple APIs & easy integration with apps
* Flexible message delivery over multiple transport protocols
* Pay as you go model with no up-front costs
* Mgmt console offers simple point/click interface
* SNS vs SQS
  + Both messaging services in AWS
  + SNS – Push
  + SQS – Polls (pulls)
* Pricing:
  + $0.50 per 1 million SNS requests
  + $0.06 per 100,000 notification deliveries over HTTP
  + $0.75 per 100 notification deliveries over SMS
  + $2.00 per 100,000 notification deliveries over email

**Elastic Transcoder**

* Media transcoder in the cloud
* Converts media file from original source format into different formats that will play on different endpoint devices
* Provides transcoding presets for popular output formats
  + Don’t need to guess about which settings work best on particular devices
* Pay based on the minutes that you transcode & the resolution at which you transcode
* [https://read.acloud.guru/easy-video-transcoding-in-aws-7a0abaaab7b8#.eepluawzo](https://read.acloud.guru/easy-video-transcoding-in-aws-7a0abaaab7b8)

**White Paper Breakdown:**

**Overview of AWS:**[**http://d0.awsstatic.com/whitepapers/aws-overview.pdf**](http://d0.awsstatic.com/whitepapers/aws-overview.pdf)

What is cloud computing? On demand delivery of IT resources and apps via the Internet w/ pay-as-you-go pricing. Cloud providers maintain the network-connected hardware while the consumer provisions and use what you need via web applications.

6 Advantages of Cloud:

1. Trade capex for “variable expense”
2. Benefit from economies of scale
3. Stop guessing about capacity
4. Increase speed & agility
5. Stop spending money running & maintaining datacenters
6. Go global in minutes

**Overview of Security Processes:**[**http://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf**](http://d0.awsstatic.com/whitepapers/Security/AWS%20Security%20Whitepaper.pdf)

* State of the art electronic surveillance and multi factor access control systems
* Staffed 24×7 by security guards
* Access is least privilege based

Shared Security Model – AWS is responsible for securing the underlying infrastructure. YOU are responsible for anything you put on or connects to the cloud

AWS responsibilities:

* Infrastructure (hardware, software, networking, facilities)
* Security configuration of it’s managed services (DynamoDB, RDS, Redshift, Elastic MapReduce, WorkSpaces)

Customer responsibilities:

* IAAS – EC2, VPC, S3
* Managed services – Amazon is responsible for patching, AV etc… but YOU are responsible for account mgmt. and user access. Recommended that MFA is implemented, SSL/TLS is used for communication, & API/user activity is logged using CloudTrail

Storage Decommissioning:

* AWS uses NIST 800-88 to destroy data. All decommed magnetic storage devices are degaussed and physically destroyed.

Network Security:

* Transmission Protection – Use HTTPS using SSL
* For customers who need additional layers of network security, AWS provides VPCs & the ability to use an IPSec VPN between their datacenter & the VPC
* Amazon Corporate Segregation – AWS production network is segregated from the Amazon corporate network by a means of a complex set of network security/segregation devices
* DDoS mitigation
* Prevent Man in the middle attacks (MITM)
* Prevent IP Spoofing – the AWS controlled, host-based firewall will not permit an instance to send traffic with a source IP or MAC other than its own.
* Prevent Port Scanning – Unauthorized port scans are a violation of T&Es. You must request a vulnerability scan in advance
* Prevent Packet Sniffing by other tenants

AWS Credentials

* Passwords
* MFA
* Access Keys
* Key Pairs
* X.509 certs

AWS Trusted Advisor

* Inspects your AWS environment & makes recommendations to save money, improve performance & close security gaps:
* Provides alerts for several of the most common security misconfigs:
  + Leaving certain ports open
  + Not creating IAM accounts for internal users
  + Allowing public access to S3 buckets
  + Not turning on user activity logging (AWS CloudTrail)
  + Not using MFA on your root AWS account

