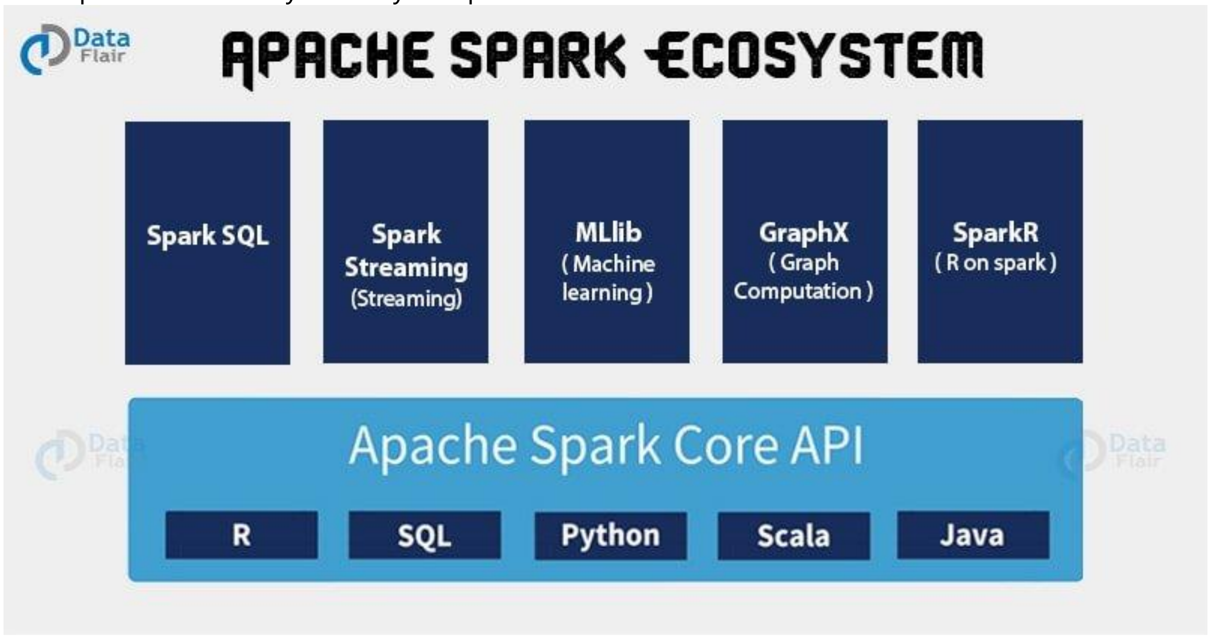
1. Explain what are various components of SPARK with block diagram? explain functionality of every components?



Spark Core is embedded with a special collection called RDD (resilient distributed dataset). RDD is among the abstractions of Spark. *Spark RDD* handles partitioning data across all the nodes in a cluster.

Transformation: It is a function that produces new RDD from the existing RDDs.

Action: In Transformation, RDDs are created from each other. But when want to work with the actual dataset, then, at that point we use Action.

Apache Spark SQL

The **Spark SQL** component is a distributed framework for *structured* data processing. Using Spark SQL, Spark gets more information about the structure of data and the computation. With this information, Spark can perform extra optimization.

Apache Spark Streaming

It is an add-on to core Spark API which allows scalable, high-throughput, fault-tolerant stream processing of live data streams. Spark can access data from sources like **Kafka**, **Flume**, **Kinesis** or **TCP socket.**

Apache Spark MLlib (Machine Learning Library)

MLlib in Spark is a scalable Machine learning library that discusses both high-quality algorithm and high speed.

Apache Spark GraphX:

GraphX in Spark is API for graphs and graph parallel execution. It is network graph analytics engine and data store. *Clustering, classification, traversal, searching, and pathfinding* is also possible in graphs.

