

Serverless | Cosmos DB

Running and publish Azure Functions project with Cosmos DB



Prerequisites and more!

An *Azure subscription* (free account).

Main supported languages:

Language	Extension
C#	C# for VS Code
Java	Debugger for Java * + Maven 3+
Typescript	.Net Core 2.2 (preview)
Javascript	Node 8.0+

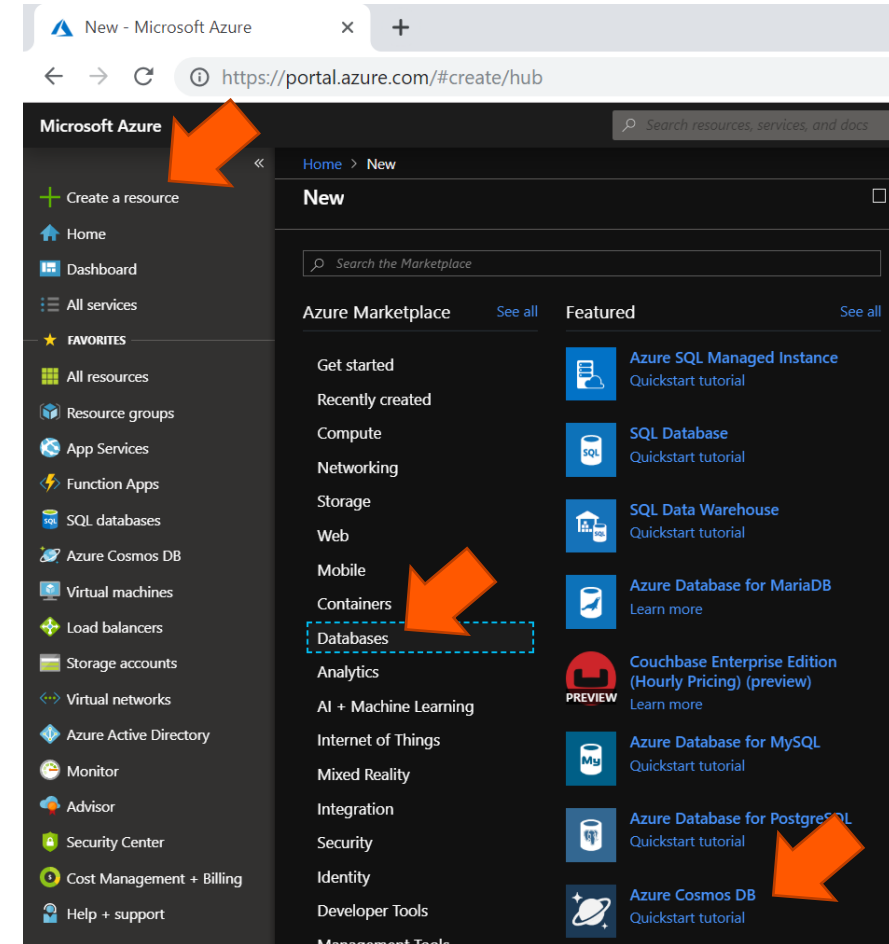
Create an Azure Cosmos DB

You must have an *Azure Cosmos DB* account that uses the *SQL API* before you create the output binding.

Sign in to the *Azure portal*.

Select:

- > *Create a resource*
- > *Databases*
- > *Azure Cosmos DB*



Create an Azure Cosmos DB

On the create *Azure Cosmos DB* page, enter the basic settings for the new *Azure Cosmos account*.

Create Azure Cosmos DB Account

[Basics](#) [Network](#) [Tags](#) [Review + create](#)

Azure Cosmos DB is a fully managed globally distributed, multi-model database service, transparently replicating your data across any number of Azure regions. You can elastically scale throughput and storage, and take advantage of fast, single-digit-millisecond data access using the API of your choice backed by 99.999 SLA. [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

Visual Studio Professional

* Resource Group

Select existing...

Create new

INSTANCE DETAILS

* Account Name

Enter account name

documents.azure.com

* API ⓘ

Core (SQL)

* Location

(South America) Brazil South

Geo-Redundancy ⓘ

Enable

Disable

Multi-region Writes ⓘ

Enable

Disable

Create an Azure Cosmos DB

Select a *resource group*, or select *Create new*, then enter a unique name for the new resource group.

Fill using: *"serverlesslabcosmodb"*

The image shows the 'Create Azure Cosmos DB Account' portal. The 'Basics' tab is selected. Under 'PROJECT DETAILS', the 'Subscription' is set to 'Visual Studio Professional'. The 'Resource Group' dropdown is open, showing 'Select existing...' and 'Create new' options. An orange arrow points from the 'Create new' link to a modal window. The modal explains that a resource group is a container for related resources and shows the name 'serverlesslabcosmodb' entered in the 'Name' field, with a green checkmark indicating it is valid. The modal has 'OK' and 'Cancel' buttons. In the background, the 'INSTANCE DETAILS' section is partially visible, showing the 'Account Name' field and a 'Next: Network' button.

Create Azure Cosmos DB Account

Basics Network Tags Review + create

Azure Cosmos DB is a fully managed globally distributed, multi-model database service, transparently replicating your data across any number of Azure regions. You can elastically scale throughput and storage, and take advantage of fast, single-digit-millisecond data access using the API of your choice backed by 99.999 SLA. [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 Visual Studio Professional

* Resource Group Select existing... Create new

INSTANCE DETAILS

* Account Name Enter account name documents.azure.com

Core (SQL)

Asia Pacific) Australia East

enable Disable

enable Disable

Next: Network

A resource group is a container that holds related resources for an Azure solution.

* Name

serverlesslabcosmodb ✓

OK Cancel

Create an Azure Cosmos DB

Enter a *name* to identify your *Azure Cosmos* account. (Because *documents.azure.com* is appended to the ID that you provide to create your URL, use a unique ID.)

Fill using: "*serverlesslabcosmodb*"

Select *Review + create*.

You can skip the *Network* and *Tags* sections.

Create Azure Cosmos DB Account

[Basics](#) [Network](#) [Tags](#) [Review + create](#)

Azure Cosmos DB is a fully managed globally distributed, multi-model database service, transparently replicating your data across any number of Azure regions. You can elastically scale throughput and storage, and take advantage of fast, single-digit-millisecond data access using the API of your choice backed by 99.999 SLA. [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: Visual Studio Professional

* Resource Group: (New) serverlesslabcosmosdb [Create new](#)

INSTANCE DETAILS

* Account Name: serverlesslabcosmodb ✓ documents.azure.com

* API ⓘ: Core (SQL)

* Location: (South America) Brazil South

Geo-Redundancy ⓘ: [Enable](#) [Disable](#)

Multi-region Writes ⓘ: [Enable](#) [Disable](#)

[Review + create](#) [Previous](#) [Next: Network](#)

Create an Azure Cosmos DB

Review the account settings, and then click *Create* button. It takes a few minutes to create the account.

Create Azure Cosmos DB Account

✓ Validation Success

Basics Network Tags **Review + create**

BASICS

Subscription	Visual Studio Professional
Resource Group	(new) serverlesslabcosmosdb
Location	(South America) Brazil South
Account Name	(new) serverlesslabcosmodb
API	Core (SQL)
Geo-Redundancy	Disable
Multi-region Writes	Disable

Create Previous Next

Create an Azure Cosmos DB

Wait for the portal page to display *your deployment is complete*. Check the resource's name.

Microsoft.Azure.CosmosDB-20190426222929 - Overview
Deployment

Search (Ctrl+/) Delete Cancel Redeploy Refresh

Overview

Inputs

Outputs

Template

✓ **Your deployment is complete**

Check the status of your deployment, manage resources, or troubleshoot deployment issues. Pin this page to your dashboard to easily find it next time.

Deployment name: Microsoft.Azure.CosmosDB-20190426222929
Subscription: [Visual Studio Professional](#)
Resource group: [serverlesslabcosmosdb](#)

DEPLOYMENT DETAILS ([Download](#))

Start time: 4/26/2019, 10:52:59 PM
Duration: 2 minutes 41 seconds
Correlation ID: 04e0e38f-44a3-4fdb-ac69-fd9e553d939b

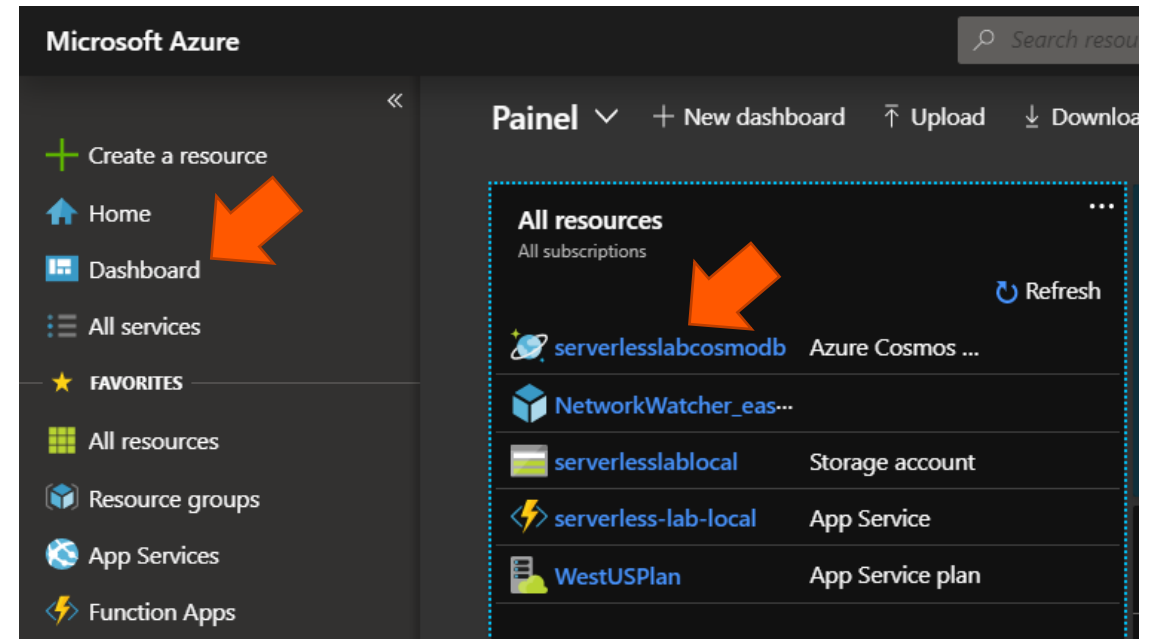
RESOURCE	TYPE	STATUS	OPERATION DETAILS
✓ serverlesslabcosmodb	Microsoft.DocumentDb/data...	OK	Operation details

Access an Azure Cosmos DB resource

You must have to access *Dashboard* and find your *Azure Cosmos DB* resource.

