AWS Lab 4: Working with EBS

Emma Matsuda

Advanced Cisco CCNP

**Purpose**

This lab focuses on key topics from Amazon Elastic Block Store (Amazon EBS), which is a base mechanism for Amazon EC2 instances – introduced in Lab 3. The purpose of this lab is to create an Amazon EBS volume, attach and mount the volume to an EC2 instance, create a snapshot of the volume, create a new volume from the snapshot, and to attach and mount the new volume to the EC2 instance.

**Background Info**

This lab focuses on the basic aspects of Amazon Elastic Block Store (Amazon EBS) Amazon EBS, in general, enables you to create individual storage volumes and attach them to an Amazon EC2 instance. It offers block-level storage and can be backed up automatically to Amazon S3 through snapshots. The volumes are automatically replicated within Availability zone. Amazon EBS use is recommended when data must be quickly accessible, require long term persistence, and when data requires an encryption solution. Some main features include the following:

Snapshots – point-in-time snapshots + recreate a new volume at any time.

Encryption - encrypted Amazon EBS volumes + no additional cost.

Elasticity – increase capacity + changeable to different types.

Amazon EBS has different volume types, all for different purposes and such. The main difference between these types is the use cases and its characteristics. They all have the same maximum volume size. Below is a chart of Amazon EBS volume types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | General Purpose | Provisioned IOPS | Throughput-optimized | Cold |
| Maximum  Volume Size | 16 TiB | 16 TiB | 16 TiB | 16 TiB |
| Maximum IPOS/Volume | 16,000 | 64,000 | 500 | 250 |
| Maximum Throughput/  Volume | 250 MiB/s | 1,000 MiB/s | 1. B/s | 250 MiB/s |
| Use Cases | - recommended  for most workloads  - System boot volumes  - Virtual desktops  - low-latency  interactive apps  - development and  test environments | - critical business  apps that require sustained IOPS  performance, or  more than 16,000  IOPS or 250 MiB/s  of throughput per  volume  - large database workloads | - streaming  workloads that  require consistent,  fast throughput at  a low price  - big data  - data warehouses  - log processing  - it cannot be a  boot volume | - throughput-oriented storage for large volumes of data that is infrequently accessed  - scenarios here the lowest storage cost is important  - it cannot be a boot volume |

**Lab Summary**

1. Access the AWS Management Console
2. Create and attach an Amazon EBS volume to an EBS volume to a new Amazon EC2 instance
3. Attach the new volume to the EC2 instance
4. Connect the Amazon EC2 instance
5. Add a new volume to a Linux instance as an ext3 file system
6. Create a snapshot of the EBS volume
7. Restore the Amazon EBS snapshot taken from earlier

**Configurations** – step by step screenshots

|  |  |
| --- | --- |
| Click on “Start Lab”  Wait until the message: “Lab status: ready” |  |
| Click on “AWS” |  |
| Click on the AWS Management Console |  |
| On the AWS Management Console, click on EC2 |  |
| Under Instances, click on the Lab instance, then make sure that the Availability zone is set to us-east-1a. |  |
| Under Volumes, click the “create volume” |  |
| Then configure the volume type to *General Purpose SSD (gp2)*, then make the Size 1 GiB, then make sure that the availability zone is set to us-east-1a. |  |
| Then, add a tag with the Key: Name and Value: My volume – then create the volume |  |
| Select “My volume” and under the actions option, choose “attach volume” |  |
| Choose the preferred instance, and then click on attach volume to confirm the attachment |  |
| On the actual lab page, click on the Details option to show the credentials window. |  |
| Download the labuser.ppk file |  |
| Open the PuTTY application |  |
| Under connections, make sure that the seconds between keepalives is set to 30 |  |
| Under session, put in the public IPv4 address from the instance |  |
| Under Connection – SSH – Credentials, select the private key file (the labuser.ppk from before) |  |
| To make sure that the device trusts and connects to the host, choose the Accept option |  |
| Login as “ec2-user" |  |
| Put in the commands shown in the pictures to:  - view the storage available on the instance  - create an ext3 file system on the new volume  - create a directory for mounting the new storage volume |  |
| - mount the new volume  - view the config file to see the setting on the last line |  |
| - view the available storage (make sure that there is an additional line)  - create a file and add some text to it  - verify that the text is on the volume |  |
| Under Volumes, select the My Volume  Under the actions option, chose Create snapshot |  |
| Add tags with the Key: Name and the Value: My Snapshot, then create the snapshot |  |
| Enter the following commands to:  - delete the file that was created on the volume  - verify that the file has been deleted |  |
| Once selecting My Snapshot, under the actions option, click the “create volume from snapshot” |  |
| Make sure that the Availability zone is us-east-1a, same as before, and make the tag with the Key: Name and Value: Restored Volume, and create volume |  |
| Under the volumes option on the left, select restored volume, then under the actions option, choose “attach volume” |  |
| Choose the (Lab) instance on the Instance field, then make sure that the device is set to what is shown on the right, then click on Attach volume. |  |
| Enter the following commands on the right to:  - create a directory for mounting the new storage volume  - mount the new volume  - verify that the volume that got mounted has the file that was created earlier. |  |

