## Week 1 Notes and Resources

## KEY TOPICS

### Relational and NoSQL databases

For decades, the predominant data model that was used for application development was the relational data model used by relational databases such as Oracle, DB2, SQL Server, MySQL, and PostgreSQL. It wasn’t until the mid to late 2000s that other data models began to gain significant adoption and usage. To differentiate and categorize these new classes of databases and data models, the term “NoSQL” was coined. Often the term “NoSQL” is used interchangeably with “nonrelational.”

Neither model is necessarily better, just more appropriate for particular application and workload requirements. More information about Relational and NoSQL databases can be found [here](https://aws.amazon.com/nosql/).

### Amazon DynamoDB

[Amazon DynamoDB](https://aws.amazon.com/dynamodb/) is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multiregion, multimaster database with built-in security, backup and restore, and in-memory caching for internet-scale applications. DynamoDB can handle more than 10 trillion requests per day and can support peaks of more than 20 million requests per second.

DynamoDB lets you offload the administrative burdens of operating and scaling a distributed database, so that you don't have to worry about hardware provisioning, setup and configuration, replication, software patching, or cluster scaling.

### Accessing and Setting up DynamoDB via the AWS Console

You can access the AWS Management Console for DynamoDB here: <https://console.aws.amazon.com/dynamodb/home>.

You can use the console to do the following in DynamoDB:

* Monitor recent alerts, total capacity, service health, and the latest DynamoDB news on the DynamoDB dashboard.
* Create, update, and delete tables. The capacity calculator provides estimates of how many capacity units to request based on the usage information you provide.
* Manage streams.
* View, add, update, and delete items that are stored in tables. Manage Time To Live (TTL) to define when items in a table expire so that they can be automatically deleted from the database.
* Query and scan a table.
* Set up and view alarms to monitor your table's capacity usage. View your table's top monitoring metrics on real-time graphs from CloudWatch.
* Modify a table's provisioned capacity.
* Create and delete global secondary indexes.
* Create triggers to connect DynamoDB streams to AWS Lambda functions.
* Apply tags to your resources to help organize and identify them.
* Purchase reserved capacity.

Additional information on using the DynamoDB with the AWS Management Console can be found [here](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/ConsoleDynamoDB.html).

### Using the CLI to access DynamoDB

You can use the [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.CLI.html) to control multiple AWS services from the command line and automate them through scripts. You can use the AWS CLI for ad hoc operations, such as creating a table. You can also use it to embed DynamoDB operations within utility scripts.

The command line format consists of a DynamoDB operation name, followed by the parameters for that operation. The AWS CLI supports a shorthand syntax for the parameter values, as well as JSON. See the [Amazon DynamoDB CLI documentation](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Tools.CLI.html" \l "Tools.CLI.UsingWithDDB) for details.

## ADDITIONAL SERVICES/CONCEPTS

### Developing in the Cloud with AWS Cloud9

[AWS Cloud9](https://docs.aws.amazon.com/cloud9/latest/user-guide/welcome.html) is a cloud-based integrated development environment (IDE) that lets you write, run, and debug your code with just a browser. It includes a code editor, debugger, and terminal. You can run this development environment on a managed Amazon Elastic Compute Cloud (Amazon EC2) instance that automatically sleeps when you are not using it.

Follow the exercise directions carefully when setting up your AWS Cloud9 instance.

### CAP Theorem

[The CAP theorem](https://en.wikipedia.org/wiki/CAP_theorem) states posits that in a distributed system such as Amazon DynamoDB, it is impossible to have more than two out of the following three:

* [Consistency](https://en.wikipedia.org/wiki/Consistency_(database_systems)
* [Availability](https://en.wikipedia.org/wiki/Availability)
* [Partition Tolerance](https://en.wikipedia.org/wiki/Network_partition)

Amazon DynamoDB provides tool to help manage [read consistency](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadConsistency.html).

## WHAT YOU ACCOMPLISHED THIS WEEK

* You created your AWS Cloud9 instance.
* You created a DynamoDB table and populated it with game data
* You began to build out your application

## Week 2 Notes and Resources

## KEY TOPICS

### Backup and Recovery, the SDK, and Monitoring and Performance

This week, we focused how to create and restore backups of your DynamoDB tables as well as dove a bit deeper in the DynamoDB SDK. We also looked into using tools such as Cloudwatch to monitor and optimize your DynamoDB performance.

### Backup and Recovery

Backups

When you create an [on-demand backup](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/backuprestore_HowItWorks.html), a time marker of the request is cataloged. The backup is created asynchronously by applying all changes until the time of the request to the last full table snapshot. Backup requests are processed instantaneously and become available for restore within minutes.