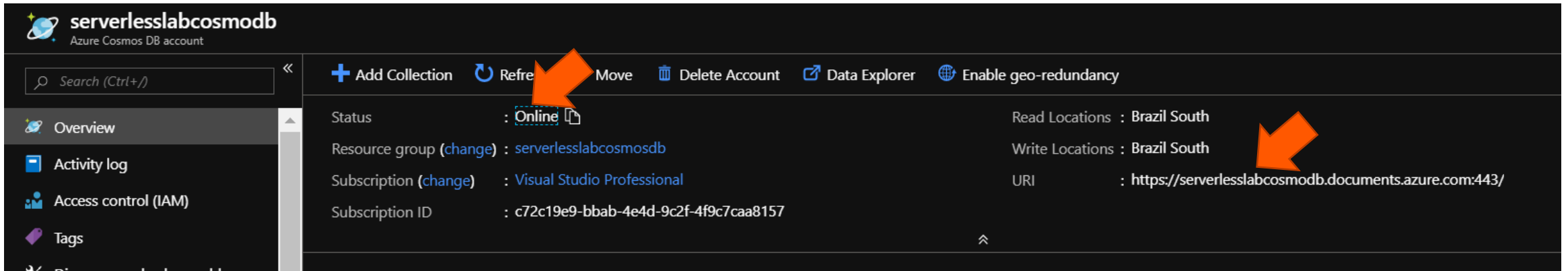
Select:

- > *Dashboard*
- > *serverlesslabcosmodb*



Access an Azure Cosmos DB resource

Some important informations to have external access (*application, AFs and etc.*) to your *Azure Cosmos DB*.



The screenshot shows the Azure Cosmos DB account overview page for 'serverlesslabcosmodb'. The left sidebar contains navigation links: Overview, Activity log, Access control (IAM), Tags, and Diagnostic logs. The main content area displays account details and actions. An orange arrow points to the 'Refresh' button in the top action bar. Another orange arrow points to the 'URI' field in the account details section.

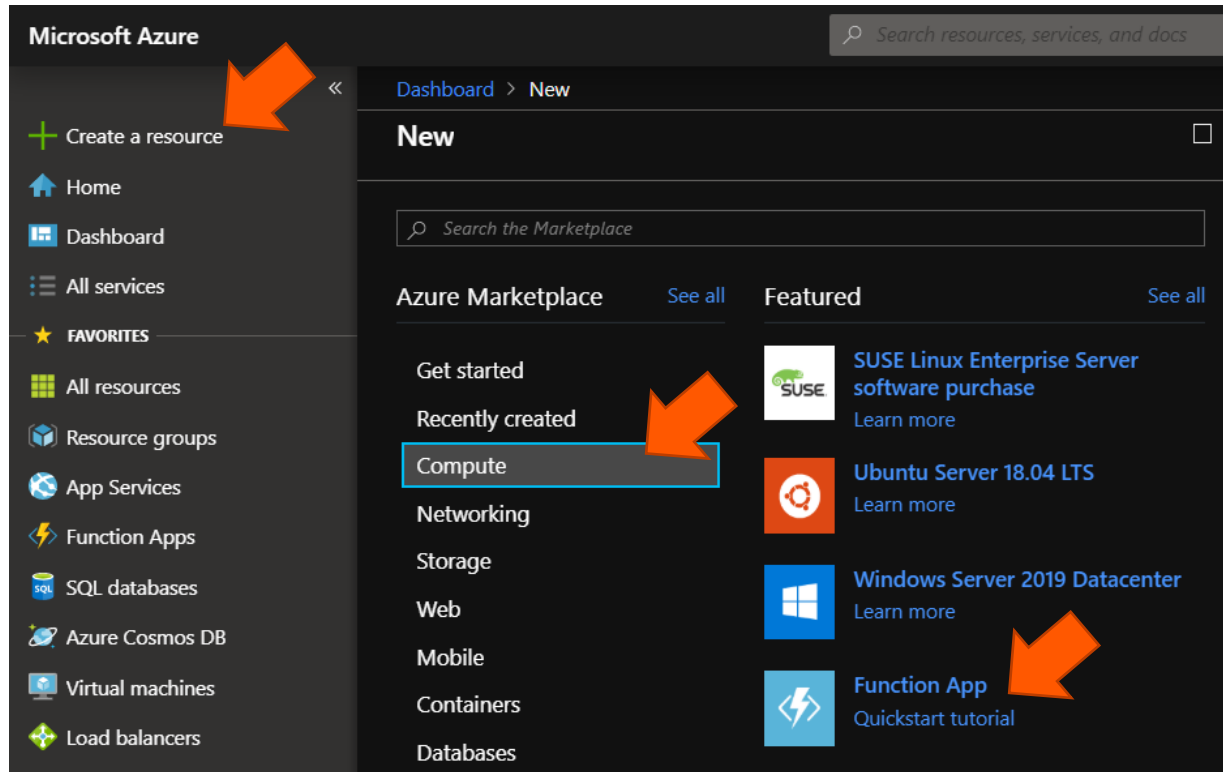
Property	Value
Status	: Online
Resource group (change)	: serverlesslabcosmosdb
Subscription (change)	: Visual Studio Professional
Subscription ID	: c72c19e9-bbab-4e4d-9c2f-4f9c7caa8157
Read Locations	: Brazil South
Write Locations	: Brazil South
URI	: https://serverlesslabcosmodb.documents.azure.com:443/

Create an Azure Function | App

Select the *Create* a resource button found on the upper left-hand corner of the *Azure portal*.

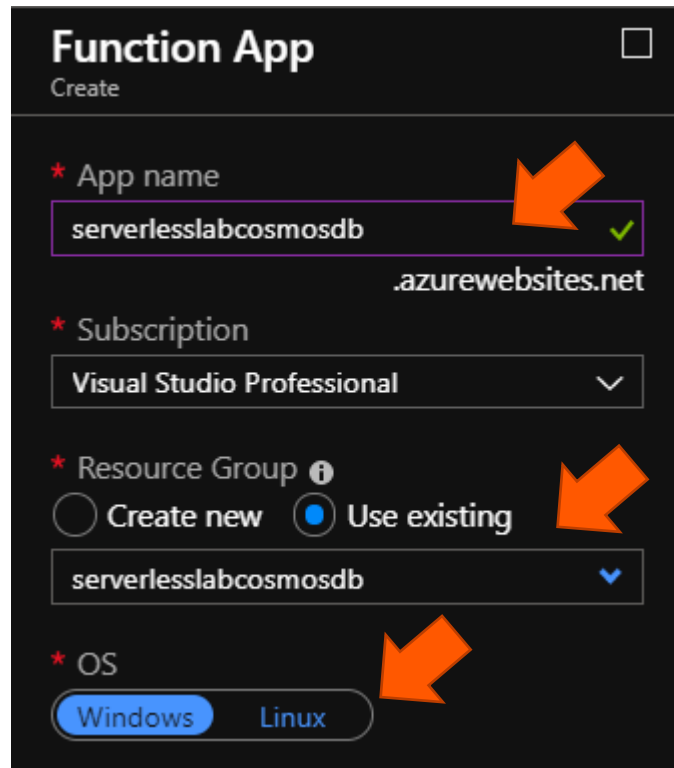
Select:

- > *Create a resource*
- > *Compute*
- > *Function App*



Create an Azure Function | App

Fill and review the *Azure Function* settings, and then click *Create* button.



