

Module 09: Develop App Service Logic Apps





Topics

- Azure logic apps

Lesson 01: Azure logic apps



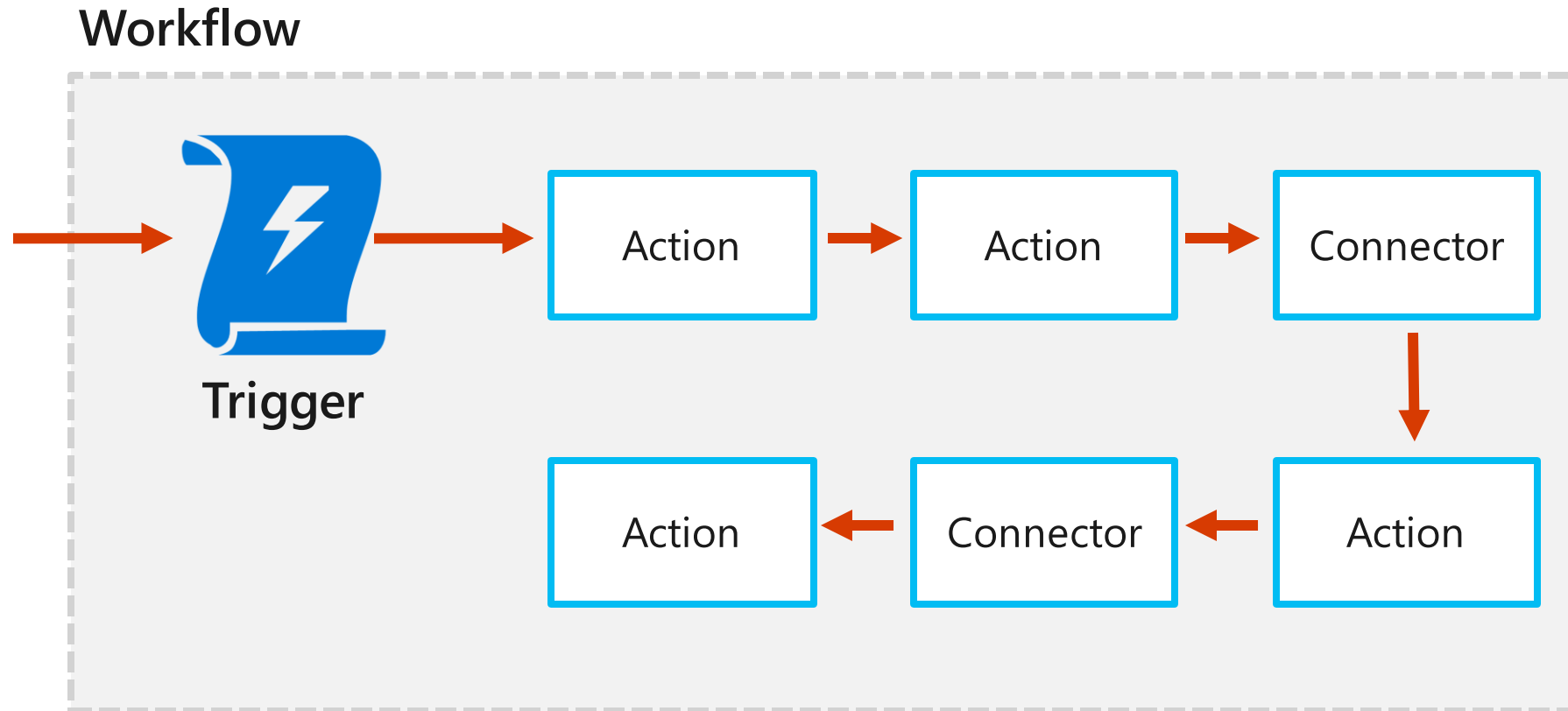
Azure logic apps

- Automation workflow solution:
 - No-code designer for rapid creation of integration solutions
 - Prebuilt templates to simplify getting started
 - Out-of-box support for popular software as a service (SaaS) and on-premises integrations
 - BizTalk APIs available to advanced integration solutions
- JSON-based workflow definition:
 - Can be deployed by using Azure Resource Manager templates

Components

- Workflow
 - The business process described as a series of steps
- Triggers
 - The step that invokes a new workflow instance
- Actions
 - A individual step in a workflow, typically a Connector or custom API app
- Connectors
 - A special case of an API app that is prebuilt and ready to integrate with a specific service or data source. For example:
 - Twitter and SQL Server Connectors

