Edit host and app settings for logic apps in single-tenant Azure Logic Apps

05/25/2021 • 18 minutes to read •

In this article

In *single-tenant* Azure Logic Apps, the *app settings* for a logic app specify the global configuration options that affect *all the workflows* in that logic app. However, these settings apply *only* when these workflows run in your *local development environment*. While running locally, the workflows can access these app settings as *local environment variables*, which are used by local development tools for values that can often change between environments. For example, these values can contain connection strings. When you deploy to Azure, app settings are ignored and aren't included with your deployment.

Your logic app also has *host settings*, which specify the runtime configuration settings and values that apply to *all the workflows* in that logic app, for example, default values for throughput, capacity, data size, and so on, *whether they run locally or in Azure*.

App settings, parameters, and deployment

In *multi-tenant* Azure Logic Apps, deployment depends on Azure Resource Manager templates (ARM templates), which combine and handle resource provisioning for both logic apps and infrastructure. This design poses a challenge when you have to maintain environment variables for logic apps across across various dev, test, and production environments. Everything in an ARM template is defined at deployment. If you need to change just a single variable, you have to redeploy everything.

In *single-tenant* Azure Logic Apps, deployment becomes easier because you can separate resource provisioning between apps and infrastructure. You can use *parameters* to abstract values that might change between environments. By defining parameters to use in your workflows, you can first focus on designing your workflows, and then insert your environment-specific variables later. You can call and reference your environment variables at runtime by using app settings and parameters. That way, you don't have to redeploy as often.

App settings integrate with Azure Key Vault. You can directly reference secure strings, such as connection strings and keys. Similar to Azure Resource Manager templates (ARM templates), where you can define environment variables at deployment time, you can define app settings within your logic app workflow definition. You can then capture dynamically generated infrastructure values, such as connection endpoints, storage strings, and more. However, app settings have size limitations and can't be referenced from certain areas in Azure Logic Apps.

For more information about setting up your logic apps for deployment, see the following documentation:

- Create parameters for values that change in workflows between environments for single-tenant Azure Logic Apps
- DevOps deployment overview for single-tenant based logic apps
- Set up DevOps deployment for single-tenant based logic apps

Visual Studio Code project structure

In Visual Studio Code, your logic app project has either of the following types:

- Extension bundle-based (Node.js), which is the default type
- NuGet package-based (.NET), which you can convert from the default type

Based on these types, your project includes slightly different folders and files. A NuGet-based project includes a .bin folder that contains packages and other library files. A bundle-based project doesn't include the .bin folder and other files. Some scenarios require a NuGet-based project for your app to run, for example, when you want to develop and run custom built-in operations. For more information about converting your project to use NuGet, review Enable built-connector authoring.

For the default bundle-based project, your project has a folder and file structure that is similar to the following example:

text Copy

```
MyBundleBasedLogicAppProjectName
| .vscode
| Artifacts
  || Maps
     ||| MapName1
     ||| ...
  || Schemas
     ||| SchemaName1
     ||| ...
| WorkflowName1
  || workflow.json
  || ...
| WorkflowName2
  || workflow.json
  || ...
| workflow-designtime
| .funcignore
| connections.json
| host.json
| local.settings.json
```

At your project's root level, you can find the following files and folders with other items:

Name	Folder or file	Description
.vscode	Folder	Contains Visual Studio Code-related settings files, such as extensions.json, launch.json, settings.json, and tasks.json files.
Artifacts	Folder	Contains integration account artifacts that you define and use in workflows that support business-to-business (B2B) scenarios. For example, the example structure includes maps and schemas for XML transform and validation operations.
	Folder	For each workflow, the folder includes a workflow.json file, which contains that workflow's underlying JSON definition.
workflow-designtime	Folder	Contains development environment-related settings files.
.funcignore	File	Contains information related to your installed Azure Functions Core Tools.

Name	Folder or file	Description
connections.json	File	Contains the metadata, endpoints, and keys for any managed connections and Azure functions that your workflows use. **Important**: To use different connections and functions for each environment, make sure that you parameterize this **connections.json** file and update the endpoints.
host.json	File	Contains runtime-specific configuration settings and values, for example, the default limits for the single-tenant Azure Logic Apps platform, logic apps, workflows, triggers, and actions. At your logic app project's root level, the **host.json** metadata file contains the configuration settings and default values that *all workflows* in the same logic app use while running, whether locally or in Azure.
local.settings.json	File	Contains app settings, connection strings, and other settings that your workflows use while running locally. In other words, these settings and values apply *only* when you run your projects in your local development environment. During deployment to Azure, the file and settings are ignored and aren't included with your deployment. This file stores settings and values as *local environment variables* that are used by your local development tools as the `appSettings` values. You can call and reference these environment variables both at runtime and deployment time by using *app settings* and *parameters*. **Important**: The **local.settings.json** file can contain secrets, so make sure that you also exclude this file from your project source control.