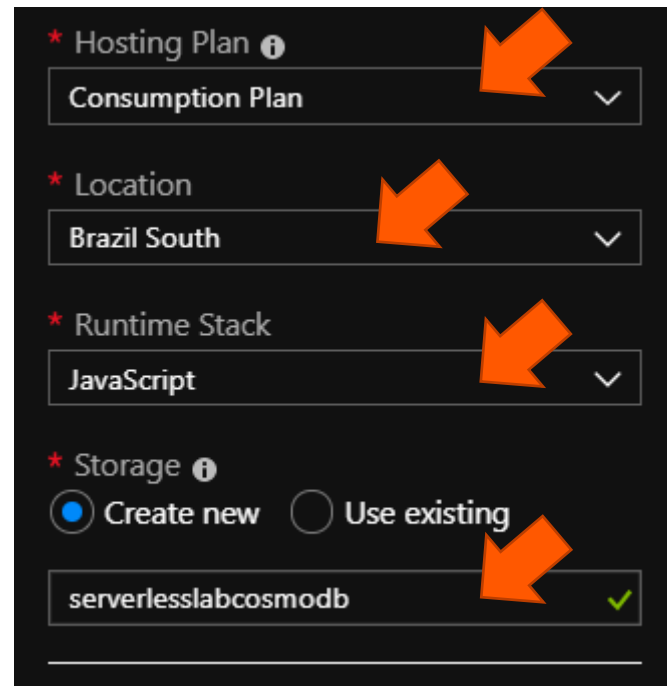
Function App Create

* App name serverlesslabcosmosdb .azurewebsites.net ✓

* Subscription Visual Studio Professional ▼

* Resource Group ⓘ Create new ☒ Use existing serverlesslabcosmosdb ▼

* OS Windows Linux

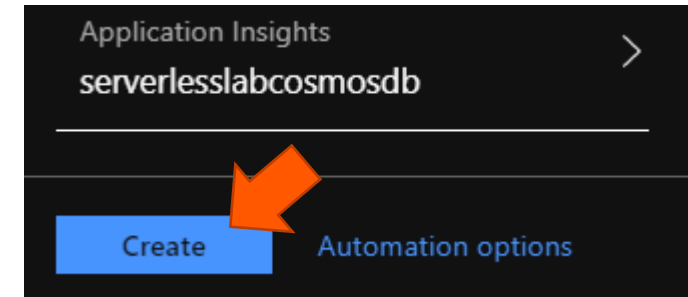


* Hosting Plan ⓘ Consumption Plan ▼

* Location Brazil South ▼

* Runtime Stack JavaScript ▼

* Storage ⓘ ☒ Create new ☐ Use existing serverlesslabcosmodb ✓

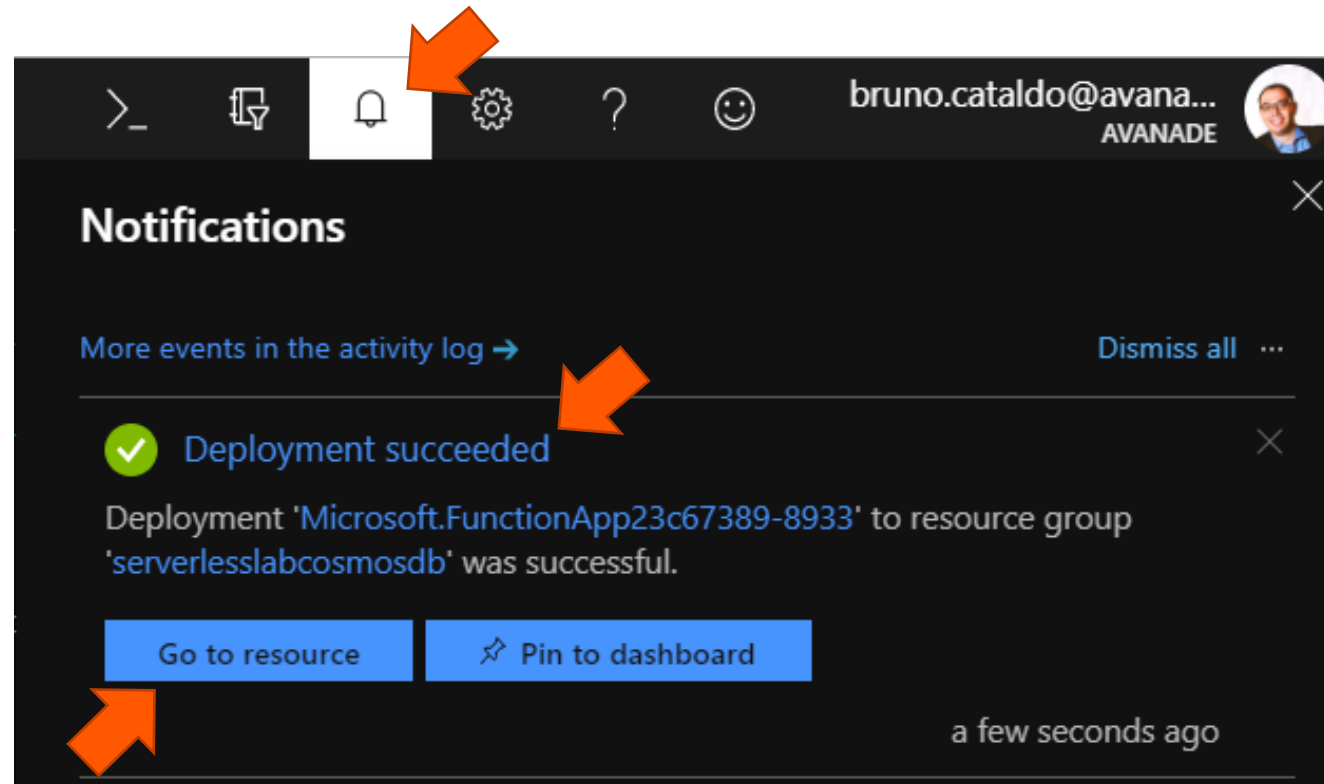


Application Insights serverlesslabcosmosdb >

Create Automation options

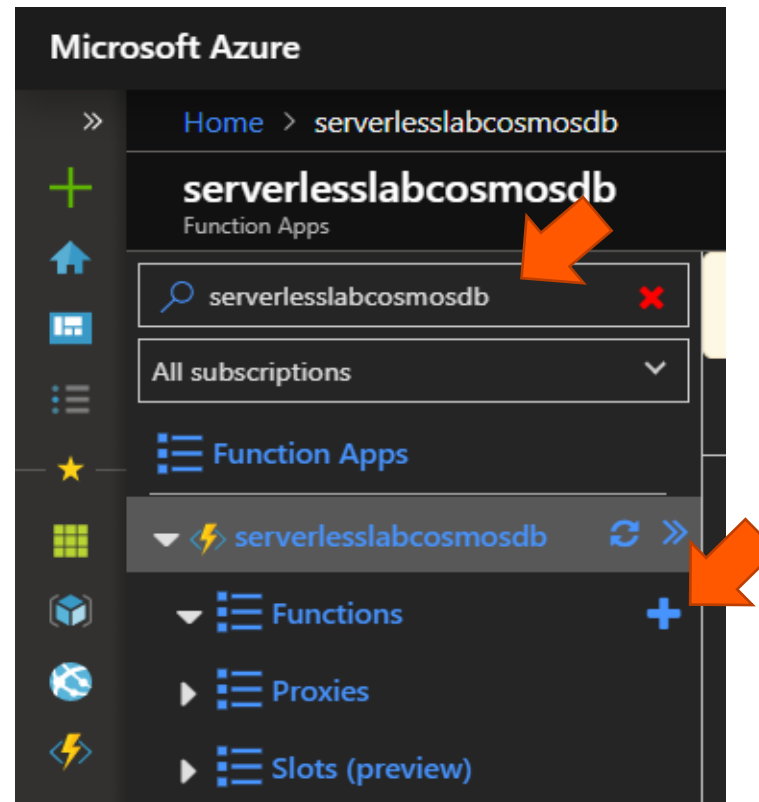
Create an Azure Function | App

Wait for the portal page notification to display *Deployment succeeded*. Click the *Go to resource* button.



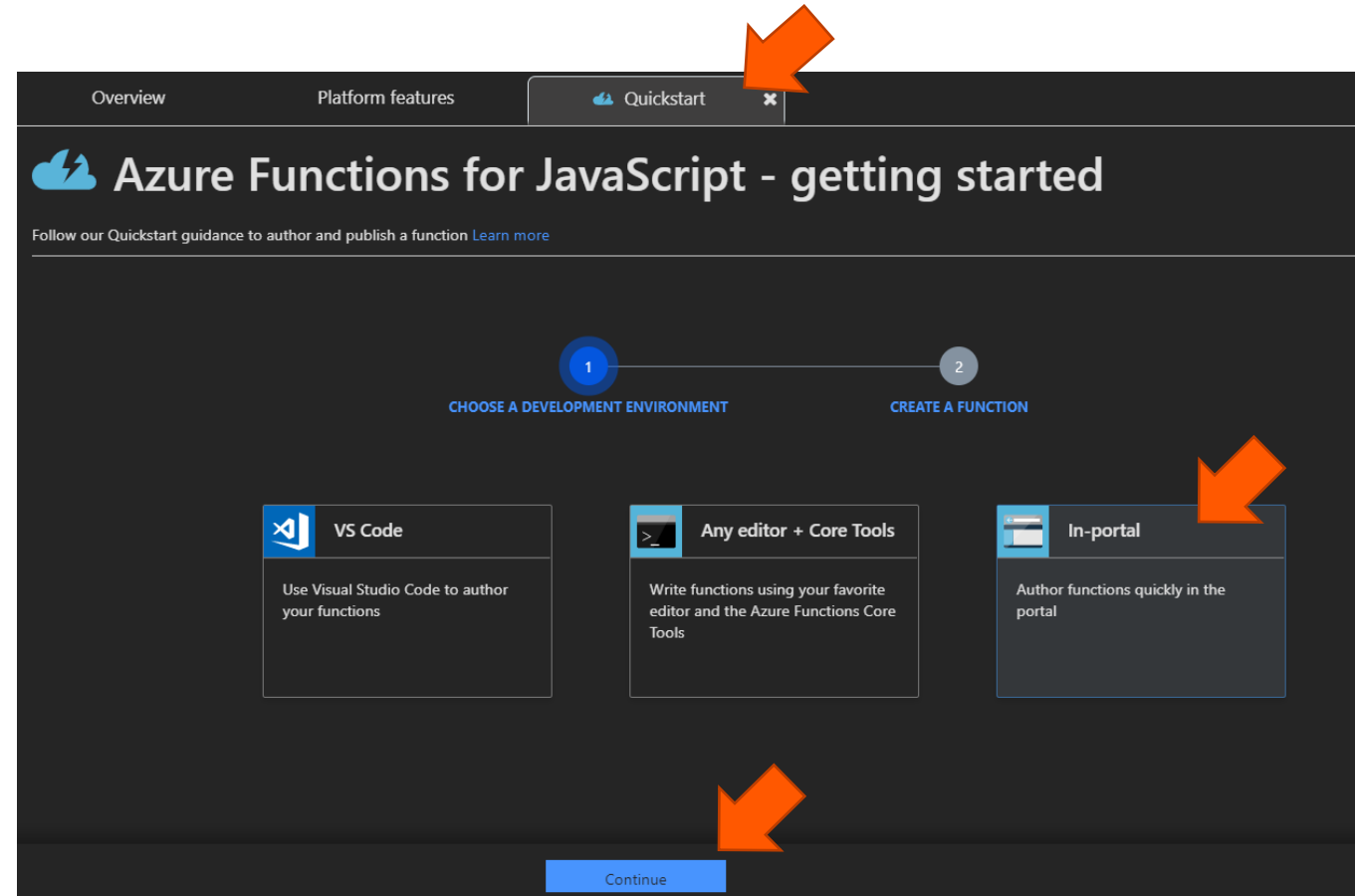
Create an Azure Function | Integrate

In the portal, navigate to the function app you created previously and expand both your function app and your function. After that click on the "+" button.