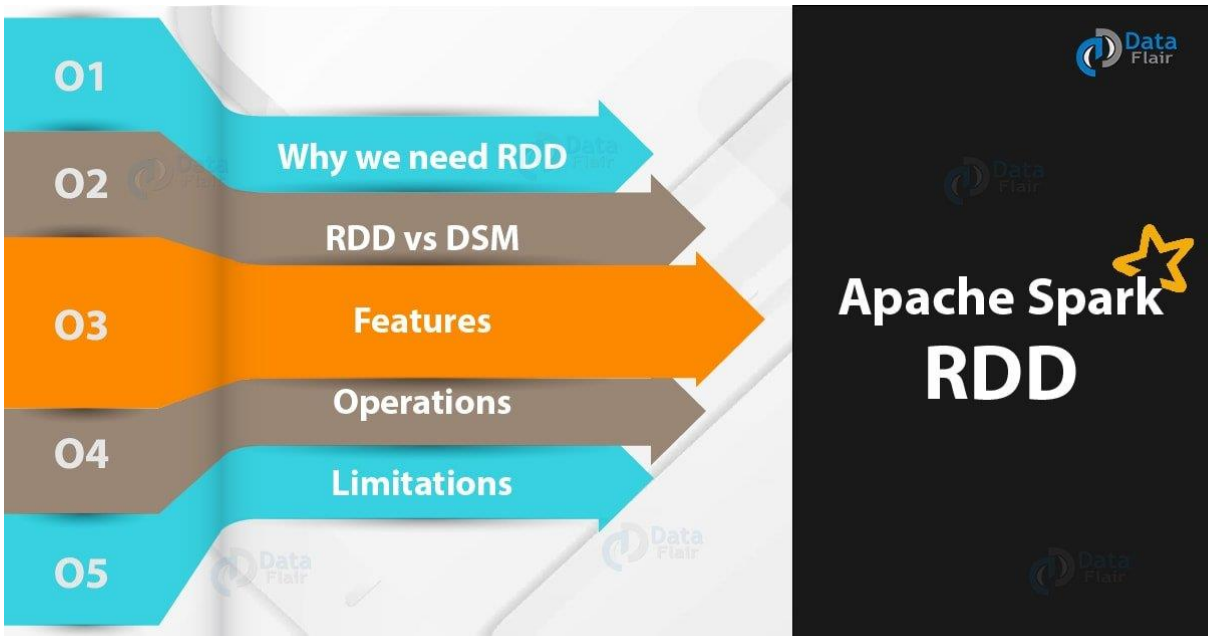
Apache SparkR

SparkR was Apache Spark 1.4 release. The key component of SparkR is SparkR DataFrame. DataFrames are a fundamental data structure for data processing in R. **The concept of DataFrames extends to other languages with libraries like *Pandas* etc.**

2. Explain Spark core in details & how RDD is related to Spark core - explain with Spark program ?

Resilient Distributed Datasets (RDDs)

Spark revolves around the concept of a *resilient distributed dataset* (RDD), which is a fault-tolerant collection of elements that can be operated on in parallel. There are two ways to create RDDs: *parallelizing* an existing collection in your driver program, or referencing a dataset in an external storage system, such as a shared filesystem, HDFS, HBase, or any data source offering a Hadoop InputFormat.



```
Val   x = sc.textile

Val   y =  x.map

Val   Z  =  y.filter
```

```
z.count()

x.count()
```

we need RDD in Spark:

- Iterative algorithms.
- Interactive data mining tools.
- **DSM** (Distributed Shared Memory) is a very general abstraction, but this generality makes it harder to implement in an efficient and fault tolerant manner on commodity clusters. Here the need of RDD comes into the picture.

3. Explain various Mlib algorithms Spark is supporting ?

Spark MLlib:

Spark MLlib is used to perform machine learning in Apache Spark. MLlib consists of popular algorithms and utilities. MLlib in Spark is a scalable Machine learning library that discusses both high-quality algorithm and high speed. The machine learning algorithms like regression, classification, clustering, pattern mining, and collaborative filtering. Lower level machine learning primitives like generic gradient descent optimization algorithm are also present in MLlib.

Spark.ml is the primary Machine Learning API for Spark. The library Spark.ml offers a higher-level API built on top of DataFrames for constructing ML pipelines.

Spark MLlib tools are given below:

1. ML Algorithms
2. Featurization
3. Pipelines
4. Persistence
5. Utilities

4. Explain benifits Spark SQL & how relational data will be inserted into SPARK ?

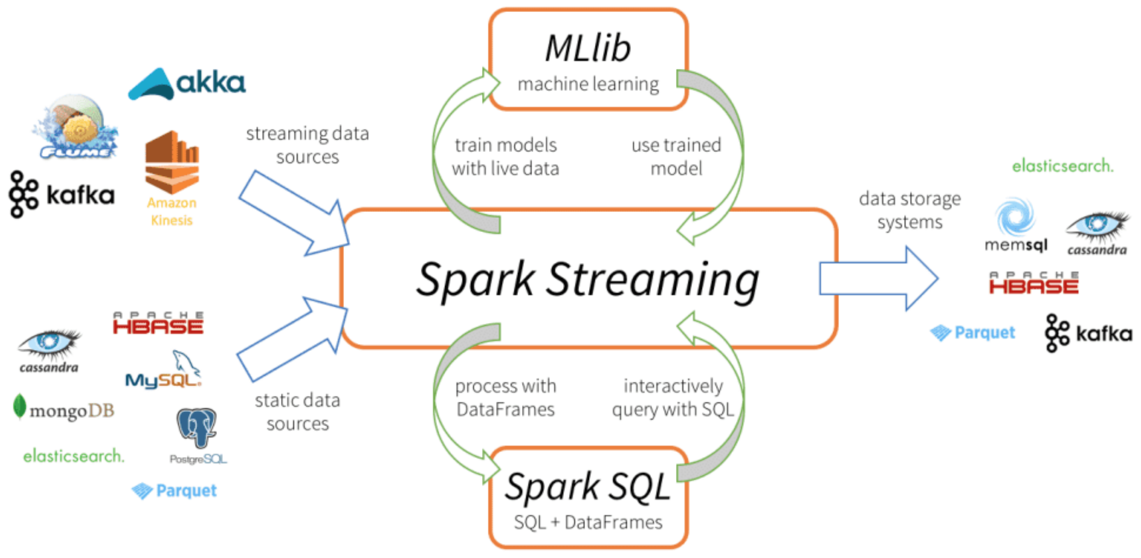
i. Simple to write. We can say it is very simple to write parallelized code, for simple problems.

- ii. Framework handles errors.

- iii. Algorithms.
- iv. Libraries.
- v. Good Local Tools.
- vi. Learning Curve.
- vii. Ease of use.
- i. Difficult to express

Spark SQL is a Spark module for structured data processing. Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. Internally, Spark SQL uses this extra information to perform extra optimizations. There are several ways to interact with Spark SQL including SQL and the Dataset API. When computing a result, the same execution engine is used, independent of which API/language you are using to express the computation. This unification means that developers can easily switch back and forth between different APIs based on which provides the most natural way to express a given transformation.

