**Amazon Redshift**

**Video transcript**

–

We just spent a lot of time discussing the kinds of workflow that require fast, reliable, current data. Databases that can handle 1,000s of transactions per second, storage that is highly available and massively durable. But sometimes, we have a business need that goes outside what is happening right now to what did happen. This data analysis is the realm of a whole different class of databases. Sure, you could use the one size fits all model of a single database for everything, but modern databases that are engineered for high speed, real time ingestion, and queries may not be the best fit.

Let me explain. In order to handle the velocity of real time read/write functionality, most relational databases tend to function fabulously at certain capacities. How much content it actually stores. The problem with historical analytics, data that answers questions like, "Show me how production has improved since we started", is the data collection never stops. In fact, with modern telemetry and the explosion of IoT, the volume of data will overwhelm even the beefiest traditional relational database.

It gets worse. Not just the volume, but the variety of data can be a problem. You want to run business intelligence or BI projects against data coming from different data stores like your inventory, your financial, and your retail sales systems? A single query against multiple databases sounds nice, but traditional databases don't handle them easily.

Once data becomes too complex to handle with traditional relational databases, you've entered the world of data warehouses. Data warehouses are engineered specifically for this kind of big data, where you are looking at historical analytics as opposed to operational analysis.

Now, let's be clear. Historical may be as soon as: show me last hour's sales numbers across all the stores. The key thing is, the data is now set. We're not selling any more from the last hour because that is now in the past. Compare that question to, "How many bags of coffee do we still have in our inventory right now?" Which could be changing as we speak. As long as your business question is looking backwards at all, then a data warehouse is the right solution for that line of business intelligence.

Now there are many data warehouse solutions out on the market. If you already have a favorite one, running it on AWS is just a matter of getting the data transferred. But beyond that, there may still be a lot of undifferentiated heavy lifting that goes into keeping a data warehouse tuned, resilient, and continuously scaling. Wouldn't it be nice if your data warehouse team could focus on the data instead of the unavoidable care and feeding of the engine?

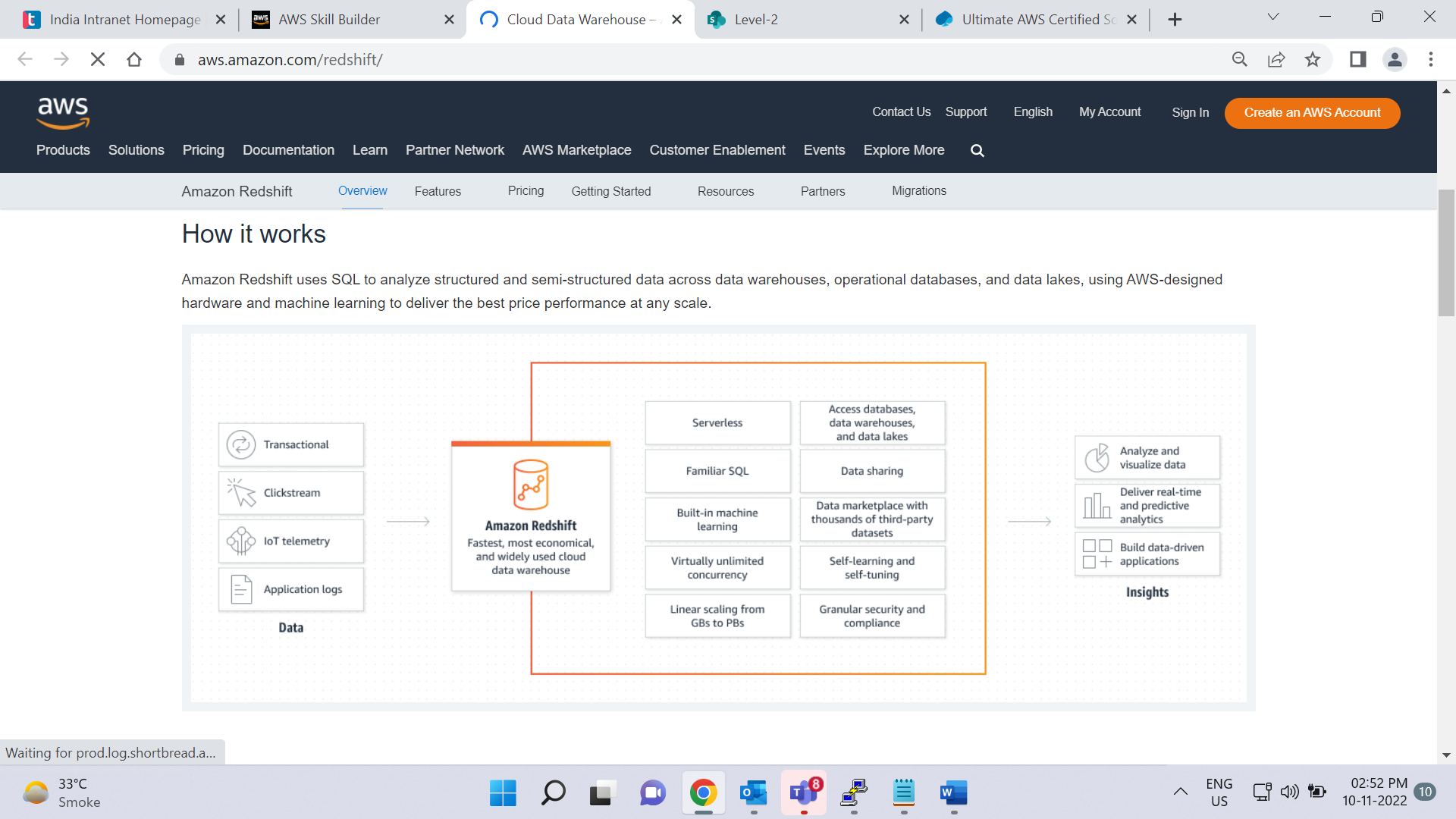
Introducing Amazon Redshift. This is data warehousing as a service. It's massively scalable. Redshift nodes in multiple petabyte sizes is very common. In fact, in cooperation with Amazon Redshift Spectrum, you can directly run a single SQL query against exabytes of unstructured data running in data lakes.