Point-in-Time Recovery for DynamoDB

You can enable [point-in-time recovery](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/PointInTimeRecovery.html) as well as create on-demand backups for your Amazon DynamoDB tables. For more information regarding on-demand backups, see [On-Demand Backup and Restore for DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/BackupRestore.html)

Point-in-time recovery helps protect your Amazon DynamoDB tables from accidental write or delete operations. With point in time recovery, you don't have to worry about creating, maintaining, or scheduling on-demand backups. For example, suppose that a test script writes accidentally to a production DynamoDB table. With point-in-time recovery, you can restore that table to any point in time during the last 35 days. DynamoDB maintains incremental backups of your table.

### Amazon DynamoDB SDK

Every [AWS SDK](https://aws.amazon.com/tools) provides one or more programmatic interfaces for working with DynamoDB. These interfaces range from simple low-level DynamoDB wrappers to object-oriented persistence layers. The available interfaces vary depending on the AWS SDK and programming language that you use. See [Getting Started with DynamoDB SDK](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GettingStarted.html) for details and examples.

### Monitoring and Performance

Understanding Read and Write Capacity

Amazon DynamoDB has two [read/write capacity modes](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html" \l "HowItWorks.requests) for processing reads and writes on your tables:

* On-demand
* Provisioned (default, free-tier eligible)

The read/write capacity mode controls how you are charged for read and write throughput and how you manage capacity. You can set the read/write capacity mode when creating a table or you can change it later.

On-Demand Mode

Amazon DynamoDB on-demand is a flexible billing option capable of serving thousands of requests per second without capacity planning. DynamoDB on-demand offers pay-per-request pricing for read and write requests so that you pay only for what you use.

When you choose on-demand mode, DynamoDB instantly accommodates your workloads as they ramp up or down to any previously reached traffic level. If a workload’s traffic level hits a new peak, DynamoDB adapts rapidly to accommodate the workload. Tables that use on-demand mode deliver the same single-digit millisecond latency, service-level agreement (SLA) commitment, and security that DynamoDB already offers. You can choose on-demand for both new and existing tables and you can continue using the existing DynamoDB APIs without changing code.

Provisioned Mode

If you choose provisioned mode, you specify the number of reads and writes per second that you require for your application. You can use auto scaling to adjust your table’s provisioned capacity automatically in response to traffic changes. This helps you govern your DynamoDB use to stay at or below a defined request rate in order to obtain cost predictability.

Provisioned mode is a good option if any of the following are true:

* You have predictable application traffic.
* You run applications whose traffic is consistent or ramps gradually.
* You can forecast capacity requirements to control costs.

Adaptive Capacity

It's not always possible to distribute read and write activity evenly all the time. When data access is imbalanced, a "hot" partition can receive such a higher volume of read and write traffic compared to other partitions. In extreme cases, throttling can occur if a single partition receives more than 3,000 RCUs or 1,000 WCUs.

To better accommodate uneven access patterns, [DynamoDB adaptive capacity](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html" \l "bp-partition-key-partitions-adaptive) enables your application to continue reading and writing to hot partitions without being throttled, provided that traffic does not exceed your table’s total provisioned capacity or the partition maximum capacity. Adaptive capacity works by automatically and instantly increasing throughput capacity for partitions that receive more traffic.

Adaptive capacity is enabled automatically for every DynamoDB table, so you don't need to explicitly enable or disable it.

As of 23-May-2019, [DynamoDB Adaptive Capacity is now instant](https://aws.amazon.com/about-aws/whats-new/2019/05/amazon-dynamodb-adaptive-capacity-is-now-instant/?sc_channel=em&sc_campaign=GLOBAL_LA_NL_Engaged-Newsletter_20190528&sc_medium=em_132692&sc_content=PA_nl_la&sc_geo=mult&sc_country=global&sc_outcome=pa&trk=em_132692_VelocityWhatsNew&mkt_tok=eyJpIjoiWkRjd05qaGtZMlV5WWprMCIsInQiOiJTcDdDWFlNbWtcL2hYMVVscWx0VnU3RENlOFpnOHlPZkw4cytCR2hmdlRBbVU4b2l6NjJnaVN6TGsyS2tRc0pzRnE0c2ZIaXV2VkVlclNSUHYzYmpOTERwWDVqVjdINnBOejh6RnRrcUVPcVlBZ3N5WlwvQXJJRTBxbWZWdHJRZzZPIn0%3D) .

Monitoring Tools

AWS provides tools that you can use to [monitor DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/monitoring-automated-manual.html). You can configure some of these tools to do the monitoring for you; some require manual intervention. We recommend that you automate monitoring tasks as much as possible.

[AWS X-Ray](https://docs.aws.amazon.com/xray/latest/devguide/aws-xray.html) is a service that collects data about requests that your application serves, and provides tools you can use to view, filter, and gain insights into that data to identify issues and opportunities for optimization. For any traced request to your application, you can see detailed information not only about the request and response, but also about calls that your application makes to downstream AWS resources, microservices, databases and HTTP web APIs.

