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)

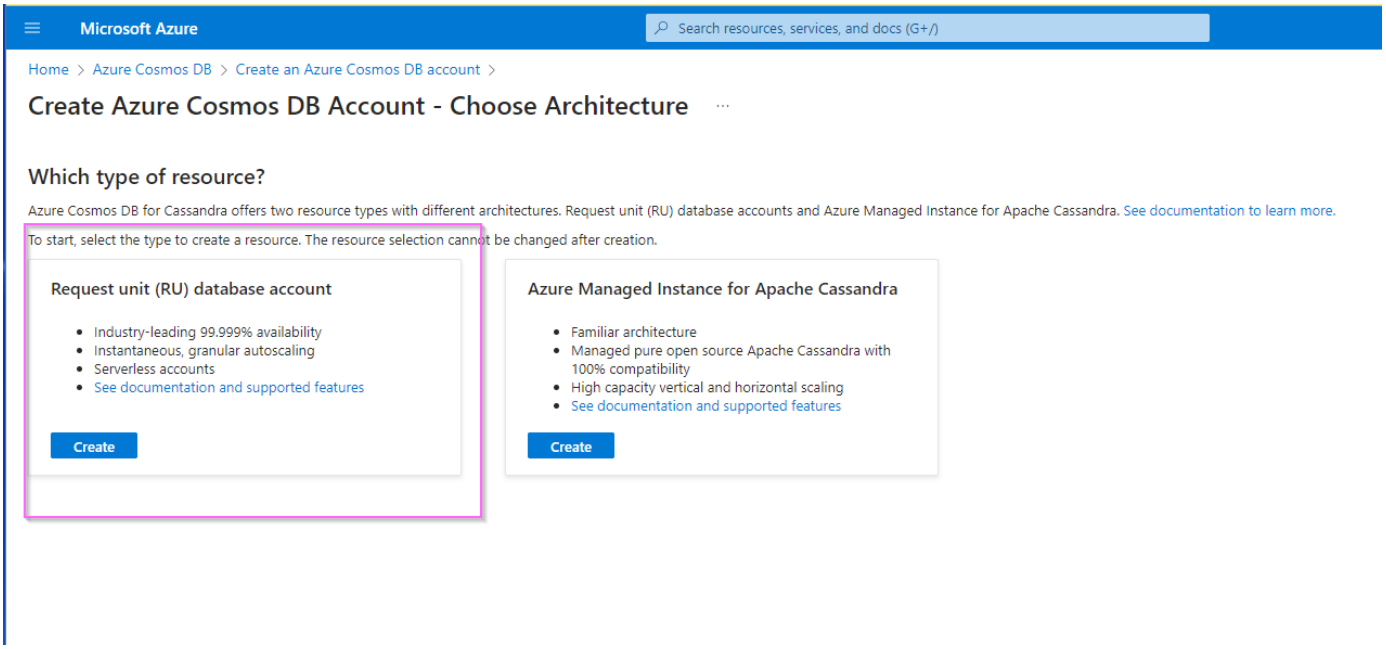
```
pip install cassandra-driver
```

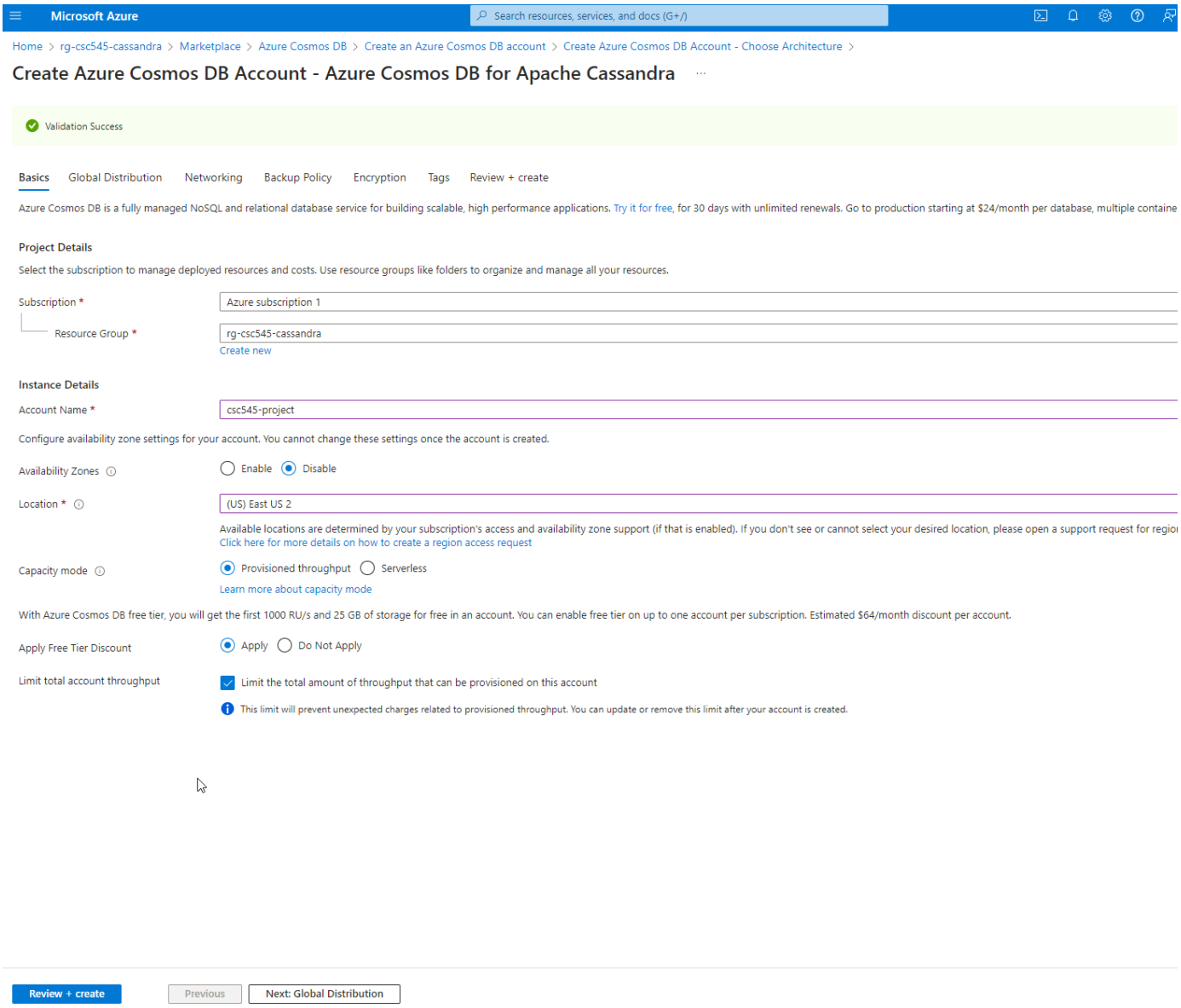
```
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider

# 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()
```

