**(Domain 1) High Availability and Business continuity**

https://d0.awsstatic.com/whitepapers/aws-disaster-recovery.pdf (Not on the Exam blueprint guideline)

RTO (Recovery Time Objective) - amount of time it takes for your business to recover from outage or disruption. TIME - time to fix the problem, recover itself and testing etc.

RPO (Recovery Point Objective) - how much data can your organization afford to loose during the outage. (an hour?, a day?, or not at all)

Services AWS provide to help with DR: (i) different regions (ii) Storage: S3 (11 - 9 durability and cross region replication), Glacier, EBS(point in time snapshot - not automatic - script it), AWS Storage gateway (Gateway-stored volumes, Gateway-cached volumes, Gateway-virtual tape library - store virtual tapes in either S3 or Glacier -min. 3 hour or longer recovery time) (iii) compute: EC2, EC2 VM Import Connector (IV) Networking: Route53, ELB, VPC, DirectConnect (V) Databases: DynamoDB(offers cross region replication), RDS (ability to snapshot from one region to another and also have a read replica running in other region), Redshift (snapshot your data warehouse to S3 in same region or copied to another region) (VI) Orchestration: CF, Elastic Beanstalk, OpsWorks.

DR Scenarios

(i) Backup and Restore: cheapest, highest RTO and RPO, S3, AWS Import Export for larger datasets, Glacier / S3 - tiered backup, S3 backup can be done either using S3 APIs or using virtual appliance like AWS Storage gateway to S3 / Glacier

key steps for backup and restore

* select appropriate tool / method for backup, ensure appropriate retention policy for data, ensure appropriate security measures are in place for the data, encryption and access policies. Storage gateway data transmission is https by default.

(ii) Pilot Light: more $ than (i), shorter RTO and RPO then (i)

* minimal version (small or most critical piece - database server) of the environment is always running in the cloud. (light it up during DR)
* AMI (preconfigured service / bundles) is ready in DR and using that spin of instances.
* networking perspective - while provisioning during DR use preallocated IP address or use preallocated ENI with preallocated mac address (for special licensing requirement) or use Route 53 DNS Failover with ELB or alias record / CNAME.
* test it out once in a while

(iii) Warm Standby: more $$ then (ii), even shorter RTO and RPO then (ii)

* scaled down version of the stack is fully up and running in the cloud. (smallest size possible)
* will not be able to take full production workload but its fully functional, during DR Autoscaling would scale it to take production work load.HZ scaling is preferred over Vertical scaling.
* use Route 53 automated health check failover.

(IV) Multi site: Active - Active, most expensive solution and RTO and RPO is almost zero.

* runs in Active Active configuration, even split. provision perfect mirror to active site.
* Route 53 weighted DNS, weight will be updated thru script or manually during DR.

DR and Business Continuity for Databases

RDS(MySQL, Oracle, SQL Server, PostgreSQL, MariaDB, Aurora) MultiAZ Failover

* in case of availability loss in primary AZ, connectivity / host failure of primary DB, software patching or rebooting the primary DB
* RDS MultiAZ deployment MySQL, Oracle, PostgreSQL - uses synchronous physical replication to achieve MultiAZ / HA.
* RDS MultiAZ deployment SQLServer - uses synchronous logical replication to achieve MultiAZ / HA.
* RDS MultiAZ Failover advantages - HA, Backups and Restores taken from secondary which avoids I/O suspensions of primary.
* RDS MultiAZ Failover is all about DR and Business continuity, its not a scaling solution. ReadReplicas for scale.

Read Replicas (Scaling Out)(MySQL 5.6 InnoDB, Postgres 9.3.5 or newer, MariaDB)

* Elastically scale out beyond the capacity of single DB instance for read - heavy workload. CreateDBInstanceReadReplicaAPI - source DB will be replicated using asynchronous replication.
* serving read traffic while source DB is unavailable, business reporting / data warehouse reporting
* while creating read replica, AWS will take a snapshot of the DB (if multi az is not enable brief I/O suspension of about 1 min, if enabled snapshot will be taken from secondary)
* new DNS endpoint for RR, RR can be promoted to primary (link will be broken), RR can’t be multi az (can’t have a DR of your read replica), for mysql you can have RR of RR (latency will be increased)

Storage Gateways

* virtual appliance you store on premise to replicate data from your on premise.
* can be deployed on premise or as an EC2 instance (to backup AWS EC2 production environment), can schedule snapshots
* can use storage gateway with Direct Connect, can implement bandwidth throttling (to slow down / limit) (e.g. allow 100Kbps for storage gateway bandwidth for replication),
* need Vmware’s ESXi or Hyper-V - hypervisors need this to run on VM, 4 or 8 vCPU, 7.5GB RAM, 75GB image / system data

(i) Gateway-Cached Volumes: primary data in S3 and frequently accessed data locally, (http://docs.aws.amazon.com/storagegateway/latest/userguide/storage-gateway-cached-concepts.html)

* low latency access to frequently accessed data, substantial cost saving for on prem storage.
* 1 volume = 32 TB, 32 volumes supported = 32 \* 32 TB = 1 PB of data can be stored (20 supported in the white paper)
* lot more economical for backups then some of the SAN (storage area network) solutions
* volume for actual data: need storage for local cache (e.g. 200 GB) and upload buffer (e.g. 300 GB)
* EBS snapshots provide durable offsite backup in S3 for the versions of your local cache

(ii) Gateway-Stored Volumes: All data locally on prem, backed up to S3 asynchronously (point in time snapshots of EBS stored in S3) (http://docs.aws.amazon.com/storagegateway/latest/userguide/storage-gateway-stored-volume-concepts.html)

* low latency access to entire dataset
* 1 volume = 16 TB, 32 volumes supported = 32 \* 16 TB = 512 TB of data can be stored
* volume for actual data: 1:1 relationship need storage for entire dataset and upload buffer (max 16 TB storage per volume)
* EBS snapshots provide durable offsite backup in S3, you can create a new EC2 instance and attach this EBS snapshot.

(ii) Gateway-Virtual Tape Library: limitless collection of virtual tapes, VTL - S3, VTS (Virtual Tape shelf) - Glacier (http://docs.aws.amazon.com/storagegateway/latest/userguide/storage-gateway-vtl-concepts.html)

VTL (backed by S3)

* Access to virtual tapes from your VTL is instantaneous.
* 1500 virtual tapes, 1 PB of total tape space(size of each tape varies between 100 GB to 2.5 TB)
* volume for actual data: need storage for local cache and upload buffer

VTS (backed by Glacier)

* Access to virtual tapes from your VTS takes about 24 hours.
* unlimited tapes

Networking: open 443 from external(inbound), 80 for activation only (can remove later), 3260 for iSCSI target, UDP 53 for DNS (outbound)

Encryption: Encrypted during transit via SSL and data is stored encrypted at rest using AES 256.

you can take point-in-time incremental snapshots of your volume(EBS snapshot) and store them in S3, also it can be scheduled snapshots. EBS volumes can be attached to the EC2 instance within the same AZ. EBS volumes are copied redundantly on multiple machines for redundancy within same AZ. From EBS volume you can create EBS snapshot which then can be used to create a new volume in a different AZ.