Reference for app settings - local.settings.json

In Visual Studio Code, at your logic app project's root level, the **local.settings.json** file contain global configuration options that affect *all workflows* in that logic app while running in your local development environment. When your workflows run locally, these settings are accessed as local environment variables, and their values can often change

between the various environments where you run your workflows. To view and manage these settings, review Manage app settings - local.settings.json.

App settings in Azure Logic Apps work similarly to app settings in Azure Functions or Azure Web Apps. If you've used these other services before, you might already be familiar with app settings. For more information, review App settings reference for Azure Functions and Work with Azure Functions Core Tools - Local settings file.

Setting	Default value	Description
`AzureWebJobsStorage`	None	Sets the connection string for an Azure storage account.
`Workflows.<workflowName>.FlowState`	None	Sets the state for *<workflowName>*.
`Workflows.<workflowName>.RuntimeConfiguration.RetentionInDays`	None	Sets the operation options for *<workflowName>*.
`Workflows.Connection.AuthenticationAudience`	None	Sets the audience for authenticating an Azure-hosted connection.
`Workflows.WebhookRedirectHostUri`	None	Sets the host name to use for webhook callback URLs.
`WEBSITE_LOAD_ROOT_CERTIFICATES`	None	Sets the thumbprints for the root certificates to be trusted.

Manage app settings - local.settings.json

To add, update, or delete app settings, select and review the following sections for Visual Studio Code, Azure portal, Azure CLI, or ARM (Bicep) template. For app settings specific to logic apps, review the reference guide for available app settings - local.settings.json.

Visual Studio Code

Azure portal Azure CLI

To review the app settings for your logic app in Visual Studio Code, follow these steps:

1. In your logic app project, at the root project level, find and open the **local.settings.json** file.

2. In the `Values` object, review the app settings for your logic app.

   For more information about these settings, review the reference guide for available app settings - local.settings.json.

To add an app setting, follow these steps:

1. In the **local.settings.json** file, find the `Values` object.

2. In the `Values` object, add the app setting that you want to apply when running locally in Visual Studio Code. Some settings enable you to add a setting for a specific workflow, for example:

JSON Copy

```
{
   "IsEncrypted": false,
   "Values": {
      "AzureWebJobsStorage": "UseDevelopmentStorage=true",
      "Workflows.WorkflowName1.FlowState" : "Disabled",
      <...>
  }
}
```

Reference for host settings - host.json

In Visual Studio Code, at your logic app project's root level, the **host.json** metadata file contains the runtime settings and default values that apply to *all workflows* in a logic app resource whether running locally or in Azure. To view and manage these settings, review Manage host settings - host.json. You can also find related limits information in the Limits and configuration for Azure Logic Apps documentation.

Job orchestration throughput

These settings affect the throughput and capacity for single-tenant Azure Logic Apps to run workflow operations.

Setting	Default value	Description
`Jobs.BackgroundJobs.DispatchingWorkersPulseInterval`	`00:00:01` (1 sec)	Sets the interval for job dispatchers to poll the job queue when the previous poll returns no jobs. Job dispatchers poll the queue immediately when the previous poll returns a job.
`Jobs.BackgroundJobs.NumWorkersPerProcessorCount`	`192` dispatcher worker instances	Sets the number of *dispatcher worker instances* or *job dispatchers* to have per processor core. This value affects the number of workflow runs per core.
`Jobs.BackgroundJobs.NumPartitionsInJobTriggersQueue`	`1` job queue	Sets the number of job queues monitored by job dispatchers for jobs to process. This value also affects the number of storage partitions where job queues exist.

Setting	Default value	Description
`Jobs.BackgroundJobs.NumPartitionsInJobDefinitionsTable`	`4` job partitions	Sets the number of job partitions in the job definition table. This value controls how much execution throughput is affected by partition storage limits.