**Conclusion**

Through this lab, I was able to do the following tasks: create an Amazon EBS volume, attach the volume to an EC2 instance, create a file system on the volume, add a file to volume, create a snapshot of the volume, create a new volume from the snapshot, attach and mount the new volume to the EC2 instance, and then verify that the file created was on the newly create volume. I was able to go through the basic steps of amazon EBs, learning how things can be applied into real world situations. I also learned that out-of-website applications like PuTTY can be used in these situations as well.

AWS Lab 5: Build a Database Server

Emma Matsuda

Advanced Cisco CCNP

**Purpose**

This lab focuses the reinforcement of the concepts of leveraging and AWS-managed database instance for solving relational database needs. The goal for this lab is to be able to: launch an Amazon RDS DB instance with high availability, configure the DB instance to permit connections from your web server, and open a web application and interact with the database.

**Background Info**

This lab focuses on databases, more specifically amazon Relational Database Service (Amazon RDS). It is a managed service that sets up and operates a relational database in the cloud. Along with being accessible via console, AWS CLI, and API calls, it is also scalable for both compute and storage. It also has features like automated redundancy and backups that are available and has the following supported DB engines: Amazon Aurora, Oracle, Microsoft SQL Server, PostgreSQL, MySQL and MariaDB.

Database in Amazon RDS/Amazon Aurora

On-premises database

Database in Amazon EC2

Common use cases involve things like web and mobile applications (key features: high throughput, massive storage capability, high availability, ecommerce applications (key features: low-cost database, data security, fully managed solution), and mobile/online games (key features: rapidly grow capacity, automatic scaling, database monitoring). The usage of Amazon RDS is recommended when the application requires the following: Complex transactions or complex queries, a medium to high query or write rate – up to 30,000 IOPS, no more than a single worker node or shared, and high durability. The use of Amazon RDS is NOT recommended when the application required the following: massive read/rite rates, sharing due to high database size or throughput demands, GET or PUT requests and queries that NoSQL database can handle, relational database management system (RDBMS) customizations.

Although relational databases are truly a useful tool when it comes to managing, there are some overall challenges to the services provided. Just to name a few: server maintenance and energy footprint, software installation and patched, database backups and high availability, limits on scalability, data security, operating system (OS) installation and patches. With Amazon RDS, AWS manages everything listed on the list above, with application optimization being the only service management responsibility given to the user.

**Lab Summary**

1. Create a security group to allow the web server to access RDS DB instance
2. Create a DB subnet group used to tell RDS which subnets can be used for the database
3. Configure and launch a Multi-AZ Amazon RDS for MySQL database instance
4. Open a web application running on web server
5. Configure the web server application to use the database