Multiple locations / office locations means - you need storage gateway at each location

AWS Import / Export

back your data to disk, carry over to amazon and they will upload / import to S3, EBS or Glacier using high speed internal connection

you can use it as export too (export only from S3). for export if versioning is enabled on S3, most latest version will be exported.

for 10Mbps internet connection to transfer 600GB- it would take 13 days (typically around 1TB or more, use AWS Import / Export, low bandwidth connection and large datasets.)

Steps

Signup to AWS —> Download Import / Export Tools —> Save Credentials to file —> create either import or export job.

import - encryption is optional, export - AWS will export with true crypt encryption, disk wipeout is your responsibility and not Amazons.

For large data backups to S3, AWS Import/Export Disk or Snowball is going to be the cheapest and potentially the fastest option.

Automated Backups

* all automated backups on S3
* services with automated backups are

(i) RDS: for MySQL you need InnoDB, delete RDS instance all automated backups are gone although manual backups will remain.performance hit when multi az is not enabled because automated backups will be taken from primary, during restore you can change the MySQL engine.

(ii) Elasticache (Redis Only): entire cluster is snapshotted so it will degrade performance, so set the proper timing / window

(iii) Redshift: by default amazon will enable automated backup with 1 day retention period, incremental snapshots take small amount of backup storage.

* EC2 doesn’t have no automated backups, you need to write scripts and schedule the incremental snapshots to be stored on S3. API call or python. schedule backup time wisely, snapshots are incremental.

**(Domain 2) Costing and Account Management**(http://docs.aws.amazon.com/IAM/latest/UserGuide/id*credentials*temp\_request.html)

Cross Account Access

* you can have multiple AWS account for Security, Billing purpose and through acquisitions, exercise “least privilege” security design method.
* both ways - allowing you to someone else’s account or allowing someone access to your account.

Steps for Delegating access to PROD account(read only s3 bucket) from DEV (http://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial\_cross-account-with-roles.html)

(i) create the policy first for Read only S3 bucket in PROD account (ii) create a role with cross account access(dev account number) in PROD (iii) apply the policy to that role in PROD account and note down the ARN, summary will also give you signinURL (iv) go to DEV account and create a custom policy with “AssumeRole” action with the ARN resource we noted down, and apply this policy to the developer group. (v) Test the account by switching the role on the console with the signinURL.

Consolidated Billing

* Paying Account can be linked to —> Test / Dev, Production and other accounts. Paying account will get one monthly bill per account. Paying account is independent and cannot access resources from any linked accounts (by default), all linked accounts are independent, 20 linked account (soft limit), when billing alert created on paying account it will be for all linked accounts together although you can still have billing alerts for individual accounts.
* S3 Pricing First 1 TB / month = $0.030 per GB, Next 49 TB = $0.0295 per GB, next 450 TB = $0.0290 per GB (calculate first 1 TB at 0.030 then next 49 at 0.0295, so on so forth = )

For 450 TB calculation would be == (1 \* 0.030 \* 1000) + (49 \* 0.0295 \* 1000) + (400 \* 0.0290 \* 1000) = $13075.50 per month

* Advantages: volume price discount, one bill per AWS account, easy to track charges and allocate costs.
* Best Practices: enable MFA for paying / root account, strong password for the paying / root account, paying account should only used for billing, don’t deploy resources here.
* CloudTrail for consolidated billing: its still going to be per AWS account / per region, can however consolidate logs using S3 bucket (turn on CloudTrail for paying account, create bucket policy that allows cross account access, Turn on CloudTrail in the other accounts and use the bucket that was created in the paying account.)

Tagging and Resource Groups

* Tags are key value pairs (metadata - data about data), tags can be inherited when AS, CloudFormation and Elastic Beanstalks create resources.
* Resource Groups: make it easy to group your resources using tags - group resources that share one or more tags. Resources groups is a collection of all the resources for those tagged and you get region, name, health checks information(alarms). EC2 = public / private ip addresses, ELB = port configurations, RDS = DB engine etc., you can export these info to CSV.
* great for keeping track of who is consuming what, also you can tag untagged resources (it provides a tag editor)

Reserved instances for EC2

* EC2 reservation is associated with instance type, instance family and Availability Zone.
* According to this blog https://aws.amazon.com/blogs/aws/ec2-reserved-instance-update-convertible-ris-and-regional-benefit/ AZ is now not required for reservation.
* On Demand = fixed hourly rate no commitment - ideal for ASG, short term spiky and unpredictable workload which can’t be interrupted, Test / QA instances

Spot = bid price, - have flexible start and stop time, large capacity needs for 3 or 4 hours (short timeframes),Spot instances are a cost-effective option for once-off compute requirements and can be accompanied with on-demand instances if there is a time-requirement.

dedicated = physically isolated hardware

Reserved = discount with capacity reservation 1 to 3 year - steady state / predictable usage, for production requirement.For servers that need to be available all the time and will be continuously running at a known capacity, reserved instances will be the most cost-effective pricing option

* Payments for RI: All upfront = Largest discount up-to 75%, Partial Upfront = Middle discount, No Upfront = Least Discount (still cheaper then On-Demand)

Spot instances are a cost-effective option for once-off compute requirements and can be accompanied with on-demand instances if there is a time-requirement.

* Modification of Reserved Instances: You can switch AZs for RI, You can change the instance type/size (e.g. from xlarge to large) within the same instance family (e.g. m4) by submitting a modification request, it will be processed providing the footprint remains the same by using the normalization factors.
* Instance Size - Normalization factor: micro = 0.5, small = 1, medium = 2, large = 4, xlarge = 8, 2xlarge = 16, 4xlarge = 32, 8xlarge = 64 , 10xlarge = 80

so 2 large instances = 2 \* normalization factor 4 = 8,

which is equivalent to 8 small instance = 8 \* normalization factor 1 = 8.

make sure its within the same instance family, small can’t be change to large. split them up in smaller reservation, so better to reserve large for access capacity, don’t reserve what you need now, think for future.

