100 XP

Exercise - Publish and subscribe to events using Pub/Sub

15 minutes

In your retail application, the software built by the inventory and shipping departments needs to respond to each other's events as part of a typical workflow. For example, the inventory microservice needs to deduct the item from its inventory once an order is shipped.

In this exercise, implement Pub/Sub in an Azure Cache for Redis instance using multiple console windows.

Open the Azure Cache for Redis console in two browser instances

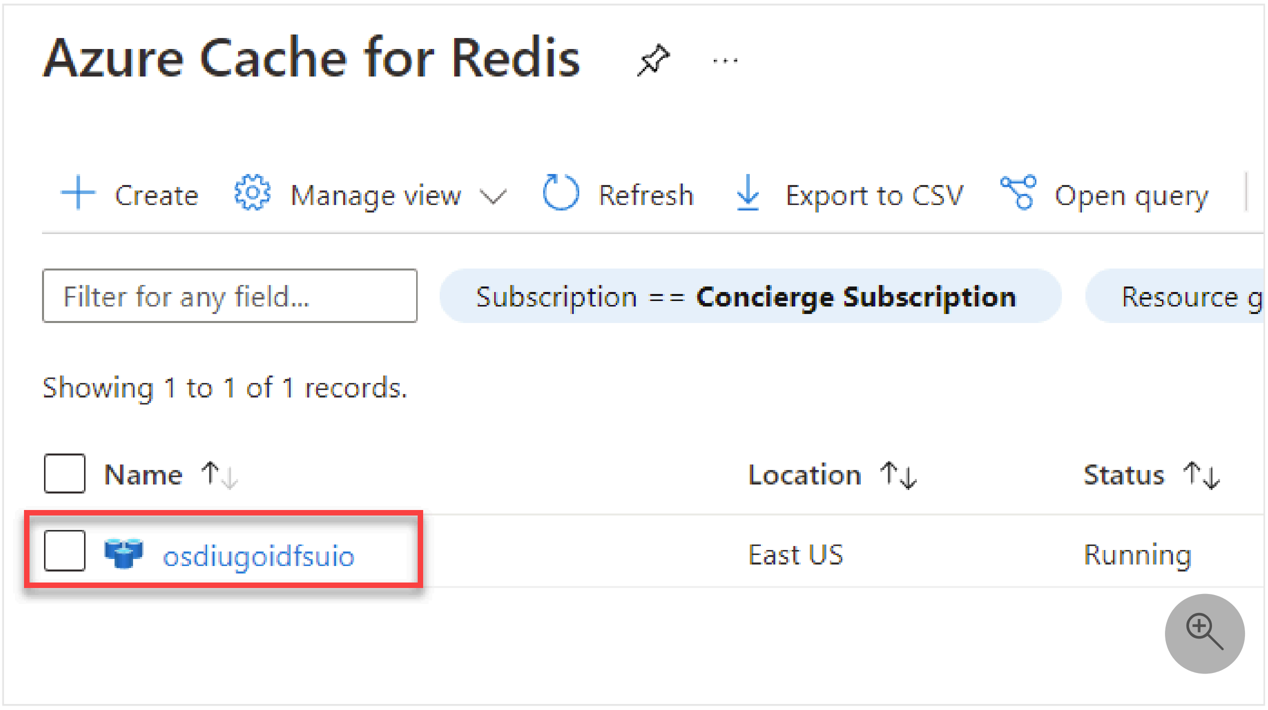
Observe two separate clients sending and receiving messages by using two unique browser instances. Each browser instance has the Redis console to help illustrate real-time Pub/Sub functionality.

Open the first browser instance

1. Sign in to the Azure portal using the same account you used to activate the sandbox.

2. Within the **Azure services** category, select **More services**, select the **Databases** category and then select **Azure Cache for Redis**.