## WHAT YOU ACCOMPLISHED THIS WEEK

* You used the AWS SDK to continue to build out your application.
* You used Cloudwatch Metrics and Alarms as well as Cloudwatch Logs.
* You used AWS x-Ray to detect and dive into performance issues.

## Week 3 Notes and Resources

## KEY TOPICS

### Security, State, and Advanced Concepts

This week we focused on security, including encryption and access, expiration of items,

### Security

Encryption and Compliance

All user data stored in Amazon DynamoDB is fully [encrypted at rest](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/EncryptionAtRest.html). DynamoDB encryption at rest provides enhanced security by encrypting all your data at rest using encryption keys stored in [AWS Key Management Service (AWS KMS)](https://aws.amazon.com/kms/). This functionality helps reduce the operational burden and complexity involved in protecting sensitive data. With encryption at rest, you can build security-sensitive applications that meet strict encryption compliance and regulatory requirements.

VPC Endpoints

For security reasons, many AWS customers run their applications within an Amazon Virtual Private Cloud environment (Amazon VPC). With Amazon VPC, you can launch Amazon EC2 instances into a virtual private cloud, which is logically isolated from other networks—including the public Internet. With an Amazon VPC, you have control over its IP address range, subnets, routing tables, network gateways, and security settings. More information about using Amazon VPC endpoints for DynamoDB can be found [here](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb.html).

Authentication and Access Control for Amazon DynamoDB

Access to Amazon DynamoDB requires credentials. Those credentials must have permissions to access AWS resources, such as an Amazon DynamoDB table or an Amazon Elastic Compute Cloud (Amazon EC2) instance. Details on DynamoDB authentication and access control can be found in the [developers guide](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/authentication-and-access-control.html).

### State, and advanced concepts

Time to Live (TTL)

[Time To Live (TTL) for DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/TTL.html) allows you to define when items in a table expire so that they can be automatically deleted from the database.

TTL is provided at no extra cost as a way to reduce storage usage and reduce the cost of storing irrelevant data without using provisioned throughput. With TTL enabled on a table, you can set a timestamp for deletion on a per-item basis, allowing you to limit storage usage to only those records that are relevant.

TTL is useful if you have continuously accumulating data that loses relevance after a specific time period. For example: session data, event logs, usage patterns, and other temporary data. If you have sensitive data that must be retained only for a certain amount of time according to contractual or regulatory obligations, TTL helps you ensure that it is removed promptly and as scheduled.

Global tables

[Amazon DynamoDB global tables](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html) provide a fully managed solution for deploying a multi-region, multi-master database, without having to build and maintain your own replication solution. When you create a global table, you specify the AWS regions where you want the table to be available. DynamoDB performs all of the necessary tasks to create identical tables in these regions, and propagate ongoing data changes to all of them.

DynamoDB Streams

Many applications can benefit from the ability to capture changes to items stored in a DynamoDB table, at the point in time when such changes occur.

[A DynamoDB stream](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Streams.html) is an ordered flow of information about changes to items in an Amazon DynamoDB table. When you enable a stream on a table, DynamoDB captures information about every modification to data items in the table.

Whenever an application creates, updates, or deletes items in the table, DynamoDB Streams writes a stream record with the primary key attribute(s) of the items that were modified. A stream record contains information about a data modification to a single item in a DynamoDB table. You can configure the stream so that the stream records capture additional information, such as the "before" and "after" images of modified items.

Using DynamoDB Streams with Amazon Elasticsearch

You can use AWS Lambda to [send data to your Amazon Elastic Search domain from Amazon DynamoDB](https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/es-aws-integrations.html" \l "es-aws-integrations-dynamodb-es). New data that arrives in the database table triggers an event notification to Lambda, which then runs your custom code to perform the indexing. See below for more information about Amazon Elasticsearch. For a user example, see this [blog post](https://aws.amazon.com/blogs/startups/combining-dynamodb-and-amazon-elasticsearch-with-lambda/).

### Other Topics

Amazon Elasticsearch

[Amazon Elasticsearch Service (Amazon ES)](https://docs.aws.amazon.com/elasticsearch-service/latest/developerguide/what-is-amazon-elasticsearch-service.html) is a managed service that makes it easy to deploy, operate, and scale Elasticsearch clusters in the AWS Cloud. Elasticsearch is a popular open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. With Amazon ES, you get direct access to the Elasticsearch APIs; existing code and applications work seamlessly with the service.

## WHAT YOU ACCOMPLISHED THIS WEEK

* You continued to build out your application using a session table and enabling TTL
* You secured your application following the least privilege principle, using AWS [IAM](https://docs.aws.amazon.com/IAM/latest/UserGuide/introduction.html), [VPC](https://docs.aws.amazon.com/vpc/latest/userguide/what-is-amazon-vpc.html), and an [VPC Endpoint for DynamoDB](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb.html).