* SWITH AZs for Reservation: - If you have RI of 10 instances in us-east-1a and you file a request to move 5 instances to us-east-1b, the modification request will result into 2 new footprints (5 instance for us-east-1a and 5 instance for us-east-1b), splitting the footprint in half and it creates 2 brand new Reservations /footprints.
* if you have 4 on demand instance for m1.small in us-east-1a and you purchase 2 reserved instance for m1.small in us-east-1a, 2 RI’s will immediately applied since instancy type, instance family and AZ are same. (if AZ is different = no discount, it will be continue to charge the on demand instance price)
* RI modifications are not supported for REDHat and SUSE linux. you can also sell RI in marketplace.

Reserved instances for RDS

* RDS reservation is associated with DB Engine, DB instance class, Deployment type, License Model, Region (remember not AZ here). any change in one of these after buying the reservation and you will be charged on demand price
* You can have Reserved Instances with RDS Multi - AZ as well as Read Replicas. For Read Replicas DB Instance class and regions must be same.

EC2 instance types

Family / Specialty / Use Case

D2 / Dense Storage, High sequential IO/ Fileservers, Data Warehousing, Hadoop / MR, parallel filesystems

I2 / High IO performance / NoSQL DBs, Data Warehousing

R3 / Memory Optimized / Memory Intensive Apps, DBs

T2 / Lowest Cost, General purpose / Webservers, small DBs

M3, M4 / General Purpose / Application Servers

C3, C4 / Compute Optimized / CPU Intensive Apps, DBs

G2 / Graphics, General purpose GPU / Video Encoding, Machine Learning, 3D Application Streaming

**(Domain 3) Deployment Management**

CloudFormation

* it gives developers and systems administrators an easy way to create and manage a collection of related AWS resources, provisioning and updating them in an orderly and predictable fashion. you can modify and update them in a controlled way like version control to your AWS infrastructure. Infrastructure as a Code (IOC)
* Services Supported:

EC2,ELB,AutoScaling,S3,Route53,VPC, CloudFront,CloudTrail,CloudWatch, RDS,DynamoDB,SimpleDB,RedShift,ElastiCache, Beanstalk,OpsWorks, SNS,SQS,IAM,Kinesis

ACL’s, ElasticIPs, security groups, RDS security groups

* Template = architectural diagram(editable), Stack = JSON document, end result of the diagram(build the template), can update the stack for new updates, dont worry interdependencies
* Elements of Template: Mandatory - “AWSTemplateFormatVersion": "2010-09-09" <— file format and version, “Resources” and its config are mandatory, everything else is optional. Template parameters - limit 60, Output values - limit 60
* Output the data: "Fn::GetAtt": [ "CertificatelabsELB", "DNSName" ] , "Value": { "Ref": "ChainTesterServerGroup" } , "Fn::Select" : ["0", { "Ref" : “TagResource” } ]
* Mappings: "CidrBlock" : { "Fn::FindInMap" : [ "SubnetConfig", "VPC", "CIDR" ]}
* Chef & Puppet & Bootstrap scripts are supported.
* Stack Creation Errors: By default “automatic rollback on error” feature is enabled but you will still be charged for the resource usage. CFT itself is free.

WaitCondition: Stack can wait creating resources further along until the completion signal is received.

* You can specify deletionPolicy, for example EBS snapshots are created and stored in S3 before the stack is deleted, if S3 creation was part of the stack resource then delete policy could prohibit that as well.
* CloudFormation can be used to create roles(AWS::IAM::Role) in IAM and you can also grant roles to resources.
* You can specify IP address range in both ways: In terms of CIDR ranges as well as Individual IP addresses, also specify pre-existing EIPs.
* VPC peering: you can create multiple VPC in one CFT and can peer across two VPCs as long as both VPCs belong to the same account., also Route53 support (create new hosted zone, update existing zone, A record, CName, MX record etc.)

RDS

ElasticBeanstalk

* it makes it easier for developers to quickly deploy and manage applications in the AWS cloud. Developers simply upload their applications and Beanstalk handles deployment details of capacity provisioning, autoscaling and application health monitoring. (portal that you could use with no prior AWS experience and it will provision everything you need)
* Its uses ASG,ELB,EC2,RDS,SNS and S3 to provision things for predefined / already set configurations

Environment Tier - Webserver, Worker

Predefined Configurations - Node.JS for Nginx / HTTP server, PHP for HTTP server, Python for HTTP server, Ruby for Passenger, Java for Tomcat, Go, .NET for IIS 7.5,

preconfigured docker: Glassfish, Python or generic docker

Environment URL - has to be unique

Dashboard - Recent events, Monitor, Logs, Alarms, Upload and Deploy and Configurations

Configuration - Scaling, Instances (dirt, key pair), Notifications, Software configuration (e.g. PHP.ini), Networking tier (ELB, VPC config), Data tier(RDS)

Environment properties (Access key and secret key as parameters) / JVM settings / caching component

Access built-in CloudWatch monitoring and get SNS on health and other events for AWS resources

Access log files for EC2 without logging into instances

* CF vs ElasticBeanstalk: CF supports ElasticBeanstalk but not the other way around, ElasticBeanstalk is for those who has no prior AWS exposure whereas CF needs AWS experience, if you have a choice always use CF.
* AWSToolkit for Eclipse / Microsoft Visual Studios, also support for GIT (only modified files will be transmitted to ElasticBeanstalk), it can track different versions of the software so that rollback is easier.
* ElasticBeanstalk stores application files to S3, also optionally stores log files to S3, if you are using AWSToolkits like eclipse / visual studio the s3 bucket for your account will be created automatically and it will copy over the application files.
* You can store application files to S3 from instance, Bundle AWS SDK for java with your application WAR file(can we do IAM roles instead?). ElasticBeanstalk can provision a new RDS instance and connection string will be exposed to your application by environmental variable.
* Configure Elastic Beanstalk to be fault tolerant between AZs but not between regions.
* Security: app is publicly available at appxxx.elasticbeanstalk.com for anyone to access, you can restrict it via security groups or NACL, it also fully supports IAM, you can ssh into instances created by ElasticBeanstalk.
* The current version of AWS Elastic Beanstalk uses the Amazon Linux AMI or the Windows Server 2012 R2 AMI (other AMIs not supported)
* Multiple environments are allowed to support version control and rollback

OpsWorks

* Definition: OpsWorks is an application management service that helps you automate operational tasks like code deployment, software configurations, package installations, database setups and server scaling using Chef. Chef turns infrastructure into code, it becomes versionable, testable. Chef server stores recipes as well as other configuration data. Chef client is installed on each server, vm, container or networking device you manage - we’ll call these nodes. Client periodically polls chef server latest policy and state of your network, if anything is out of date, client brings it up to date.
* GUI to deploy and configure your infrastructure quickly. OpsWorks consists of two elements, Stacks and Layers.
* Stack: A group of collection of resources are called Stacks (ELBs, EC2 instances, RDS etc.)
* Layers: A layer exists within a stack and consists of things like (e.g Web Application layer, Database layer and so on), Opswors will take care of installing and configuring PHP and PHP.ini for example so that you don’t have to manually do it. its basically tiers in your stack. you can deploy the entire environment and not have to worry about ssh into it. Instance must be assigned to at least 1 layer / it can belong to multiple layers(e.g. web layer / admin layer). Preconfigured layers include: Applications, Databases, ELBs, Caching layer
* You can choose Role, select whether you want SSH access or not
* You can clone stacks, ideal for moving from dev to test to production. security groups will have to be manually deleted (annoying)
* Lifecycle events: Lifecycle events cause specific recipes to be run on all associated instances (e.g. Setup, configure, deploy, undeploy, shutdown etc.) each layer has multiple lifecycle events (chef recepies to run). blue / green deployment etc.