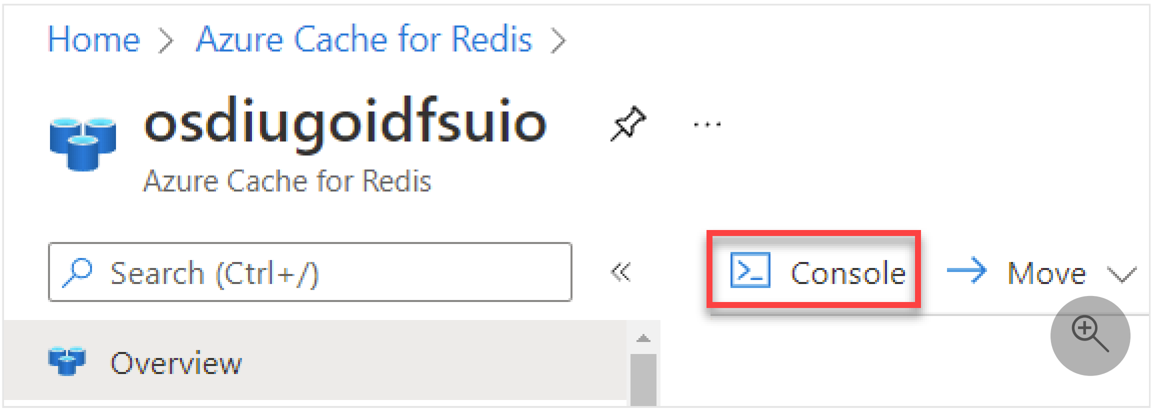
3. Select the Azure Cache for Redis instance you created in a previous exercise.



Showing 1 to 1 of 1 records.

Name	Location	Status
osdiugoidfsuio	East US	Running

4. In the resource pane, select **Console** to open the Redis console.



5. Leave the console open to complete subsequent tasks in this exercise.

Open the second browser instance

1. Sign in to the Azure portal using the same account you used to activate the sandbox and a separate browser window or tab.

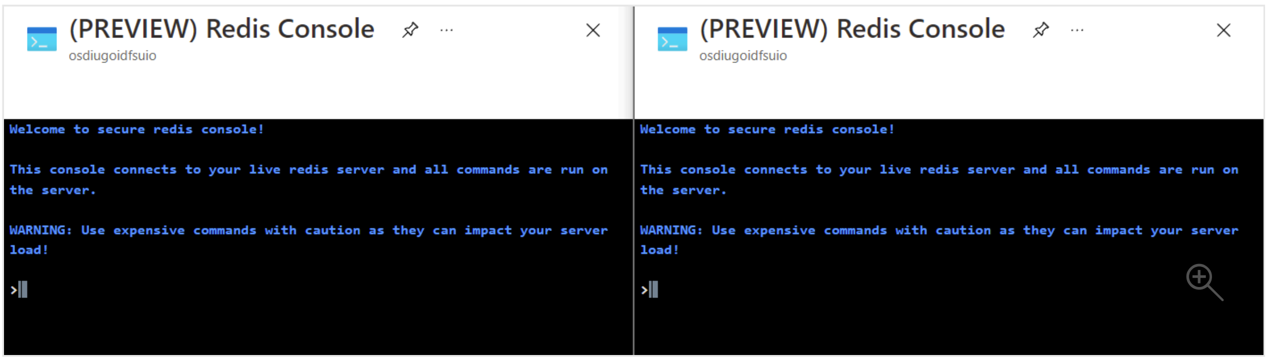
2. Within the **Azure services** category, select **More services**, select the **Databases** category and then select **Azure Cache for Redis**.

3. Select the Azure Cache for Redis instance you created in a previous exercise.

4. In the resource pane, select **Console** to open the Redis console.

5. Leave the console open to complete subsequent tasks in this exercise.

6. At this point, you should have two browser instances open each with an instance of the Redis console.



```
Welcome to secure redis console!

This console connects to your live redis server and all commands are run on
the server.

WARNING: Use expensive commands with caution as they can impact your server
load!

>
```

```
Welcome to secure redis console!

This console connects to your live redis server and all commands are run on
the server.

WARNING: Use expensive commands with caution as they can impact your server
load!

>
```

> **Tip**
>
> If your operating system supports it, we recommend *docking* the browser windows side-by-side to simplify the remainder of this exercise.

Subscribe and publish messages to a known channel

Subscribe to channels using the `SUBSCRIBE` command and then publish messages using the `PUBLISH` command.