**Configurations** – step by step screenshots

|  |  |
| --- | --- |
| Click on “Start Lab”  Wait until the message: “Lab status: ready” |  |
| Click on “AWS” |  |
| Click on the AWS Management Console |  |
| Under Services, choose “VPC” |  |
| Under security groups, choose “create security group” |  |
| Configure the security group name, description, and VPC |  |
| Choose the type: MySQL/Aurora and the prefix list of Web Security Group |  |
| Under services, choose the RDS option |  |
| Under security group, create DB subnet group |  |
| Configure the name, description, and VPC as shown on the left |  |
| Under Ass subnets, choose the Availability Zones and Subnets on the left, make sure that it displays the same information on the table below, then create the subnet group |  |
| Under Databases, choose the “create database” option |  |
| Select MySQL as the engine option |  |
| Under templates, choose the Dev/Test, and the Multi-AX DB instance for the Availability and durability |  |
| Configure the following settings:  DB instance identifier  Master username  Master password  Confirm password  (password is set to lab-password) |  |
| Then, configure the DB instance class to Burstable classes |  |
| Configure the storage type and allocated storage |  |
| Make sure that the VPC is the Lab VPC |  |
| Unselect “default” and instead select “DB Security Group” for the Existing VPA security groups list |  |
| Name the database name to “lab” and uncheck the following:  Enable automatic backups  Enable encryption |  |
| Uncheck enable enhanced monitoring |  |
| When the database is created, select the lab-db database and copy the endpoint to use later |  |
| On the lab page itself, select the details to show the WebServer IP address |  |
| Copy the WebServer IP address, then enter it onto a new web browser to visit the page |  |
| Under RDS, configure the following:  Endpoint – the endpoint that was copied earlier  Database  Username  Password  Then, submit |  |
| An address book should pop up, where you are able to edit information as you want. |  |

**Conclusion**

Through this lab, I was able to do the following tasks: launch an Amazon RDS DB instance with high availability, Configure the DB instance to permit connections from the web server, and open a web application and interact with the database. Through this experience of learning these skills, I found that one simple click could lead to a whole different outcome. Originally, when first completing this lab, I made the mistake of not using the newest way to create a database, which led me to not have certain options that I had to configure. On the second try, I didn’t have to go through the same problem because I figures out the pop-up that offered the newest process.

AWS Lab 6 : Scale and Load Balance Your Architecture

Emma Matsuda

Advanced Cisco CCNP

**Purpose**

This lab goes through the process of using the Elastic Load Balancing (ELB) an Auto Scaling services for the purpose of load balancing and automized scaling of the infrastructure. The main goal of this lab is: creating an amazon Machine Image (AMI) from a running instance, creating a load balancer, creating a launch template and an Auto Scaling group, automatically scaling new instances, and creating Amazon CloudWatch alarms and monitoring the performance of the infrastructure.

**Background Info**

The two main topics that this lab focuses on it Elastic Load Balancing and Auto Scaling. Below are key points and information about these two topics.

Elastic Load Balancing (ELB) – ELB is an AWS service that distributes incoming application or network traffic across multiple targets (such as Amazon EC2 instances, containers, IP addresses, and Lambda functions) in a single Availability Zone or across multiple Availability Zones. ELB scaled your load balancer as traffic to your application changes over time, and it can automatically scale to most workloads.

Elastic Load Balancing is available in 3 types:

|  |  |  |
| --- | --- | --- |
| Application Load Balancer | Network Load Balancer | Classic Load Balancer  (Previous Generation) |
| * load balancing of HTTP and HTTPS traffic * routes traffic to targets based on content of request * provides advances request routing targeted at the delivery of modern application architecture, including microservices and containers * provides advances request routing targeted at the deliver of modern application architecture, including microservices and containers * operated at the application layer (OSI model layer 7) | * load balancing of TCP, UDP, and TLS traffic where extreme performance is required * routes traffic to targets based on Ip protocol data * can handle millions of requests per second while maintaining ultra-low latencies * Is optimized to handle sudden and volatile traffic patterns * Operated at the transport layer (OSI model layer 4) | * Load balancing of HTTP, HTTP, TCP, and SSL traffic * Load balancing across multiple EC2 instances * Operated at both the application and transport layers |

Some use cases for these include: highly available and fault-tolerant applications, containerized applications, elasticity and scalability, virtual private cloud, hybrid environments, and to invoke Lambda functions over HTTP(S)

