**Amazon SES**

* Cloud based e-mail service
* Used for sending transactional and mass emails, marketing content, other high-quality content to your customers/audience
* Offers wide list of possible integrations
* Uses SMTP
* Can integrate SES with Amazon SDK or other email clients

**Why SES?**

* High deliverability rate
* Focuses on white-listing by supporting authentication mechanisms such as DKIM

(**DomainKeys Identified Mail** is an email authentication method designed to detect forged sender addresses in email, a technique often used in phishing and email spam. DKIM allows the receiver to check that an email claimed to have come from a specific domain was indeed authorized by the owner of that domain)

* SPF (Sender Policy Framework) allows users to choose which IP Addresses are allowed to send mails from a particular domain
* As a sender, you authenticate your email using DKIM and SPF, and you publish a DMARC record in your DNS.

(DMARC (Domain-based Message Authentication, Reporting and Conformance) is **an email authentication protocol. It is designed to give email domain owners the ability to protect their domain from unauthorized use, commonly known as email spoofing). It uses SPF.**

* SES offers content personalization, i.e. mails tailored to a specific user based on various factors such as DOB, username, location, behavior, etc.

E.g.: Sending a newsletter mail with a subscriber’s first name in the greeting

* SES can even be used for email receiving, not just sending
* SES can integrate with other services offered by AWS such as cloudwatch, IAM, EC2, Beanstalk, Lambda, Route 53, S3, SNS, Workmail.
* (Check this) If your app is hosted using AWS EC2 or Beanstalk, first 62k mails of the month are free.
* Pricing: https://aws.amazon.com/ses/pricing/

**Pros and Cons of SES (Based on User reviews)**

Pros:

* High deliverability and reliability
* No required additional maintenance
* Pricing
* Comprehensive set of tools for email receiving and management

Cons:

* Complicated initial configuration
* Initial limitations before you get approved and verify your sending domains
* Not a marketing platform (Lack of template building)
* No E-Mail list storage

More details on how SES works: <https://docs.aws.amazon.com/ses/latest/dg/send-email-concepts-process.html>

Hands-on (Sending mail using SMTP Interface):

* Docs: <https://docs.aws.amazon.com/ses/latest/dg/send-email-smtp.html>
* Create an AWS account
* Go to AWS Management Console, search for SES and click
* Create SMTP credentials (username and password)
  + Docs: <https://docs.aws.amazon.com/ses/latest/dg/smtp-credentials.html>
  + An IAM user can create Amazon SES SMTP credentials, but the IAM user's policy must give them permission to use IAM itself, because Amazon SES SMTP credentials are created by using IAM
  + Your IAM policy must allow you to perform the following IAM actions: iam:ListUsers, iam:CreateUser, iam:CreateAccessKey, and iam:PutUserPolicy
  + **To create your SMTP credentials**

1. Sign in to the AWS Management Console and open the Amazon SES console at <https://console.aws.amazon.com/ses/>.
2. Choose **Account dashboard** in the left navigation pane.
3. In the **Simple Mail Transfer Protocol (SMTP) settings** container, choose **Create SMTP Credentials** in the lower-left corner - the IAM console will open.
4. For **Create User for SMTP**, type a name for your SMTP user in the **IAM User Name** field. Alternatively, you can use the default value that is provided in this field. When you finish, choose **Create** in the bottom-right corner.
5. Expand **Show User SMTP Security Credentials** - your SMTP credentials are shown on the screen.
6. Download these credentials by choosing **Download Credentials** or copy them and store them in a safe place, because you can't view or save your credentials after you close this dialog box.
7. Choose **Close Window**.

* Add and verify Email address
* Go to SES console, click on create identity
* Go to the inbox on the ID that we wish to verify, and click on the link in the mail that you would have received with the subject Amazon Web Services – Email Address Verification Request
* Note: Email addresses are case sensitive. Also, keep the regions in mind when using SES, as the email addresses that we verify are linked to the region
* Add a couple more email addresses
* Domain verification is a standard procedure, for which you need to add a TXT record to your domain’s DNS
* We need to start sending mails in Sandbox mode, i.e. we can send emails to verified addresses and domains or to Amazon SES mailbox simulator only

(Limit: 200 mails/24 hours, 1 mail/sec)

* To send mails to people without verifying their email addresses, you need to submit a request via AWS Management Console, describing how you will use SES
* To send a test mail follow: <https://docs.aws.amazon.com/ses/latest/dg/send-an-email-from-console.html#send-email-simulator>