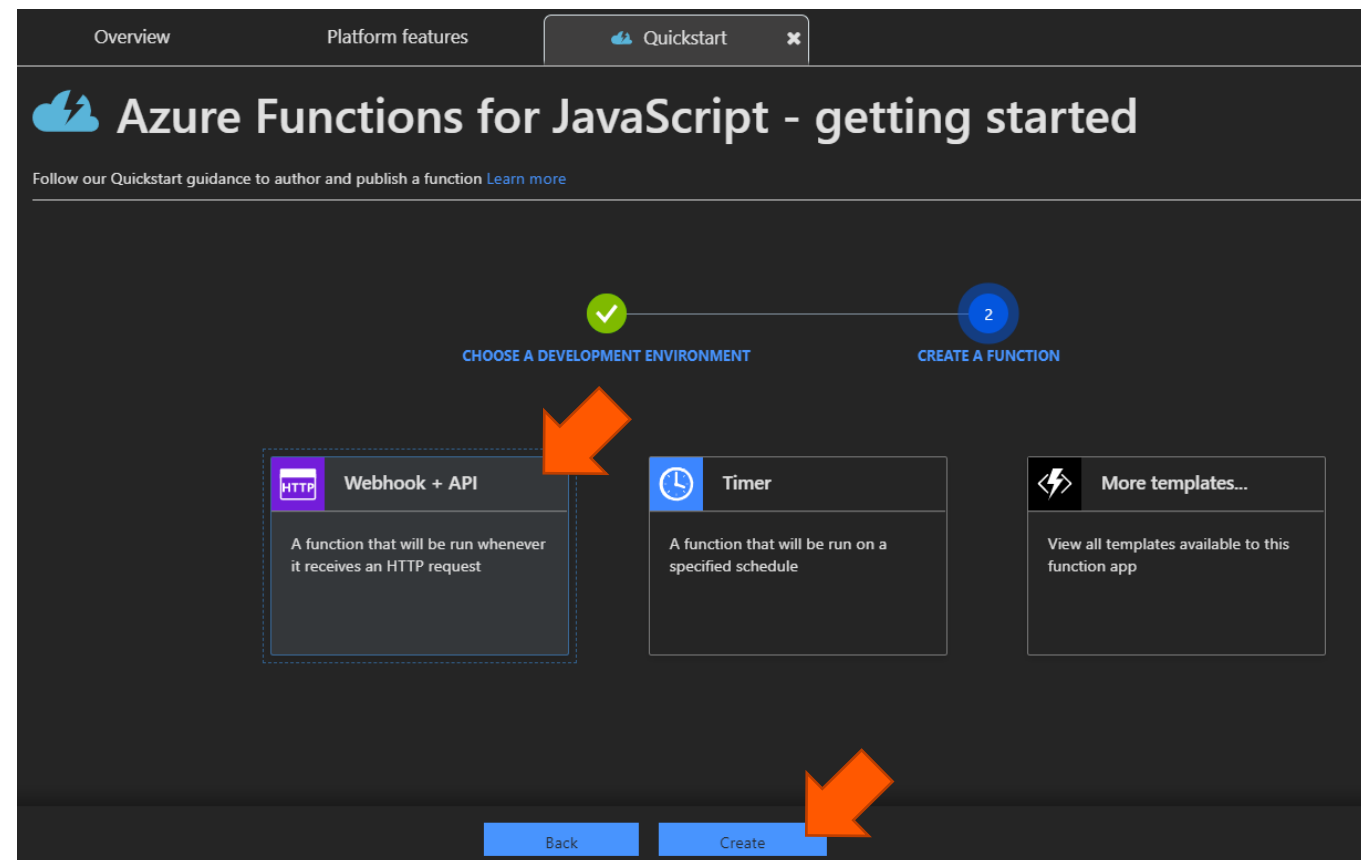
Create an Azure Function | Integrate

Click the *Quickstart* button, select the *In-portal* option and click *Continue* button.



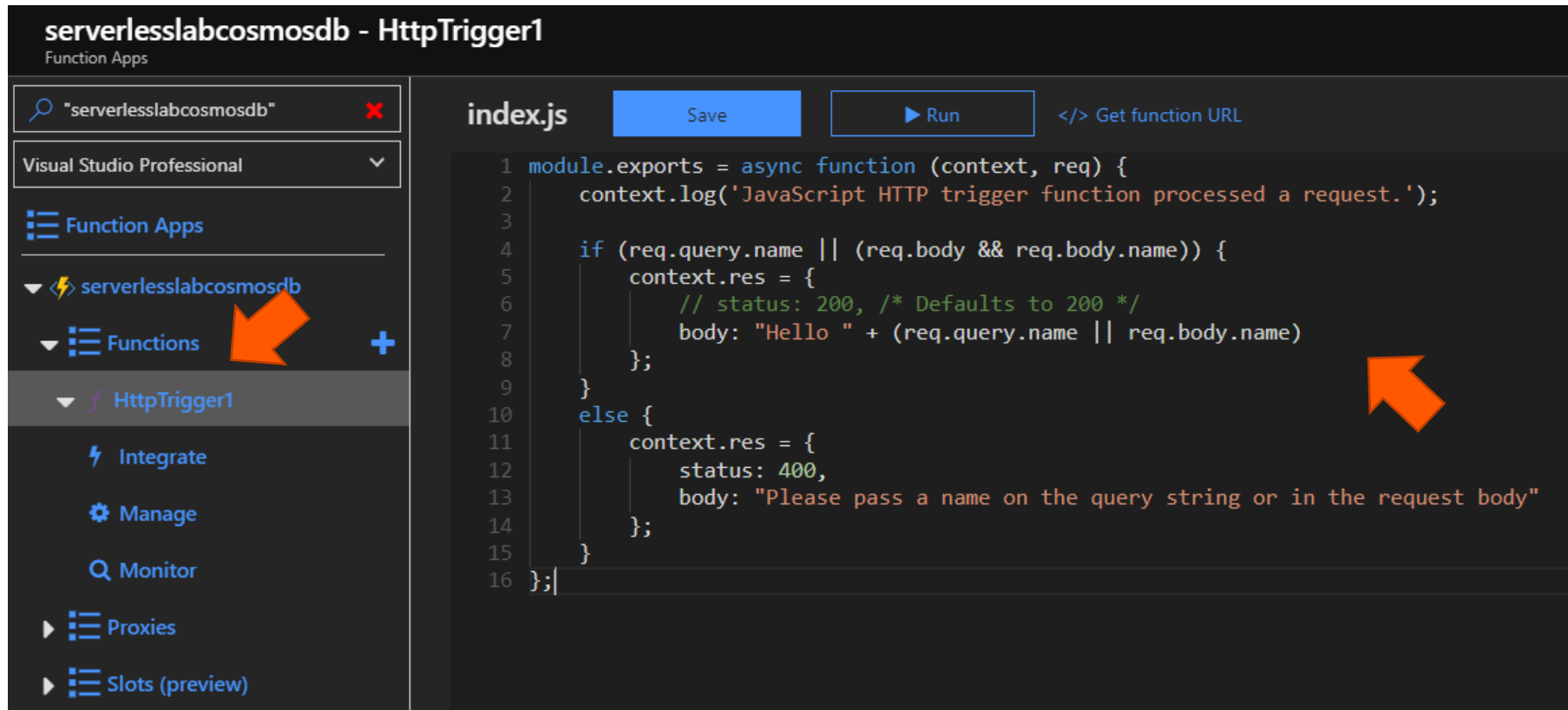
Create an Azure Function | Integrate

Select the *Webhook + API* option, click *Create* button and refresh the page.



Create an Azure Function | Integrate

After page refresh select the *Function* option in the list and *HttpTrigger1* to view your *Azure Function* script.



The screenshot displays the Azure Functions portal interface for a function named 'HttpTrigger1' within the 'serverlesslabcosmosdb' app. The left-hand navigation pane shows the 'Functions' section expanded, with 'HttpTrigger1' selected. An orange arrow points to this selection. Below the function name, there are buttons for 'Integrate', 'Manage', and 'Monitor'. The main area on the right shows the 'index.js' file with its code. At the top of this area are buttons for 'Save', 'Run', and 'Get function URL'. The code in 'index.js' is as follows:

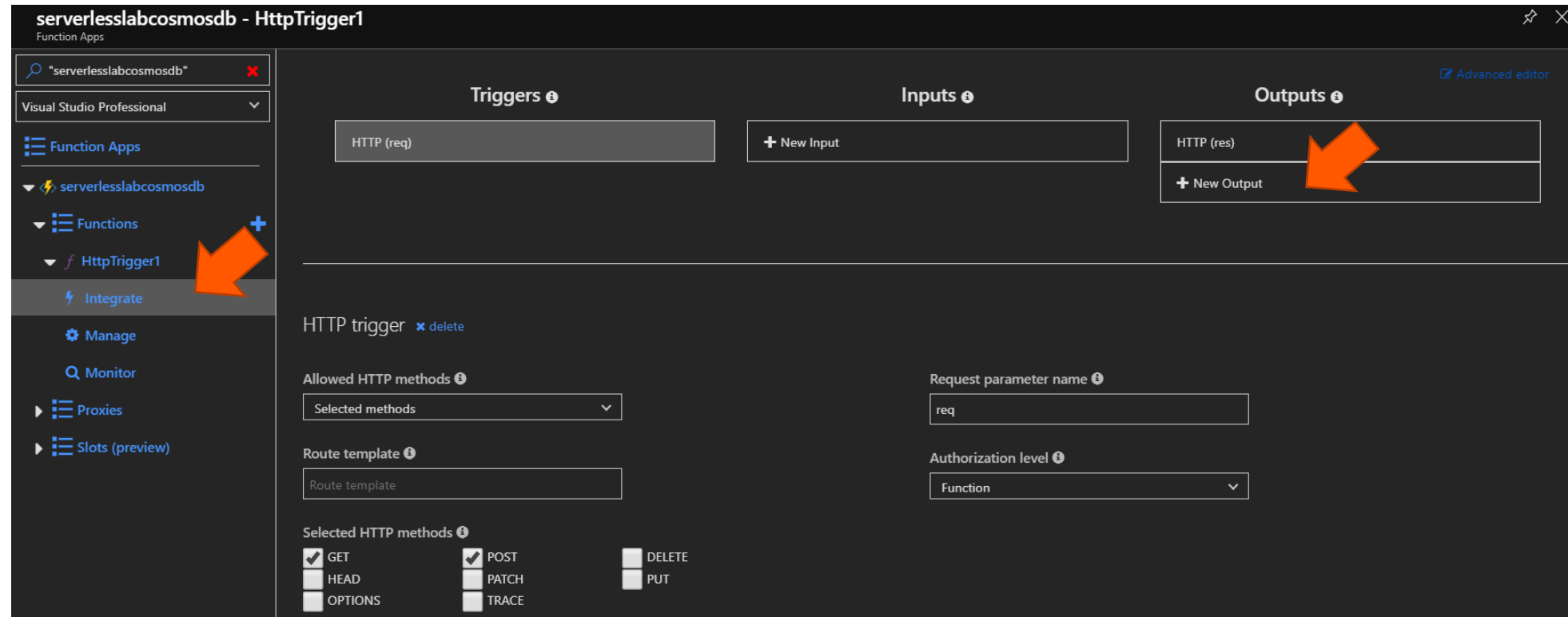
```
1 module.exports = async function (context, req) {
2   context.log('JavaScript HTTP trigger function processed a request.');
```

```
3
4   if (req.query.name || (req.body && req.body.name)) {
5     context.res = {
6       // status: 200, /* Defaults to 200 */
7       body: "Hello " + (req.query.name || req.body.name)
8     };
9   }
10  else {
11    context.res = {
12      status: 400,
13      body: "Please pass a name on the query string or in the request body"
14    };
15  }
16 };
```

An orange arrow points to the closing brace of the function definition on line 16.