But it's more than just being able to handle massively larger data sets. Redshift uses a variety of innovations that allow you to achieve up to 10 times higher performance than traditional databases, when it comes to these kinds of business intelligence workloads.

I'm not gonna go into details of how the magic works. We have whole classes that you or your data teams can take that explain how it is built, and why it can return such improved results. The key for you is to understand that when you need big data BI solutions, Redshift allows you to get started with a single API call. Less time waiting for results, more time getting answers.

**Amazon Redshift**

[**Amazon Redshift**](https://aws.amazon.com/redshift) is a data warehousing service that you can use for big data analytics. It offers the ability to collect data from many sources and helps you to understand relationships and trends across your data.



# AWS Database Migration Service

Play Video

**Video transcript**

–

We've been talking about databases and the various database options on AWS. But what happens if you have a database that's on-premises or in the cloud already? Does that mean you have to start from scratch or does AWS have a magical way to help you migrate your existing database?

Thankfully, AWS offers a service called Amazon Database Magic. I mean, Amazon Database Migration Service, or DMS, to help customers do just that. DMS helps customers migrate existing databases onto AWS in a secure and easy fashion. You essentially migrate data between a source and a target database. The best part is that the source database remains fully operational during the migration, minimizing downtime to applications that rely on that database. Better yet is that the source and target databases don't have to be of the same type.

But let's start with databases that are of the same type. These migrations are known as homogenous and can be from MySQL to Amazon RDS for MySQL, Microsoft SQL Server to Amazon RDS for SQL Server or even Oracle to Amazon RDS for Oracle. The process is fairly straightforward since schema structures, data types, and database code is compatible between source and target.

As I mentioned, the source database can be located on-premises, running on Amazon EC2 Instances, or it can be an Amazon RDS database. The target itself can be a database in Amazon EC2 or Amazon RDS. In this case, you create a migration task with connections to the source and target databases. Then start the migration with a click of the button. AWS Database Migration Service takes care of the rest.

The second type of migration occurs when source and target databases are of different types. This is called heterogeneous migrations, and it's a two-step process. Since the schema structures, data types, and database code are different between source and target, we first need to convert them using the AWS Schema Conversion Tool. This will convert the source schema and code to match that of the target database. The next step is then to use DMS to migrate data from the source database to the target database.

But these are not the only use cases for DMS. Others include development and test database migrations, database consolidation, and even continuous database replication.