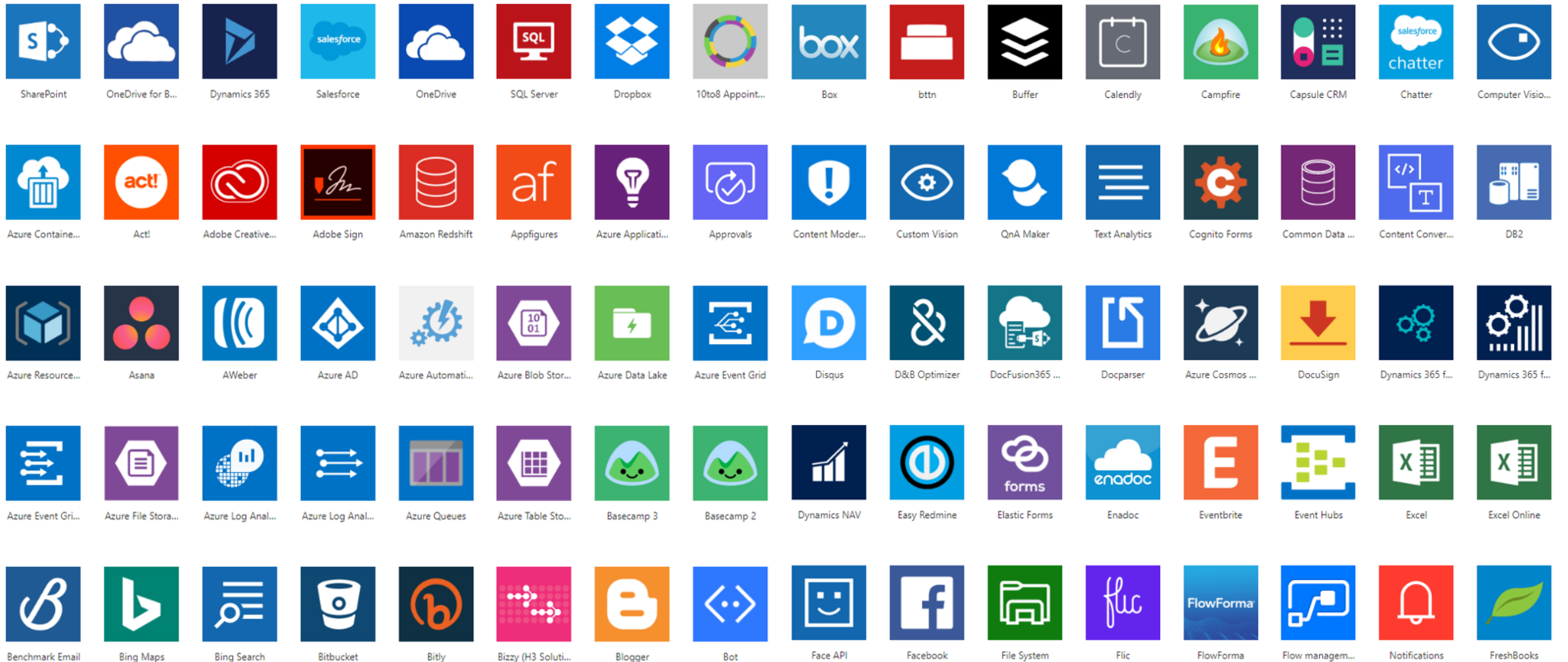
Workflow components



Connectors



Connector ecosystem

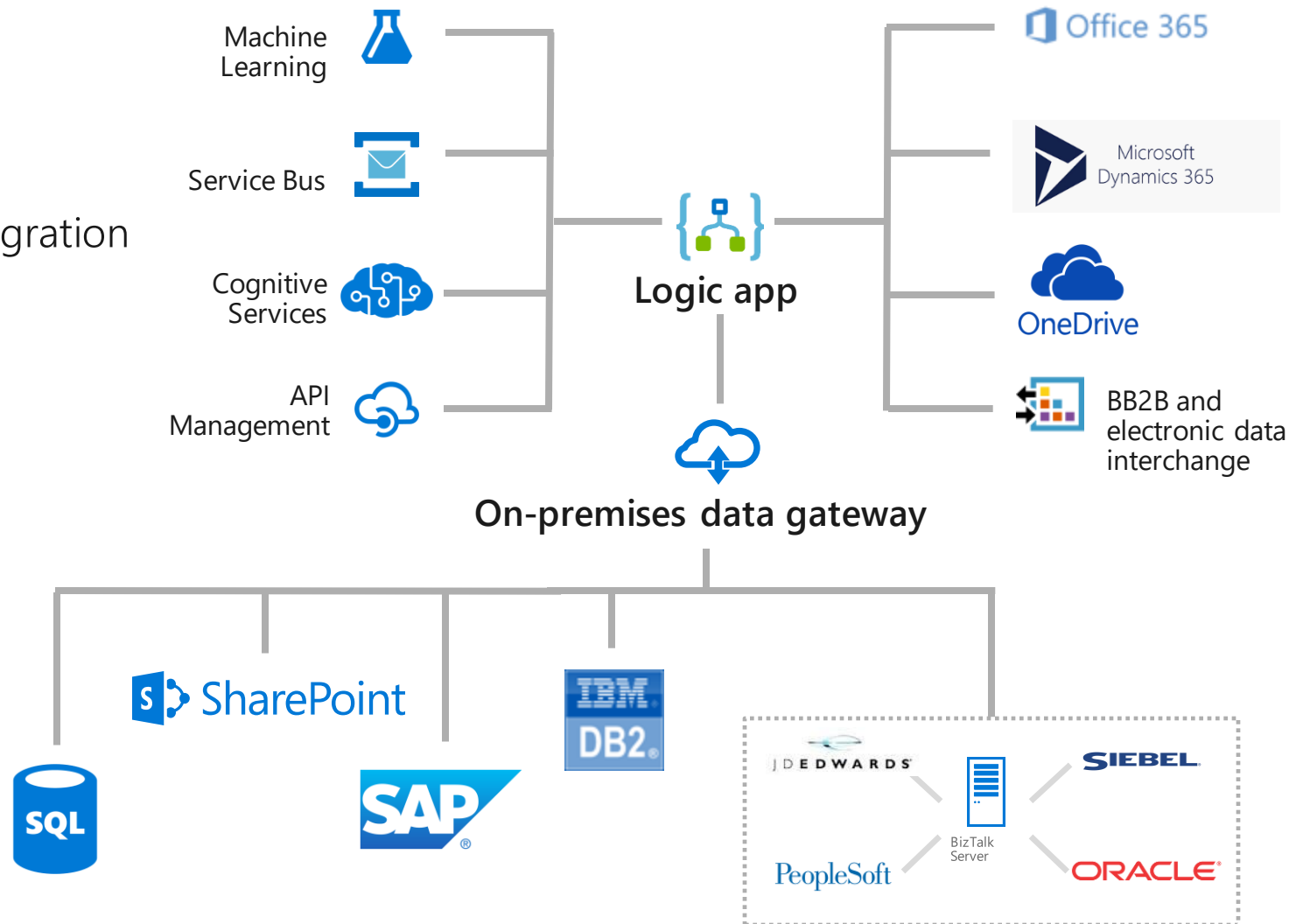


Connector components

- Connectors are composed of
 - Actions
 - Changes directed by a user
 - Triggers
 - Notify your app when events occur
- There are two types of Triggers
 - Polling triggers
 - Push triggers

Hybrid connectivity

- Connect on-premises, hybrid and cloud applications
- Run mission-critical, complex integration scenarios

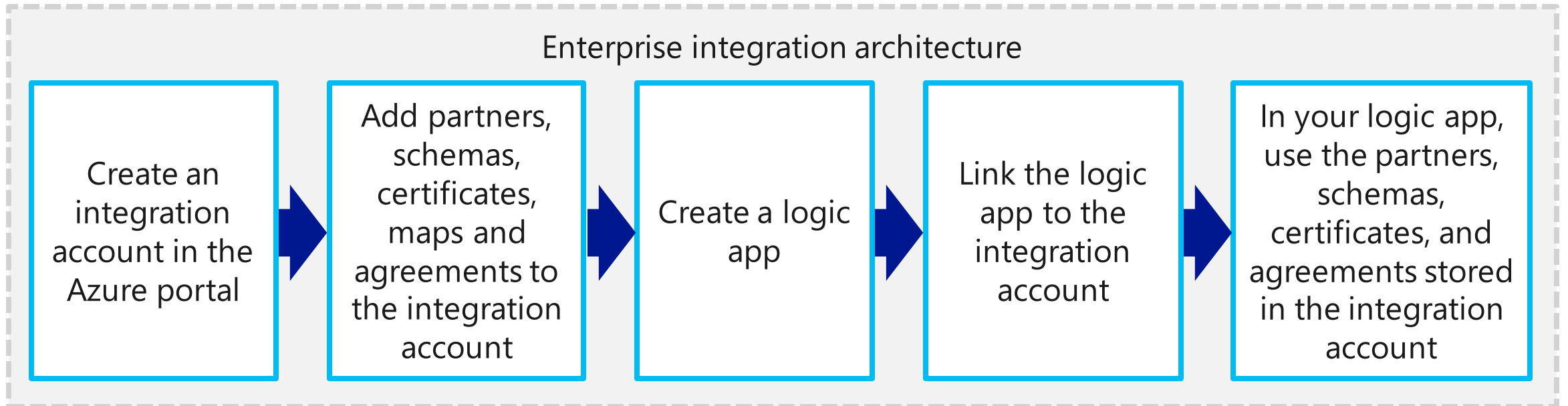


B2B scenarios and the Enterprise Integration Pack

- A special pack that transforms different formats
 - Communicate seamlessly between organizations
 - Secure messages with encryption and digital signatures
 - Based on familiar BizTalk concepts
- Why should you use Enterprise Integration?
 - With enterprise integration, you can store all your artifacts in one place—your integration account
 - You can build B2B workflows and integrate with third-party software as a service (SaaS) apps, on-premises apps, and custom apps by using the Azure logic apps engine and all its connectors
 - You can create custom code for your logic apps with Azure Functions