**(Domain 4) Network Design**

VPC Refresher - logical datacenters in AWS

* VPC Can span multiple AZ, but can’t span multiple regions, Custom VPC has to be /16 can’t go higher then that /8 is not allowed
* when you create Custom VPC it creates default security group, default network ACL and default route table., it doesn’t create default Subnet

One Subnet == one AZ, you can have security group spanning multiple AZ, ACL’s span across AZ (assign sg and ACL to two different subnets)

any CIDR block 5 reserved IPs (.0, .1, .2, .3, .255), so for CIRD block /24: 2^8 - 5 = 256 - 5 = 251 available IP address space (from any CIDR first 4 ip and last IP is reserved - work on CIRD calculation : https://www.youtube.com/watch?v=ls1mMyfnaC0&list=PLEmikcqMNQvO1iDKqccPY5A1fNGFJGIAO&index=42)

when you create internet gateway, by default its detached, attach it to VPC then, only 1 IGW per VPC

When you create a VPC Default Routetable(Main Routable) is created where the default Routes are 10.0.0.0/16 Local <— all subnets inside VPC will be able to talk to each other, Don’t touch Main route table

Create another routetable for route out to internet (0.0.0.0/0 IGW) <— route out to the internet, Last thing you associate this new route table to one of the subnet which will make it public. (you can enable auto assign public IP for the public subnet)

1 subnet can have 1 routetable , ICMP is for ping / monitor

NAT instance and NAT gateway

NAT Instance - disable source / destination check., always behind security group, must be in public subnet, must have an EIP, ,must be a route out of the private subnet to NAT

Increase the instance size if bottleneck

Change the main route table - add a route (0.0.0.0/0 NAT Instance target)

NAT Instance is a single point of failover (put it behind a ASG),

NAT gateway - released in 2016 - amazon handled

Amazon maintains it for you, no need to handle yourself. (security patches applied by AWS)

You can just create the gateway and assign EIP (put it in public subnet) (automatically assigned)

Change the main route table - add a route (0.0.0.0/0 NAT gateway target)

No need for disable source/destination check or no need to put it behind a security group - it handles it for you.

Highly available / redundancy no need for ASG.

NAT gateways are little bit costly - always use it in production scale automatically up to 10Gbps

VPC Peering

* Connection between two VPCs that enables you to route traffic between them using private IP addresses. VPC peering across regions are not possible but across AZ is possible. Peering with your own VPCs or across another account’s VPC as long as they are in the same region, transitive peering is not possible. AWS uses existing infrastructure of a VPC to create a VPC peering connection; it is neither a gateway nor a VPN connection (so traffic is not encrypted), and does not rely on a separate piece of physical hardware, There is no single point of failure for communication or a bandwidth bottleneck. you can only peer if the CIDR blocks doesn’t collide / overlap. (it will break the peer if its already established.)
* VPC Peering Limitations: soft limit of 50 VPC peer per VPC, can be risen up to 125 by request. A placement group can span peered VPCs (but you will get degraded performance); however you will not get full bandwidth between instances in the peered VPCs. In ingress / outguess security group you cannot reference peer VPC’s security group - instead use CIDR block in the source. Private DNS values cannot be resolved between instances in the peered VPCs. there is no charge for peering, however data transfer across the peering connections is charged.

Steps to setup VPC peering

* Owner of the local VPC sends a request to the owner of the second VPC to peer
* Owner of the second VPC has to accept.
* Owner of the local VPC adds a route to their route table allowing their subnets to route out to the peer VPC
* Owner of the peer VPC adds a route to their route table allowing their subnets to route back to the local VPC
* Security groups in both VPCs have to both allow traffic.
* If you are using NACL’s make sure they are allowing the traffic through.

Troubleshooting (important)

* If you can’t create a VPC peer, make sure if they are in the same region or check if the CIDR blocks are not overlapping.
* After the peer is setup, make sure security groups and NACLs allow traffic and check if the route is created in BOTH VPC’s routing tables.

DirectConnect (http://jayendrapatil.com/category/aws/direct-connect/)(https://aws.amazon.com/directconnect/faqs/) <— must read FAQ

* makes it easy to establish a dedicated network connection from your on premise to AWS. you can establish private connectivity between AWS and your datacenter which can reduce your network cost while using large volumes of traffic, increase bandwidth throughput and provide more consistent network experience than internet based connections.
* dedicated circuit / network connections from your network and one of the AWS Direct Connect locations using 802.1q VLANs. This dedicated connection can be partitioned into multiple VIFs (virtual interfaces). VIFs allow you to use the public VIF connection to access public resources such as objects in S3 or DynamoDB or EC2 instances by public IP address (e.g any public HTTPS service endpoints-S3,SES,SQS,SNS), and private VIF connection to private resources such as Amazon EC2 instances running in VPC using private IP space (its virtual interface but connection is same) maintaining the network separation between public and private environments.
* Direct Connect Benefits: site to site VPN is unreliable because it uses software tunnels and the connection could drop out (size of router), DC gives your increased bandwidth, increased reliability by giving consistent network experience, it reduces cost when using large volumes of traffic.
* How is it different from VPN ? VPN can be configured in minutes, if you have low to modest bandwidth requirements, and can tolerate the inherent variability in internet - based connectivity. AWS Direct Connect does not involve internet, it uses dedicated, private network connection between your intranet and Amazon VPC.
* Direct Connect Connections are available in 10Gbps, 1 Gbps or sub 1 Gbps (can be purchased through AWS Direct Connect Partners)
* diagram: http://jayendrapatil.com/category/aws/direct-connect/
* Public VIF = public ip address / Public end points, Private VIF = private ip address within VPC (both are separate)
* redundancy / failover: Direct connect itself is not redundant, get redundancy by option 1) two connections - two routers and two direct connects or 2) site to site VPN in place as backup, you can use BGP to failover automatically from direct connect to a site to site VPN. Bidirectional Forwarding Detection (BFD) detects the failover. CGW(customer gateway at your datacenter) vs VPG(virtual private gateway to AWS VPN)
* In US you only need 1 direct connect connection to connect in all 4 US regions. once established data transfer goes over AWS lines and not public internet.
* layer 2 connections are not supported with direct connect.
* Amazon CloudFront supports custom origins including origins you run outside of AWS. With AWS Direct Connect, you will pay AWS Direct Connect data transfer rates for origin transfer
* Direct Connect doesn’t guarantee any SLA.
* Route propagation is enabled on Virtual Private Gateway and not on CGW.