5. Explain Spark streaming in detail ?



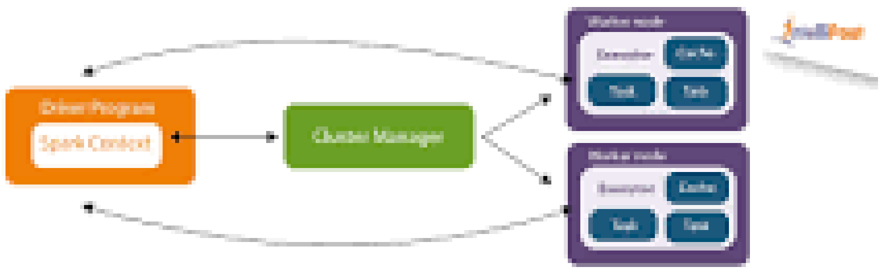
Spark Streaming is the previous generation of Apache Spark's streaming engine. There are no longer updates to Spark Streaming and it's a legacy project. There is a newer and easier to use streaming engine in Apache

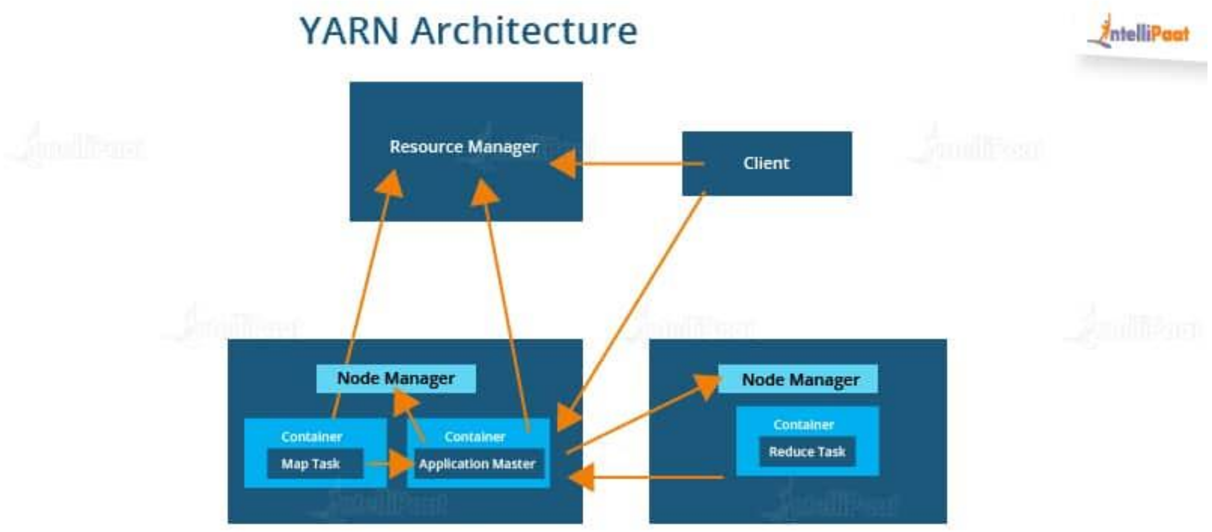
Spark called Structured Streaming. You should use Spark Structured Streaming for your streaming applications and pipelines.

Four Major Aspects of Spark Streaming

- Fast recovery from failures and stragglers
- Better load balancing and resource usage
- Combining of streaming data with static datasets and interactive queries
- Native integration with advanced processing libraries (SQL, machine learning, graph processing)

6. Explain SPARK architecure? what is Master - Slave architecure ?





Spark Architecture Overview:-

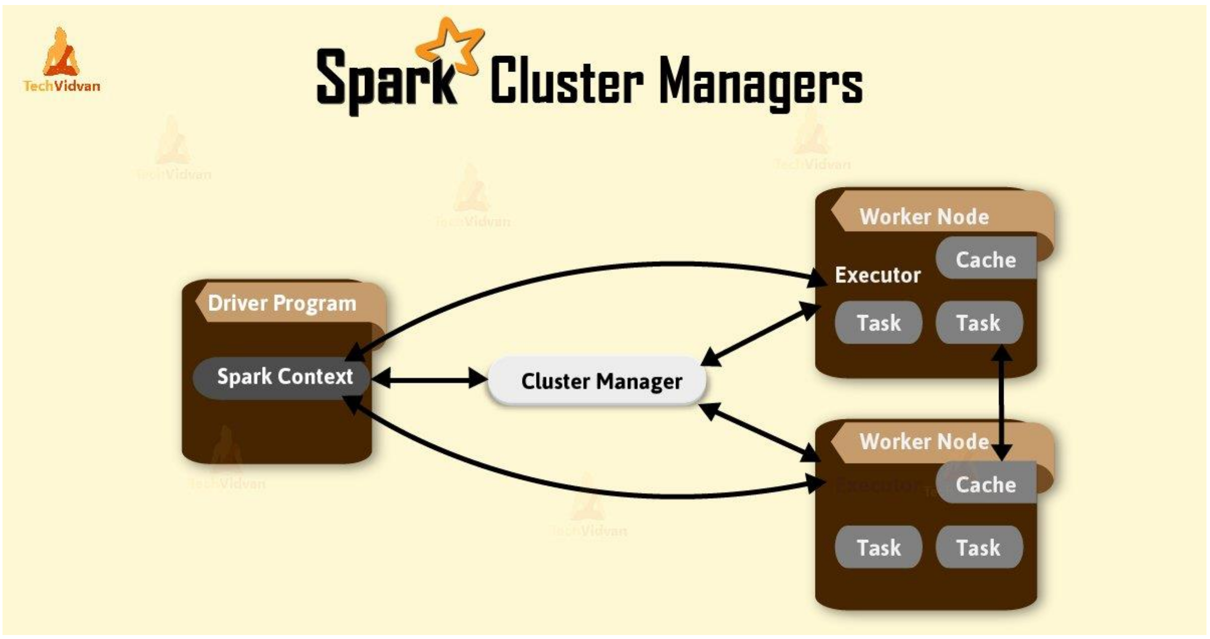
Apache Spark has a well-defined layered architecture where all the spark components and layers are loosely coupled. This architecture is further integrated with various extensions and libraries. Apache Spark Architecture is based on two main abstractions: Resilient Distributed Dataset (RDD)

10x faster on disk. 100x faster in memory.

