

Setups with Azure

```
pip install azure-eventhub
```

To send events:

```
from azure.eventhub import EventHubProducerClient, EventData

# Replace the following with your own values
CONNECTION_STR = 'YourEventHubConnectionString'
EVENTHUB_NAME = 'YourEventHubName'

# Create a producer client to send messages to the event hub
producer = EventHubProducerClient.from_connection_string(conn_str=CONNECTION_STR,
eventhub_name=EVENTHUB_NAME)

try:
    # Create a batch.
    event_data_batch = producer.create_batch()

    # Add a message to the batch.
    event_data_batch.add(EventData('First message'))

    # Send the batch of events to the event hub.
    producer.send_batch(event_data_batch)
    print("Message sent successfully.")
except Exception as ex:
    print(f"Exception: {ex}")
finally:
    # Close the producer.
    producer.close()
```

To Receive Events:

```
from azure.eventhub import EventHubConsumerClient

# Replace the following with your own values
CONNECTION_STR = 'YourEventHubConnectionString'
EVENTHUB_NAME = 'YourEventHubName'
CONSUMER_GROUP = '$Default' # Use the name of your consumer group

def on_event_batch(partition_context, events):
    for event in events:
        # Do something with the event
        print(f"Received event: '{event.body_as_str()}' from partition: {partition_context.partition_id}")

    # Update the checkpoint so that the program doesn't read these events again
    partition_context.update_checkpoint()

def on_error(partition_context, error):
    # Put your code here to deal with errors
    print(f"An error occurred on partition {partition_context.partition_id}: {error}")

consumer_client = EventHubConsumerClient.from_connection_string(
    conn_str=CONNECTION_STR,
    consumer_group=CONSUMER_GROUP,
```

```

    eventhub_name=EVENTHUB_NAME,
)

try:
    with consumer_client:
        # The on_event_batch function will be called for each batch of events received
        consumer_client.receive_batch(
            on_event_batch=on_event_batch,
            on_error=on_error,
            starting_position="-1", # "-1" is from the beginning of the partition.
        )
except KeyboardInterrupt:
    print("Stopped receiving.")
except Exception as ex:
    print(f'Exception: {ex}')

```

To Connect to CosmosDB (using the cassandra API)

```

pip install cassandra-driver

```

```

from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider

# Replace the following with your Azure Cosmos DB's Cassandra API details
CONTACT_POINTS = ['your-cosmos-db-cassandra-api-contact-point']
PORT = 10350
USERNAME = 'your-cosmos-db-username'
PASSWORD = 'your-cosmos-db-password'

# Configure the provider for authentication
auth_provider = PlainTextAuthProvider(username=USERNAME, password=PASSWORD)

# Create a session
cluster = Cluster(contact_points=CONTACT_POINTS, port=PORT, auth_provider=auth_provider, ssl_options=True)
session = cluster.connect()

# Create a keyspace - note that replication strategies should align with Cosmos DB's requirements
session.execute("""
    CREATE KEYSPACE IF NOT EXISTS cosmosdb_test
    WITH REPLICATION = {
        'class' : 'NetworkTopologyStrategy',
        'datacenter1' : 1
    };
""")

# Use the new keyspace
session.set_keyspace('cosmosdb_test')

# Create a table
session.execute("""
    CREATE TABLE IF NOT EXISTS users (
        user_id int PRIMARY KEY,
        name text,
        email text
    );
""")

# Insert a record
session.execute("""

```

```

INSERT INTO users (user_id, name, email)
VALUES (1, 'John Doe', 'john.doe@example.com');
"""

# Query the record
rows = session.execute('SELECT name, email FROM users WHERE user_id = 1')

for row in rows:
    print(row.name, row.email)

# Clean up (comment out if you want the table to persist)
# session.execute("DROP TABLE users")
# session.execute("DROP KEYSPACE cosmosdb_test")

# Close the session and cluster connection
cluster.shutdown()

```

Supplemental Instructions for Creating Cassandra in Azure

Pick the "REquest unit (RU) database account" option

Microsoft Azure

Search resources, services, and docs (G+/I)

Home > Azure Cosmos DB > Create an Azure Cosmos DB account >

Create Azure Cosmos DB Account - Choose Architecture ...

Which type of resource?

Azure Cosmos DB for Cassandra offers two resource types with different architectures. Request unit (RU) database accounts and Azure Managed Instance for Apache Cassandra. [See documentation to learn more.](#)

To start, select the type to create a resource. The resource selection cannot be changed after creation.

Request unit (RU) database account

- Industry-leading 99.999% availability
- Instantaneous, granular autoscaling
- Serverless accounts
- [See documentation and supported features](#)

Create

Azure Managed Instance for Apache Cassandra

- Familiar architecture
- Managed pure open source Apache Cassandra with 100% compatibility
- High capacity vertical and horizontal scaling
- [See documentation and supported features](#)

Create

Microsoft Azure

Search resources, services, and docs (G+)

Home > rg-csc545-cassandra > Marketplace > Azure Cosmos DB > Create an Azure Cosmos DB account > Create Azure Cosmos DB Account - Choose Architecture >

Create Azure Cosmos DB Account - Azure Cosmos DB for Apache Cassandra

Validation Success

Basics

Global Distribution

Networking

Backup Policy

Encryption

Tags

Review + create

Azure Cosmos DB is a fully managed NoSQL and relational 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 container

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 subscription 1

Resource Group *

rg-csc545-cassandra

Create new

Instance Details

Account Name *

csc545-project

Configure availability zone settings for your account. You cannot change these settings once the account is created.

Availability Zones ⓘ

Enable

Disable

Location *

(US) East US 2

Available locations are determined by your subscription's access and availability zone support (if that is enabled). If you don't see or cannot select your desired location, please open a support request for region. [Click here for more details on how to create a region access request](#)

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

Limit total account throughput

Limit the total amount of throughput that can be provisioned on this account

This limit will prevent unexpected charges related to provisioned throughput. You can update or remove this limit after your account is created.

Review + create

Previous

Next: Global Distribution

Open up the resource and then get the connection info from here:

portal.azure.com/#@southernct.edu/resource/subscriptions/654a4826-24c3-4f48-9fdb-5ad55267267c/resourceGroups/rg-csc545-cassandra/providers/Microsoft.DocumentDb/databaseAccounts/csc545-...

AppsSouthernHomeHNSelectiveRetirementMiroAdops

Microsoft Azure

Search resources, services, and docs (G+)

suvern1@soutSOUTHERN CONNEC

Home > Microsoft.Azure.CosmosDB-20240401224041 | Overview > csc545-project

DB

csc545-project | Connection strings

☆ ...

Search

RefreshFeedback

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Cost Management

Quick start

Notifications

Data Explorer

Settings

Features

Replicate data globally

Default consistency

Backup & Restore

Networking

Connection strings

Advisor Recommendations

Identity

Preview Features

Locks

CONTACT POINT

csc545-project.cassandra.cosmos.azure.com

PORT

10350

USERNAME

csc545-project

Read-write Keys

Read-only Keys

PRIMARY PASSWORD

Last regenerated: 4/1/2024 (0 days ago). Learn more

SECONDARY PASSWORD

Last regenerated: 4/1/2024 (0 days ago). Learn more

PRIMARY CONNECTION STRING

SECONDARY CONNECTION STRING

Notifications

More events in the activity log →

Deployment succeeded

Deployment 'Microsoft.Azure.CosmosDB-20240401224041' t group 'rg-csc545-cassandra' was successful.

Go to resourcePin to dashboard

a few