HPC (High performance computing and enhanced networking)

* Batch processing with large and compute intensive workloads (pharmaceutical and automotive industries), demands high performance CPU, Network and Storage, Usually Jumbo frames (ethernet frame with 9000 bytes of payload - regular frame is 1500 bytes of payload) are required. Shared filesystems such as Lustre and NFS use jumbo frames. Enhanced networking results in higher bandwidth, higher packet per second (PPS) performance, lower latency, consistency, scalability and lower jitter
* SR-IOV: is a method of device virtualization which provides higher I/O performance and lower CPU utilization, Use of jumbo frames is supported in AWS through enhanced networking. Enhanced networking is available using Single Root I/O Virtualization (SR-IOV): SR-IOV is supported on HVM(Hardware virtulization) VM and not PV(para virtualization). Type of supported instances are (C3, C4, D2, I2, M4, R3). Enhanced networking is only supported within VPC and not in EC2- Classic.

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/enhanced-networking.html

Placement groups

* Logical grouping of instances within a single AZ, it enables applications to participate in a low network latency or high network throughput of up to 10Gbps network and you must use instance type that supports enhanced networking (C3, C4, D2, I2, M4, R3)
* 1 placement group = 1 AZ, it can span across different subnets but the subnets must be in same AZ, can span across different VPC within same AZ / subnet - but not recommended due to bad performance. Existing instance cannot be moved to placement group. Amazon recommends to use homogenous(identical) instance types for all your instances to be placed in the placement group. Best practice to provision your placement group for peak load and launch all your ec2 instances in the placement group at the same time (you may and may not be able to add instances later on)

ELB: distributes incoming traffic across multiple ec2 instances in the cloud (distribute traffic / fault tolerance)

* ELB ports supported: 25(SMTP), 80/443(HTTP/HTTPS), 1024 - 65535. can’t assign and EIP to ELB. IPV4 and IPV6 supported although VPCs do not support IPV6 at this time. you can load balance the Zone Apex of your domain(ankitshah.com, aclouguru.com <— naked domains is zone apex) using alias record of Route 53. ELB ports in EC2 classic: 25, 80, 443, 465, 587, 1024-65535
* By turning on CloudTrail you can get ELB API calls history (usually in S3 bucket) for security analysis and troubleshooting. 1 ELB = 1 SSL certificate unless you have wildcard certificates.

Scaling NATs: Bottleneck can occur if too much traffic is being passed through NAT.

Scale up: increase your instance size, use the instance types that support enhanced networking(C3, C4, D2, I2, M4, R3)

Scale out: you can only have 1 NAT / each public subnet, so scaling out means add additional NAT by creating new public subnet. HA can be provisioned here as well but at one time only 1 NAT per public subnet.

*\*(Domain 5) Data Storage(15%) \**

(Must Read whitepaper: https://d0.awsstatic.com/whitepapers/AWS%20Storage%20Services%20Whitepaper-v9.pdf )

Read S3 from AWS Developer notes.

Optimize of S3

* Optimize for PUTS: For Strong Networks(fast internet connection - fibre) we want to take the advantage of the network and make the network itself a bottleneck. For Weaker Networks(slow / less reliable internet connection - edge, 3g) prevent large files having to restart their uploads <— for both scenarios Parallelizing the PUT (or even GET), divide large file into small files and push them simultaneously. chunk sizes: High bandwidth - 25 to 50 MB and low bandwidth - 10 MB, need to balance - too many chunks increases the connection overhead and too few doesn’t use enough bandwidth.
* Optimize for GETS: Two Options: (i) Use CloudFront - Multiple endpoints globally, low latency high transfer speed, caches objects from s3, Two varieties - RTMP and Web Distributions. (ii) Range Based GETS to get multithreaded performance - Using Range HTTP header in a GET request, you can retrieve a specific range of bytes in an object stored in Amazon S3. (e.g first 100 bytes) so it allows you to use multiple get requests simultaneously and maximized the bandwidth throughput.
* S3 is Lexographical: stored in dictionary order - introduce a random prefix into your objects within bucket so that your data is spread around S3 in different blocks / machines etc instead of being in a specific disk all together for multiple put request (large volumes). More randomness in your file structure within a bucket = better performance.

S3 Best Practice

* Versioning: no penalty in terms of performance, easy to delete, rollback versions (once you turn on versioning can’t go back, you can suspend though)
* Securing S3: (i) use bucket policy to restrict delete, turn on MFA for delete, also use MFA to change the versioning in your bucket, (ii) Ideally to secure S3 use another account, versioning protects you against individual file level and not against entire bucket, bucket deletion can be accidental and malicious —> so backup your bucket to another bucket owned by another account using cross account access. (gives an extra copy)

Database Design

RDS(MySQL, Oracle, SQL Server, PostgreSQL, MariaDB, Aurora): Multi AZ: Synchronous - commits to two different AZ during write, DR only not for scaling. RDS is compatible to existing SQL engines on premise, RDS is optimal for new application with structured data that requires sophisticated querying and joining capabilities.

RDS for ACID properties: Atomicity: either all commit or none in a transaction, Consistency: transaction creates a new valid data or rollbacks to previous, Isolation: not committed transactions must remain isolated, Durability: once committed to disk, its durable and in case of failure / restart you have the copy of that data.

Where not to use RDS: (i) index and query focused data = DynamoDB, (ii) BLOBs(Binary Large Objects) : audio, video, images, flat flies to be stored in S3 (iii) RDS provides push button scalability (i.e. you have to take the snapshot and scale up when needed, its not automated), so when the term Automated Scalability comes use DynamoDB (scale out == DynamoDB)

(iv) another DB platforms like IBM DB2, Sybase, informix etc then no RDS, you can use EC2 by having a custom db ami and installing db software on the instance. (iv) complete control: if you need OS level control , need root level control (e.g. require to install some 3rd party app etc.) — RDS doesn’t work.