**DynamoDB**

**What is DynamoDB?**

* Managed NoSQL Database optimized for performance at scale
  + It is like a key-value look-up store
  + If we had our own hardware, we would have to maintain its hardware, security patches, etc. but that is not the case with DynamoDB, where everything is managed by AWS
  + You can scale out horizontally by adding more nodes to a cluster and separating data outside your nodes
* Ideal for applications with known access patterns

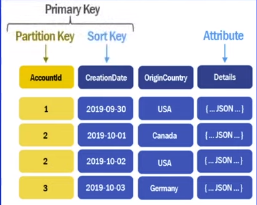
- Important to build a schema prior to getting started with DynamoDB

* Access DynamoDB using an API or an ORM, and enable authorization using IAM (which makes it easier to revoke permissions in case someone leaves the your organization)
* 2 cost models:
  + On-demand: For known access patterns, less cost effective
  + Provisioned mode: For when we know the capacity of our cluster

**Core concepts - Tables, Items, Attributes, Indexes**

* Tables are a collection of Items
* Items (typically known as a row in a database) are collections of Attributes, or key/value pairs

E.g.: Creation date is the key name and “2019-09-30” is the value for Item with account ID “1”



* Q: What if we have to find out all the accounts with the origin country of “USA”?

A: One way is to use a scan operation, which will search every row and check for the OriginCountry. This is a bad way to go about it as we will be iterating over the entire table, and we will incur Read costs for every row, even though “USA” is present in only 2 rows.

To avoid that, we can use **Global Secondary Index** (GSI)



This means that instead of querying on the Partition or Sort key, we can query on any other column.

**Hands-on tutorial for DynamoDB:**

1. Search for DynamoDB on AWS console, and go to the service page
2. To create a new table, click on the “Create Table” button
3. Choose a table name (Note: A table name is region-specific), eg: “Orders”
4. Name the partition key, say “OrderId” with type “String”
5. Name the Sort Key (Note: You cannot add a sort key after the table is created). To decide whether we need a Sort key or not, we should check whether the partition key will be duplicated or not. If it can be duplicated, then we need a partition key and a sort key, otherwise we just need a partition key.

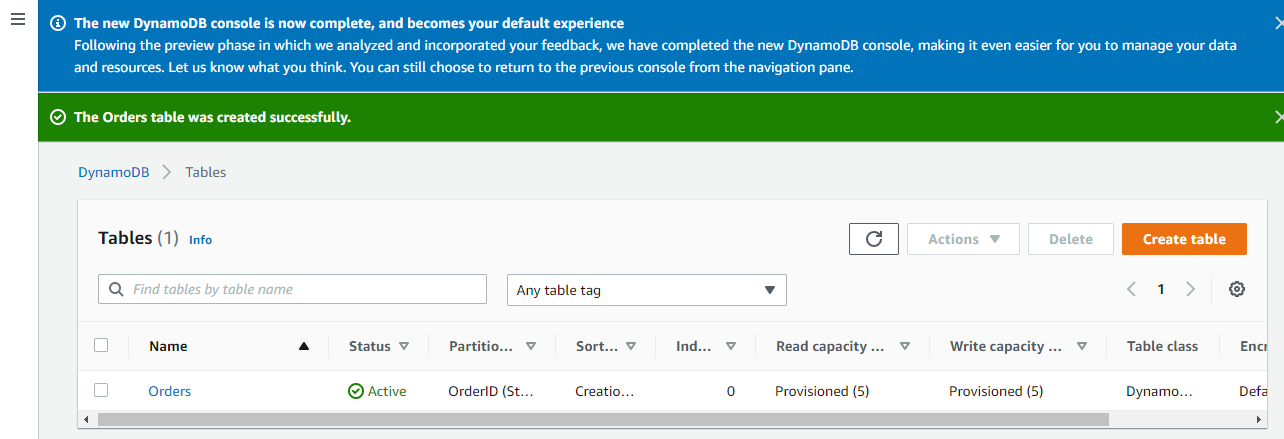
In this example, we are naming the Sort key “Creation Date”

1. Next, we will select Customize settings
2. Next, in the “**Read/write capacity settings**” section, we choose the type of capacity that we want: 1. On-demand 2. Provisioned
3. Provisioned capacity is chosen when traffic patterns are predictable, as it costs less
4. Set auto-scaling for Read Capacity and Write Capacity to Off. if you want to have it adjust automatically, you can set the required Capacity to On and set the minimum capacity and maximum capacity according to your requirements
5. In the secondary index section, we see two options; “Create local index” and “Create Global Index”. The former can only be created at the time of table creation, which is used to query on range-like attributes, (such as dates)