Run duration and history

Setting	Default value	Description
`Runtime.FlowRetentionThreshold`	`90.00:00:00` (90 days)	Sets the amount of time to keep workflow run history after a run starts.
`Runtime.Backend.FlowRunTimeout`	`90.00:00:00` (90 days)	Sets the amount of time a workflow can continue running before forcing a timeout. **Important**: Make sure this value is less than or equal to the `Runtime.FlowRetentionThreshold` value. Otherwise, run histories can get deleted before the associated jobs are complete.

Inputs and outputs

Setting	Default value	Description

Setting	Default value	Description
`Runtime.FlowRunActionJob.MaximumActionResultSize`	`209715200` bytes	Sets the maximum size in bytes that the combined inputs and outputs can have in an action.
`Runtime.ContentLink.MaximumContentSizeInBytes`	`104857600` characters	Sets the maximum size in characters that an input or output can have in a trigger or action.

Pagination

Setting	Default value	Description
`Runtime.FlowRunRetryableActionJobCallback.MaximumPageCount`	`1000` pages	When pagination is supported and enabled on an operation, sets the maximum number of pages to return or process at runtime.

Chunking

Setting	Def valu
`Runtime.FlowRunRetryableActionJobCallback.MaximumContentLengthInBytesForPartialContent`	`107` byte

Setting	Def valu
`Runtime.FlowRunRetryableActionJobCallback.MaxChunkSizeInBytes`	5242 byte
`Runtime.FlowRunRetryableActionJobCallback.MaximumRequestCountForPartialContent`	1000 requ

Trigger concurrency

Setting	Default value	Description
`Runtime.Trigger.MaximumRunConcurrency`	`100` runs	Sets the maximum number of concurrent runs that a trigger can start. This value appears in the trigger's concurrency definition.
`Runtime.Trigger.MaximumWaitingRuns`	`200` runs	Sets the maximum number of runs that can wait after concurrent runs meet the maximum. This value appears in the trigger's concurrency definition.

For each loops

Setting	Default value	Description
`Runtime.Backend.FlowDefaultForeachItemsLimit`	`100000` (100K array items)	For a *stateful workflow*, sets the maximum number of array items to process in a `For each` loop.
`Runtime.Backend.Stateless.FlowDefaultForeachItemsLimit`	`100` items	For a *stateless workflow*, sets the maximum number of array items to process in a `For each` loop.

Setting	Default value	Description
`Runtime.Backend.ForeachDefaultDegreeOfParallelism`	`20` iterations	Sets the default number of concurrent iterations, or degree of parallelism, in a `For each` loop. To run sequentially, set the value to `1`.
`Runtime.Backend.FlowDefaultSplitOnItemsLimit`	`100000` (100K array items)	Sets the maximum number of array items to debatch or split into multiple workflow instances based on the `SplitOn` setting.

Until loops

Setting	Default value	Description
`Runtime.Backend.MaximumUntilLimitCount`	`5000` iterations	For a *stateful workflow*, sets the maximum number possible for the `Count` property in an `Until` action.
`Runtime.Backend.Stateless.MaximumUntilLimitCount`	`100` iterations	For a *stateless workflow*, sets the maximum number possible for the `Count` property in an `Until` action.
`Runtime.Backend.Stateless.FlowRunTimeout`	`00:05:00` (5 min)	Sets the maximum wait time for an `Until` loop in a stateless workflow.

Variables

Setting	Default value	Description
`Runtime.Backend.DefaultAppendArrayItemsLimit`	`100000` (100K array items)	Sets the maximum number of items in a variable with the Array type.
`Runtime.Backend.VariableOperation.MaximumVariableSize`	Stateful workflow: `104857600` characters Stateless workflow: `1024` characters	Sets the maximum size in characters for the content that a variable can store.

HTTP operations

Setting	Default value	Description
`Runtime.Backend.HttpOperation.RequestTimeout`	`00:03:45` (3 min and 45 sec)	Sets the request timeout value for HTTP triggers and actions.
`Runtime.Backend.HttpOperation.MaxContentSize`	`104857600` bytes	Sets the maximum request size in bytes for HTTP triggers and actions.
`Runtime.Backend.HttpOperation.DefaultRetryCount`	`4` retries	Sets the default retry count for HTTP triggers and actions.

