**Question 1 (of 7): Amazon Glacier is designed for: (Choose 2 answers)**

* A. active database storage.
* B. infrequently accessed data.
* C. data archives.
* D. frequently accessed data.
* E. cached session data.

**Answer:** B. infrequently accessed data. C. data archives.

Think “cold storage” and the name [Glacier](http://aws.amazon.com/glacier/faqs/) makes a bit more sense.  AWS includes a number of storage solutions and as per the to pass the exam, [you are expected](http://awstrainingandcertification.s3.amazonaws.com/production/AWS_certified_solutions_architect_associate_blueprint.pdf) to know the appropriate use of all of them.

I picture them on the following scale:

Instance (aka ephemeral, aka local) storage is a device like a RAM disk physically attached to your server (your EC2 instance) and characteristically it gets completely wiped every reboot.  Naturally this makes it suitable for temporary storage, but nothing that needs to survive something as simple as a reboot. You can store the Operating System on there if nothing important gets stored there after the instance is started (and bootstrapping completes).  Micro-sized [instance types](http://aws.amazon.com/ec2/instance-types/)(low specification servers) don’t have ephemeral storage.  Some larger more expensive instance types come with SSD instance storage for higher performance.

[Elastic Block Store](http://aws.amazon.com/ebs/) (EBS) is a service where you buy devices more akin to a hard disk that can be attached to one (and only one -at the time of writing) EC2 instance.  They can be set to persist after an instance is restarted.  They can be easily “snapshotted”, i.e. backed up in away that you can create a new identical device and attach that to the same or another EC2 instance.  One other thing to know about EBS is that you can pay extra money for what is known as provisioned IOPS which means guaranteed (and very high if you like) disk read and write speeds.

[S3](http://aws.amazon.com/s3/) is a cloud file storage service more akin to DropBox or GoogleDrive.  It *is* possible to attach a storage volume created and stored in S3 to an EC2 instance, but this is no longer recommended (EBS is preferable).  S3 is instead for storing things like your EC2 server images (Amazon Machine Images aka AMIs), static content e.g. for a web site, input or output data files (like you’ve use an SFTP site), or anything that you’d treat like a file.

An S3 store is called a bucket whilst living in one specified global region, has a globally unique name.  S3 integrates extremely will with the [CloudFront](http://aws.amazon.com/cloudfront/faqs/) content distribution service which offers caching of content to a much more globally distributed set of edge locations (thus improving performance and saving bandwidth costs).

Glacier comes next as basically a variant on S3 where you expect to want to view the files either hardly ever or never again.  For example old backups, old data only kept for compliance purposes.  Instead of a bucket, Glacier files are stored in a Vault. Instead of getting instant access to files, you have to make a retrieval request and wait a number of hours. S3 and Glacier play very nicely together because you can set up [Lifecycles](http://docs.aws.amazon.com/AmazonS3/latest/dev/object-lifecycle-mgmt.html) for S3 objects which cause them to be moved to Glacier after a certain trigger e.g. a certain elapsed “expiry” time passing.

**Wrong answers:**

A. active database storage.

Obviously databases are written to regularly i.e. the polar (excuse the pun) opposite of Glacier.

Amazon offer a 5 different options for databases.

[RDS](http://aws.amazon.com/rds/faqs/)is the Relational Database Service. This allows Amazon to handle the database software for you (including licenses, replication, backups and more). You aren’t given access to any underlying EC2 servers and instead you simply connect to the database using your preferred method (e.g. JDBC). NB. currently this supports MySQL, Oracle, PostGreSQL and Microsoft SQL Slug.

[SimpleDB](http://aws.amazon.com/simpledb/faqs/) is a non-relational database service that works in a similar way to RDS.

[Redshift](http://aws.amazon.com/redshift/) is Amazon’s relational data warehouse solution capable of much larger (and efficient at large scale) storage.

[DynamoDB](http://aws.amazon.com/dynamodb/faqs/) is Amazons NoSQL managed database service. For this storage Amazon apparently uses Solid Stage Devices for high performance.

Finally of course, you can create servers with EC2 and install the database software yourself and work as you would in your data centre. This is the only time that you would need to consider what storage solution you actually want to use for a database.  EBS would be most appropriate.  Clearly Instance storage is a very risky option due to not persisting after restarts.  S3 is inappropriate for databases especially for Oracle which can efficiently manage raw storage devices rather than writing files to a file system.

D. frequently accessed data.

Clearly this is the opposite of Glacier.  Obviously if your data doesn’t need to persist after restarts, Instance storage would be the best choice for Frequently accessed data. Otherwise EBS is the choice if you applications are reading and writing the data. S3 (plus CloudFront) is the option if end users access your data over the www.

E. cached session data.

[ElasticCache](http://aws.amazon.com/elasticache/faqs/) is the AWS that provides a  [Memcached](http://www.memcached.org/) or [Redis](http://www.redis.io/) compliant caching server that your applications can make use of.  Your web application front end consists of multiple EC2 instances behind an Elastic Load Balancer.