Interactive shell. Less code. More operators. Write programs quickly

Build applications combining different processing model. Batch Analytics, Streaming Analytics and Interactive Analytics.

7. Explain various cluster managers in SPARK?
   Apache Spark Standalone Cluster Manager. Standalone mode is a simple cluster manager incorporated with Spark. It makes it easy to setup a cluster that Spark itself manages and can run on Linux, Windows, or Mac OSX. Often it is the simplest way to run Spark application in a clustered environment.

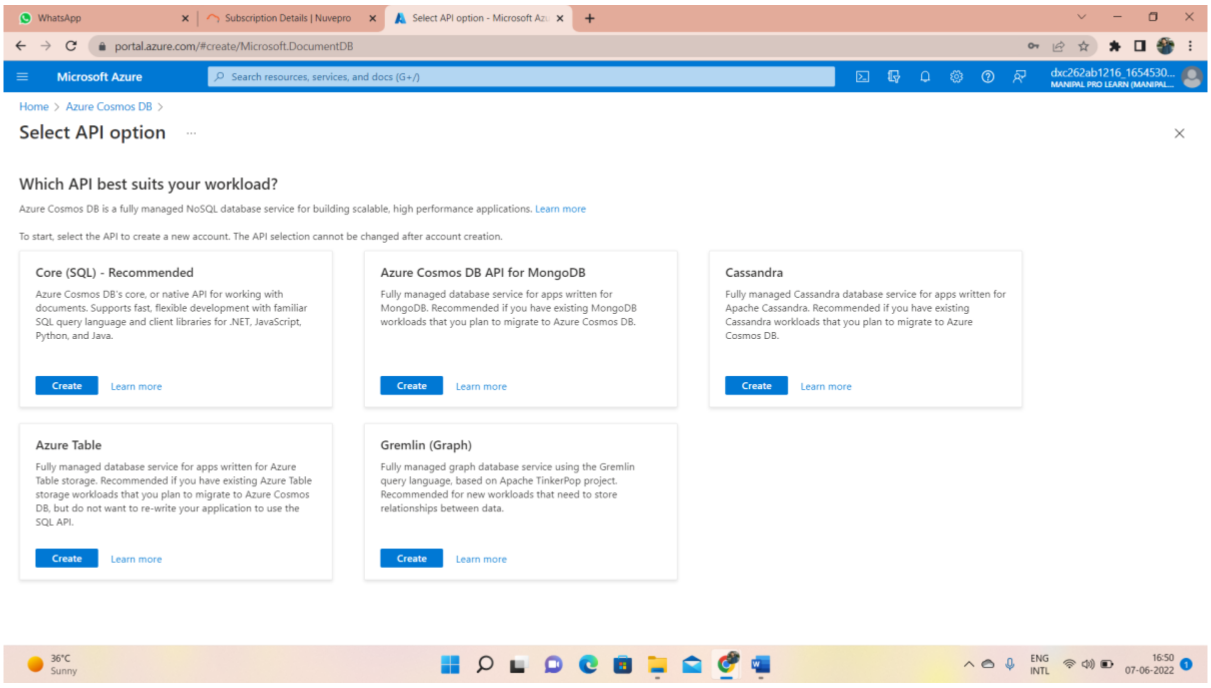


- **Standalone**: A simple cluster manager included within Spark can access HDFS and is easier to set up as it has a lot of online support. The cluster manager is resilient in nature and can successfully handle failures. It has the capability to manage resources according to the requirements of the applications.

- **Apache Mesos**: A general manager that can also run Hadoop MapReduce and service applications. It is a distributed cluster manager that can manage resources per application. We can easily run spark jobs, Hadoop MapReduce, or any other service applications. Apache has API for most programming languages.

- **Hadoop YARN:** A general manager in Hadoop. It acts as a distributed computing framework that maintains job scheduling as well as resource management. There are executors and pluggable schedulers that are readily available.
- **Kubernetes:** A system for the automation, deployment, scaling, and management of containerized applications. It makes use of a native Kubernetes scheduler that has been added to Spark; however, the Kubernetes scheduler is currently experimental.

8. Explain with sceenshots & steps how to create Cosmos DB ?


WhatsApp
Subscription Details | Nuvepro
Select API option - Microsoft Azu
portal.azure.com/#create/Microsoft.DocumentDB
Microsoft Azure
Search resources, services, and docs (G+/)
dxc262ab1216_1654530...
MANIPAL PRO LEARN (MANIPAL...
Home > Azure Cosmos DB >
36°C
Sunny
ENG
INTL
16:50
07-06-2022

Select API option

Which API best suits your workload?

Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. Learn more

To start, select the API to create a new account. The API selection cannot be changed after account creation.