Create an Azure Function | Integrate

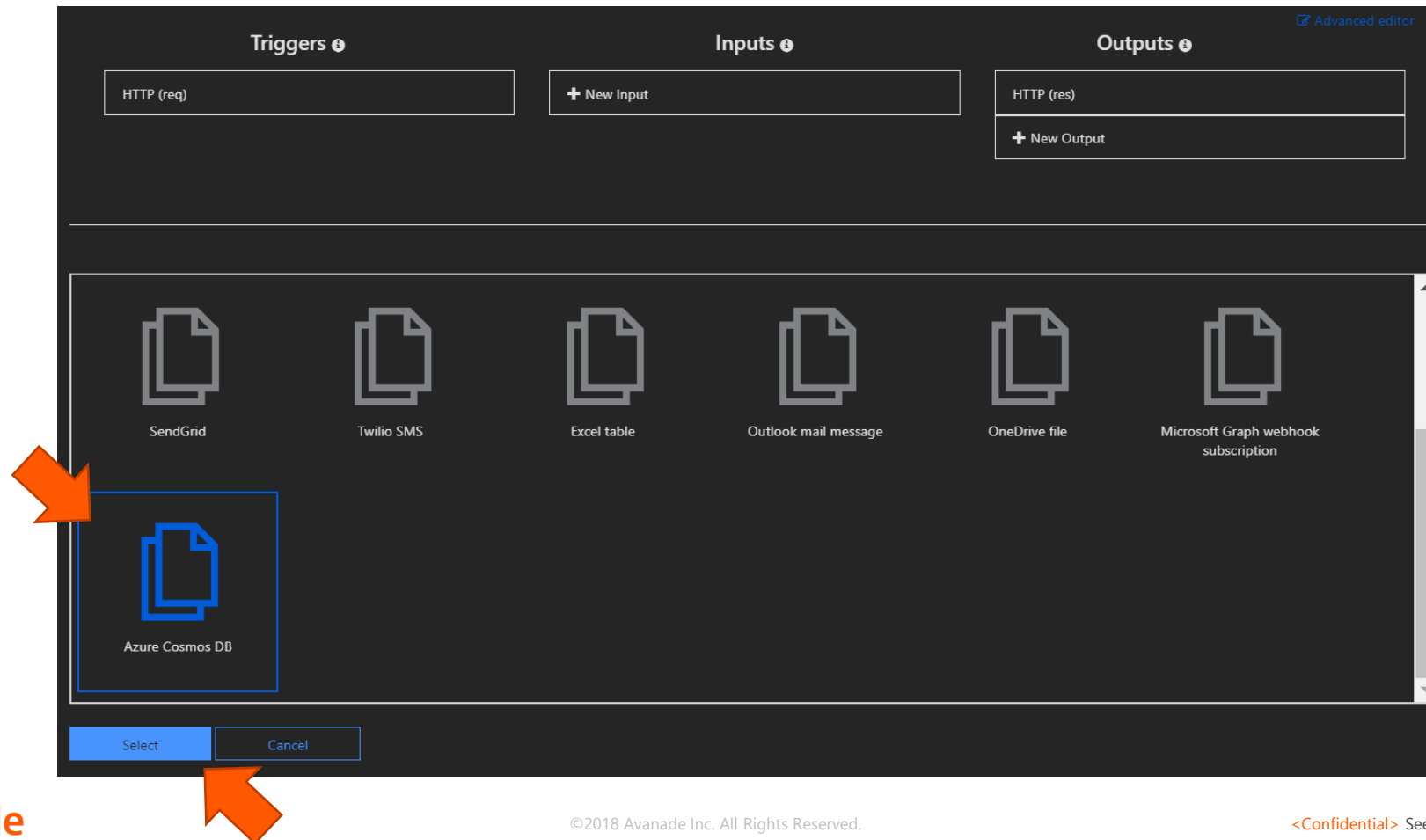
Select the *Integrate* item in the list to navigate to the next page. Click + *New Output* button.



The screenshot displays the Azure Functions portal interface for a function named 'HttpTrigger1' within the 'serverlesslabcosmosdb' Function App. The left-hand navigation pane shows the 'Integrate' tab selected, indicated by an orange arrow. The main content area is divided into three sections: 'Triggers', 'Inputs', and 'Outputs'. The 'Triggers' section shows 'HTTP (req)'. The 'Inputs' section has a '+ New Input' button. The 'Outputs' section shows 'HTTP (res)' and a '+ New Output' button, with an orange arrow pointing to it. Below these sections, the 'HTTP trigger' configuration is visible, including 'Allowed HTTP methods' (set to 'Selected methods'), 'Route template' (set to 'Route template'), 'Request parameter name' (set to 'req'), and 'Authorization level' (set to 'Function'). The 'Selected HTTP methods' section shows checkboxes for GET, POST, DELETE, HEAD, PATCH, PUT, and OPTIONS, with GET and POST checked.

Create an Azure Function | Integrate

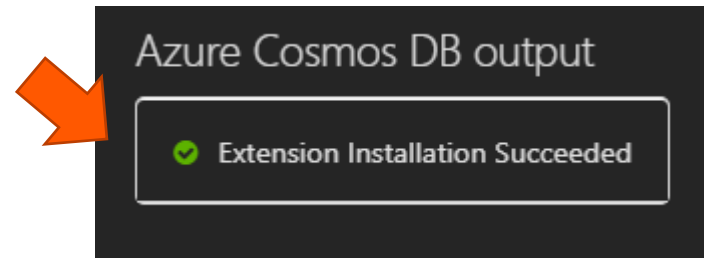
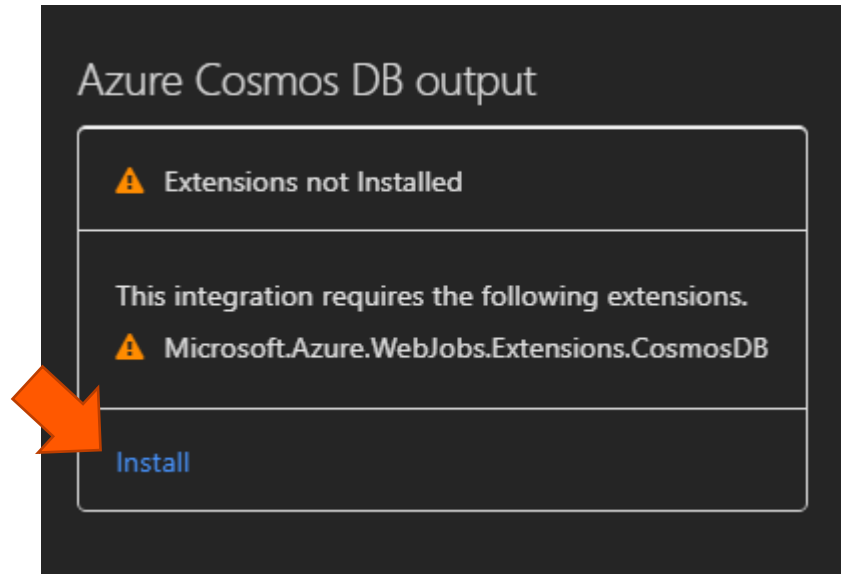
Select the *Azure Cosmos DB* item in the list and click *Select* button.



Create an Azure Function | Integrate

After reload page, scroll down page.

If you get an *Extensions* not installed message, choose Install to install the *Azure Cosmos DB* bindings extension in the function app. Installation may take a minute or two.



Create an Azure Function | Integrate

Use the *Azure Cosmos DB output settings* as specified in the table

Azure Cosmos DB output

✓ Extension Installation Succeeded

Document parameter name ⓘ
taskDocument

☐ Use function return value

Collection Name ⓘ
taskCollection

Azure Cosmos DB account connection ⓘ show value
[Redacted] new

Collection throughput (optional) ⓘ
Collection throughput (optional)

Database name ⓘ
taskDatabase

If true, creates the Azure Cosmos DB database and collection ⓘ
☒

Partition key (optional) ⓘ
Partition key (optional)

Save Cancel

Detailed description: This is a screenshot of the 'Azure Cosmos DB output' configuration window in a dark-themed application. At the top, a green box indicates 'Extension Installation Succeeded'. The window is divided into two main sections. The left section contains fields for 'Document parameter name' (set to 'taskDocument'), a checkbox for 'Use function return value' (unchecked), 'Collection Name' (set to 'taskCollection'), 'Azure Cosmos DB account connection' (with a redacted value and a 'new' button), and 'Collection throughput (optional)'. The right section contains 'Database name' (set to 'taskDatabase'), a checkbox for 'If true, creates the Azure Cosmos DB database and collection' (checked), and 'Partition key (optional)'. Four orange arrows point to the 'Document parameter name', 'Collection Name', 'Database name', and the 'If true...' checkbox. At the bottom are 'Save' and 'Cancel' buttons.

Create an Azure Function | Integrate

Select *New* to create an application setting for your account connection. Click *Select* button.