Dynamo DB Use cases: ideal for existing or new application that need a flexible NoSQL DB with low read and write latencies, and the ability to scale storage and throughput up and down as needed without code changes or downtime.(e.g. mobile apps, gaming, digital ad serving, live voting and audience interactions for live events, sensor networks, log ingestion, access control for web based content, metadata for s3, e-commerce shopping carts, and session managment), it scales out automatically based on the demand / highly available / scalable database. Dynamo DB optimized for workloads which require high I/O rate per GB stored.

Where not to use DynamoDB: (i) pre-written / existing app that is already tied to relational db (ii) have JOINS and/or complex transactions or other relational infrastructure (ii) BLOB Data - use S3 but you can store metadata in dynamo db (iii) large data with very low I/O requirement / infrequently accessed data then S3 may be a better choice.

RDS with Read Replicas: Asynchronous(take time - replica lag), Not DR but scaling out, to improve performance use read replicas, remember enabling Multi AZ in RDS with actually decrease performance because of the synchronous replication.

**(Domain 6) Security (20%)**

AWS Directory Service: Managed service that allows you to connect your AWS resources with an existing on premise Microsoft Active Directory or to setup a new one in AWS Cloud.

Two Flavors/services: (i) AD Connector - connect existing Active Directory to AWS cloud without synchronization or federation infrastructure, you can use the existing corporate credentials to log on to AWS such as workspaces, workdays and manage AWS resources via IAM role based access. AD connector supports MFA.

(ii) Simple AD: new Active Directory service from scratch, doesn’t support MFA, can’t add existing AD servers, no trust relationship allowed and cannot transfer FSMO roles.

Security Token Service (STS): Grants users limited temporary access to AWS resources, 3 methods.

(i)Federation(with Active Directory): Uses SAML, grants temporary access based off the user’s AD credentials, doesn’t need to be a user in IAM. AssumeRoleWithSAML.

(ii)Federation(with Web Identity / Mobile Apps): Use Facebook / Amazon / Google or other OpenID providers (e.g. Salesforce) to log in. AssumeRoleWithWebIdentity

(iii)Cross Account Access: Let’s users from one AWS account access resources in another. AssumeRole

Federation: combining or joining list of users in one domain(IAM) with list of users in another domain(AD, Facebook, Google etc.)

Identity Broker: A service that allows you to take and identity from point A to join it (federate it) to point B

Identity Store / Domain: Services like AD, Facebook, Google etc.

Identities: A user of a service like Facebook, Google

Federation Steps for a scenario to access S3 bucket with Active directory user.

(i) Employee enters the username / password

(ii) Application calls identity broker which captures the username / password

(iii) Identity broker uses Org’s LDAP directory to validate the employee’s identity

(iv) Identity broker calls the new GetFederationToken API with specified IAM policy

(v) STS confirms the IAM policy and creates a new security credential that has 4 values: Access Key, Secret Access Key, a token, duration of token.

(vi) Identity broker returns the temporary security credentials to the application

(vii) application uses the temporary security credentials to make a request to S3.

(viii) S3 uses IAM to validate the credential to allow the requested operation on the S3 bucket

(IX) IAM provides S3 the go ahead to perform the requested operation.

Identity broker always authenticates with LDAP first and then AWS STS.

Monitoring in the Cloud:

CloudTrail: enables you to retrieve a history of API calls and other events for all the regions in your account. API calls, calls made by console, command line tools, SDKs etc. its for auditing and not logging. who did what, what date - time etc. CloudTrail can dump multiple account’s trail into single S3 bucket by cross account access.

CloudWatch / CloudWatch Logs: monitoring service for AWS resources(EC2, DynamoDB, RDS), CPU/DISK/Network metrics by default, custom metrics generated by your applications. it also monitors log files and set alarms(SNS). CloudWatch alarm history are only stored for 14 days. By default CloudWatch Logs will store your log data indefinitely for each log group. Logs are cheaper to store in S3 then CloudWatch logs (look for cost effective solution) but it depends on the environment.

CloudWatch to monitor CloudTrail: CloudTrail(api calls / audit trails) could send logs from all the regions to CloudWatch Logs log group. with CloudWatch logs metric filter (pattern like “ERROR”) and then you can set an alarm for that pattern. CloudWatch can also monitor multiple AWS account.

best place to store logs would be persistent or safe place - S3(if you want to make logs public to the world) or CloudWatch logs(Generally). You can send your logs to CloudWatch logs, S3 or third party (spunk / sumo logic etc..)

CloudHSM: physical device which safeguards and manages digital keys for strong authentication. CloudHSM service allows you to protect encryption keys within HSMs. You can securely generate, store and manage cryptographic keys used for data encryption such that they are only accessible by you. price: upfront fee / flat fee + hourly fee.

* Single Tenanted(1 physical device for you) / not shared, must be within VPC / VPC peering, you can use EBS Volume encryption, S3 Object encryption and key management with CloudHSM but it requires scripting (not out of the box).
* To achieve fault tolerance you need cluster of HSMs (at least two). Can integrate with RDS(Oracle and SQL server) as well as redshift. Monitor via syslog.

DDoS: Distributed Denial of Service attack. (makes the app unavailable using large packet flooding by using Amplification / reflection or botnets - PC’s or server that are compromised)

* Amplification / Reflection attack: attacker send a 3rd party server (DNS / NTP server) a request using spoofed IP address. Server will then respond to that request with a greater payload (28 to 54 times the original request). attacker can use multiple NTP server by making it distributed.
* Layer 7 attack, Attacker can use use BOTNETS(PCs or Server that are compromized multiple IPs) to initiate Flood of Get requests to web server or Slowloris (Partial Get requests - slow get request) which will keep the number of connections open and that means legitimate users can’t get the response.
* How to Mitigate DDoS ?

(i) Minimize the Attack Surface Area: use bastion host to limit the exposure to access your instances using RDP / SSH or DB servers and move web servers to private subnets. keep the bastion IP secret, don’t use CNAME for bastion to expose them. blacklist everything and whitelist specific source IP’s.

WAF sandwich (Web application firewall) in ASG before ELB in private subnet. (ELB—> WAF(ASG) —> ELB) — looking for anything malicious. AWS also has a managed WAF service as well WAF on the marketplace.

external ELB in public subnet —> WAF (ASG) —> internal ELB in private subnet—> web server. (http://jayendrapatil.com/category/aws/waf/)

(ii) Be Ready to Scale to Absorb the attack: Scale HZ and Vertically to defeat the strategy of DDoS attack. (HZ is better). Scaling will distribute the DDoS traffic to larger area which will then force attacker to counter attack with more resources which will buy the time to analyze the attack.

(iii) Safeguard Exposed Resources: In situation where you cannot eliminate entry points to your app, you can potentially use CloudFront, Route53 or WAFs which will safeguard the origins.