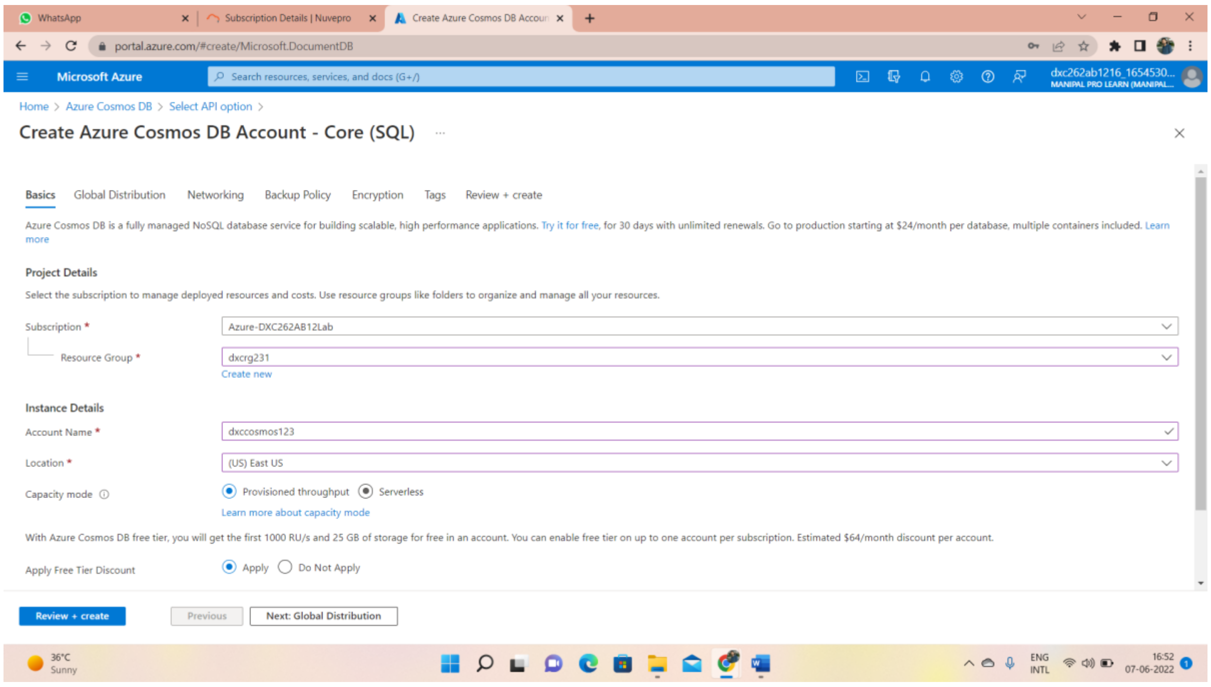
Core (SQL) - Recommended
Azure Cosmos DB's core, or native API for working with documents. Supports fast, flexible development with familiar SQL query language and client libraries for .NET, JavaScript, Python, and Java.
Create Learn more

Azure Cosmos DB API for MongoDB
Fully managed database service for apps written for MongoDB. Recommended if you have existing MongoDB workloads that you plan to migrate to Azure Cosmos DB.
Create Learn more

Cassandra
Fully managed Cassandra database service for apps written for Apache Cassandra. Recommended if you have existing Cassandra workloads that you plan to migrate to Azure Cosmos DB.
Create Learn more

Azure Table
Fully managed database service for apps written for Azure Table storage. Recommended if you have existing Azure Table storage workloads that you plan to migrate to Azure Cosmos DB, but do not want to re-write your application to use the SQL API.
Create Learn more

Gremlin (Graph)
Fully managed graph database service using the Gremlin query language, based on Apache TinkerPop project. Recommended for new workloads that need to store relationships between data.
Create Learn more


portal.azure.com/#create/Microsoft.DocumentDB

Home > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Core (SQL)

Basics | Global Distribution | Networking | Backup Policy | Encryption | Tags | Review + create

Azure Cosmos DB is a fully managed NoSQL database service for building scalable, high performance applications. Try it for free, for 30 days with unlimited renewals. Go to production starting at $24/month per database, multiple containers included. Learn more

Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Azure-DXC262AB12Lab
Resource Group *	dxcrg231

Create new

Instance Details

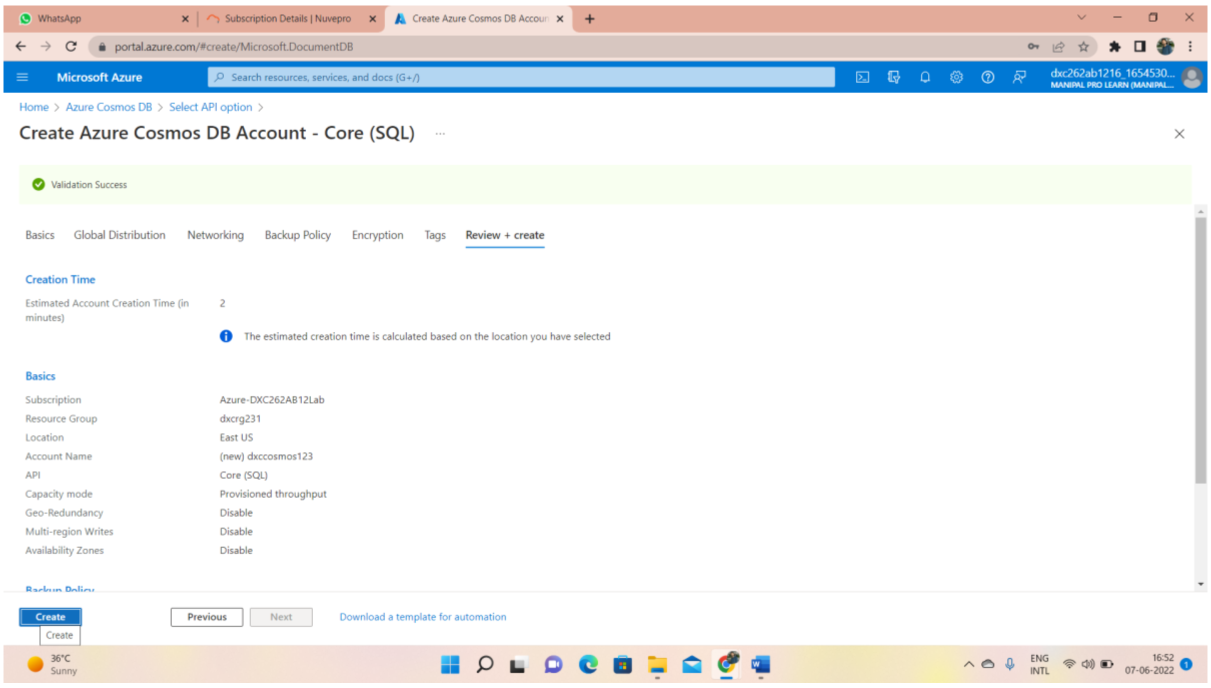
Account Name *	dxccosmos123
Location *	(US) East US
Capacity mode	[✓ Provisioned throughput] Serverless