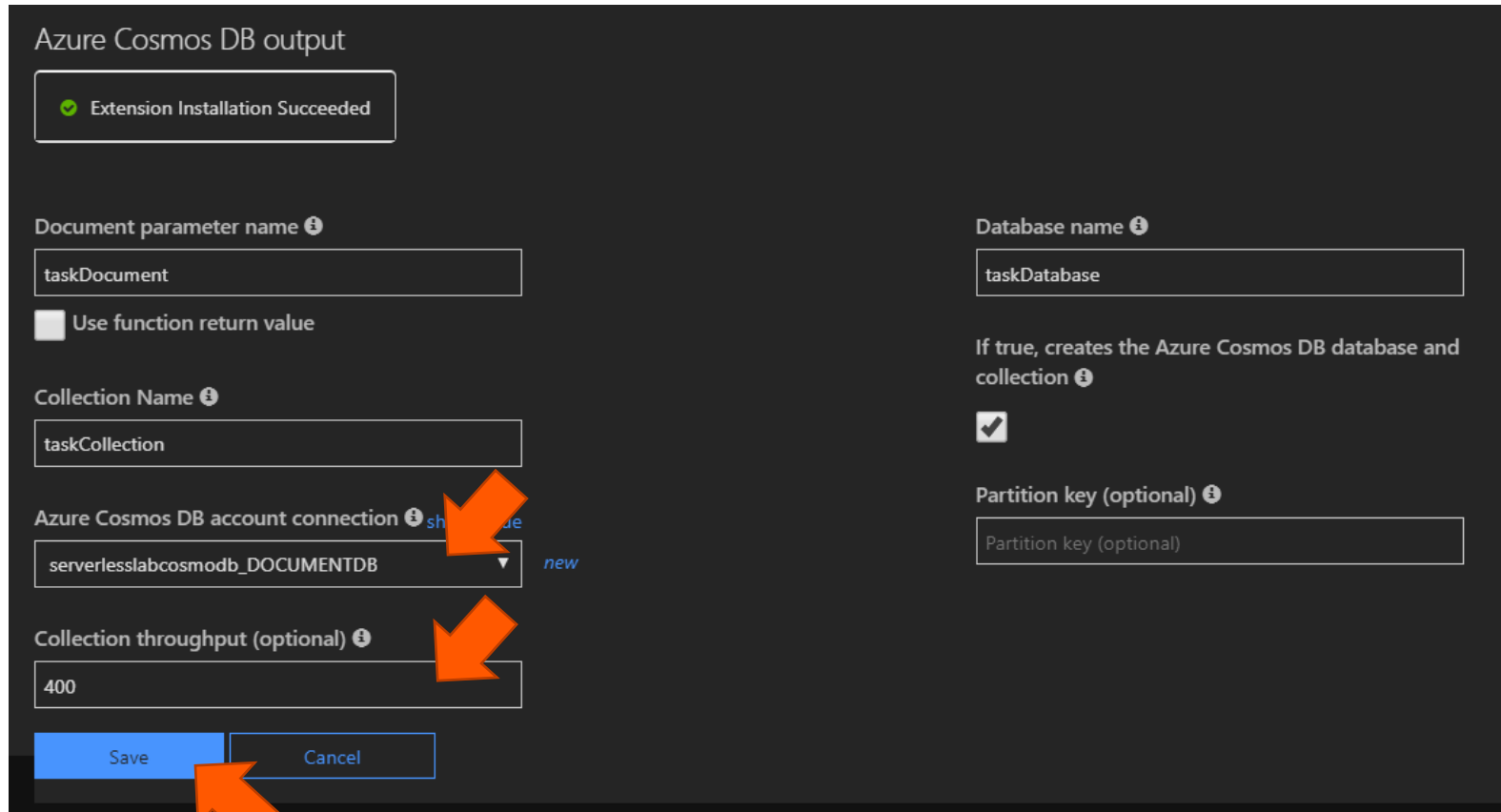
The main interface is titled "Azure Cosmos DB output". It features a green checkmark icon and the text "Extension Installation Succeeded". Below this, there are several input fields: "Document parameter name" with the value "taskDocument", "Database name" with the value "taskDatabase", "Collection Name" with the value "taskCollection", and "Azure Cosmos DB account connection" with a red exclamation mark icon and the word "new" next to it. There is also a checkbox for "Use function return value" and a field for "Collection throughput (optional)". At the bottom are "Save" and "Cancel" buttons.

An inset dialog box titled "Connection" is overlaid on the right side. It contains two tabs: "Azure Cosmos DB account" (selected) and "Custom". Below the tabs are two dropdown menus: "Subscription" with the value "Visual Studio Professional" and "Database Account" with the value "serverlesslabcosmodb". At the bottom of the dialog is a blue "Select" button.

Two orange arrows point to the "new" text in the "Azure Cosmos DB account connection" field and the "Select" button in the "Connection" dialog.

Create an Azure Function | Integrate

After all the configuration has been completed, click the *Save* button.



Azure Cosmos DB output

✓ Extension Installation Succeeded

Document parameter name ⓘ
taskDocument

☐ Use function return value

Collection Name ⓘ
taskCollection

Azure Cosmos DB account connection ⓘ show ⓘ hide
serverlesslabcosmodb_DOCUMENTDB new

Collection throughput (optional) ⓘ
400

Database name ⓘ
taskDatabase

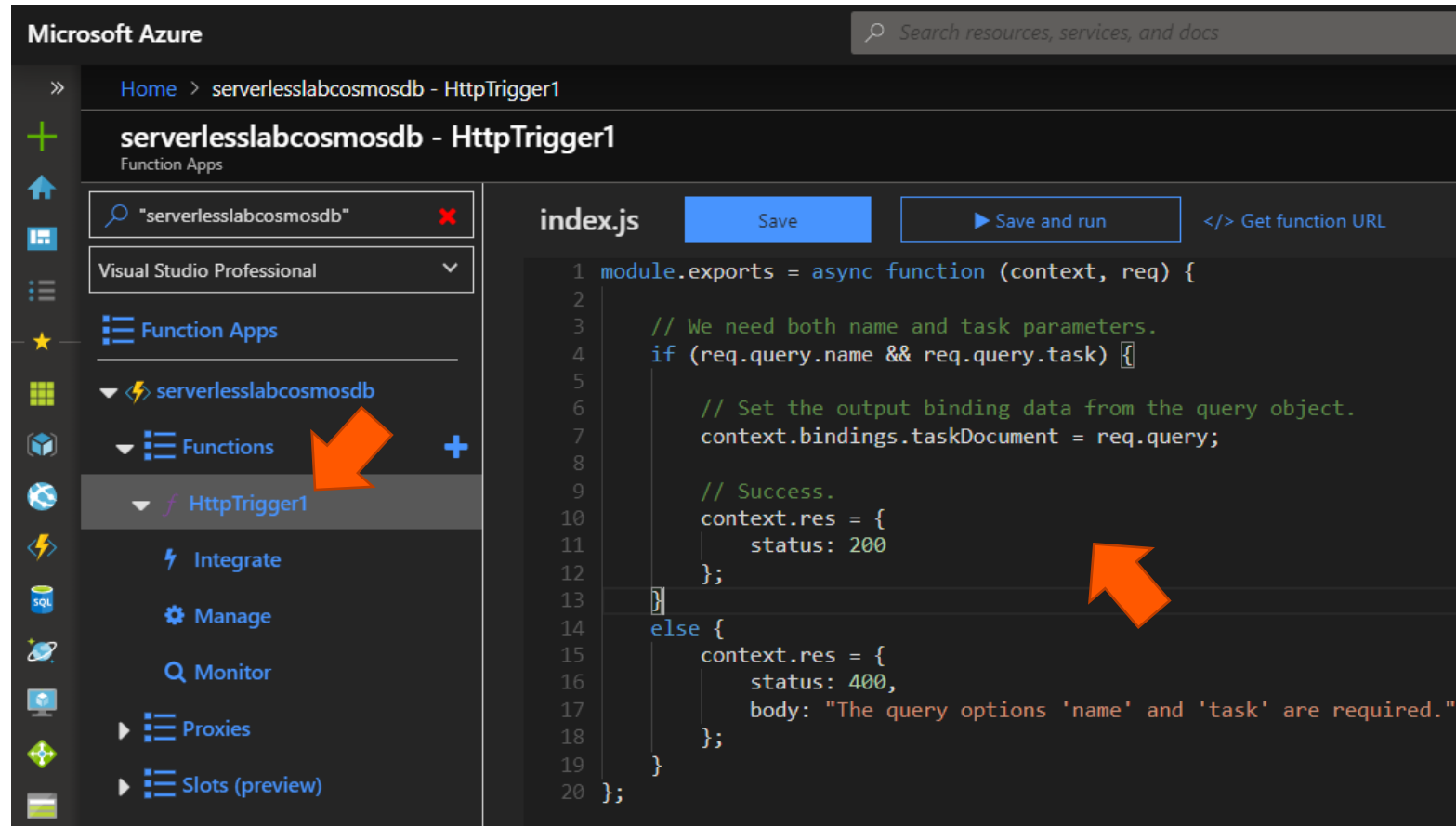
If true, creates the Azure Cosmos DB database and collection ⓘ
☒

Partition key (optional) ⓘ
Partition key (optional)

Save Cancel

Create an Azure Function | Integrate

Click *HttpTrigger1* link and replace the existing *JavaScript* function with the following *code*:



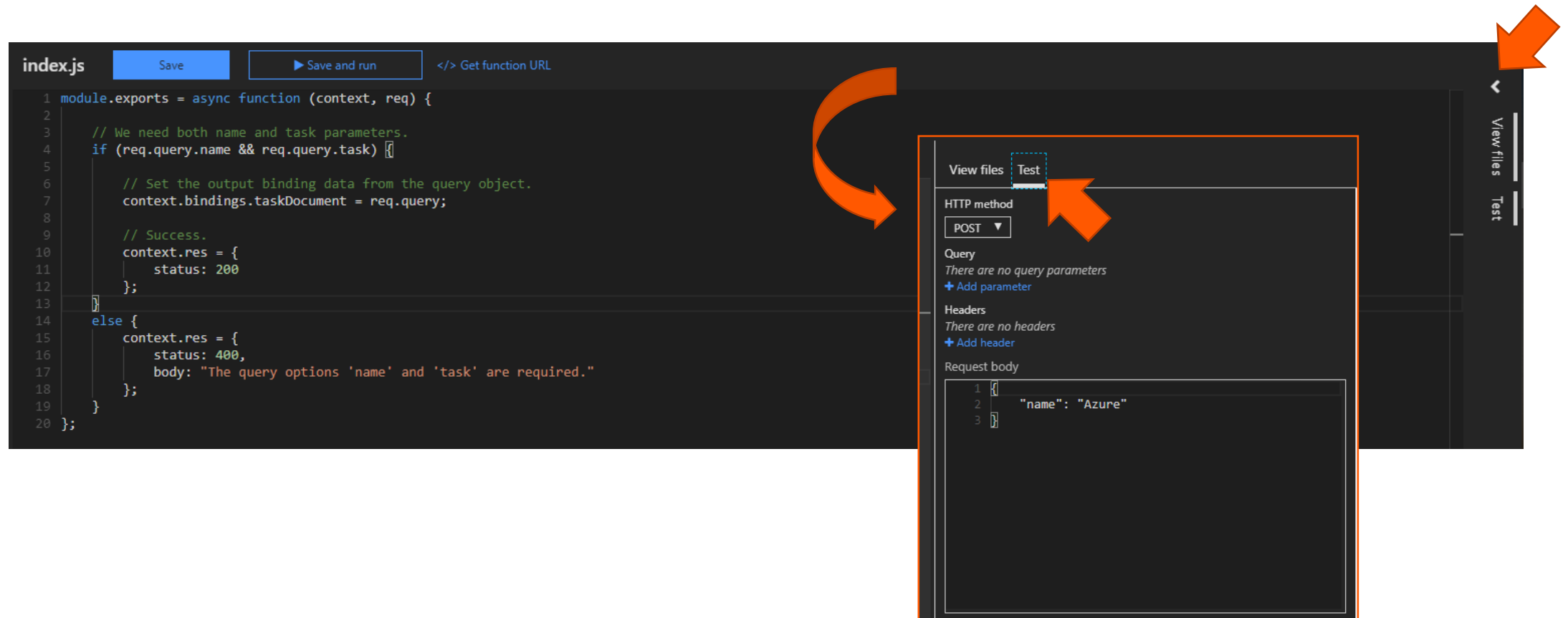
The screenshot shows the Microsoft Azure portal interface. The left sidebar contains a navigation menu with options like Home, Function Apps, Functions, and HttpTrigger1. The 'Functions' section is expanded, and 'HttpTrigger1' is selected, highlighted by an orange arrow. The main area displays the 'index.js' file with the following JavaScript code:

```
1 module.exports = async function (context, req) {
2
3   // We need both name and task parameters.
4   if (req.query.name && req.query.task) {
5
6     // Set the output binding data from the query object.
7     context.bindings.taskDocument = req.query;
8
9     // Success.
10    context.res = {
11      status: 200
12    };
13  }
14  else {
15    context.res = {
16      status: 400,
17      body: "The query options 'name' and 'task' are required."
18    };
19  }
20 }
```

An orange arrow points to the closing curly brace of the 'if' statement in the code.

Test an Azure Function | Integrate

Click on the *right* button to open the panel and click the *Test* tab to open the test panel:



The screenshot displays the Azure Functions portal interface. On the left, the 'index.js' file is open in the code editor, showing an async function that checks for 'name' and 'task' query parameters. The function returns a 200 status if both are present and a 400 status with an error message if they are missing. Above the code editor are buttons for 'Save', 'Save and run', and 'Get function URL'. On the right, the 'Test' tab is selected in the sidebar, and the 'Test' panel is open. The panel shows the HTTP method set to 'POST', and the request body is a JSON object: { "name": "Azure" }. An orange arrow points from the 'right' button in the code editor to the 'Test' tab in the sidebar. Another orange arrow points from the 'Test' tab to the 'Test' panel. A third orange arrow points from the top right corner of the interface to the 'Test' panel.

```
1 module.exports = async function (context, req) {
2
3   // We need both name and task parameters.
4   if (req.query.name && req.query.task) {
5
6     // Set the output binding data from the query object.
7     context.bindings.taskDocument = req.query;
8
9     // Success.
10    context.res = {
11      status: 200
12    };
13  }
14  else {
15    context.res = {
16      status: 400,
17      body: "The query options 'name' and 'task' are required."
18    };
19  }
20 };
```

Test an Azure Function | Integrate

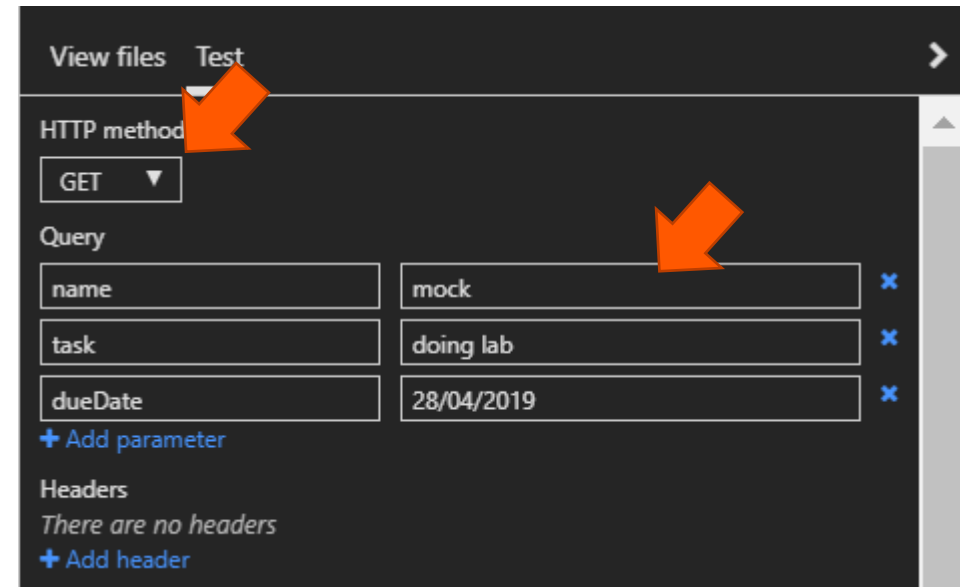
Select *HTTP method* with *GET* option.

Use the *Query fields* to fill the json request that going to be used to add an item and test.

Add 3 fields like image beside.

Click *Save and run* button.

* You can use a *Request body* to fill writing too.



View files Test

HTTP method GET

Query

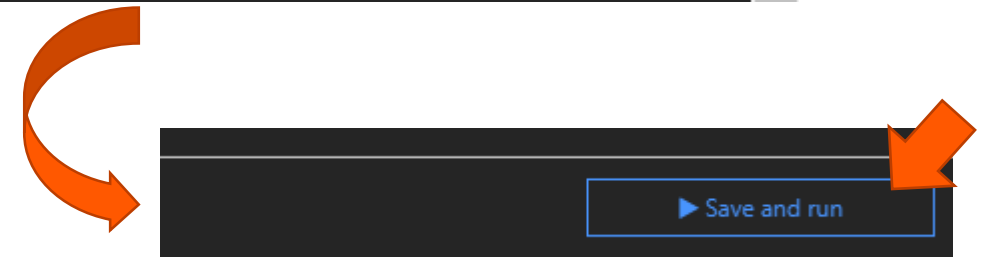
name	mock	×
task	doing lab	×
dueDate	28/04/2019	×

+ Add parameter

Headers

There are no headers

+ Add header



Test an Azure Function | Integrate

Wait for the test page to display *Status: 200 OK* in the output console (right side). For more details check the *Logs* and *Console* panel.



The screenshot displays the Azure Functions test interface. On the left, the 'Logs' and 'Console' panels are visible, with an orange arrow pointing to the 'Logs' tab. The logs show the following information:

```
2019-04-27T03:34:35 No new trace in the past 11 min(s).  
2019-04-27T03:35:11.085 [Information] Executing 'Functions.HttpTrigger1' (Reason='This function was programmatically called via the host APIs.', Id=5caeaecf-f23b-46e0-85b3-93d1920db94f)  
2019-04-27T03:35:13.809 [Information] Executed 'Functions.HttpTrigger1' (Succeeded, Id=5caeaecf-f23b-46e0-85b3-93d1920db94f)  
2019-04-27T03:36:35 No new trace in the past 1 min(s).  
2019-04-27T03:37:35 No new trace in the past 2 min(s).  
2019-04-27T03:38:35 No new trace in the past 3 min(s).  
2019-04-27T03:39:35 No new trace in the past 4 min(s).  
2019-04-27T03:40:35 No new trace in the past 5 min(s).  
2019-04-27T03:41:35 No new trace in the past 6 min(s).  
2019-04-27T03:42:35 No new trace in the past 7 min(s).  
2019-04-27T03:43:35 No new trace in the past 8 min(s).  
2019-04-27T03:44:35 No new trace in the past 9 min(s).
```

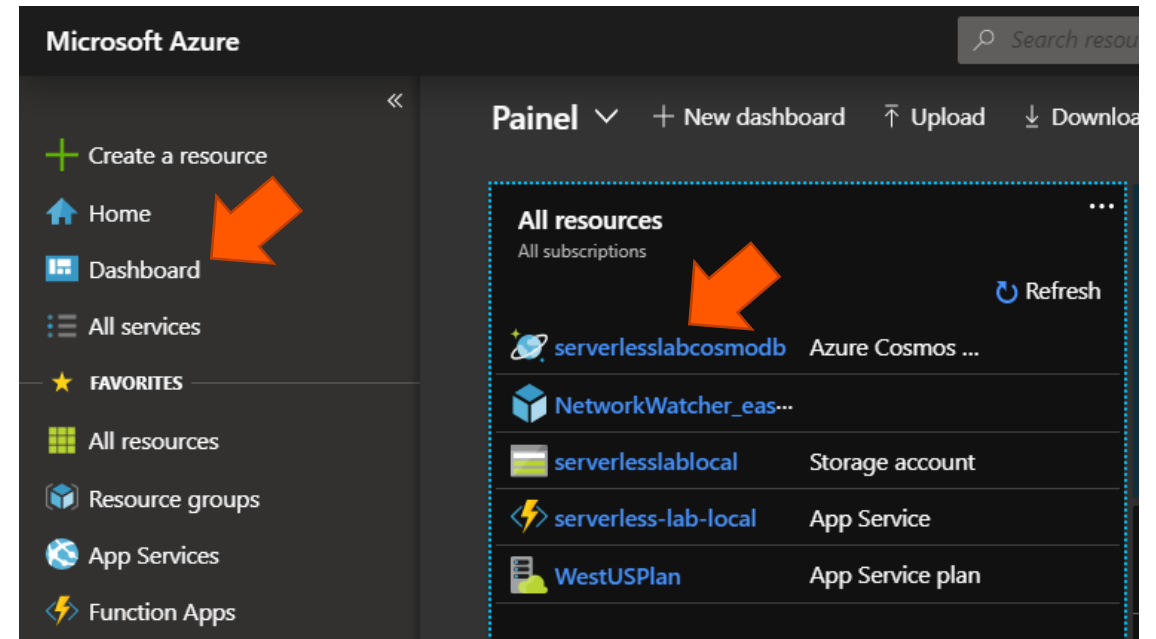
On the right, the 'Output' panel shows the status 'Status: 200 OK' with a green checkmark, indicated by an orange arrow. Below the output panel is a 'Run' button.