1. In the console of the **first** browser instance, perform the following actions:

   a. Enter the following command and use the **ENTER** key to begin listening for messages on the **org.shipping.alerts** channel.

   Redis

   ```
   SUBSCRIBE org.shipping.alerts
   ```

   b. Observe the response from the console indicating that it's now listening on the **org.shipping.alerts** channel.

   Redis

   ```
   Reading messages... (press ENTER to quit)
   1) "subscribe"
   2) "org.shipping.alerts"
   3) (integer) 1
   ```

2. In the console of the **second** browser instance, perform the following actions:

    a. Enter the following command and use the **ENTER** key to send a new message with the content **labelprint-sdf9878** to the **org.shipping.alerts** channel.

    Redis

    ```
    PUBLISH org.shipping.alerts labelprint-sdf9878
    ```

    b. Enter the following command and use the **ENTER** key to send a new message with the content **labelprint-sdf9878** to the **org.shipping.alerts** channel.

    Redis

    ```
    PUBLISH org.shipping.alerts packagesent-sdf9878
    ```

3. Back in the console of the **first** browser instance, perform the following actions:

    a. Observe the first response from the console indicating that it has received a new message on the **org.shipping.alerts** channel.

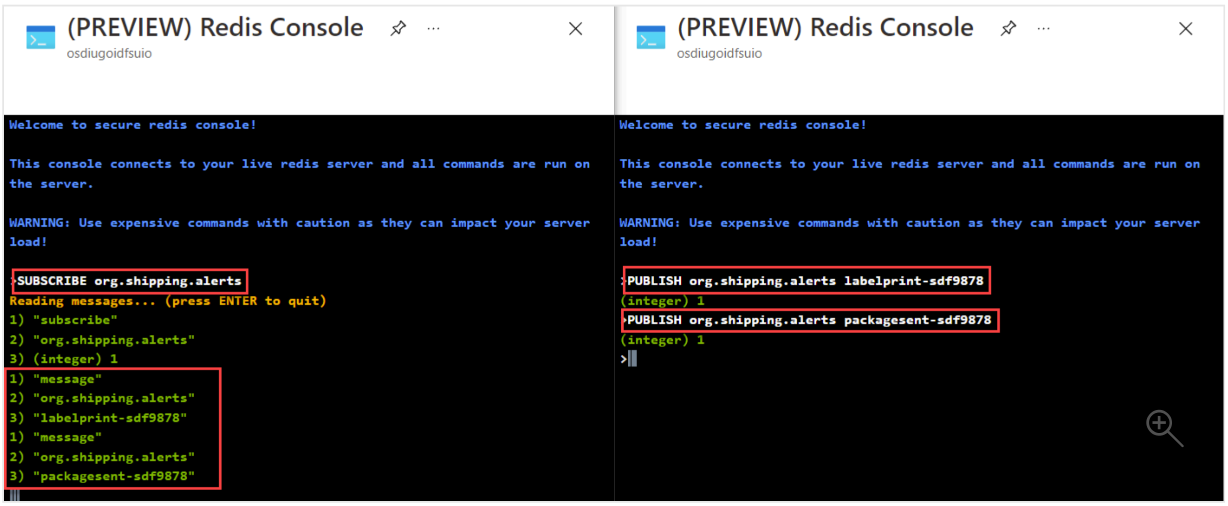
    Redis

    ```
    1) "message"
    2) "org.shipping.alerts"
    3) "labelprint-sdf9878"
    ```

    b. Observe the second response from the console indicating that it has received a new message on the **org.shipping.alerts** channel.