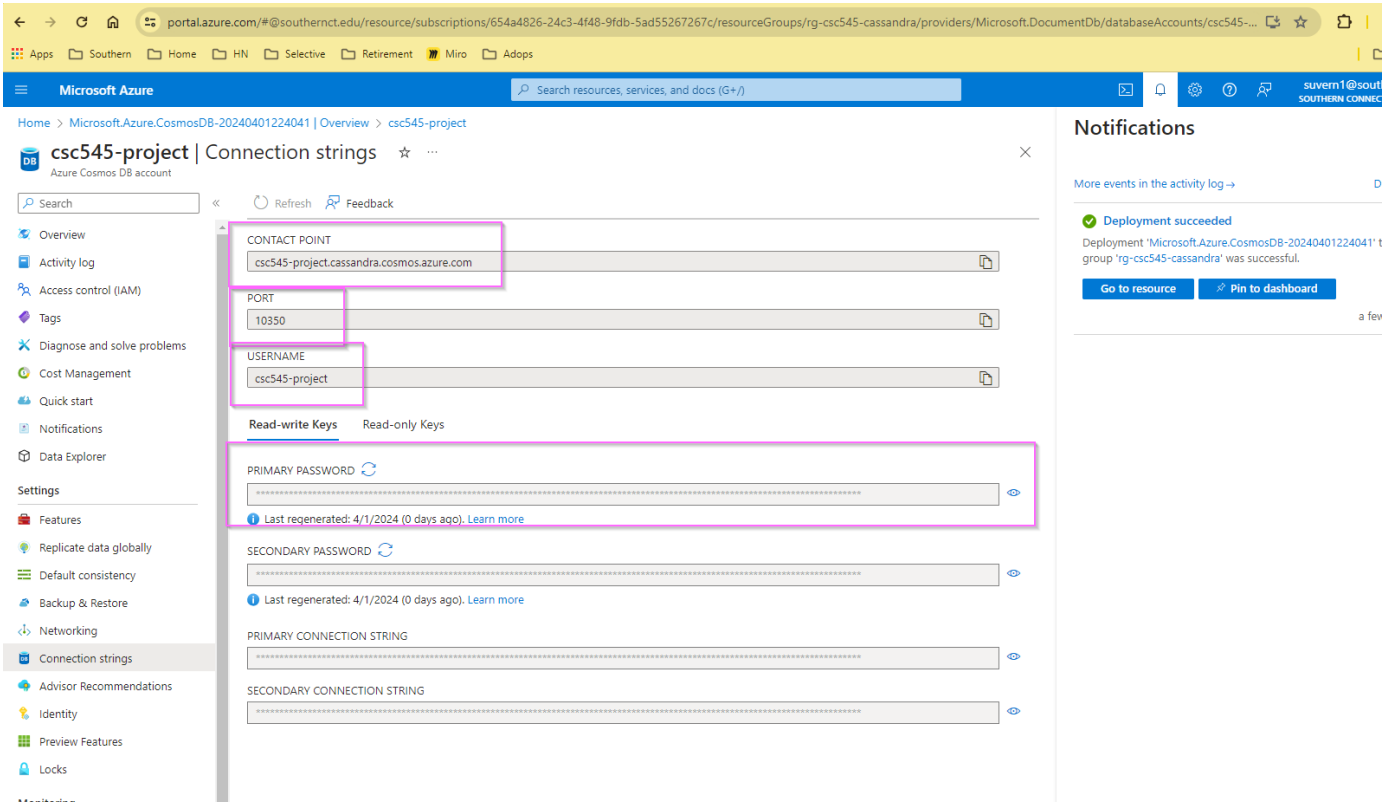
Supplemental Instructions for Creating Cassandra in Azure

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





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-...
Apps
Southern
Home
HN
Selective
Retirement
Miro
Adops
Microsoft Azure
Search resources, services, and docs (G+/)
suvern1@sout
SOUTHERN CONNEC
Home > Microsoft.Azure.CosmosDB-20240401224041 | Overview > csc545-project
csc545-project | Connection strings
Azure Cosmos DB account
Search
Refresh
Feedback
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 resource
Pin to dashboard
a few