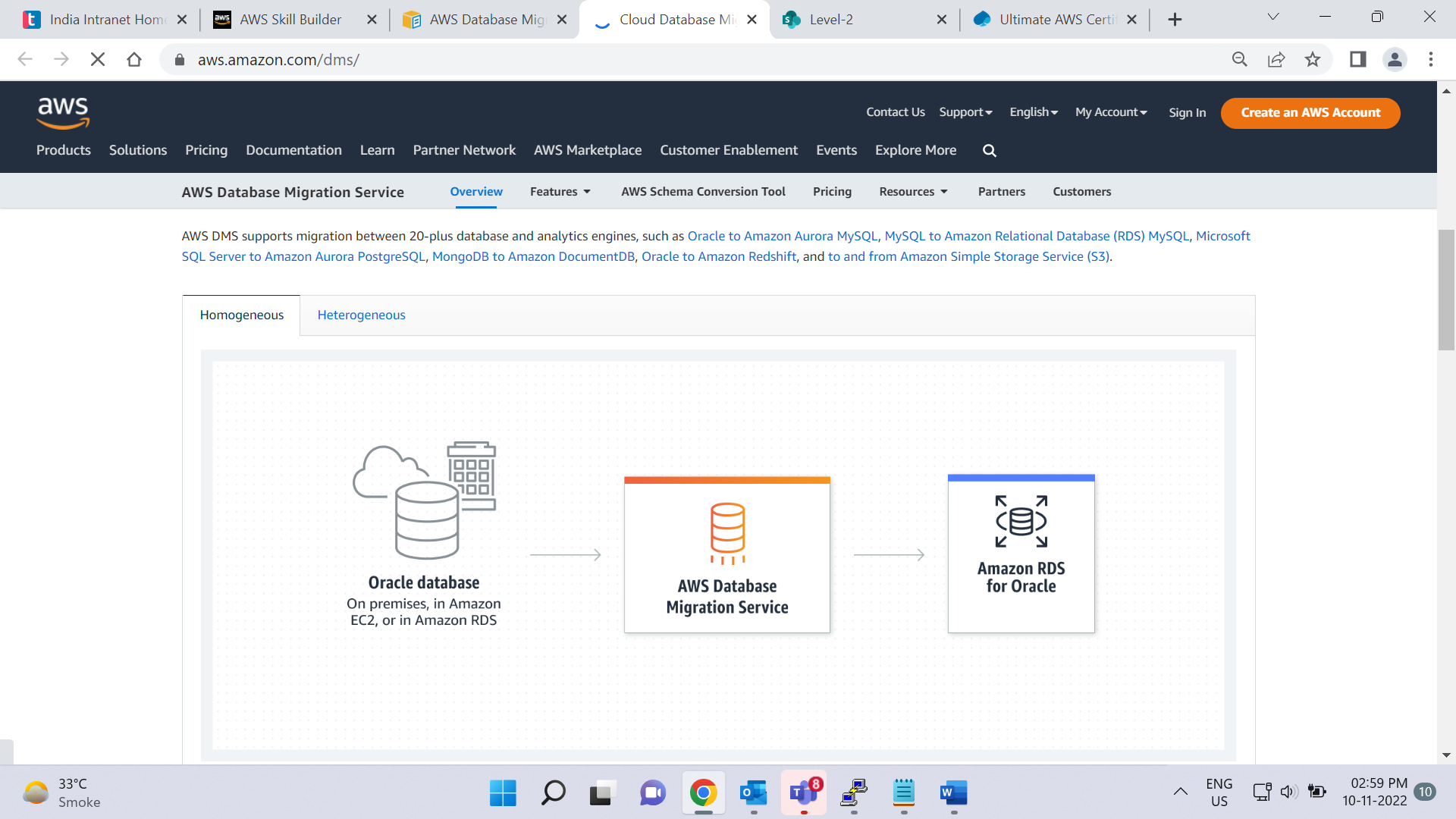
Development and test migration is when you want to develop this to test against production data, but without affecting production users. In this case, you use DMS to migrate a copy of your production database to your dev or test environments, either once-off or continuously.

Database consolidation is when you have several databases and want to consolidate them into one central database.

Finally, continuous replication is when you use DMS to perform continuous data replication. This could be for disaster recovery or because of geographic separation.

If you'd like to learn more about any of these, please check out our resources section. And there you have it, folks, DMS in a nutshell.

**AWS Database Migration Service (AWS DMS)**



[**AWS Database Migration Service (AWS DMS)**](https://aws.amazon.com/dms/) enables you to migrate relational databases, nonrelational databases, and other types of data stores.

With AWS DMS, you move data between a source database and a target database. [The source and target databases](https://aws.amazon.com/dms/resources) can be of the same type or different types. During the migration, your source database remains operational, reducing downtime for any applications that rely on the database.

For example, suppose that you have a MySQL database that is stored on premises in an Amazon EC2 instance or in Amazon RDS. Consider the MySQL database to be your source database. Using AWS DMS, you could migrate your data to a target database, such as an Amazon Aurora database.

# (AWS Database Migration Service Documentation

AWS Database Migration Service (AWS DMS) is a web service you can use to migrate data from your database that is on-premises, on an Amazon Relational Database Service (Amazon RDS) DB instance, or in a database on an Amazon Elastic Compute Cloud (Amazon EC2) instance to a database on an AWS service. These services can include a database on Amazon RDS or a database on an Amazon EC2 instance. You can also migrate a database from an AWS service to an on-premises database. You can migrate between source and target endpoints that use the same database engine, such as from an Oracle database to an Oracle database. You can also migrate between source and target endpoints that use different database engines, such as from an Oracle database to a PostgreSQL database.

)

**Other use cases for AWS DMS**

Select each card to flip it.