## Week 4 Notes and Resources

## KEY TOPICS

### Diving Deep and Best Practices

This week we look at concurrent access to tables using Optimistic Locking, Dive into Global and Local Secondary indexes, and then look at optimizing queries and schema design.

### Optimistic Locking

[Optimistic locking](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/DynamoDBMapper.OptimisticLocking.html) is a strategy to ensure that the client-side item that you are updating (or deleting) is the same as the item in DynamoDB. If you use this strategy, then your database writes are protected from being overwritten by the writes of others — and vice-versa.

With optimistic locking, each item has an attribute that acts as a version number. If you retrieve an item from a table, the application records the version number of that item. You can update the item, but only if the version number on the server side has not changed. If there is a version mismatch, it means that someone else has modified the item before you did; the update attempt fails, because you have a stale version of the item. If this happens, you simply try again by retrieving the item and then attempting to update it. Optimistic locking prevents you from accidentally overwriting changes that were made by others; it also prevents others from accidentally overwriting your changes.

### LSI and GSI

Amazon DynamoDB provides fast access to items in a table by specifying primary key values. However, many applications might benefit from having one or more [secondary (or alternate) keys](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/SecondaryIndexes.html) available, to allow efficient access to data with attributes other than the primary key. To address this, you can create one or more secondary indexes on a table, and issue Query or Scan requests against these indexes.

A secondary index is a data structure that contains a subset of attributes from a table, along with an alternate key to support Query operations. You can retrieve data from the index using a Query, in much the same way as you use Query with a table. A table can have multiple secondary indexes, which gives your applications access to many different query patterns.

Amazon DynamoDB supports [two types](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-indexes-general.html) of secondary indexes:

* [Global secondary index](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html) An index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions. A global secondary index has no size limitations and has its own provisioned throughput settings for read and write activity that are separate from those of the table.
* [Local secondary index](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/LSI.html) An index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value. As a result, the total size of indexed items for any one partition key value can't exceed 10 GB. Also, a local secondary index shares provisioned throughput settings for read and write activity with the table it is indexing.

### Writing Better Queries

In this section we discussed writing conditional expressions to improve query performance. You can use comparisons (such as "a >b"), functions (such as "attribute\_exists" ) and logical evaluations. Details can be found in the [Developer Guide](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.OperatorsAndFunctions.html" \l "Expressions.OperatorsAndFunctions.LogicalEvaluations).

Single Table

Modeling data in DynamoDB is different than when modeling data in a relational database. For details on data models see this section of the [Developer Guide](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-modeling-nosql-B.html).

Time Series

General design principles in DynamoDB recommend that you keep the number of tables you use to a minimum. For most applications, a single table is all you need. However, for [time-series data](https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-time-series.html), you can often best handle it by using one table per application per period.

### Other Topics

There are other database options available in AWS.

* [Amazon Aurora](https://aws.amazon.com/rds/aurora/) is a MySQL and PostgreSQL-compatible [relational database](https://aws.amazon.com/relational-database/) built for the cloud, that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases.
* [Amazon DocumentDB](https://aws.amazon.com/documentdb/) (with MongoDB compatibility) is a fast, scalable, highly available, and fully managed document database service that supports MongoDB workloads.
* [Amazon Neptune](https://aws.amazon.com/neptune/) is a fast, reliable, fully managed graph database service that makes it easy to build and run applications that work with highly connected datasets.
* [Amazon Timestream](https://aws.amazon.com/timestream/) is a fast, scalable, fully managed time series database service for IoT and operational applications that makes it easy to store and analyze trillions of events per day at 1/10th the cost of relational databases.

## WHAT YOU ACCOMPLISHED THIS WEEK

* You continued to build out your application using conditional updates and transactions.
* You completed the course!

Please do complete the course survey! We hope you enjoyed the class.

## LEARN MORE ABOUT AWS SERVICES

A variety of online learning resources are available. Here are a few recommendations to help you build on the knowledge you gained in this course:

* For unlimited access to foundational AWS training, visit our digital library at [aws.amazon.com/training](https://www.aws.training/LearningLibrary?filters=digital%3A1&search=&tab=digital_courses). We currently offer over 400 free online digital courses.
* AWS Training and Certification offers several courses that cover DynamoDB. You can access the courses at [aws.amazon.com/training](https://www.aws.training/LearningLibrary?&search=dynamodb&tab=view_all).
* Would you like to get AWS certified? Visit the AWS Certification page at [aws.amazon.com/certification](http://aws.amazon.com/certification) to get more information about exams, requirements, and prep courses.

Finally, remember to check back on [edX.org](https://www.edx.org/school/aws) for additional courses about AWS services and cloud skills.

Thanks for joining us! We look forward to seeing you in future AWS classes.