Setting	Default value	Description
`Runtime.Backend.HttpOperation.DefaultRetryInterval`	`00:00:07` (7 sec)	Sets the default retry interval for HTTP triggers and actions.
`Runtime.Backend.HttpOperation.DefaultRetryMaximumInterval`	`01:00:00` (1 hour)	Sets the maximum retry interval for HTTP triggers and actions.
`Runtime.Backend.HttpOperation.DefaultRetryMinimumInterval`	`00:00:05` (5 sec)	Sets the minimum retry interval for HTTP triggers and actions.

HTTP Webhook operations

Setting	Default value	Description
`Runtime.Backend.HttpWebhookOperation.RequestTimeout`	`00:02:00` (2 min)	Sets the request timeout value for HTTP webhook triggers and actions.
`Runtime.Backend.HttpWebhookOperation.MaxContentSize`	`104857600` bytes	Sets the maximum request size in bytes for HTTP webhook triggers and actions.

Setting	Default value	Description
`Runtime.Backend.HttpWebhookOperation.DefaultRetryCount`	4 retries	Sets the default retry count for HTTP webhook triggers and actions.
`Runtime.Backend.HttpWebhookOperation.DefaultRetryInterval`	`00:00:07` (7 sec)	Sets the default retry interval for HTTP webhook triggers and actions.
`Runtime.Backend.HttpWebhookOperation.DefaultRetryMaximumInterval`	`01:00:00` (1 hour)	Sets the maximum retry interval for HTTP webhook triggers and actions.
`Runtime.Backend.HttpWebhookOperation.DefaultRetryMinimumInterval`	`00:00:05` (5 sec)	Sets the minimum retry interval for HTTP webhook triggers and actions.
`Runtime.Backend.HttpWebhookOperation.DefaultWakeUpInterval`	`01:00:00`	Sets the

Setting	Default value	Description
`Runtime.Backend.HttpWebhookOperation.DefaultWakeupInterval`	`01:00:00` (1 hour)	Sets the default wakeup interval for HTTP webhook trigger and action jobs.

Built-in Azure Functions operations

Setting	Default value	Description
`Runtime.Backend.FunctionOperation.RequestTimeout`	`00:03:45` (3 min and 45 sec)	Sets the request timeout value for Azure Functions actions.
`Runtime.Backend.FunctionOperation.MaxContentSize`	`104857600` bytes	Sets the maximum request size in bytes for Azure Functions actions.
`Runtime.Backend.FunctionOperation.DefaultRetryCount`	`4` retries	Sets the default retry count for Azure Functions actions.

Setting	Default value	Description
`Runtime.Backend.FunctionOperation.DefaultRetryInterval`	`00:00:07` (7 sec)	Sets the default retry interval for Azure Functions actions.
`Runtime.Backend.FunctionOperation.DefaultRetryMaximumInterval`	`01:00:00` (1 hour)	Sets the maximum retry interval for Azure Functions actions.
`Runtime.Backend.FunctionOperation.DefaultRetryMinimumInterval`	`00:00:05` (5 sec)	Sets the minimum retry interval for Azure Functions actions.

Built-in SQL operations

Setting	Default value	Description
`Runtime.ServiceProviders.Sql.QueryExecutionTimeout`	`00:00:30` (30 sec)	Sets the request timeout value for SQL service provider operations.

Built-in Azure Service Bus operations

Setting	Default value	De

Setting	Default value	De
`Runtime.ServiceProviders.ServiceBus.MessageSenderPoolSizePerProcessorCount`	`64` message senders	Set nu Azu Ser me ser pro co in t me ser po

Managed API connector operations

Setting	Default value	Descripti
`Runtime.Backend.ApiConnectionOperation.RequestTimeout`	`00:02:00` (2 min)	Sets the request timeout value for managed API connecto triggers a actions.
`Runtime.Backend.ApiConnectionOperation.MaxContentSize`	`104857600` bytes	Sets the maximum request si

Setting	Default value	Descripti
		in bytes fo managed API connector triggers a actions.
`Runtime.Backend.ApiConnectionOperation.DefaultRetryCount`	`4` retries	Sets the default re count for managed API connector triggers a actions.
`Runtime.Backend.ApiConnectionOperation.DefaultRetryInterval`	`00:00:07` (7 sec)	Sets the default re interval fo managed API connector triggers a actions.
`Runtime.Backend.ApiWebhookOperation.DefaultRetryMaximumInterval`	`01:00:00` (1 day)	Sets the maximum retry interval fo managed API connector webhook triggers a actions.
`Runtime.Backend.ApiConnectionOperation.DefaultRetryMinimumInterval`	`00:00:05` (5 sec)	Sets the minimum retry