**Question 2 (of 7): You configured ELB to perform health checks on these EC2 instances. If an instance fails to pass health checks, which statement will be true?**

* A. The instance is replaced automatically by the ELB.
* B. The instance gets terminated automatically by the ELB.
* C. The ELB stops sending traffic to the instance that failed its health check.
* D. The instance gets quarantined by the ELB for root cause analysis.

**Answer:** C. The ELB stops sending traffic to the instance that failed its health check.

This question tests that you properly understand how auto-scaling works. If you don’t, you might take a guess that load balancers take the more helpful sounding option A, i.e. automatically replacing a failed server.

The fact is, an elastic load balancer is still just a load balancer. Arguably when you ignore the elastic part, it is quite a simple load balancer in that (currently) it only supports round robin routing as opposed to anything more clever (perhaps balancing that takes into account the load on each instance).

The elastic part just means that when new servers are added to an “auto-scaling group”, the load balancer recognises them and starts sending them traffic. In fact to make answer A above, you need the following:

* *A launch configuration* This tells AWS how to stand up a bootstrapped server that once up is ready to do work without any human intervention
* *An auto-scaling group* This tells AWS where it can create servers (could be subnets in different Availability Zones in one region (NB. subnets can’t span AZ’s), but not across multiple regions).  Also: which launch configuration to use, the minimum and maximum allowed servers in the group, and how to scale up and down. By how to scale up and down, it means for example 1 at a time, 10% more and various other things.  With both of these configured, the when an instance fails the heath checks (presumably because it is down), it is the auto scaling group that will decide whether we now need to add another server t to compensate.

Just to complete the story about auto scaling, it is worth mentioning the [CloudWatch](http://aws.amazon.com/cloudwatch/) service. This is the name for the monitoring service in AWS. You can add custom checks and use these to trigger scaling policies to expand or contract your group of servers (and of course the ELB keeps up and routes traffic appropriately).

**Wrong answers:**

A. The instance is replaced automatically by the ELB.

As described above, you need an Auto Scaling group to handle replacements.

B. The instance gets terminated automatically by the ELB.

As discussed above, load balancers aren’t capable of manipulating EC2 like this.

D. The instance gets quarantined by the ELB for root cause analysis.

There is no concept of quarantining.

**Question 3 (of 7):You are building a system to distribute confidential training videos to employees. Using CloudFront, what method could be used to serve content that is stored in S3, but not publically accessible from S3 directly?**

* A. Create an Origin Access Identity (OAI) for CloudFront and grant access to the objects in your S3 bucket to that OAI.
* B. Add the CloudFront account security group “amazon-cf/amazon-cf-sg” to the appropriate S3 bucket policy.
* C. Create an Identity and Access Management (IAM) User for CloudFront and grant access to the objects in your S3 bucket to that IAM User.
* D. Create a S3 bucket policy that lists the CloudFront distribution ID as the Principal and the target bucket as the Amazon Resource Name (ARN).

**Answer:** A. Create an Origin Access Identity (OAI) for CloudFront and grant access to the objects in your S3 bucket to that OAI.

An Origin Access Identity is a special user that you will set up the CloudFront service to use to access you restricted content, see [here](http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-restricting-access-to-s3.html).

**Wrong Answers:**

B. Add the CloudFront account security group “amazon-cf/amazon-cf-sg” to the appropriate S3 bucket policy.

The CloudFront OAI solution is more tightly integrated with S3 and you don’t need to know implementation level details like the actual user name as that gets handled under the covers by the service.

C. Create an Identity and Access Management (IAM) User for CloudFront and grant access to the objects in your S3 bucket to that IAM User.

[IAM](http://aws.amazon.com/iam/faqs/) is the service for controlling who can do what within your AWS account. The fact is that an AWS account is so incredibly powerful, that it would be far too dangerous to have many people in a company with full access to create servers, remove storage, etc. etc.

IAMs allows you to create that fine grained access to use of services. It doesn’t work down to the level suggested in this answer of specific objects. IAMs could stop a user accessing S3 admin functions, but not specific objects.

D. Create a S3 bucket policy that lists the CloudFront distribution ID as the Principal and the target bucket as the Amazon Resource Name (ARN). When configuring Bucket policies, a Principal is one or more named individuals in receipt of a particular policy statement. For example, you could be listed as a principal so that you can be denied access to delete objects in an S3 bucket. So the terminology is misused.

**Question 4 (of 7): Which of the following will occur when an EC2 instance in a VPC (Virtual Private Cloud) with an associated Elastic IP is stopped and started? (Choose 2 answers)**

* A. The Elastic IP will be dissociated from the instance
* B. All data on instance-store devices will be lost
* C. All data on EBS (Elastic Block Store) devices will be lost
* D. The ENI (Elastic Network Interface) is detached
* E. The underlying host for the instance is changed

**Answers:** B. All data on instance-store devices will be lost

(See storage explanations above)

E. The underlying host for the instance is changed

Not a great answer here.  You are completely abstracted from underlying hosts.  So you have no way of knowing this.  But by elimination, I picked this.