Learn more about capacity mode

With Azure Cosmos DB free tier, you will get the first 1000 RU/s and 25 GB of storage for free in an account. You can enable free tier on up to one account per subscription. Estimated $64/month discount per account.

Apply Free Tier Discount	[✓ Apply] Do Not Apply

Review + create | Previous | Next: Global Distribution


portal.azure.com/#create/Microsoft.DocumentDB

Home > Azure Cosmos DB > Select API option >

Create Azure Cosmos DB Account - Core (SQL)

Validation Success

Basics | Global Distribution | Networking | Backup Policy | Encryption | Tags | Review + create

Creation Time

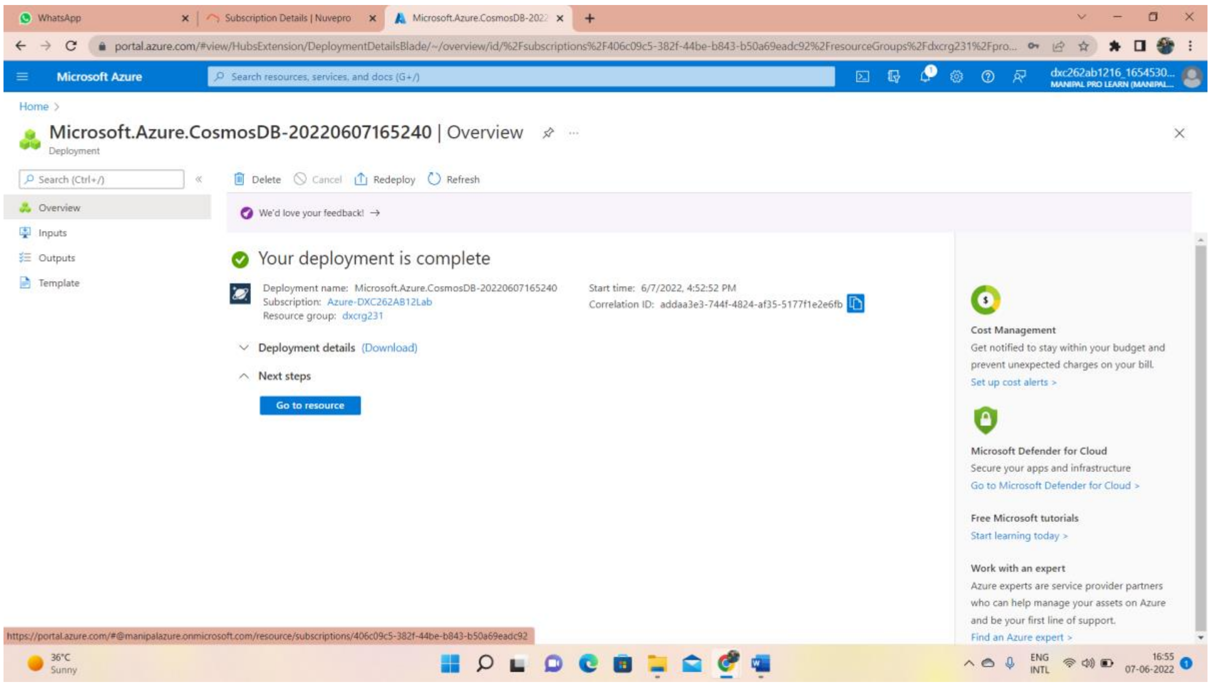
Estimated Account Creation Time (in minutes)	2

The estimated creation time is calculated based on the location you have selected

Basics

Subscription	Azure-DXC262AB12Lab
Resource Group	dxcrg231
Location	East US
Account Name	(new) dxccosmos123
API	Core (SQL)
Capacity mode	Provisioned throughput
Geo-Redundancy	Disable
Multi-region Writes	Disable
Availability Zones	Disable

Create | Previous | Next | Download a template for automation



Microsoft.Azure.CosmosDB-20220607165240 | Overview

Deployment

Your deployment is complete

Deployment name: Microsoft.Azure.CosmosDB-20220607165240
Subscription: Azure-DXC262AB12Lab
Resource group: dxcrg231

Start time: 6/7/2022, 4:52:52 PM
Correlation ID: addaa3e3-744f-4824-af35-5177f1e2e6fb

Deployment details (Download)

