High Availability and Autoscaling

14 March 2024 15:33

What do we scale?

- Determine the resource that needs to be scaled, such as EC2 instances, databases, or other AWS services.

2. Where do we scale?

- Decide where the scaling should occur within the AWS infrastructure, considering factors like VPC configuration, availability zones, and load balancer placement.

3. When do we need to scale?

- Identify the triggers or conditions that indicate when scaling is necessary, such as CloudWatch alarms, performance metrics, or user demand patterns.

These three questions provide a framework for making scaling decisions in AWS environments and are crucial to consider when designing scalable and highly available architectures.
Vertical scaling involves increasing the capacity of individual resources, like resizing an EC2 instance to a larger size, while horizontal scaling involves adding more instances or resources to distribute the workload. Horizontal scaling is typically preferred for its scalability, fault tolerance, and cost-effectiveness.
In the context of AWS, CloudWatch alarms are often used to trigger scaling actions based on predefined thresholds or conditions. These alarms can be set up to monitor various metrics, such as CPU utilization, network traffic, or database connections, and automatically initiate scaling actions when certain thresholds are exceeded.
Understanding these concepts and questions will help you effectively design, deploy, and manage scalable architectures on AWS.

Launch Templates:

- Launch templates allow you to specify all the settings required to build an EC2 instance, such as AMI, instance type, networking, security groups, storage, and more.
- They are a collection of configuration settings that can be reused to deploy similar instances without going through the EC2 instance creation wizard repeatedly.
- Launch templates support versioning, allowing you to create multiple versions and select a default version for deployment.
- They offer more granularity and flexibility compared to launch configurations.
- Launch templates are recommended over launch configurations due to their support for all EC2 auto-scaling features and versioning capabilities.

Launch Configurations:

- Launch configurations are the predecessor to launch templates.
- They have limited configuration options compared to launch templates and do not support versioning.
- Launch configurations are immutable, meaning any changes require creating a new launch configuration.
- While launch configurations are still supported, AWS recommends using launch templates for their enhanced features and flexibility.

Demo Highlights:

- In the console demo, a launch template was created step-by-step, specifying settings such as AMI, instance type, networking, storage, user data, and tagging.
- Versioning of launch templates was demonstrated, allowing for the creation of multiple versions with different configurations.

Exam Tips:

- Understand the customization options available in launch templates, including image selection, instance type, security groups, networking, and more.
- Launch templates are the preferred method for creating EC2 instance templates, offering more

features and flexibility compared to launch configurations.

- User data can be included in launch templates to execute scripts or commands upon instance launch.
- Launch templates support versioning, while launch configurations are immutable.
- Networking information can be optionally included in launch templates, although it's typically specified at the auto-scaling group level.

By mastering launch templates and launch configurations, you'll be better prepared to design, deploy, and manage EC2 instances in AWS environments.

From <https://chat.openai.com/c/d6fe90a1-a00b-4374-9bfc-54c5891880f3

Auto-Scaling Groups:

- Auto-scaling groups are resources in AWS that contain a collection of EC2 instances treated as a collective group for scaling and management purposes.
- They allow for scaling up and down based on demand, providing elasticity to your infrastructure.
- Auto-scaling groups leverage launch templates or launch configurations to define the configuration settings for EC2 instances.
- High availability is achieved by distributing instances across multiple availability zones (AZs) within the selected VPC.
- Instances within auto-scaling groups can be registered behind a load balancer for traffic distribution and load balancing.
- Auto-scaling groups support setting capacity limits, including minimum, maximum, and desired instance counts.

Capacity Limits:

- Minimum: Specifies the lowest number of EC2 instances that the auto-scaling group will maintain.
- Maximum: Specifies the highest number of instances that the auto-scaling group will provision.
- Desired: Specifies the initial number of instances to launch when the group is created.

Lifecycle Hooks:

- Lifecycle hooks allow for performing custom actions on instances during specific lifecycle events, such as launching or terminating instances.
- Lifecycle hooks can be configured to wait for up to two hours before proceeding with the lifecycle action, enabling custom configuration or validation tasks.
- Examples of lifecycle hook use cases include configuring proprietary software during instance launch or capturing application logs before instance termination.

Exam Tips:

- Understand the importance of high availability in auto-scaling groups, achieved through distributing instances across multiple availability zones and leveraging load balancers.
- Be familiar with configuring capacity limits, including minimum, maximum, and desired instance counts.
- Learn about lifecycle hooks and their use cases, such as custom configuration tasks during instance launch or capturing logs before instance termination.
- Understand how auto-scaling groups distribute instances across availability zones for better resilience and availability.

By mastering auto-scaling groups, you'll be better prepared to design and implement scalable and highly available architectures in AWS

From <https://chat.openai.com/c/d6fe90a1-a00b-4374-9bfc-54c5891880f3>

Step Scaling:

- Step scaling applies stepped adjustments based on the size of the alarm breach.
- Different steps are defined for scaling out (adding instances) and scaling in (removing instances) based on specified metric thresholds.
- Each step defines the number of instances to add or remove at different ranges of the metric.

  Simple Scaling:
- Simple scaling relies on metrics for scaling needs, such as CPU utilization.
- When a metric threshold is breached, a fixed number or percentage of instances is added or removed from the auto-scaling group.

  Target Tracking:
- Target tracking maintains a specified metric value (e.g., CPU utilization) by dynamically adjusting the number of instances in the auto-scaling group.
- The auto-scaling group adds or removes instances to maintain the target metric value.