Development and test database migrations

Enabling developers to test applications against production data without affecting production users

Database consolidation

Combining several databases into a single database

Continuous replication

Sending ongoing copies of your data to other target sources instead of doing a one-time migration

# Additional database services

**Video transcript**

–

Before we wrap up databases and storage, I wanna loop back to the topic that we started all this with. Choosing the right database, choosing the right storage platform to fit your business needs, rather than forcing your data to fit your database's requirements. No matter what a database vendor might try to tell you, there is no one-size-fits-all database for all purposes. We've covered quite a few database flavors already, but there are even more databases AWS offers for special business requirements, that we don't have time to cover. But its worth just knowing they are there in case you need them.

For example, we talked about DynamoDB, and that's great for key-value pair databases. But what if you need more than just small attributes? What if you need a full content management system? Introducing Amazon DocumentDB, Great for content management, catalogs, user profiles.

What if you add a social network that you wanted to track for those kind of social webs, who is connected to who, is very clunky to manage in a traditional relational database so Amazon Neptune: a graph database, engineered for social networking and recommendation engines, also great for fraud detection needs.

Or perhaps you have a supply chain, that you have to track with assurances that nothing is lost. Or you have banking or financial records that require 100% immutability, or some people will tell you, oh, that's what blockchain is all about. Well, perhaps, I mean now, If you do need a blockchain solution, wouldn't you know it? We offer Amazon Managed Blockchain. But that's probably not what you really need here. It solves part of the equation, but adds a huge decentralization component, that's not what financial regulators wanna see. What you really need is an immutable ledger, so Amazon QLDB, or Quantum Ledger Database. An immutable system of record where any entry can never be removed from the audits.

Databases by themselves are great but if there is a way to make them faster, wouldn't that be greater? But you know I wouldn't be saying that, if there weren't some accelerator options that can be used in a number of unique scenarios. Starting with adding caching layers on top of your databases that can help improve read times of common requests from milliseconds to microseconds Amazon ElastiCache can provide those caching layers without your team needing to worry about the heavy lifting of launching, uplift, and maintenance. And it comes in both Memcached and Redis flavors.

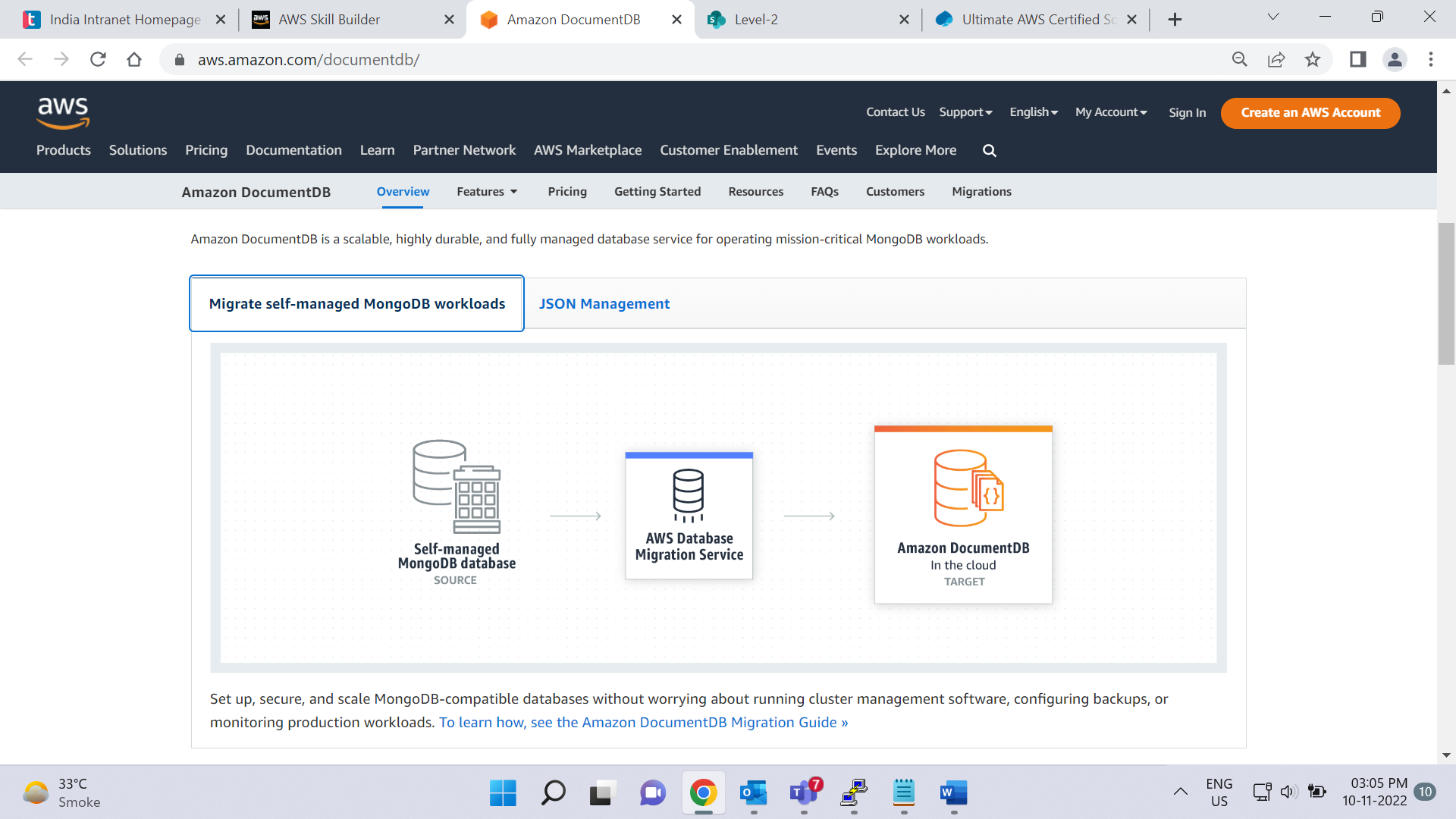
Or if you are using DynamoDB, try using DAX, the DynamoDB Accelerator, a native caching layer designed to dramatically improve read times for your nonrelational data.