Enterprise integration steps

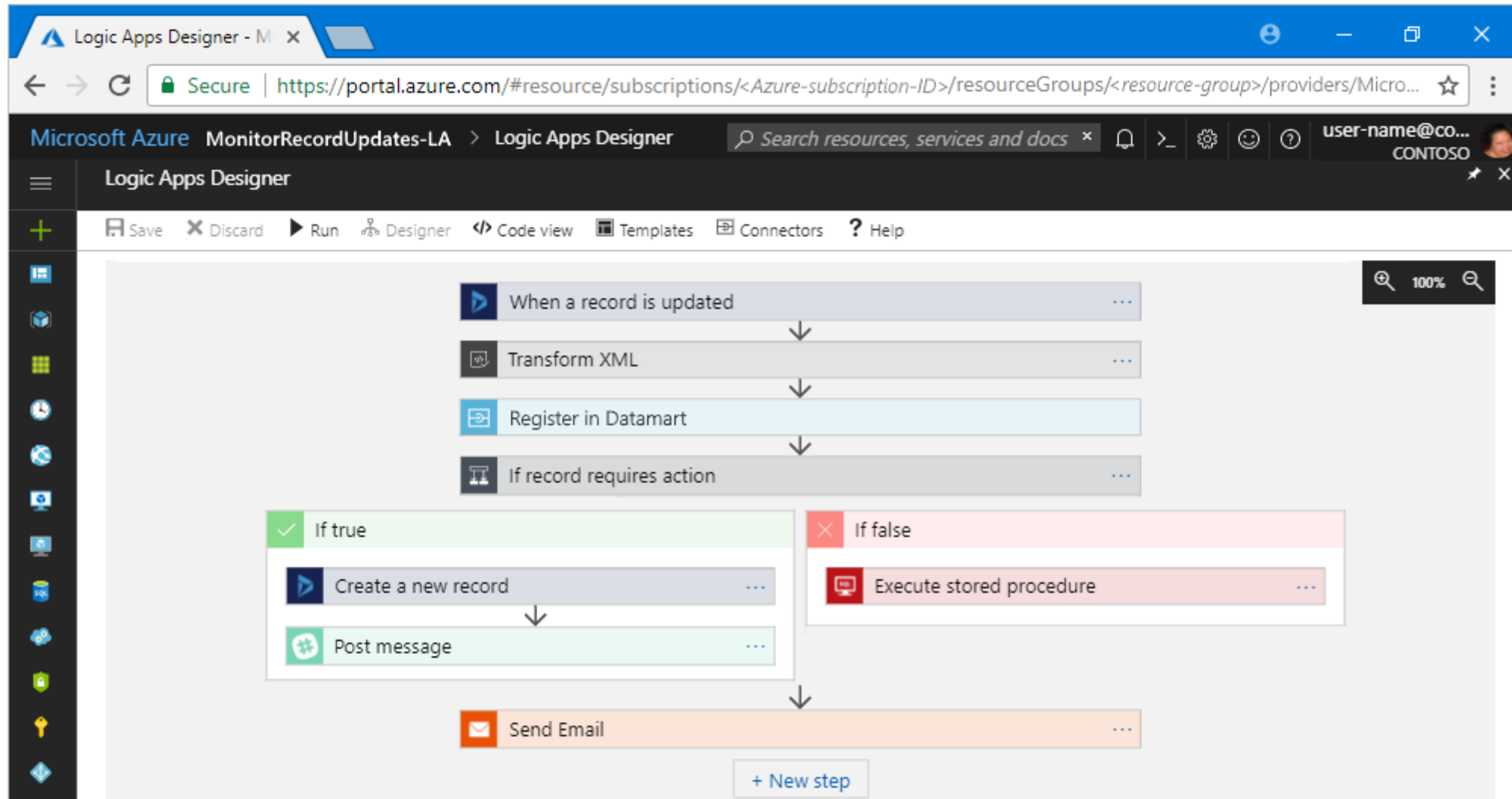
1. Create an integration account in the Azure portal
2. Add partners, schemas, certificates, maps, and agreements to the integration account
3. Create a logic app
4. Link the logic app to the integration account
5. In your logic app, use the components stored in the integration account



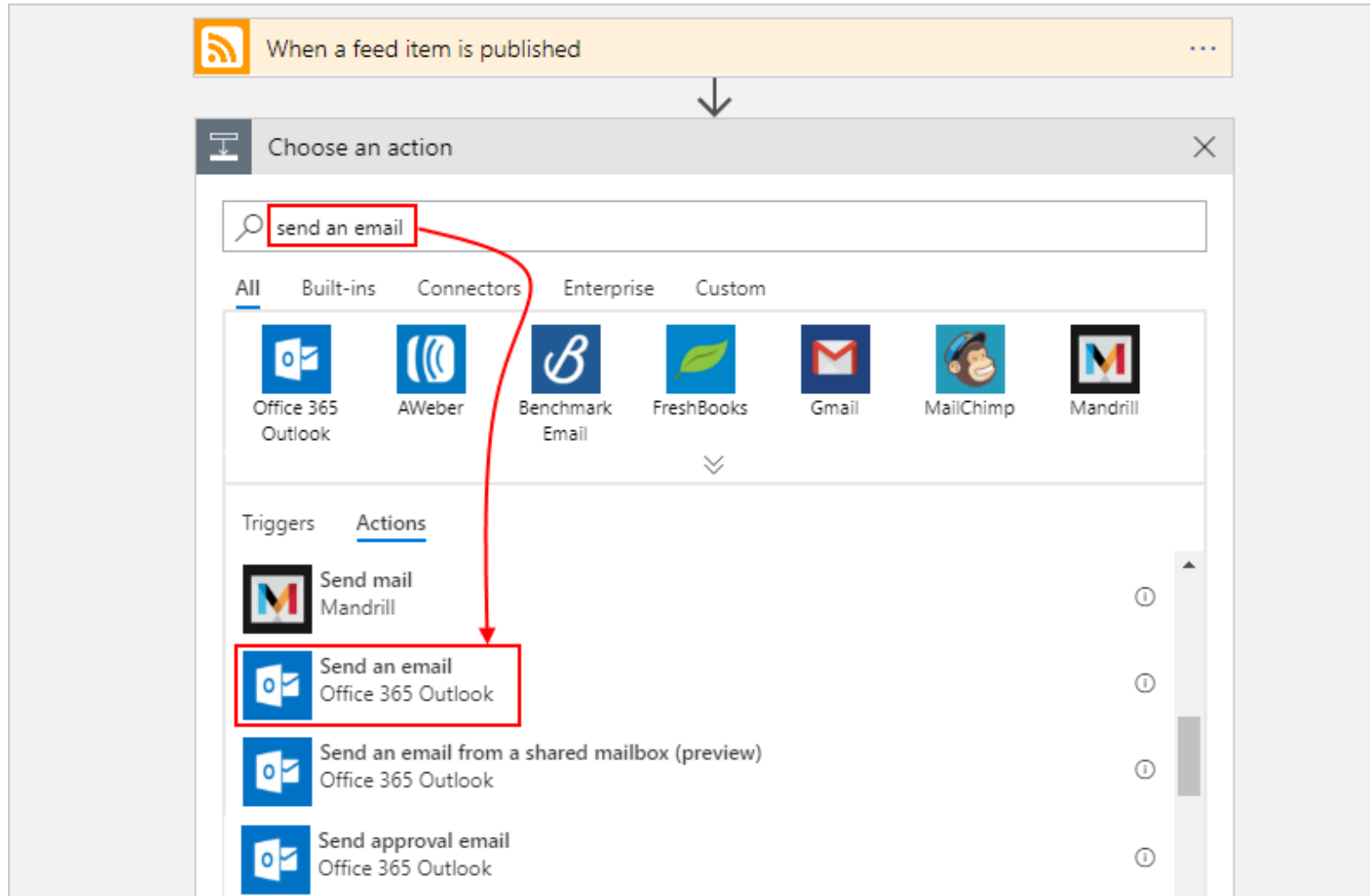
Create logic apps by using Visual Studio