CloudFront - GeoRestriction (blacklist / whitelist)

Route 53 - With Alias recordset change the ELB to CloudFront distribution or to another ELB with higher capacity running WAFs. you don’t have to wait for DNS propagation while using Alias record. Another pattern is to spin up a new DDoS resilient environment (ASG, WAFs) using CloudFormation when under attack and change the Alias record. (expensive environment)

Private DNS - allows you to manage internal DNS names for your application resources, so move everything to private DNS.

WAF (Web Application Firewall) : to mitigate commonly targeted attacks with low volume use WAF. either use the WAF managed service or buy other WAF’s on the AWS Market place. (trend micro) - WAF protect agains sql injection etc / Layer 7 attack.

(iv) Learn Normal Behavior: If you learn normal behavior (expected and unexpected resource spikes) then you can spot abnormalities, you can create alarms to alert abnormal behavior, helps you to collect forensic data to understand attack.

(v) Create a Plan for attack: Having a plan ensures you have validated the architecture, you understand the cost of increased resiliency when user under attack, you know whom to contact

So DDoS related AWS resources: CloudFront, ELB, AutoScaling, WAF, CloudWatch, Route 53.

DDoS white paper: https://d0.awsstatic.com/whitepapers/Security/DDoS*White*Paper.pdf

IDS and IPS:

IDS(Intrusion Detection System): inspects all inbound and outbound network activity and identifies suspicious pattern that may indicate a network or system attack from someone. Only detects if somebody tries to intrude your network, it doesn’t prevent.

IPS(Intrusion Prevention System): is a network security / threat prevention technology that examines network flows to detect and prevent vulnerability exploits. stop malicious traffic. Both detect and prevention of intrusion.

IPS / IDS appliance in public subnet and agents installed on all EC2 instances, Agents reports the data to the appliance which then sent these logs to SOC (Security Operation Center) or S3.

**(Domain 7) Scalability and Elasticity**

CloudFront (FAQ: https://aws.amazon.com/cloudfront/faqs/): can be used to deliver your entire website including static, dynamic, streaming and interactive content using global network of edge locations. Content is then served from these edge locations for lowest latency. can work with S3, EC2, Route 53, ELB. It consist of CloudFront Distribution (URI with which end users interact) and Origin (you origin e.g. S3, EC2)

* Distribution types: Web Distributions (HTTP), RTMP Distribution(Stream Adobe RTMP - video etc.)
* Geo Restrictions / Geo Blocking: allow you to restrict contents to countries using whitelist or blacklists using console or API. viewer from blacklists get 403 errors and or / custom html page.
* Support GET, HEAD, PUT, POST, PATCH, DELETE, OPTIONS: Doesn’t cache response to PUT, POST, PATCH and DELETE. Caches response for GET, HEAD and may be OPTIONS.
* SSL: CloudFront can be used with HTTP or HTTPS. With SSL you can use default CloudFront URL or your own Custom URL with your own SSL certificate.
* Dedicated IP with Custom SSL: dedicated IP / every edge location, very expensive ($600 per SSL certificate per edge location per month) however it can support older browsers.
* SNI Custom SSL: relies on SNI extension of TSL protocol - allows one IP to serve multiple certificates based on hostname passed in during TLS handshake. no support for older browsers.save money.
* CNAME support: 100 CNAME / distribution. Also support Wildcard CNAMEs.
* Invalidate Requests: Similar to purge in AKAMAI. With CloudFront two options - (i) delete the content in origin and after TTL expires edge will fetch the deleted entry (ii) forcefully remove using invalidate API request (like malicious or must do quickly for whatever reasons) - you pay for this though.
* Zone Apex: Naked domain or zone apex can be aliased to your CloudFront distribution e.g. http://aaa.com ===> http://d1134234.cloudfront.net using Route 53 alias record. Route 53 doesn’t charge for the queries to Alias record.
* Dynamic Content Support: CloudFront supports dynamic content delivery using HTTP cookies (personalized contents). you can also log cookie values in the CloudFront Access logs.
* with signed URL, Signed cookies and OAI (Origin Access Identity) you can secure CloudFront to deliver the content to allow user or not. Private Content delivery etc.

ElasticCache (Memcached - simpler engine vs Redis)

Redis is more advanced caching engine.

Requirements Memcached/Redis

Simple Cache to Offload DB(Mostly Memcached ) Yes/Yes

Ability to Scale Horizontally(scale out / add remove nodes on demand) Yes/No

Multi-threaded performance(large core with multicore/multithreads) Yes/No

Sharded data across multiple Nodes Yes/No

Advanced data types (strings, hashes, lists, sets) No/Yes

Ranking / Sorting data sets (sort or rank in memory) No/Yes

Publisher / Subscriber capabilities(client being informed on events) No/Yes

Persistence (persist your keystone) No/Yes

Multi-AZ (automatic failover) No/Yes

Backup and Restore Capabilities () No/Yes

HA / replicate from primary to one or more read replica No/Yes

Kinesis (Kinesis Streams)(Any kind of scenario where you are streaming large amounts of data that needs to be processed quickly- Not a persistent storage)

* Enables you to build custom applications that process or analyze streaming data for specialized needs. you can continuously add various types of data such as clickstreams, application logs, and social media to Amazon Kinesis streams from hundred of thousands of sources. Within seconds the data will be available for your Amazon Kinesis applications to read and process the stream. Streaming of data = Kinesis Streams
* Data in Kinesis is stored for 24 hours by default, you can increase this to 7 days. (temporary store - not permanent)

Data Producers (EC2, client, mobile, servers with agents) can ingest data into Kinesis streams into one of the shard. and it will be processed by Data Consumer on EC2 instances (Kinesis stream Applications) process it and in turn will be persisted to S3, DynamoDB, Redshift or EMR.

Shards Stream contains multiple Shards. unit of measurement of data when referring to Kinesis. Shard provides capacity of 1 MB/second for data input and 2 MB/second for data output. Specify the number of shards to be created when you create stream. one shard supports 1000 PUT records / second. you can dynamically add / remove shards live (reshard your shard)

Shards contains Records: which has (i) sequence number, (ii) partition key and (iii) data itself (Blob)

Data producers:

Kinesis Streams API - PutRecord (single record), PutRecords(multiple data records),

KPL (Kinesis Producer Library) - open source software on GitHub for data ingestion.

Amazon Kinesis Agent - java agent to be installed on producers

Records: contains following 3 things

(i) Partition Key: shards are grouped by partition key. Partition key tell you which shard the data belongs to. its is specified by the applications putting the data into a stream.

(ii) Sequence Number: each data record has a unique sequence number (unique key). its is assigned by the stream after you write using client.putRecord or client.putRecords. you can’t use sequence numbers to logically separate data in terms of what shards they have come from, you can only do this using partition keys.