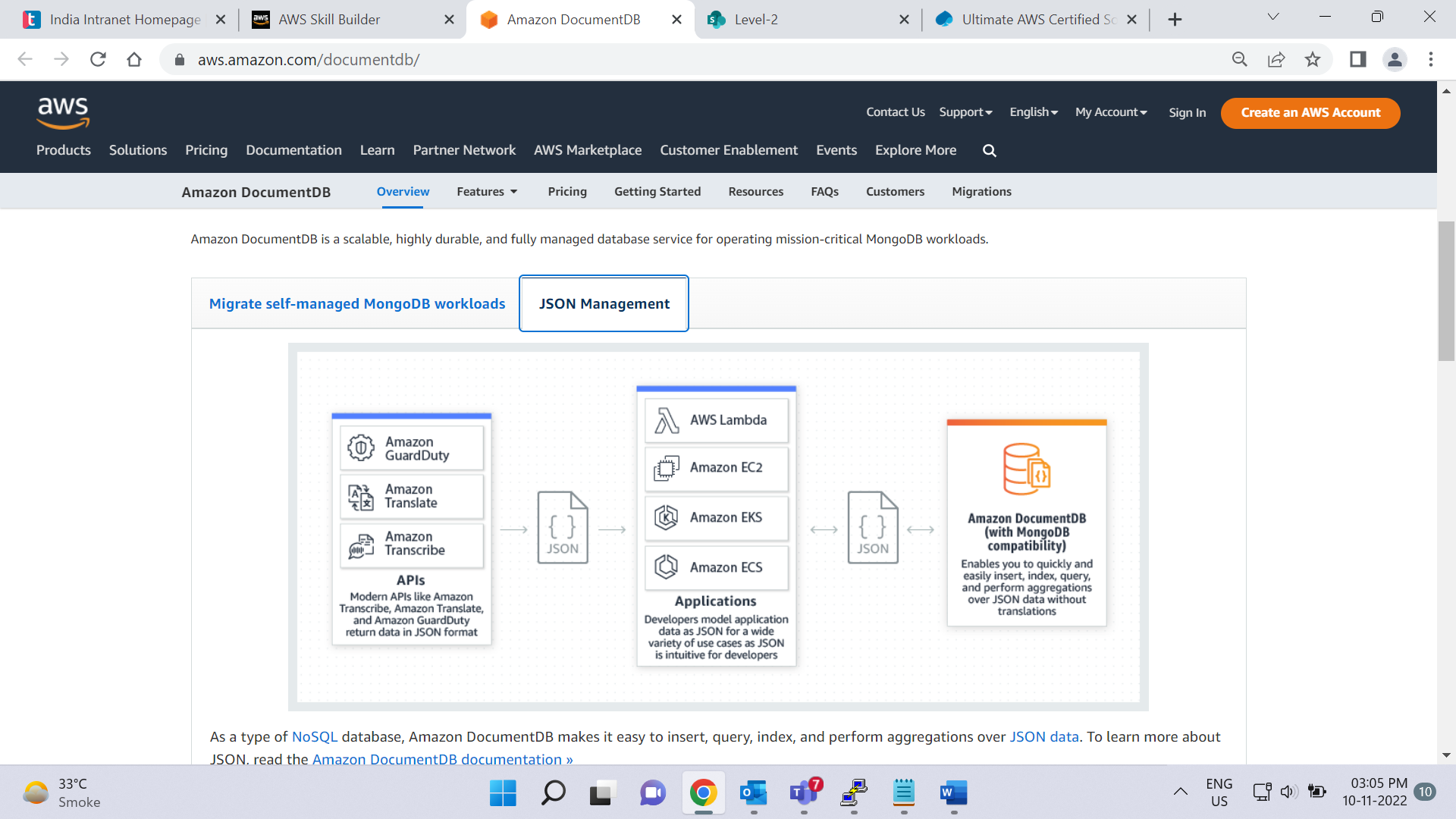
The key thing to understand is, AWS wants to make sure that you using the best tool for the job.