- Save time and simplify complex processes with visual design tools
- Build logic apps from start to finish by using the logic apps Designer
 - Through your browser in the Azure portal
 - Through Visual Studio
- Start workflow with a trigger and actions directly in Visual Studio
- View, edit, and revise templates quickly by using existing code-based tools

logic apps Designer



logic apps Designer – action search



logic apps Designer – action configuration

The screenshot displays the Logic Apps Designer interface. At the top, a trigger action 'When a feed item is published' is shown in an orange box. An arrow points down to the 'Send an email' action, which is highlighted in a blue box. The 'Send an email' action configuration is as follows:

- To:** <your-email-address@domain>
- Subject:** New RSS item: Feed title x
- Body:**
 - Title: Feed title x
 - Date published: Feed publishe... x
 - Link: Primary feed l... x

A red rectangular box highlights the 'Body' section of the configuration.

logic apps Designer – dynamic content

The screenshot displays the Logic Apps Designer interface. At the top, a trigger action 'When a feed item is published' is shown. Below it, the 'Send an email' action is configured. The 'Subject' field contains the text 'New RSS item:'. A red arrow points from the 'Feed title' dynamic content option in the right-hand pane to the end of the 'Subject' field text. The right-hand pane, titled 'Add dynamic content from the apps and connectors used in this flow.', shows a list of dynamic content options under the 'Dynamic content' tab. The options include 'categories - Item', 'Feed copyright information', 'Feed ID', 'Feed published on', 'Feed summary', and 'Feed title'. The 'Feed title' option is highlighted with a red box. The 'Expression' tab is also visible.

When a feed item is published

Send an email

* To: <your-email-address@domain>

* Subject: New RSS item: [Add dynamic content](#)

* Body: Specify the body of the mail

[Show advanced options](#)

Connected to <your-email-address@domain> [Change connection.](#)

