1. difference between scalability and elasticity in aws?

|  |  |
| --- | --- |
|  | Scalability is the ability of the system to accommodate larger loads just by adding resources either making hardware stronger (scale up) or adding additional nodes (scale out).  Elasticity is the ability to fit the resources needed to cope with loads dynamically usually in relation to scale out. So that when the load increases you scale by adding more resources and when demand wanes you shrink back and remove unneeded resources. Elasticity is mostly important in Cloud environments where you pay-per-use and don't want to pay for resources you do not currently need on the one hand, and want to meet rising demand when needed on the other hand. |
|  |  |

1. Connection draining in aws?
2. **Connection Draining** feature for your Elastic Load Balancers. You can do this from the [AWS Management Console](https://aws.amazon.com/blogs/aws/elb-connection-draining-remove-instances-from-service-with-care/), the [AWS Command Line Interface](https://aws.amazon.com/cli/), or by calling the **[ModifyLoadBalancerAttributes](https://docs.aws.amazon.com/ElasticLoadBalancing/latest/APIReference/API_ModifyLoadBalancerAttributes.html" \t "_self)** function in the [Elastic Load Balancing API](https://docs.aws.amazon.com/ElasticLoadBalancing/latest/APIReference/Welcome.html). You simply enable the feature and enter a timeout between one second and one hour. Connection Draining is enabled by default for load balancers that are created using the Console.

When you enable connection draining, you can specify a maximum time for the load balancer to keep connections alive before reporting the instance as de-registered. The maximum timeout value can be set between 1 and 3,600 seconds (the default is 300 seconds). When the maximum time limit is reached, the load balancer forcibly closes connections to the de-registering instance.

**To enable connection draining using the console**

1. Open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. On the navigation pane, under **LOAD BALANCING**, choose **Load Balancers**.
3. Select your load balancer.
4. On the **Instances** tab, for **Connection Draining**, choose **(Edit)**.
5. On the **Configure Connection Draining** page, select **Enable Connection Draining**.
6. (Optional) For **Timeout**, type a value between 1 and 3,600 seconds.
7. Choose **Save**.
8. RDS retention period?
9. You can set the backup retention period when you create a DB instance. If you don't set the backup retention period, the default backup retention period is one day if you create the DB instance using the Amazon RDS API or the AWS CLI, or seven days if you create the DB instance using the AWS Console. For Amazon Aurora DB clusters, the default backup retention period is one day regardless of how the DB cluster is created. After you create a DB instance, you can modify the backup retention period. You can set the backup retention period to between 1 and 35 days. You can also set the backup retention period to 0, which disables automated backups
10. **What is the difference between a spot instance and a demand instance on EC2?**
    1. On-Demand" instances allow the user to use the compute by hour without requiring long term commitment. There are no guarantees that  the user will always be able to launch specific instance types in an availability zone, though AWS tries it's best to meet the needs. This service is preferable for POCs and they do not suffer an interruption of the service(by AWS) like Spot instances.  
       "Spot" instances are a bid\_for\_low\_price version of On-Demand instances, but could be shut down by AWS anytime the Spot instance price goes higher than bid price. Spot price fluctuates based on the supply and demand of the capacity. It's essentially the leftover capacity of AWS to be used. There is no difference in the performance compared to On-Demand instances and is usually cheaper than On-demand instances as there is no guarantee provided over the availability. The user can choose a start time and end time for the instances or can make a persistent request(no end time specified) for this service. This service is preferable for computing needs which are not tied to any deadlines, computing needs are large and the interruption of service is acceptable.

You can provision Spot Instances directly using Amazon EC2. You can also provision Spot Instances using other services in AWS.

Auto Scaling and Spot Instances

You can create launch configurations with a bid price so that Auto Scaling can launch Spot Instances.

Amazon EMR and Spot Instances

There are scenarios where it can be useful to run Spot Instances in an Amazon EMR cluster.

1. encryption methods in aws?

**Use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)** – Each object is encrypted with a unique key employing strong multi-factor encryption. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. Amazon S3 server-side encryption uses one of the strongest block ciphers available, 256-bit Advanced Encryption Standard (AES-256), to encrypt your data.

**Use Server-Side Encryption with AWS KMS-Managed Keys (SSE-KMS)** – Similar to SSE-S3, but with some additional benefits along with some additional charges for using this service. There are separate permissions for the use of an envelope key (that is, a key that protects your data's encryption key) that provides added protection against unauthorized access of your objects in S3. SSE-KMS also provides you with an audit trail of when your key was used and by whom. Additionally, you have the option to create and manage encryption keys yourself, or use a default key that is unique to you, the service you're using, and the region you're working in..

**Use Server-Side Encryption with Customer-Provided Keys (SSE-C)** – You manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects.

1. [Change Region For An Amazon RDS Instance](https://webmasters.stackexchange.com/questions/42959/change-region-for-an-amazon-rds-instance)**?**

To launch an instance from a snapshot in a different region, you have to first **copy** the snapshot from the region where it was created and stored, into the target region.

In the RDS console, from the origin region, choose "Snapshots," then select the snapshot you want to copy, then click "Copy Snapshot." You will be given a choice of the destination region for the snapshot copy.

After the copy is complete, you'll see the snapshot under "Snapshots" in the target region. From there, you should be able to use that snapshot to create a new instance.

1. Ansible advantages?

Agent-less

Verylow overhead

Good performance

1. Ansible main components?
2. Inventory

Playbook

Modules

Ansible.cfg