    Redis

    ```
    1) "message"
    2) "org.shipping.alerts"
    3) "packagesent-sdf9878"
    ```



```
Welcome to secure redis console!

This console connects to your live redis server and all commands are run on
the server.

WARNING: Use expensive commands with caution as they can impact your server
load!

>SUBSCRIBE org.shipping.alerts
Reading messages... (press ENTER to quit)
1) "subscribe"
2) "org.shipping.alerts"
3) (integer) 1
1) "message"
2) "org.shipping.alerts"
3) "labelprint-sdf9878"
1) "message"
2) "org.shipping.alerts"
3) "packagesent-sdf9878"
```

```
Welcome to secure redis console!

This console connects to your live redis server and all commands are run on
the server.

WARNING: Use expensive commands with caution as they can impact your server
load!

>PUBLISH org.shipping.alerts labelprint-sdf9878
(integer) 1
>PUBLISH org.shipping.alerts packagesent-sdf9878
(integer) 1
>
```

c. Use the **ENTER** key to stop the console from listening to events.

d. Enter the following command and use the **ENTER** key to clear the console output.

Redis

```
clear
```

4. Back in the console of the **second** browser instance, perform the following actions:

   a. Enter the following command and use the **ENTER** key to clear the console output.

Redis

```
clear
```

Subscribe to a channel pattern and listen for messages

Subscribe to a pattern of channels using the `PSUBSCRIBE` command and then publish a message using the `PUBLISH` command.

1. In the console of the **first** browser instance, perform the following actions:

   a. Enter the following command and use the **ENTER** key to begin listening for messages on the **org.shipping.alerts** channel.

Redis

```
PSUBSCRIBE org.inventory.*
```

b. Observe the response from the console indicating that it's now listening on the **org.inventory.*** channel pattern.

Redis

```
Reading messages... (press ENTER to quit)
1) "psubscribe"
2) "org.inventory.*"
3) (integer) 1
```

2. In the console of the **second** browser instance, perform the following actions:

   a. Enter the following command and use the **ENTER** key to send a new message with the content **item-sku-318947** to the **org.inventory.empty** channel.

   Redis

   ```
   PUBLISH org.inventory.empty item-sku-318947
   ```

   b. Enter the following command and use the **ENTER** key to send a new message with the content **order-dsy3821** to the **org.shipping.sent** channel.

   Redis

   ```
   PUBLISH org.shipping.sent order-dsy3821
   ```

   Note

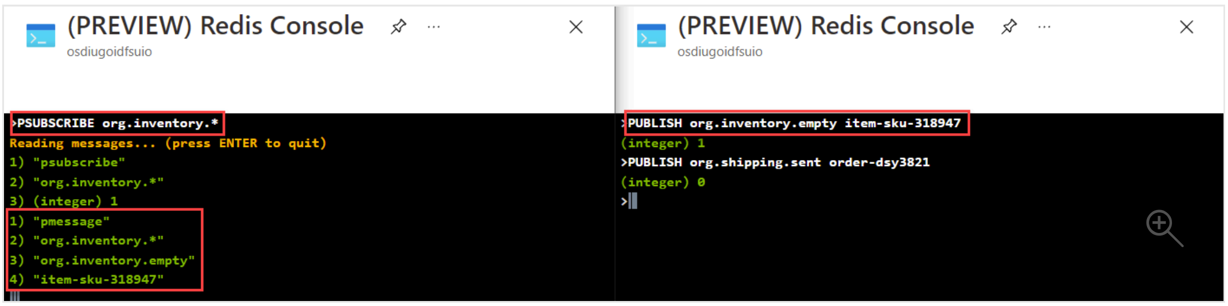
   Since this channel does not match the **org.inventory.*** pattern, you shouldn't expect this command to send a message that your other client will receive.

3. Back in the console of the **first** browser instance, perform the following actions:

   a. Observe the response from the console indicating that it has received only a single new message using the **org.inventory.*** channel pattern.

   Redis

   ```
   1) "pmessage"
   2) "org.inventory.*"
   3) "org.inventory.empty"
   4) "item-sku-318947"
   ```



```
>PSUBSCRIBE org.inventory.*
Reading messages... (press ENTER to quit)
1) "psubscribe"
2) "org.inventory.*"
3) (integer) 1
1) "pmessage"
2) "org.inventory.*"
3) "org.inventory.empty"
4) "item-sku-318947"
```

```
>PUBLISH org.inventory.empty item-sku-318947
(integer) 1
>PUBLISH org.shipping.sent order-dsy3821
(integer) 0
>
```

b. Use the **ENTER** key to stop the console from listening to events.

Next unit: Streams

Continue >