Add dynamic content from the apps and connectors used in this flow. Hide

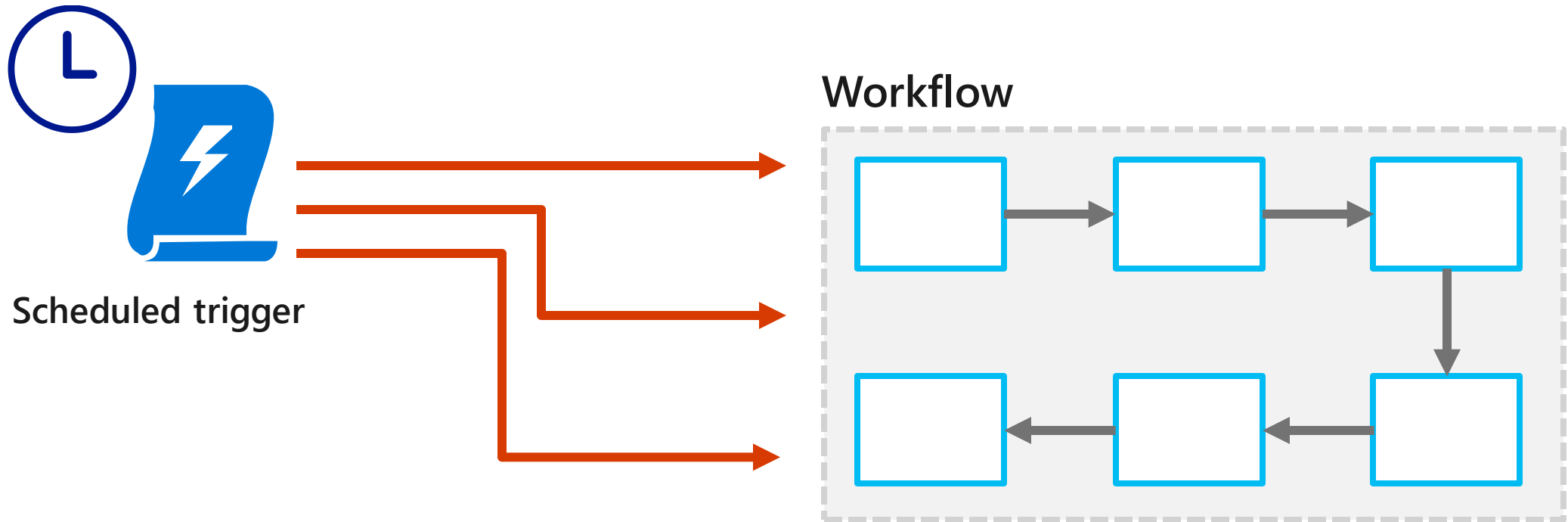
Dynamic content Expression

Search dynamic content

When a feed item is published

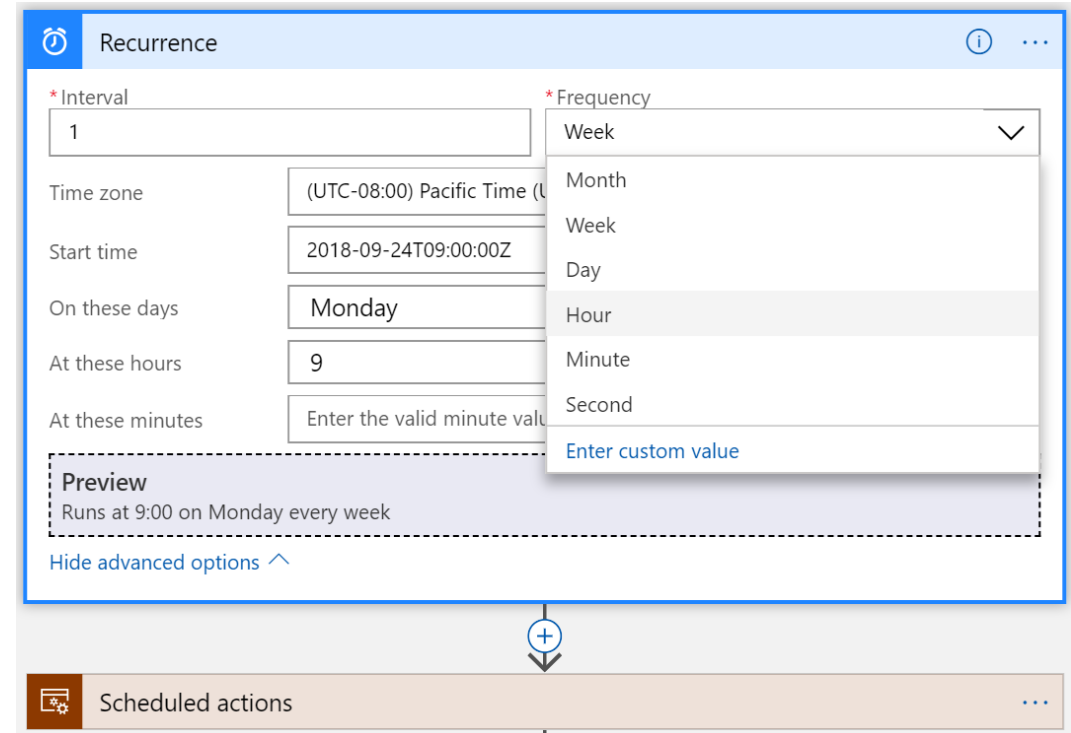
- categories - Item
- Feed copyright information
Copyright information
- Feed ID
Feed ID
- Feed published on
Feed published date
- Feed summary
Feed item summary
- Feed title**
Feed title

Schedule triggers



Scheduling recurrence

- Simple recurrence:
 - Interval and frequency
 - Examples: every 30 seconds, 5 minutes, or 1 month
- Start time:
 - Date and time for the first execution
 - Not earlier than the date and time for prescribed schedules
- Complex schedules:
 - Specify minutes, hours, weekdays, or days of the month of the recurrence
 - Examples: every Sunday at noon or every 15 minutes during work hours



The screenshot shows the 'Recurrence' tab in the Windows Task Scheduler. The 'Interval' is set to 1, and the 'Frequency' dropdown is open, showing options: Week, Month, Week, Day, Hour, Minute, Second, and 'Enter custom value'. The 'Time zone' is set to (UTC-08:00) Pacific Time (U...), 'Start time' is 2018-09-24T09:00:00Z, 'On these days' is Monday, 'At these hours' is 9, and 'At these minutes' is 'Enter the valid minute value'. A 'Preview' section shows 'Runs at 9:00 on Monday every week'. A 'Hide advanced options' link is visible. Below the recurrence settings is a 'Scheduled actions' section with a plus icon and a dropdown arrow.

Recurrence

* Interval: 1

* Frequency: Week

Time zone: (UTC-08:00) Pacific Time (U...)