Setting	Default value	Descripti
		retry interval fo managed API connector triggers a actions.
`Runtime.Backend.ApiWebhookOperation.DefaultWakeUpInterval`	`01:00:00` (1 day)	Sets the default wakeup interval fo managed API connector webhook trigger an action job

Blob storage

Setting	Default value	Description
`Runtime.ContentStorage.RequestOptionsServerTimeout`	`00:00:30` (30 sec)	Sets the timeout value for blob requests from the Azure Logic Apps runtime.
`Runtime.DataStorage.RequestOptionsMaximumExecutionTime`	`00:02:00` (2 min)	Sets the operation timeout value, including retries, for table and queue storage requests from the Azure Logic Apps runtime.

Setting	Default value	Description
`Runtime.ContentStorage.RequestOptionsDeltaBackoff`	`00:00:02` (2 sec)	Sets the backoff interval between retries sent to blob storage.
`Runtime.ContentStorage.RequestOptionsMaximumAttempts`	`4` retries	Sets the maximum number of retries sent to table and queue storage.

Store content inline or use blobs

Setting	Default value	Des
`Runtime.FlowRunEngine.ForeachMaximumItemsForContentInlining`	`20` items	Wh eac run eac valu sto eith with met tab sto sep

Setting	Default value	Des
`Runtime.FlowRunRetryableActionJobCallback.MaximumPagesForContentInlining`	`20` pages	Sets max num pag stor inlir con tab stor bef stor blo stor
`Runtime.FlowTriggerSplitOnJob.MaximumItemsForContentInlining`	`40` items	Wh `Spl` sett deb arra into mu wor inst eac valu stor

Setting	Default value	Des
		eith with met tab stor sep in b stor Sets num iten stor
`Runtime.ScaleUnit.MaximumCharactersForContentInlining`	`8192` characters	Sets max num ope inpu out cha to s inli tab stor bef stor blo stor

Table and queue storage

Setting	Default value	Description
`Runtime.DataStorage.RequestOptionsServerTimeout`	`00:00:16` (16 sec)	Sets the timeout value for table and queue storage requests from the Azure Logic Apps runtime.

Setting	Default value	Description
`Runtime.DataStorage.RequestOptionsMaximumExecutionTime`	`00:00:45` (45 sec)	Sets the operation timeout value, including retries, for table and queue storage requests from the Azure Logic Apps runtime.
`Runtime.DataStorage.RequestOptionsDeltaBackoff`	`00:00:02` (2 sec)	Sets the backoff interval between retries sent to table and queue storage.
`Runtime.DataStorage.RequestOptionsMaximumAttempts`	`4` retries	Sets the maximum number of retries sent to table and queue storage.

Retry policy for all other operations

Setting	Default value	Description
`Runtime.ScaleMonitor.MaxPollingLatency`	`00:00:30` (30 sec)	Sets the maximum polling latency for runtime scaling.
`Runtime.Backend.Operation.MaximumRetryCount`	`90` retries	Sets the maximum number of retries in the retry policy definition for a workflow operation.
`Runtime.Backend.Operation.MaximumRetryInterval`	`01:00:00:01` (1 day and 1 sec)	Sets the maximum interval in the retry policy definition for a workflow operation.

Setting	Default value	Description
`Runtime.Backend.Operation.MinimumRetryInterval`	`00:00:05` (5 sec)	Sets the minimum interval in the retry policy definition for a workflow operation.

Manage host settings - host.json

You can add, update, or delete host settings, which specify the runtime configuration settings and values that apply to *all the workflows* in that logic app, such as default values for throughput, capacity, data size, and so on, *whether they run locally or in Azure*. For host settings specific to logic apps, review the reference guide for available runtime and deployment settings - host.json.

Visual Studio Code - host.json

To review the host settings for your logic app in Visual Studio Code, follow these steps:

1. In your logic app project, at the root project level, find and open the **host.json** file.
2. In the `extensions` object, under `workflows` and `settings`, review any host settings that were previously added for your logic app. Otherwise, the `extensions` object won't appear in the file.

   For more information about host settings, review the reference guide for available host settings - host.json.