**Additional database services**

To learn more, select the **+** symbol next to each category.

**Amazon DocumentDB**

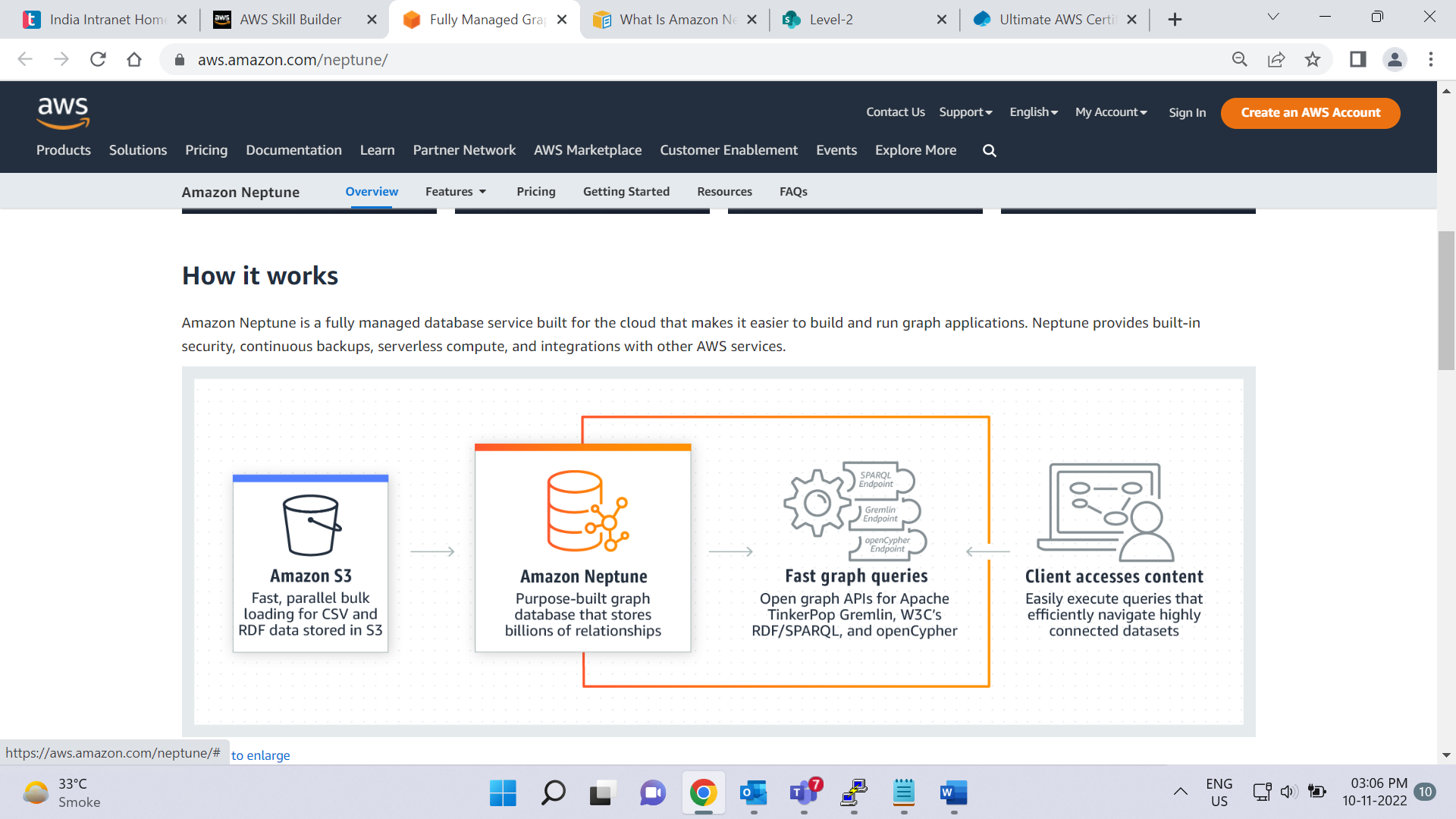




–

[**Amazon DocumentDB**](https://aws.amazon.com/documentdb) is a document database service that supports MongoDB workloads. (MongoDB is a document database program.)