**Wrong Answers:**

A. The Elastic IP will be dissociated from the instance

This is the opposite of the truth. Elastic IPs are sticky until re-assigned for a good reason (such as the instance has been terminated i.e. it is never coming back).

C. All data on EBS (Elastic Block Store) devices will be lost

EBS devices are independent of EC2 instances and by default outlive them (unless configured otherwise). All data on Instance storage however will be lost and also on the root (/dev/sda1) partition of S3 backed servers.

D. The ENI (Elastic Network Interface) is detached

As far as I know, just as silly answer!

**Question 5 (of 7): In the basic monitoring package for EC2, Amazon CloudWatch provides the following metrics:**

* A. web server visible metrics such as number failed transaction requests
* B. operating system visible metrics such as memory utilization
* C. database visible metrics such as number of connections
* D. hypervisor visible metrics such as CPU utilization

**Answer:** D. hypervisor visible metrics such as CPU utilization

Amazon needs to know this anyway to provide IaaS, so it seems natural that they share it.

**Wrong Answers:**

A. web server visible metrics such as number failed transaction requests

Too detailed for EC2 – Amazon don’t even want to know whether you have or haven’t even installed a web server.

B. operating system visible metrics such as memory utilization

Too detailed for EC2 – Amazon don’t want to interact with your operating system.

C. database visible metrics such as number of connections

Too detailed for EC2 – Amazon don’t even want to know whether you have or haven’t even installed a web server.  NB. the question states Ec2 monitoring, RDS monitoring does include this.

**Question 6 (of 7): Which is an operational process performed by AWS for data security?**

* A. AES-256 encryption of data stored on any shared storage device
* B. Decommissioning of storage devices using industry-standard practices
* C. Background virus scans of EBS volumes and EBS snapshots
* D. Replication of data across multiple AWS Regions E. Secure wiping of EBS data when an EBS volume is un-mounted

**Answer:** B. Decommissioning of storage devices using industry-standard practices

Clearly there is no way you could do this, so AWS take care.

**Wrong Answers:**

A. AES-256 encryption of data stored on any shared storage device

Encryption of storage devices (EBS) is your concern.

C. Background virus scans of EBS volumes and EBS snapshots

Too detailed for EC2 – Amazon don’t want to interact with your data.

D. Replication of data across multiple AWS Regions

No, you have to do this yourself.

E. Secure wiping of EBS data when an EBS volume is un-mounted

An un-mount doesn’t cause an EBS volume to be wiped.

**Question 7 (of 7): To protect S3 data from both accidental deletion and accidental overwriting, you should:**

* A. enable S3 versioning on the bucket
* B. access S3 data using only signed URLs
* C. disable S3 delete using an IAM bucket policy
* D. enable S3 Reduced Redundancy Storage
* E. enable Multi-Factor Authentication (MFA) protected access

**Answer:** A. enable S3 versioning on the bucket

As the name suggests, S3 versioning means that all versions of a file are kept and retrievable at a later date (by making a request to the bucket, using the object ID and also the version number). The only charge for having this enabled is from the fact that you will incur more storage. When an object is deleted, it will still be accessible just not visible.

**Wrong Answers:**

B. access S3 data using only signed URLs

Signed URLs are actually part of CloudFront which as I mentioned earlier is the content distribution service. These protect content from un-authorised access.

C. disable S3 delete using an IAM bucket policy

No such thing as an IAM bucket policy.  There are IAM policies and there are Bucket policies.

D. enable S3 Reduced Redundancy

Reduced Redundancy Storage RRS is a way of storing something on S3 with a lower durability, i.e. a lower assurance from Amazon that they won’t lose the data on your behalf. Obviously this lower standard of service comes at a lower price. RRC is designed for things that you need to store for convenience e.g. software binaries, but if they got deleted you could recreate (or re-download). So with this in mind enabling RRC reduces the level of protection rather than increases it. It is worth noticing the incredible level of durance that S3 provides. Without RRC enabled, durability is 11 9s, which equates to

“If you store 10,000 objects with us, on average we may lose one of them every 10 million years or so. This storage is designed in such a way that we can sustain the concurrent loss of data in two separate storage facilities.”

(see [here](http://aws.typepad.com/aws/2010/05/new-amazon-s3-reduced-redundancy-storage-rrs.html), thanks to [here](https://blog.cloudsecurityalliance.org/2010/05/24/amazon-aws-11-9s-of-reliability/)).

With RRC, this drops to 4 9s which is still probably probably better than most IT departments can offer.

E. enable Multi-Factor Authentication (MFA) protected access

This answer is of little relevance. As I mentioned accounts on AWS are incredibly powerful due to the logical nature of what they control. In the physical world it isn’t possible for someone to press a button and delete an entire data centre (servers, storage, backups and all). In AWS, you could press a few buttons and do that, not just in one data centre, but in ever data centre you’ve used globally. So MFA is a mechanism for increasing security over people accessing your AWS account. As I mentioned earlier IAMS is the mechanism for further restricting what authenticated people are authorised to do.

**Good Luck!!**