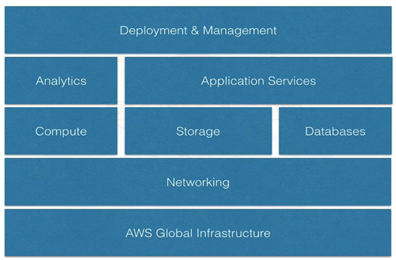
Instance Isolation

* Instances on same physical machine are isolated from each other via the Xen hypervisor.
* The AWS firewall resides within the hypervisor layer, between the physical network interface & the instances virtual interface.
  + All packets must pass through this firewall
* Physical RAM is separated using similar mechanisms
* Customer instances have no access to raw disk devices, only virtual disks
* AWS proprietary disk virtualization layer resets every block of storage used by the customer
  + Ensures customer X data isn’t exposed to customer Y
* Mem allocated to guest is scrubbed (zeroed out) by hypervisor when it becomes unprovisioned
  + Mem not returned to pool of free mem until scrubbing is complete
* Guest OS
  + Instances are completely controlled by customer. AWS does not have any access rights or back doors to guest OSes
  + AWS provides the ability to encrypt EBS volumes & their snapshots with AES-256
* Firewall:
  + EC2 provides a complete firewall solution. By default inbound is DENY-ALL
* ELB – SSL Termination on the load balancer is supported
  + Allows you to ID the originating IP address of a client connecting to your servers, whether you are using HTTPS or TCP load balancing
* Direct Connect:
  + Slower to provision than a VPN because it’s a physical connection
  + Bypass ISPs in your network path (if you don’t want traffic to traverse Internet)
  + Procure rack space within the facility housing the AWS Direct Connect location & deploy your equipment nearby.
  + Connect this equipment to AWS Direct Connect using a cross-connect
  + Use VLANs (802.1q) to use 1 connection to access both public (S3) and private (EC2 in a VPC) AWS resources
  + Available in
    - 10Gbps
    - 1Gbps
    - Sub 1Gbps groups purchased through AWS Direct Connect Partners

**Risk and Compliance**: <http://d0.awsstatic.com/whitepapers/compliance/AWS_Risk_and_Compliance_Whitepaper.pdf>

* AWS mgmt. has a strategic business plan which includes risk identification & mitigation plans. This is re-evaluated at least bi-annually.
* AWS security regularly scans all Internet facing service endpoint IP addresses for vulnerabilities (these scans do not include customer instances)
* Independent external vulnerability threat assessments are performed regularly by 3rd party security firms.
  + Not meant to replace a customer’s own vulnerability scans
* SOC 1/SSAE 16/ISAE 3402
* SOC2
* SOC3
* FISMA, DIACAP, & FedRAMP
* PCI DSS Level 1   
  **can take credit card information with PCI compliance (software needs to be compliant too)**
* ISO 27001
* ISO 9001
* ITAR
* FIPS 140-2
* HIPAA
* Cloud Security Alliance (CSA)
* Motion Picture Association of America (MPAA)

AWS Platform:



**Storage Options in the Cloud: (2 docs?)**

[**http://media.amazonwebservices.com/AWS\_Storage\_Options.pdf**](http://media.amazonwebservices.com/AWS_Storage_Options.pdf)

[**http://d0.awsstatic.com/whitepapers/AWS%20Storage%20Services%20Whitepaper-v9.pdf**](http://d0.awsstatic.com/whitepapers/AWS%20Storage%20Services%20Whitepaper-v9.pdf)

**Architecting for the Cloud – Best Practices:**[**http://d0.awsstatic.com/whitepapers/AWS\_Cloud\_Best\_Practices.pdf**](http://d0.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf)

Business Benefits:

* Almost 0 upfront infrastructure investment
* JIT infrastructure
* More efficient resource utilization
* Usage-based pricing
* Reduced time to market

Technical Benefits:

* Automation – “Scriptable infrastructure”
* Auto-scaling
* Proactive scaling
* More efficient dev lifecycle
* Improved testability
* DR/BC baked in
* “Overflow” traffic to the cloud

Design for Failure:

* Assume that hardware will fail & outages will occur
* Assume that you will be overloaded with requests
* By being a pessimist, you think about recovery strategies during design time, which helps you design an overall better system

Decouple your components:

* Build components that do not have tight dependencies so that if 1 component dies/sleeps/is busy, the other components are built so as to continue work as if no failure is happening.
* If you see decoupling in exam, think SQS
* WebServer – SQS – AppServer – SQS – DBServer

Implement Elasticity:

* Proactive Cyclic Scaling – periodic scaling that occurs @ fixed intervals (daily, weekly, monthly, quarterly) i.e. “Payroll Monday”
* Proactive Event Scaling – when you are expecting a big surge of traffic (Black Friday, new product launch, marketing campaign)
* Auto-scaling based on demand – Create triggers in monitoring to scale up/down resources

Secure Your Application:

* Only have the ports open to/from your various stacks to allow communication, no more (duh)

Consolidated Billing

* 1 paying account for all linked accounts in an org
* Paying account gets 1 monthly bill
* Paying account cannot access resources of the linked accounts
* All linked accounts are independent of each other
* 20 linked accounts for consolidated billing (soft limit)
* Easy to track charges & allocate costs
* Volume pricing discount, resources of all your linked accounts are added up for discounts

Resource Groups & Tagging

* Tags = Key/Value pairs attached to AWS resources
* Metadata
* Tags can be inherited sometimes:
  + Autoscaling, CloudFormation, Elastic Beanstalk can create other resources
* Resource Groups
  + Make it easy to group resources using the tags that are assigned to them
  + Contain info like:
    - Region
    - Name
    - Health checks
    - For EC2 – Public & Private IP addresses
    - For ELB – Port configs
    - For RDS – Database engine, etc.
  + Use tag editor to find/modify resources in large volumes

Active Directory Integration:

* User browses to ADFS URL
* User authenticates against AD
* User receives a SAML assertion
* User’s browser posts the SAML assertion to the AWS sign-in endpoint for SAML
* User’s browser receives the sign-in URL and is redirected to the console