**Amazon Neptune**

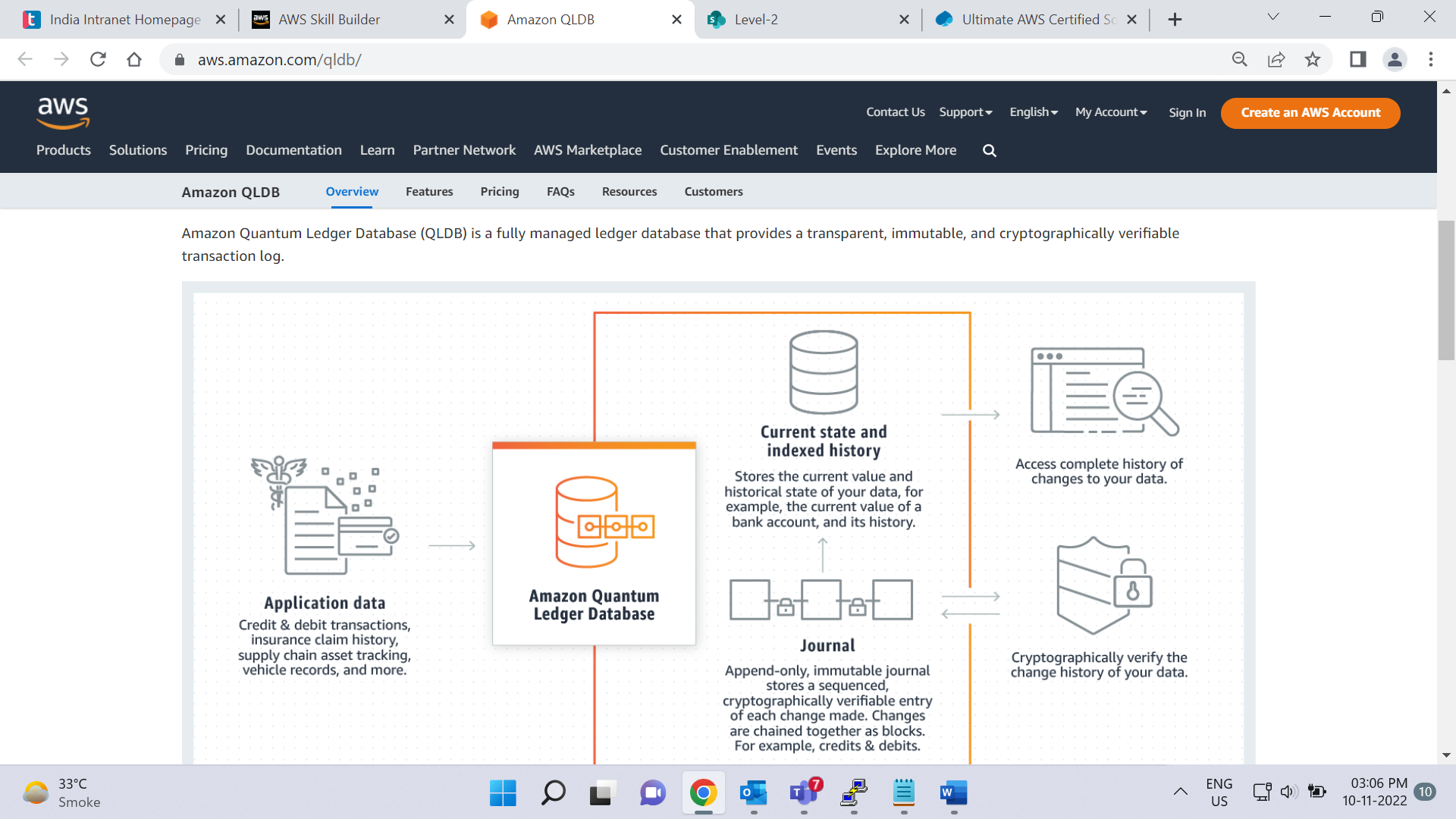


–

[**Amazon Neptune**](https://aws.amazon.com/neptune) is a graph database service.

You can use Amazon Neptune to build and run applications that work with highly connected datasets, such as recommendation engines, fraud detection, and knowledge graphs.

**Amazon Quantum Ledger Database (Amazon QLDB)**



–

[**Amazon Quantum Ledger Database (Amazon QLDB)**](https://aws.amazon.com/qldb) is a ledger database service.

You can use Amazon QLDB to review a complete history of all the changes that have been made to your application data.