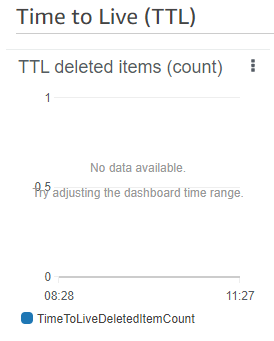
The latter can be created at any time but there is additional cost implication to it as it duplicates and re-formats the table in a way that makes querying faster, therefore it takes up more space

1. Create Table

It will look something like this:

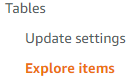


1. On clicking the table name, we will be able to see a dashboard for our table, in which, if you click on the **Monitor** tab, you will see a lot of metrics which are important when it comes to tracking usage.
2. One such graph is that of a Time to Live, which look like

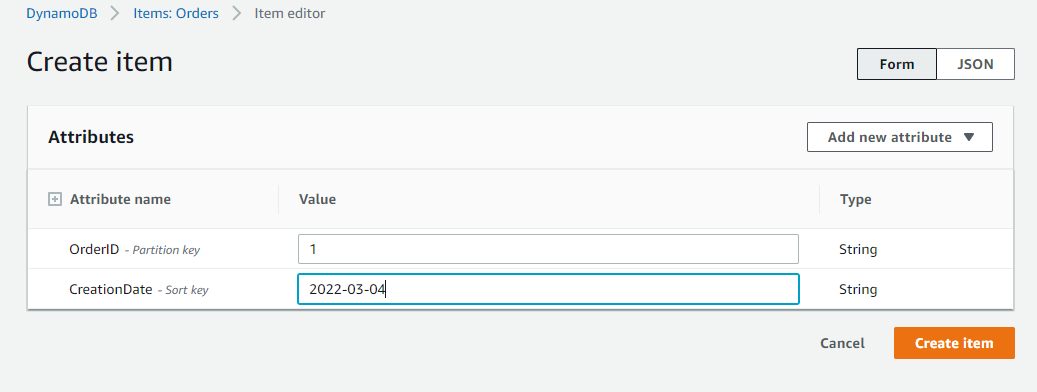


TTL is used for records on a table so that you can delete them at the specified timestamp

1. We can also create alarms, which notify us when a certain criteria is met in the metrics, e.g. a certain number of read units, write units, billing amount because of the table, number of errors, etc.
2. Another tab is **Global Tables**. As we saw earlier, a table is region specific, so if someone from a different region wishes to access our table, the process will be very slow due to the distance involved. To solve this, we are given the option to replicate our table, which will sync data across regions
3. Explore **Backups** tab
4. Moving on the create some data for our table, click Explore Items

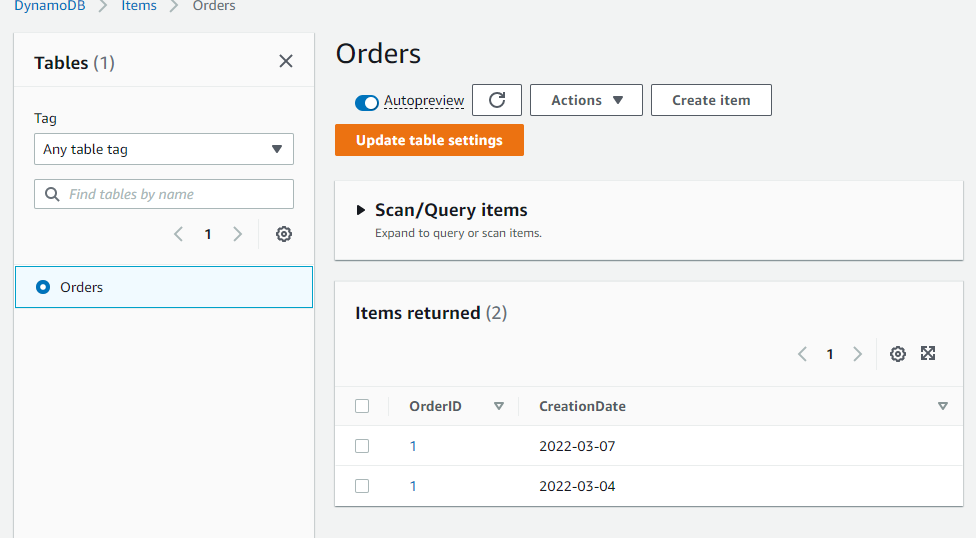


1. Click on the table in which you wish to enter items. In this case, it will be “Orders”. Then click on “Create Item”
2. You can choose to enter data in JSON format, or via a form.