**Lab Summary**

1. Create an AMI from the existing server – saves the contend of the boot disk so new instances can be launched identically
2. Create a load balancer – can balance traffic across EC2 instances and Availability Zones
3. Create launch template for Auto Scaling group – used by Auto Scaling group to launch EC2 instances
4. Verify that Load Balancing is working correctly
5. Increase the load to cause Auto Scaling to add additional instances
6. Terminate web server 1 – no longer needed

**Configurations** – step by step screenshots

|  |  |
| --- | --- |
| Click on “Start Lab”  Wait until the message: “Lab status: ready” |  |
| Click on “AWS” |  |
| Click on the AWS Management Console |  |
| Under Services, choose “EC2” |  |
| Under Instances, check that the instance is running by making sure that the status displays 2/2 checks passed, then select the server. |  |
| Under Actions, choose Image and temples > Create image |  |
| Name the image as “WebServerAMI” and the description as “Lab AMI for Web Server”, then create the image. |  |
| Under Target Groups, create target group. |  |
| Choose the target type as Instances, then enter the name LabGroup |  |
| Select LabVPC from the VPC drop-down menu, then select Next. |  |
| Under Load Balancers, create the load balancer. |  |
| Select the Application Load Balancer for the load balancer type. |  |
| Name the balancer LabELB |  |
| Choose the Lab VPC, and then choose the two subnets as the Publics |  |
| Choose the Web Security Group for the security group, unselect the default one. Then, make sure that the Default action for the HTTP:80 listener is LabGroup. |  |
| Create the Load Balancer and View the load balancer. |  |
| Under Launch Templates, create launch template. |  |
| Name is LabConfig, and then select the auto scaling guidance. |  |
| Under MyAMIs, choose the “WebServerAMI” |  |
| Select the t2.micro instance type, and then choose the vockey key pair name. Then, choose the “Select existing security group” option. |  |
| Enable the Detailed CloudWatch monitoring under Advanced details |  |
| Create launch template. |  |
| Click on the labconfig in the green banner that shows up. |  |
| Under actions, create auto scaling group. |  |
| Name it as “Lab Auto Scaling Group, then select the LabConfig launch template. |  |
| Select the Lan VPN, and the two subnets from earlier, just this time, choose the private subnets. |  |
| Select the “attach to an existing load balancer” option for load balancing, then choose the LabGroup HTTP balancer target group. |  |
| Enable group metrics collection within Cloudwatch, then go to the next step. |  |
| Use the group size 2 capacity, then make sure that the minimum is 2 and maximum is 6. |  |
| Choose the target scaling polity, then name is LabScalingPolicy, then use the metric type Average CPU utilization |  |
| Skip the next step |  |
| The Key should be Name, then the value as Lab Instance, then choose next. |  |
| Under instances, then see that there are 2 new Lab Instances. |  |
| Under the target groups, select the LabGroup, and then see that there are two registered targets that are names Lab Instance. |  |
| Make sure that both of them are Healthy. |  |
| Then. Under Load Balancers, select the LabELB and copy the DNS name |  |
| Paste the name into the URL of another browser tab, ad see that the application appears. |  |
| Back on AWS, under services, choose CloudWatch |  |
| Under All alarms. Make sure that the AlarmHigh is with the OK status. |  |
| Back on the DNS URL webpage, click on Load Test |  |
| Then, wait until the AlarmHigh is shown as “In alarm” |  |
| On AWS under Services, choose EC2 |  |
| Under instances, check that there are now 4 Lab Instances. |  |
| Click on Web Server 1, then under Instance state, select Terminate instance. |  |
| Click on Terminate to confirm the termination of the instance. |  |

**Conclusion**

Through this lab, I was able to do the following tasks: Create an Amazon Machine Image (AMI) from a running instance, create a load balancer, create a launch template and an Auto Scaling group, automatically scale new instances, and create Amazon CloudWatch alarms and monitor performance of your infrastructure. Through this experience, I learned that patience is important for the applications to work well. Multiple times throughout this lab, there ere times when I had to wait for a status of something to turn into another, and I noticed that if I ever tried to skip over the step and continue to the net steps without paying attention to them, they just simply do not work. This made me realize that the small details are important for going through these processes.