Next steps

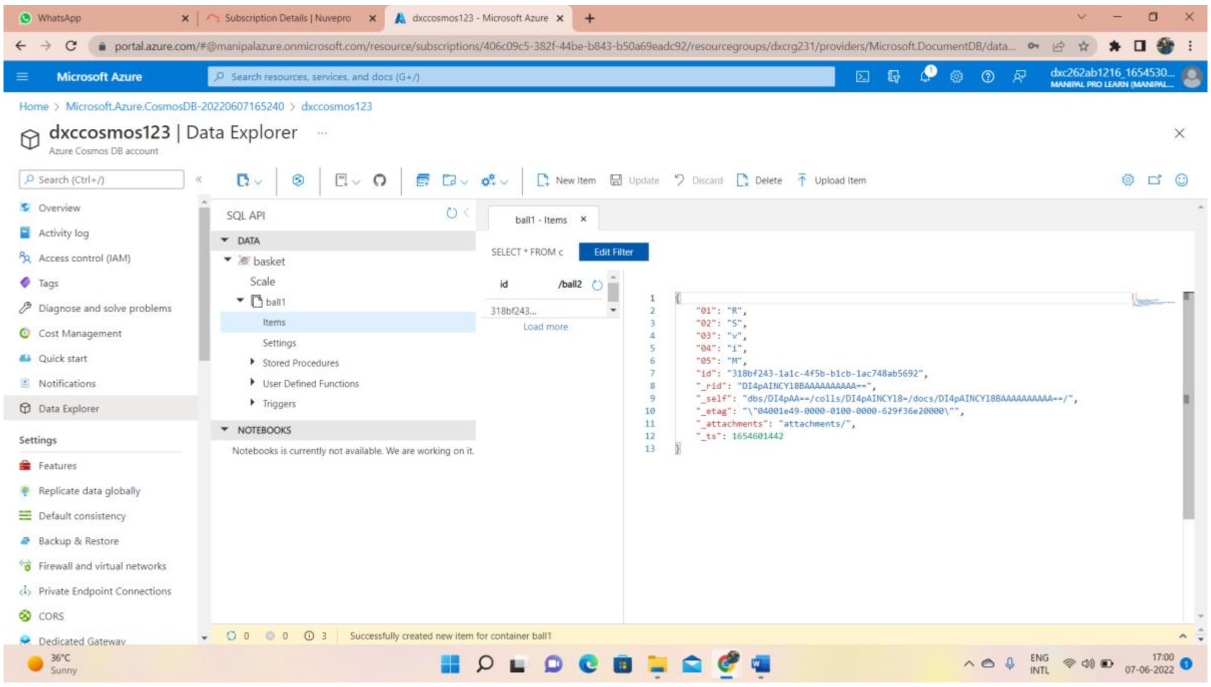
Go to resource

Cost Management
Get notified to stay within your budget and prevent unexpected charges on your bill.
Set up cost alerts >

Microsoft Defender for Cloud
Secure your apps and infrastructure
Go to Microsoft Defender for Cloud >

Free Microsoft tutorials
Start learning today >

Work with an expert
Azure experts are service provider partners who can help manage your assets on Azure and be your first line of support.
Find an Azure expert >



dxccosmos123 | Data Explorer

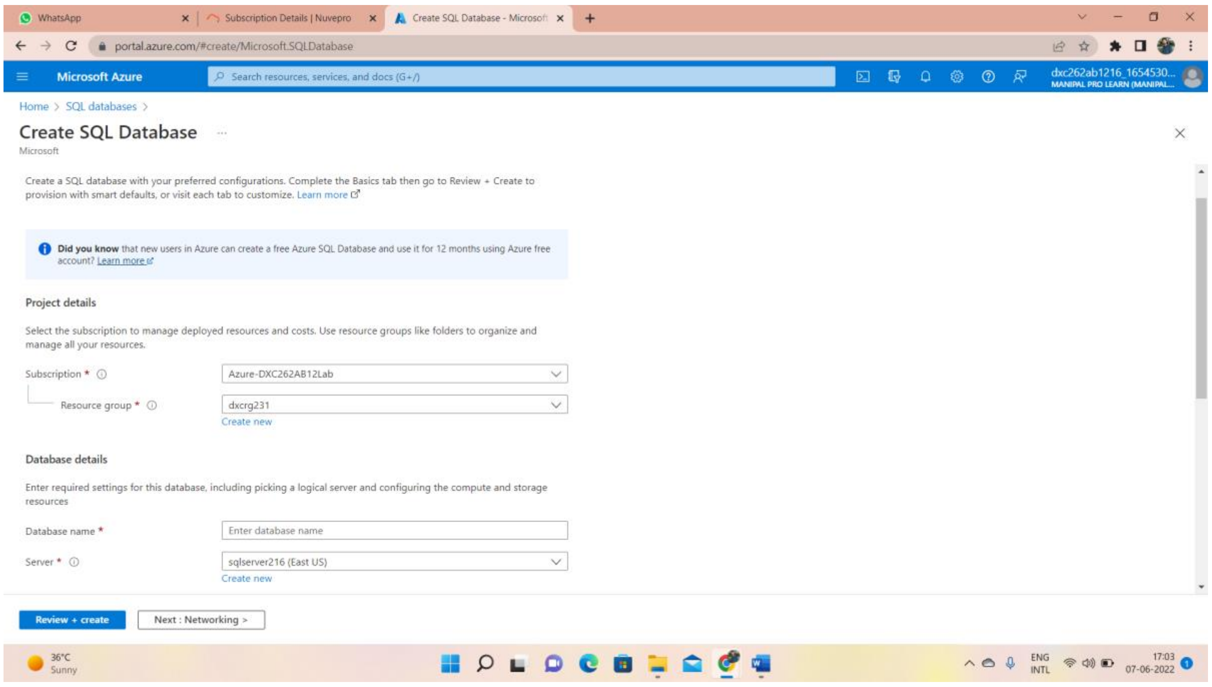
Azure Cosmos DB account

SELECT * FROM c

```
{
    "01": "R",
    "02": "S",
    "03": "v",
    "04": "i",
    "05": "M",
    "id": "318bf243-1a1c-4f5b-b1cb-1ac748ab5692",
    "_rid": "DI4pAINCY18BAAAAAAAAAA==",
    "_self": "dbs/DI4pAA==/colls/DI4pAINCY18=/docs/DI4pAINCY18BAAAAAAAAAA==/",
    "_etag": "\"04001e49-0000-0100-0000-629f36e20000\"",
    "_attachments": "attachments/",
    "_ts": 1654601442
}
```