Check data on Azure Cosmos DB | Integrate

You must have to access *Dashboard* and find your *Azure Cosmos DB* resource.

Select:

- > *Dashboard*
- > *serverlesslabcosmodb*



Check data on Azure Cosmos DB | Integrate

Select *Data Explorer* item, expand *taskDatabase*, expand *taskCollection*, click on *Documents* and *hash indetificator* like image. Check the data.

The screenshot shows the Azure Cosmos DB Data Explorer interface. On the left, the navigation pane includes 'Overview', 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problem...', 'Quick start', 'Notifications', 'Data Explorer' (highlighted with an orange arrow), and 'Settings'. The 'Data Explorer' section is expanded, showing 'taskDatabase' and 'taskCollection' (both highlighted with orange arrows). Under 'taskCollection', 'Documents' is selected. The main area displays a SQL query 'SELECT * FROM c' and a list of document IDs. One ID is highlighted with an orange arrow. The document view on the right shows a JSON object with fields like 'code', 'name', 'task', 'dueDate', 'id', '_rid', '_self', '_etag', '_attachments', and '_ts'. An orange arrow points to the '_ts' field value '1556336113'.

Test an Azure Function | Integrate + Browser

Click the "</> *Get function URL*" link like the image below.

Using the button to *copy* the URL.

The screenshot displays the Azure Functions portal interface for a function named 'HttpTrigger1' under the 'serverlesslabcosmosdb' app. The left sidebar shows the 'Functions' list with 'HttpTrigger1' selected. The main area shows the 'index.js' file with the function code. The 'Get function URL' link is highlighted with an orange arrow. A modal window titled 'Get function URL' is open, showing the URL 'https://serverless-lab-local.azurewebsites.net/api/HttpTriggerJS' and a 'Copy' button, also highlighted with an orange arrow.

```
index.js
1 module.exports = async function (context, req) {
2   context.log('JavaScript HTTP trigger function processed a request.');
```

Get function URL

Key	URL
default (Function key)	https://serverless-lab-local.azurewebsites.net/api/HttpTriggerJS

Copy

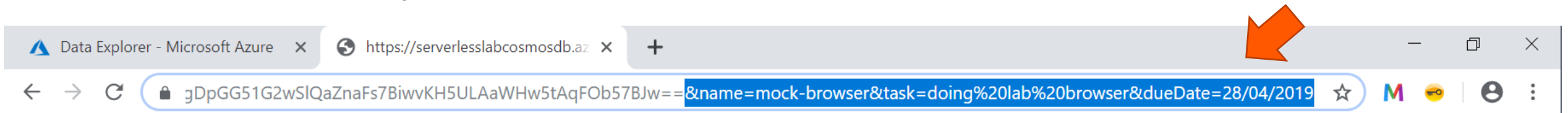
Test an Azure Function | Integrate + Browser

Paste the URL for the HTTP request into your browser's address bar. Append the query string in the end and execute the request.

&name=<yourname>

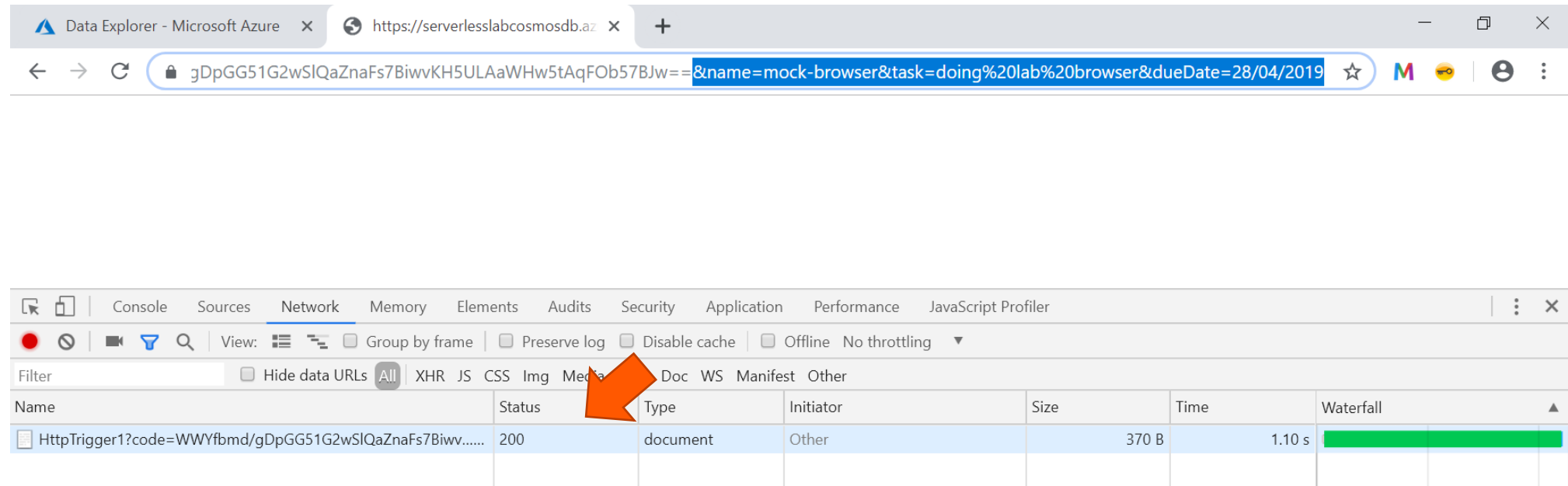
&task=<yourname>

&dueDate=<yourname>



Test an Azure Function | Integrate + Browser

The following shows the response *status 200* in the browser to the *GET* request using *F12* or *Network* console of your browser.

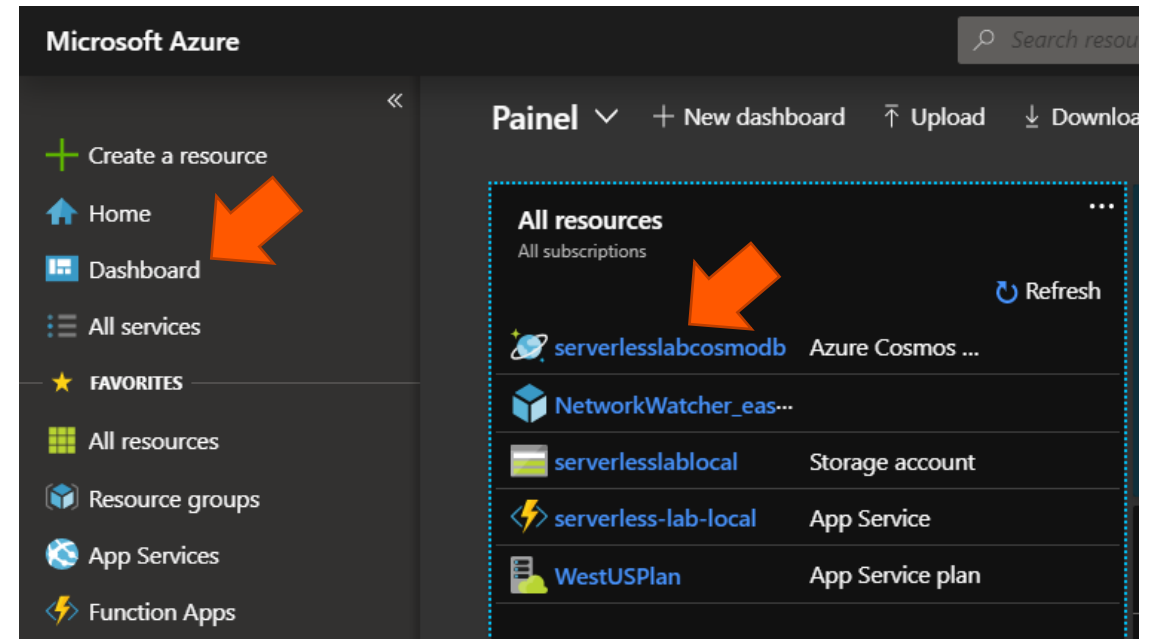


Check data on Azure Cosmos DB | Integrate

You must have to access *Dashboard* and find your *Azure Cosmos DB* resource.

Select:

- > *Dashboard*
- > *serverlesslabcosmodb*



Check data on Azure Cosmos DB | Integrate

Select *Data Explorer* item, expand *taskDatabase*, expand *taskCollection*, click on *Documents* and *hash indetificator* like image. Check the data.

The screenshot shows the Microsoft Azure portal interface for an Azure Cosmos DB account named 'serverlesslabcosmodb'. The left sidebar contains navigation options like 'Create a resource', 'Home', 'Dashboard', 'All services', 'FAVORITES', 'All resources', 'Resource groups', 'App Services', 'Function Apps', 'SQL databases', 'Azure Cosmos DB', 'Virtual machines', and 'Load balancers'. The main area displays the 'Data Explorer' for the 'serverlesslabcosmodb' account. The breadcrumb path is 'Dashboard > serverlesslabcosmodb - Data Explorer'. The 'taskDatabase' is expanded, showing 'taskCollection' and 'Documents'. The 'Documents' tab is selected, displaying a list of documents. A document is selected, showing its JSON content. Orange arrows highlight the navigation path: from the 'Data Explorer' menu item to 'taskCollection', then to 'Documents', and finally to a specific document.

```
SELECT * FROM c
```

id
9c29f29e-de51-4a0e-a493-8b381...
951caab9-bcc0-4563-87f7-d8785...
f0a20fee-a32c-46da-8021-0194b...
d24d0ea4-2491-4776-b9f3-f7349...

```
{
  "code": "WwYfbmd/gDpGG51G2wS1QaZnaFs7BiwvKH5ULAaHw5tAqF0b57BJw==",
  "name": "mock-browser",
  "task": "doing lab browser",
  "dueDate": "28/04/2019",
  "id": "f0a20fee-a32c-46da-8021-0194bd05a53f",
  "_rid": "JF1uAK4+aAMDAAAAAAAAA==",
  "_self": "dbs/JF1uAA==/colls/JF1uAK4+aAM=/docs/JF1uAK4+aAMDAAAAAAAAA==/",
  "_etag": "\"0000571b-0000-0b00-0000-5cc3d62b0000\"",
  "_attachments": "attachments/",
  "_ts": 1556338219
}
```