To add a host setting, follow these steps:

1. In the **host.json** file, under the `extensionBundle` object, add the `extensions` object, which includes the `workflow` and `settings` objects, for example:

JSON Copy

```
{
   "version": "2.0",
   "extensionBundle": {
      "id": "Microsoft.Azure.Functions.ExtensionBundle",
```

```
      "version": "[1.*, 2.0.0)"
   },
   "extensions": {
      "workflow": {
         "settings": {
         }
      }
   }
}
```

2. In the `settings` object, add a flat list with the host settings that you want to use for all the workflows in your logic app, whether those workflows run locally or in Azure, for example:

JSON Copy

```
{
   "version": "2.0",
   "extensionBundle": {
      "id": "Microsoft.Azure.Functions.ExtensionBundle",
      "version": "[1.*, 2.0.0)"
   },
   "extensions": {
      "workflow": {
         "settings": {
            "Runtime.Trigger.MaximumWaitingRuns": "100"
         }
      }
   }
}
```

Azure portal - host.json

To review the host settings for your single-tenant based logic app in the Azure portal, follow these steps:

1. In the Azure portal search box, find and open your logic app.
2. On your logic app menu, under **Development Tools**, select **Advanced Tools**.
3. On the **Advanced Tools** page, select **Go**, which opens the **Kudu** environment for your logic app.
4. On the Kudu toolbar, from the **Debug console** menu, select **CMD**.
5. In the Azure portal, stop your logic app.

a. On your logic app menu, select **Overview**.

b. On the **Overview** pane's toolbar, select **Stop**.

6. On your logic app menu, under **Development Tools**, select **Advanced Tools**.

7. On the **Advanced Tools** pane, select **Go**, which opens the Kudu environment for your logic app.

8. On the Kudu toolbar, open the **Debug console** menu, and select **CMD**.

   A console window opens so that you can browse to the **wwwroot** folder using the command prompt. Or, you can browse the directory structure that appears above the console window.

9. Browse along the following path to the **wwwroot** folder: `...\home\site\wwwroot`.

10. Above the console window, in the directory table, next to the **host.json** file, select **Edit**.

11. After the **host.json** file opens, review any host settings that were previously added for your logic app.

    For more information about host settings, review the reference guide for available host settings - host.json.

To add a setting, follow these steps:

1. Before you add or edit settings, stop your logic app in the Azure portal.
   a. On your logic app menu, select **Overview**.
   b. On the **Overview** pane's toolbar, select **Stop**.

2. Return to the **host.json** file. Under the `extensionBundle` object, add the `extensions` object, which includes the `workflow` and `settings` objects, for example:

JSON Copy

```
{
   "version": "2.0",
   "extensionBundle": {
      "id": "Microsoft.Azure.Functions.ExtensionBundle",
      "version": "[1.*, 2.0.0)"
   },
   "extensions": {
      "workflow": {
         "settings": {
```

```
            }
        }
    }
}
```

3. In the `settings` object, add a flat list with the host settings that you want to use for all the workflows in your logic app, whether those workflows run locally or in Azure, for example:

JSON Copy

```
{
   "version": "2.0",
   "extensionBundle": {
      "id": "Microsoft.Azure.Functions.ExtensionBundle",
      "version": "[1.*, 2.0.0)"
   },
   "extensions": {
      "workflow": {
         "settings": {
            "Runtime.Trigger.MaximumWaitingRuns": "100"
         }
      }
   }
}
```

4. When you're done, remember to select **Save**.

5. Now, restart your logic app. Return to your logic app's **Overview** page, and select **Restart**.

Next steps

- Create parameters for values that change in workflows between environments for single-tenant Azure Logic Apps
- Set up DevOps deployment for single-tenant Azure Logic Apps

Is this page helpful?

Yes No

Recommended content

DevOps deployment for single-tenant Azure Logic Apps - Azure Logic Apps

Learn about DevOps deployment for single-tenant Azure Logic Apps.

Create workflows with single-tenant Azure Logic Apps (Standard) in the Azure portal - Azure Logic Apps

Create automated workflows to integrate apps, data, services, and systems with single-tenant Azure Logic Apps (Standard) in the Azure portal.

Set up DevOps for single-tenant Azure Logic Apps - Azure Logic Apps

How to set up DevOps deployment for workflows in single-tenant Azure Logic Apps.

Create workflows with single-tenant Azure Logic Apps (Standard) in Visual Studio Code - Azure Logic Apps

Create automated workflows to integrate apps, data, services, and systems with single-tenant Azure Logic Apps (Standard) in Visual Studio Code.

Show more