Successfully created new item for container ball1

10. Explain with sceenshots & step how to create Azure SQL Db & also explain how to insert data into Azure SQL D?


Home > SQL databases >

Create SQL Database

Microsoft

Create a SQL database with your preferred configurations. Complete the Basics tab then go to Review + Create to provision with smart defaults, or visit each tab to customize. Learn more

Did you know that new users in Azure can create a free Azure SQL Database and use it for 12 months using Azure free account? Learn more

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *: Azure-DXC262AB12Lab

Resource group *: dxcrg231

Database details

Enter required settings for this database, including picking a logical server and configuring the compute and storage resources

Database name *: Enter database name

Server *: sqlserver216 (East US)

WhatsApp
Subscription Details | Nuvepro
Create SQL Database - Microsoft
portal.azure.com/#create/Microsoft.SQLDatabase
Microsoft Azure
Search resources, services, and docs (G+/)
Create new
Create new
Review + create
Next : Networking >
36°C
Sunny
ENG
INTL
17:03
07-06-2022

CREATE TABLE address(address_id INTEGER NOT NULL, address_building_number VARCHAR(55) NOT NULL, address_street VARCHAR(55) NOT NULL, address_locality VARCHAR(55), address_city VARCHAR(55) NOT NULL, address_zip_postal VARCHAR(55) NOT NULL, address_state_province_county VARCHAR(55) NOT NULL, address_country VARCHAR(55) NOT NULL , CONSTRAINT PK_Address PRIMARY KEY (address_id));
CREATE TABLE email_address(email_address_id INTEGER NOT NULL, email_address_person_id INTEGER, email_address VARCHAR(55) NOT NULL, CONSTRAINT PK_email_address PRIMARY KEY (email_address_id));
CREATE TABLE person(person_id INTEGER NOT NULL, person_first_name VARCHAR(55) NOT NULL, person_last_name VARCHAR(55) NULL, person_contacted_number INTEGER NOT NULL, person_date_last_contacted DATETIME NOT NULL,person_date_added DATETIME NOT NULL, CONSTRAINT PK_person PRIMARY KEY (person_id));
CREATE TABLE person_address(person_address_id INTEGER NOT NULL, person_address_person_id INTEGER NOT NULL, person_address_address_id INTEGER NOT NULL, CONSTRAINT PK_person_address PRIMARY KEY (person_address_id));
CREATE TABLE phone_number(phone_number_id INTEGER NOT NULL, phone_number_person_id INTEGER NOT NULL, phone_number VARCHAR(55) NOT NULL, CONSTRAINT PK_phone_number PRIMARY KEY (phone_number_id));

INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_zip_postal, address_state_province_county, address_country) VALUES (4, '555', 'azure203Demo', NULL, 'San Francisco', '91001', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, address_locality, address_city, address_zip_postal, address_state_province_county, address_country) VALUES (1, '555', 'azure203Demo', NULL, 'Los Angeles', '91001', 'California', 'US');
INSERT INTO address (address_id, address_building_number, address_street, address_locality,

```
address_city, address_zip_postal, address_state_province_county, address_country) VALUES (2,
'555', 'azure203Demo', NULL, 'Toronto', '7777', 'Ontario', 'Canada');
INSERT INTO address (address_id, address_building_number, address_street, address_locality,
address_city, address_zip_postal, address_state_province_county, address_country) VALUES (3,
'555', 'azure203Demo', 'Boonies', 'Somewhere', '11111', 'Maine', 'US');


INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES
(1, 1, 'jon.flanders@mail.com');
INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES
(2, 1, 'jonf@anothermail.com');


INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES
(4, 3, 'fritz@mail.com');
INSERT INTO email_address (email_address_id, email_address_person_id, email_address) VALUES
(5, NULL, 'aaron@mail.com');


INSERT INTO person (person_id, person_first_name, person_last_name,
person_contacted_number,person_date_last_contacted,person_date_added ) VALUES (1, 'Jon',
'Flanders', 5,'2013-09-14 11:43:31','2013-01-14 11:43:31');
INSERT INTO person (person_id, person_first_name, person_last_name,
person_contacted_number,person_date_last_contacted,person_date_added) VALUES (2,
'Shannon', 'Ahern', 0,'2013-08-14 11:43:31','2013-02-14 11:43:31');
INSERT INTO person (person_id, person_first_name, person_last_name,
person_contacted_number,person_date_last_contacted,person_date_added) VALUES (3, 'Fritz',
'Onion', 1,'2013-07-14 11:43:31','2013-03-14 11:43:31');


INSERT INTO person_address (person_address_id, person_address_person_id,
person_address_address_id) VALUES (1, 1, 1);
INSERT INTO person_address (person_address_id, person_address_person_id,
person_address_address_id) VALUES (3, 2, 1);
INSERT INTO person_address (person_address_id, person_address_person_id,
person_address_address_id) VALUES (4, 2, 2);
INSERT INTO person_address (person_address_id, person_address_person_id,
person_address_address_id) VALUES (5, 3, 3);


INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_number)
VALUES (1, 1, '555-1212');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_number)
```

```
VALUES (2, 2, '555-1213');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_number)
VALUES (3, 3, '555-1214');
INSERT INTO phone_number (phone_number_id, phone_number_person_id, phone_number)
VALUES (4, 3, '555-1215');
```