(iii) Blob: Data blob is the data your data producer adds to a stream. maximum size of a data blob is 1 MB.

With Kinesis stream - stores rolling buffer of data, data is only removed after timeout - no matter how many application consumes it.

SNS Mobile Push (Subset of SNS service - read SNS,SQS,SWF from developer course)

* SNS Mobile Push: Push your message out to Custom Mobile Apps (not just SMS text) and even desktops using one of the following supported push notification service (Don’t have to Remember): Amazon Device Messaging (ADM), Apple push Notification Service (APNS) for both iOS and Mac OS X, Baidu Cloud Push, Google cloud messaging for Android (GCM), Microsoft Push Notification service for Windows Phone(MPNS -windows phone), Windows Push Notification Service(WNS - windows itself)
* Steps (try to remember)

(i) Request Credentials for Mobile Platforms (Windows / Apple / Google) (ii) Use this credentials to generate a token (iii) create platform application object (iv) create platform endpoint object (v) publish message to mobile endpoint.

**(Domain 8) Cloud Migration and Hybrid**

Integrating with VMware

* AWS Management Portal for VMware vCenter plugin enables you to manage your AWS resources. The portal installs as a vCenter plug-in within your existing vCenter environment. AWS is doing lot more than just virtualization. right click > Migrate.
* Common Use Cases: (i) Migrate VMware VMs to Amazon EC2 (ii) Reach New Geographies, From vCenter console, use AWS as backup and disaster recovery region instead of spinning up another VMWare production DR (iii) Self Service AWS portal within vCenter (iv) Leverage vCenter experience while getting started with AWS.

Migrating using Storage Gateway

* Storage Gateway can be used a mechanism to migrate existing VM / production environment / Infrastructure to AWS Cloud. Data Center could be consist of hyper-v host, ESX host, xen host and app could be running on them and they use SAN or DAS as the storage. Using AWS Storage gateway it can take the snapshots of the vmdk files of the root device volumes and its going to replicate the snapshot over https to the AWS Storage Gateway Service (over https) and store the snapshots in S3. we can then create EBS volumes from those snapshots which can be attached to the EC2 instances. (migration tool - mount the snapshots)
* Consistent snapshots: Snapshots must be consistent while using Storage Gateway for migration to cloud. Snapshots only capture data that has been written to your storage volumes which can exclude data that has been buffered by either the client or OS.
* Easiest way to guarantee that your OS and file system have flushed their buffered data to disk prior to taking a snapshot is to take your storage volume offline. Now take the snapshots and turn the storage volume back online, other tools available to make sure nothing in the buffer as well.

Data Pipeline (released before lamda, with lambda data pipeline might deprecate in future.)

* AWS Data Pipeline is a web services that helps you reliably process and move data between different AWS compute and storage services as well as on-premise data sources to AWS at specified intervals. (LAB must watch before exam)
* Data Pipeline - Key Concepts: Pipeline is name of container that consist of datanodes, Activity, Precondition, Schedule required in order to move your data from one location to another.
* Resource: A pipeline can run on either EC2 instance or EMR instance.(Resource) <— preprogrammed instance - can’t login / automatically provisioned. ( EMR job, hive job, a sql job, instance that copies data and running some commands.) Pipeline will provision at the start and terminate these resources for you automatically once the task is finished.
* AWS Data Pipeline supplies a Task Runner package that can be installed on your on-premise hosts. it constantly polls AWS Data Pipeline Service for work to perform. When its time to run a particular activity (take DB snapshots and copy to S3, executing a db stored procedure etc) it will issue appropriate command to the Task Runner. (e.g. dump every night from on-premise mysql server to S3) - used in hybrid cloud mode.
* Pipeline consist of :
* Datanode: source/ destination for your data. e.g. Amazon S3 path ( s3://example-bucket/my-logs/logdata-#{ScheduleStartTime(‘YYYY-MM-dd-HH’)}.tgz ) (supports S3, SQL (RDS), DynamoDB, Redshift datanodes)
* Activity: action that AWS pipeline initiates, (e.g. EMR, hive jobs, copies, SQL queries, command line scripts) activity which either run on EC2 or EMR instance. custom activity using ShellCommandActivity
* Precondition (readiness check - before kick of data pipeline) : Optional pre condition check must complete successfully before activity is run. e.g. DynamoDBDataExists - does data exist, DynameDBTableExists - Does table exist , S3KeyExist ? - S3 path exist , S3PrefixExists - Does a file exist within S3 path, ShellCommandPrecondition - Custom precondtion
* Schedule: for pipeline activities run time and the frequency. only way to schedule cron before lambda was announced.

Network Migrations

* Allowable block sizes within your VPC subnets: between /28(smallest) netmask and /16(biggest) netmask. For all Subnets within the VPC, CIDR blocks must not overlap.
* CIDR Reservations: The first four IP addresses and the last IP address in each subnet CIDR block are not available for you to use.

For Instance in a subnet with CIDR block 10.0.0.0/24, the following 5 IP addresses are reserved:

10.0.0.0 : Network address

10.0.0.1 : Reserved by AWS for VPC router

10.0.0.2 : Reserved by AWS for mapping to the Amazon provided DNS

10.0.0.3 : Reserved by AWS for future use

10.0.0.255 : Network broadcast address. no support for broadcast in VPC, therefor its reserved.

* VPN to DirectConnect Migrations: Once DirectConnect is installed you configure is so that your VPN connection and your DirectConnect connection is within the same BGP community. You then configure BGP so that your VPN connection has a higher BGP cost than the direct connect connection, so now all the traffic will be routed through Direct Connect. VPN Connection is there for redundancy but it has a higher cost (BGP will failover to site to site VPN)

**My Video playlist on youtube (watch this at-least 2 times)**

<https://goo.gl/Ff4KCS>

**Other Notes**

AWS Architect professional exam will throw a lot of buzzwords at you, e.g. Fault Tolerant, High Availability etc. Each of these buzz words means one or more AWS services. Here is a list of buzzwords and the services that you should always think:

High Availability: Multi AZ resiliency, AS, CloudFront

Cost management: AS

Fault tolerant: ELB, AS

Bandwidth: hi bandwidth = lo latency

Raid 0(no redundancy / fault tolerance, high speed - low cost) - high I/O performance, Raid 1 - mirror two volumes together (disaster recovery, redundant , no performance improvement, writes latency increase) , Raid 5(R/W operation will continue, more popular, combination of performance, fault tolerance)

CloudFront FAQ

**Surprise in the exam**

I got at-least 4 to 5 questions on Docker (which i didn't know - now I will read), a few questions on modeling data around dynamoDB (which i happened to know).