  Instance Warmup and Cool-down:
- Instance warmup allows time for newly launched instances to initialize and pass health checks before being placed behind a load balancer.
- Cool-down periods pause auto-scaling for a specified time after a scaling activity to prevent rapid scaling events and avoid excessive cost.

  Other Scaling Types:
- Reactive Scaling: Scaling based on real-time metrics to respond to increased or decreased demand.
- Scheduled Scaling: Preemptively scaling resources based on predictable workload patterns, such as anticipated traffic spikes during events like Black Friday.
- Predictive Scaling: Using machine learning algorithms to forecast resource needs and scale proactively.

  Exam Tips:
- Understand how to configure minimum, maximum, and desired capacity in auto-scaling groups.
- Consider cost implications and adjust scaling strategies accordingly, scaling out aggressively but scaling in conservatively.
- Utilize EC2 reserved instances and compute saving plans to save costs for predictable workloads.
- Pay attention to provisioning times and optimize by pre-baking AMIs.
- Use CloudWatch for monitoring and alerting to trigger auto-scaling actions based on defined policies.

  By mastering these auto-scaling concepts, you'll be better prepared to design and implement scalable and cost-efficient architectures in AWS.>

Four Ways to Scale Relational Databases:

1. Vertical Scaling: Increase the compute power of your database by resizing to a larger instance type. This offers greater performance but may result in higher costs.
2. Scaling Storage: Storage for RDS and Aurora databases can be scaled up but not down. Careful planning is needed to balance performance needs with cost considerations.
3. Read Replicas: Create read-only copies of your database to offload read-intensive workloads. Read replicas cannot be used for writing data and are not meant for high availability or failover purposes.
4. Aurora Serverless: Offload scaling responsibilities to AWS with Aurora Serverless, which automatically adjusts capacity based on demand. Ideal for unpredictable workloads.

Exam Tips:

- Understand the trade-offs between scaling and refactoring to a NoSQL solution like

DynamoDB.

- Consider implementing read replicas for read-intensive workloads, but remember they are not for high availability.
- Be cautious when scaling storage as it only scales up.
- Don't hesitate to vertically scale database compute for improved performance, but be mindful of cost implications.
- Enable Multi-AZ deployments for production environments but note that standby instances cannot be directly used for reads and writes.
- Prefer Aurora over other relational database options when possible, as AWS favors its own services.

By mastering these concepts and tips, you'll be better prepared to design scalable and efficient relational database architectures in AWS.

Scaling Options for DynamoDB:

1. Provisioned Capacity: Suitable for generally predictable workloads. Requires setting upper and lower scaling boundaries based on past or predicted usage. Provisioned capacity is often the most cost-effective model.
2. On-Demand Scaling: Ideal for sporadic workloads with unpredictable usage patterns. Easy to enable by selecting on-demand capacity mode, but may be less cost-effective compared to provisioned capacity for accurately predicted workloads.

Capacity Units in DynamoDB:

- Read Capacity Units (RCUs): Measure reads per second for items up to 4 KB in size. Each RCU supports one strongly consistent read per second or two eventually consistent reads per second.
- Write Capacity Units (WCUs): Measure writes per second for items up to 1 KB in size. Each WCU supports one write per second.

Console Demo - Creating a DynamoDB Table:

- Demonstrated how to create a DynamoDB table using provisioned capacity and switch to on-demand capacity mode.
- Emphasized the importance of understanding access patterns and considering cost when choosing between capacity modes.

Exam Tips:

- Consider the predictability of access patterns when choosing between provisioned and on-demand capacity modes.
- Design DynamoDB tables to avoid hotkeys for better performance.
- You can switch between provisioned and on-demand capacity modes, but only twice per 24 hours per table.
- Keep cost optimization in mind and be aware of the trade-offs between different capacity modes.

By understanding these concepts and tips, you'll be better equipped to design scalable and cost-effective DynamoDB databases in AWS.

Key Questions for High Availability and Scaling:

1. Is it highly available? Always prioritize solutions that include high availability, unless the question explicitly states otherwise.
2. Horizontal or Vertical Scaling? Favor horizontal scaling for flexibility and better availability, but consider vertical scaling when necessary, such as for increasing network throughput.
3. Is the solution cost-effective? Keep cost in mind when selecting solutions, even if not explicitly asked in the question.
4. Could we switch the database? Consider switching between relational and non-relational databases if it better suits the workload, even if migration would be complex in

reality.
Auto Scaling Tips:

- Auto scaling is only for EC2 instances and cannot be used for other services like RDS or load balancers.
- Aim to get ahead of the workload by scaling out instances before peak times, rather than reacting after.
- Bake everything into the Amazon Machine Image (AMI) to reduce provisioning time and improve scalability.
- Spread resources across multiple availability zones for better fault tolerance and high availability.
- Utilize steady state groups for legacy instances that can only have one instance running at a time.
- Use load balancers in conjunction with auto scaling groups to distribute traffic and replace unhealthy instances.

Scaling Databases:

- For RDS, scaling options include vertical scaling, additional storage, and read replicas for horizontal scaling.
- Consider read replicas for read-heavy workloads in RDS.
- For DynamoDB, choose between provisioned and on-demand capacity based on the access pattern of the workload.

By keeping these tips in mind, you'll be better prepared to tackle questions related to high availability and scaling on the AWS certification exams.
That wraps up this lesson on exam tips for high availability and scaling.

From <https://chat.openai.com/c/d6fe90a1-a00b-4374-9bfc-54c5891880f3>