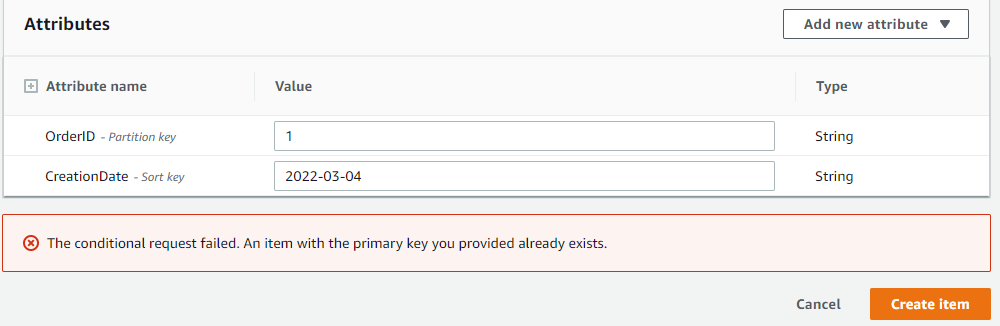


Click on “Create Item”

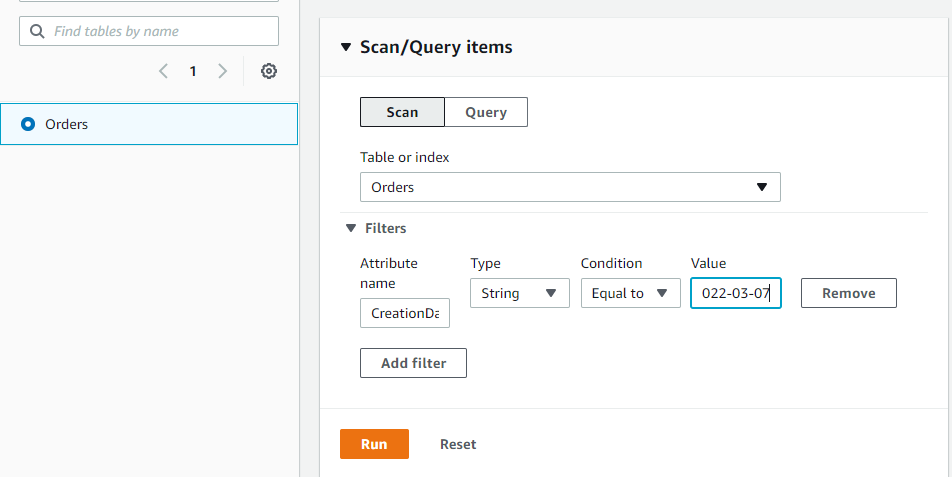
Create another such row. The table should look like this:

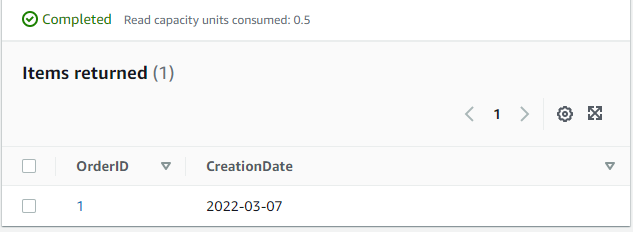


Notice that we have two items with the same OrderID, which was the Partition Key, but if we try to create an item where both the OrderId and CreationDate are the same as another item, it will fail to create the item.



1. We can add filters to view data, just like we do so programmatically





1. Do the same using Query tab

It will ask for the partition key, which will help look up data in constant time, as it knows exactly where the required data is.

1. After making a few such calls, have a look at the Monitor dashboard and try to see if there are any changes in any of the graphs
2. Make sure that you delete the table after you are done experimenting.

**Programming hands-on:**

Node.js + DynamoDB

Docs: <https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.NodeJs.html>

* Code: <https://github.com/arindamkeswani/aws/tree/main/nodeXdynamodb>

(Follow the commits for a step-by-step guide)

Note: Instead of using the complete database, as required in the tutorial, use only 1 or 2 JSON objects for testing to avoid incurring unnecessary costs