Start time: 2018-09-24T09:00:00Z

On these days: Monday

At these hours: 9

At these minutes: Enter the valid minute value

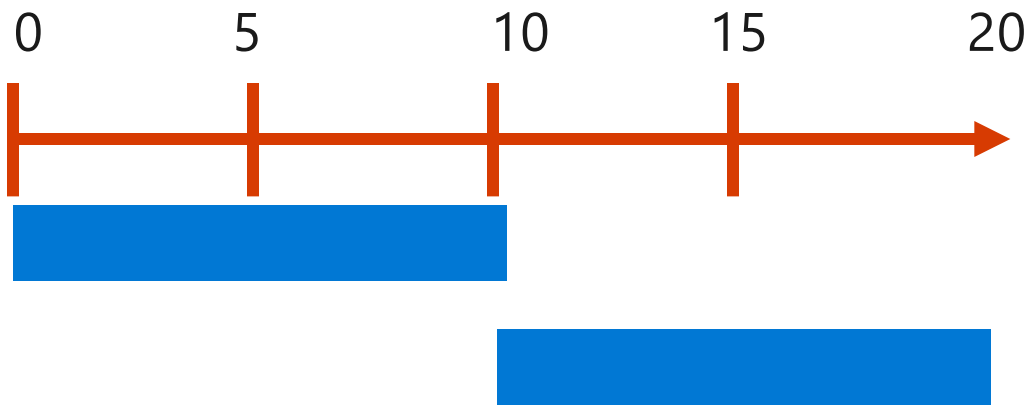
Preview: Runs at 9:00 on Monday every week

Hide advanced options ^

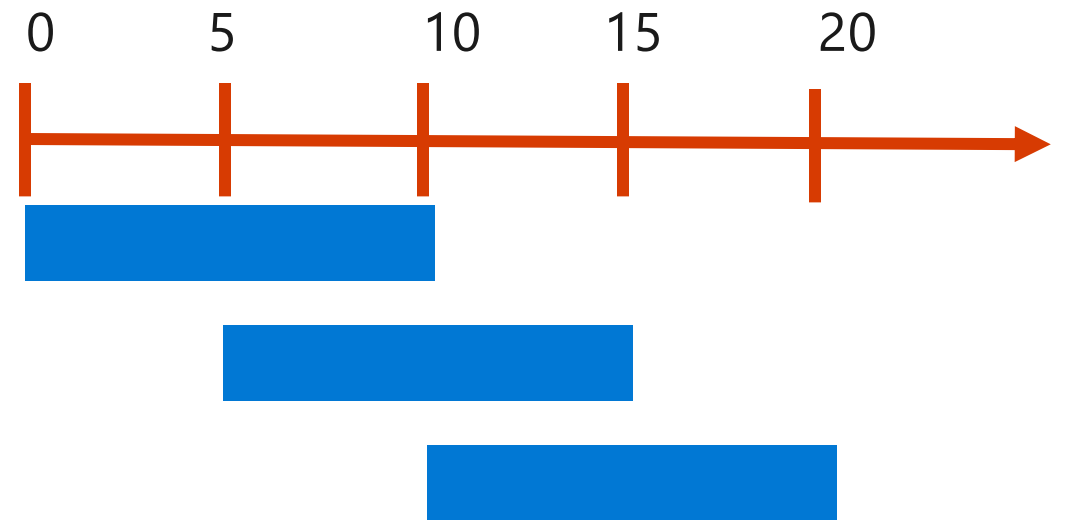
Scheduled actions

Schedule trigger types

Recurrence



Sliding window



Single execution (run once)

- Delay
 - Specify the wait duration
- Delay until
 - Specify the startTime property
- Execute actions

The screenshot displays the Azure Logic Apps workflow editor interface. The workflow consists of three sequential actions connected by downward arrows:

- When a HTTP request is received** (Trigger):
 - Header: When a HTTP request is received
 - Message: You can update request schema to pass in custom values for actions to be executed.
 - HTTP POST URL: URL will be generated after save
 - Request Body JSON Schema:

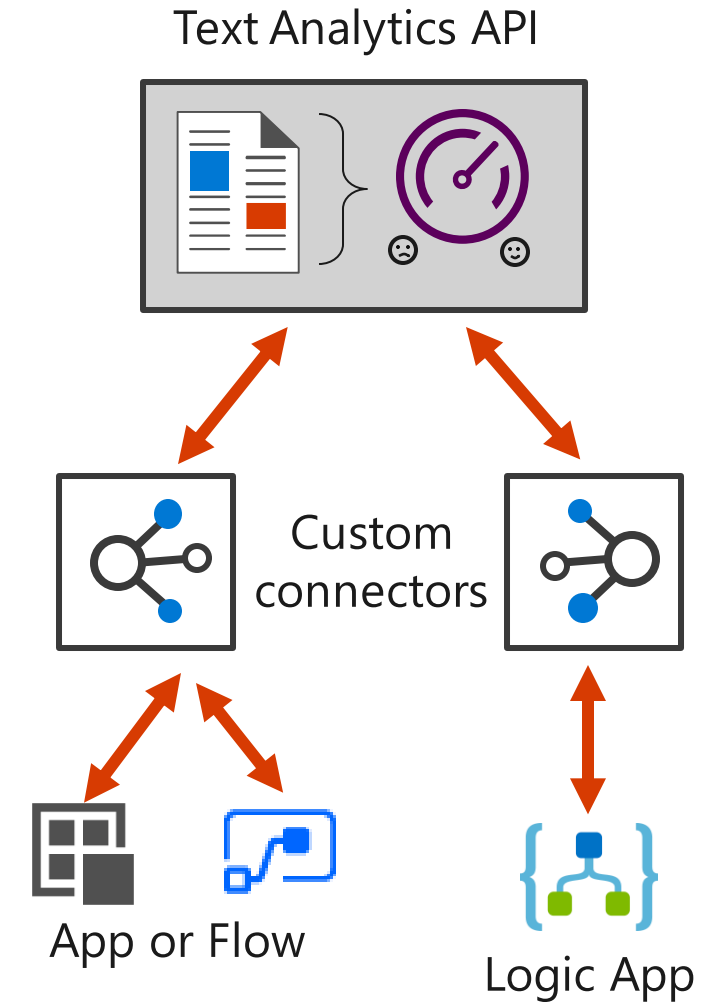
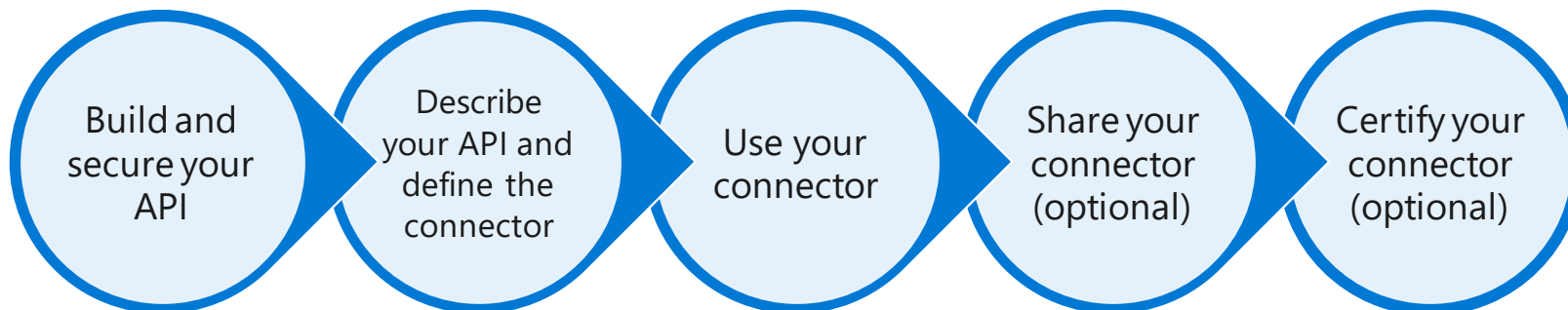
```
{  "properties": {    "startTime": {      "type": "string"    }  },  "type": "object"}
```
 - Links: Use sample payload to generate schema, Show advanced options
- Delay until** (Action):
 - Header: Delay until
 - Timestamp: *Timestamp (Field contains: startTime)
- Scheduled actions** (Action):
 - Header: Scheduled actions

Demonstration: Creating a logic app by using the Azure portal



Custom connectors

- Some APIs, services, and systems are available by using prebuilt connectors
- Build custom connectors:
 - Function-based
 - Custom defined triggers and actions



Deployment templates

- JSON template to build a logic app workflow
 - Three basic components of a logic app represented as JSON objects
 - Logic app resource
 - Workflow definition
 - Connections
- Can be extracted from existing workflows
- Templates can be deployed by using Azure Resource Manager templates

Template code

```
{
  "$schema": "http://schema.management.azure.com/providers/Microsoft.Logic/schemas/2016-06-01/workflowdefinition.json#",
  "contentVersion": "1.0.0.0",
  "parameters": {
    "testUri": {
      "type": "string",
      "defaultValue": "[parameters('testUri')]"
    }
  },
  "triggers": {
    "recurrence": {
      "type": "recurrence",
      "recurrence": {
        "frequency": "Hour",
        "interval": 1
      }
    }
  },
}
```



Template code (continued)

```
"actions": {  
  "http": {  
    "type": "Http",  
    "inputs": {  
      "method": "GET",  
      "uri": "@parameters('testUri')"  
    },  
    "runAfter": {}  
  }  
},  
"outputs": {}  
}
```



Create a deployment template

- Visual Studio tools for logic apps
 - Either generate from a visual workflow or author JSON directly
- Using the **Code** tab in the Azure portal
 - Generate from existing visual workflow
- Use a logic app template creator PowerShell module

Create a deployment template – Azure PowerShell

```
Install-Module -Name LogicAppTemplate
```

```
armclient token $SubscriptionId | Get-LogicAppTemplate -LogicApp MyApp -ResourceGroup  
MyRG -SubscriptionId $SubscriptionId -Verbose | Out-File C:\template.json
```



Template parameters – Function App example

```
"parameters": {  
  "functionName": { "type": "string", "minLength": 1, "defaultValue": "<FunctionName>" },  
},  
...  
"MyFunction": {  
  "type": "Function",  
  "inputs": {  
    "body": {},  
    "function": {  
      "id": "[resourceid('Microsoft.Web/sites/functions','functionApp',  
parameters('functionName'))]"  
    }  
  },  
  "runAfter": {}  
}
```



Template parameters – Service Bus example

```
"Send_message": {  
  "type": "ApiConnection",  
  "inputs": {  
    "host": {  
      "connection": {  
        "name": "@parameters('$connections')['servicebus']['connectionId']"  
      }  
    },  
    "method": "post",  
    "path": "[concat('/@{encodeURIComponent('', parameters('queueuname'),  
    '' )}/messages')]",  
    "body": { "ContentData": "@{base64(triggerBody())}" },  
    "queries": { "systemProperties": "None" }  
  },  
  "runAfter": {}  
}
```



Lab: Automating business processes by using logic apps

Duration



Lab sign-in information