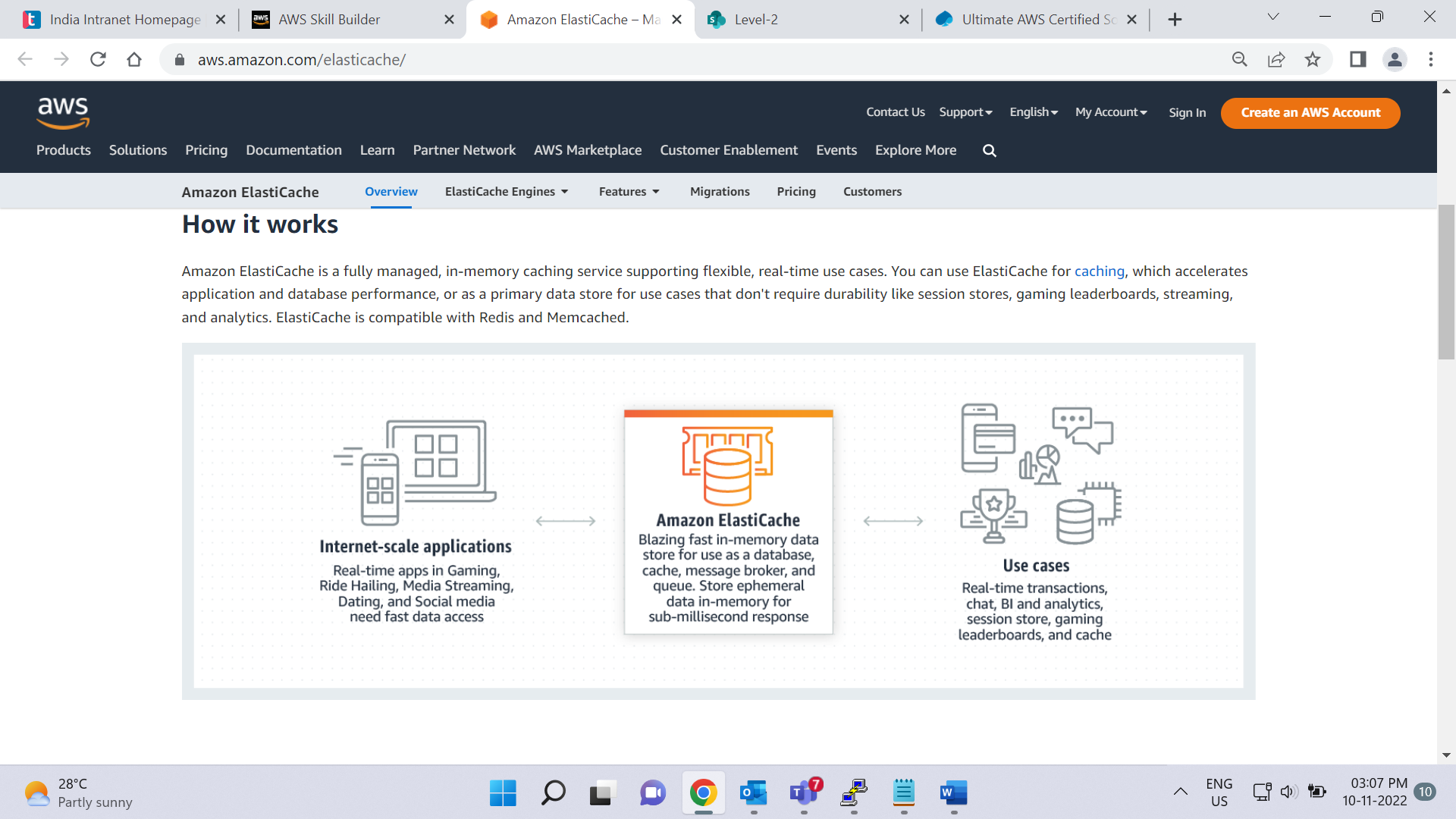
**Amazon Managed Blockchain**

–

[**Amazon Managed Blockchain**](https://aws.amazon.com/managed-blockchain) is a service that you can use to create and manage blockchain networks with open-source frameworks.

Blockchain is a distributed ledger system that lets multiple parties run transactions and share data without a central authority.

**Amazon ElastiCache**



–

[**Amazon ElastiCache**](https://aws.amazon.com/elasticache) is a service that adds caching layers on top of your databases to help improve the read times of common requests.

It supports two types of data stores: Redis and Memcached.

**Amazon DynamoDB Accelerator**

–

[**Amazon DynamoDB Accelerator (DAX)**](https://aws.amazon.com/dynamodb/dax/) is an in-memory cache for DynamoDB.

It helps improve response times from single-digit milliseconds to microseconds.

# Module 5 summary

In Module 5, you learned about the following concepts:

* Amazon EC2 instance store and Amazon EBS
* Amazon S3
* Amazon EFS
* Relational databases and Amazon RDS
* Nonrelational databases and DynamoDB
* Amazon Redshift
* AWS DMS
* Additional database services and accelerators

**Video transcript**

–

Another module completed. Awesome stuff. You learned about all the different types of AWS storage mechanisms. Let's recap them, shall we?

The first one we learned about is Elastic Block Store volumes, and you attach those to EC2 instances so you have local storage that is not ephemeral.

You learned about how S3 and how you can store objects in AWS with the click of a button or call of an API.

We even discussed the various relational database options available on AWS. Or for the workloads that just need a key-value pair, we have the non-relational offering called DynamoDB.

Next up was EFS for file storage use cases.

We then have Amazon Redshift for all our data warehouse needs.

And to aid in migration of existing databases, we have DMS or Database Migration Service.

We also touched upon the lesser known storage services, like DocumentDB, Neptune, QLDB, and Amazon Managed Blockchain.

Lastly, we talked about how caching solutions like ElastiCache and DynamoDB Accelerator can be used.

That's a lot of places to store different types of data, and hopefully you've learned the correct place to store each type.