1. Enterprise Workflows with Azure Logic Apps and Azure Functions

Creating automated enterprise workflows with the power of Azure Serverless solutions.

Fábio Dario – [fabio.dario@gft.com](mailto:fabio.dario@gft.com)

Abstract

This article demonstrates how you can quickly create enterprise workflows with the power of Azure Logic Apps and Azure Functions.

We’ll understand the key concepts of the Serverless architecture and Modern Workflow tools as we go through a demonstration of an employee onboarding process, created only with Logic Apps and Azure Functions.

Introduction

Before we start this quick journey inside the Serverless world, here it goes a question. Why Serverless?

As a technology, it abstracts away the most menial parts of building an application, leaving you free to actually spend your days coding.

What this means is that you, the developer, can rapidly build apps that handle production-ready traffic. You don’t have to actively manage scaling for your applications. You don't have to provision servers, or pay for resources that go unused.

Ultimately, Serverless is about focusing your efforts on what provides value to your users. This means using managed services for databases, search indexes, queues, SMS messaging, and email delivery. It means tying these services together using stateless, ephemeral compute like the various FaaS (Function as a Service) offerings.

Installing windows updates does not provide value to your users. Managing your MSMQ servers does not provide value to your users. Shipping product provides value to your users.

Focus on your business logic, not your servers!

Azure Functions

Azure Functions is a serverless compute service that enables you to run code on-demand without having to explicitly provision or manage infrastructure.

It is a solution for easily running small pieces of code, or “functions” in the cloud. You can write just the code you need for the problem at hand, without worrying about a whole application or the infrastructure to run it. Functions can make development even more productive, and you can use your development language of choice. Pay only for the time your code runs and trust Azure to scale as needed.

Some features:

* Choice of language - Write functions using your choice of C#, F#, JavaScript or PHP.
* Pay-per-use pricing model - Pay only for the time spent running your code.
* Bring your own dependencies - Functions supports NuGet and NPM, so you can use your favorite libraries.
* Integrated security - Protect HTTP-triggered functions with OAuth providers such as Azure Active Directory, Facebook, Google, Twitter, and Microsoft Account.
* Simplified integration - Easily leverage Azure services and software-as-a-service (SaaS) offerings.
* Flexible development - Code your functions right in the portal or set up continuous integration and deploy your code through GitHub, Azure DevOps Services, and other supported development tools.
* Open-source - The Functions runtime is open-source and available on GitHub.

Azure Functions is a great solution for processing data, integrating systems, working with the internet-of-things (IoT), and building simple APIs and microservices. Consider Functions for tasks like image or order processing, file maintenance, or for any tasks that you want to run on a schedule.

Azure Functions has two kinds of pricing plans, so you can choose the one that best fits your needs:

* Consumption plan - When your function runs, Azure provides all of the necessary computational resources. You don't have to worry about resource management, and you only pay for the time that your code runs.
* App Service plan - Run your functions just like your web, mobile, and API apps. When you are already using App Service for your other applications, you can run your functions on the same plan at no additional cost.

Logic Apps

Azure Logic Apps is a cloud service that helps you automate and orchestrate tasks, business processes, and workflows when you need to integrate apps, data, systems, and services across enterprises or organizations.

Logic Apps simplifies how you design and build scalable solutions for app integration, data integration, system integration, enterprise application integration (EAI), and business-to-business (B2B) communication, whether in the cloud, on premises, or both.

For example, here are just a few workloads you can automate with logic apps:

* Process and route orders across on-premises systems and cloud services.
* Send email notifications with Office 365 when events happen in various systems, apps, and services.
* Move uploaded files from an SFTP or FTP server to Azure Storage.
* Monitor tweets for a specific subject, analyze the sentiment, and create alerts or tasks for items that need review.

To build enterprise integration solutions with Azure Logic Apps, you can choose from a growing gallery with 200+ connectors, which include services such as Azure Service Bus, Functions, and Storage; SQL, Office 365, Dynamics, Salesforce, BizTalk, SAP, Oracle DB, file shares, and more. Connectors provide triggers, actions, or both for creating logic apps that securely access and process data in real time.

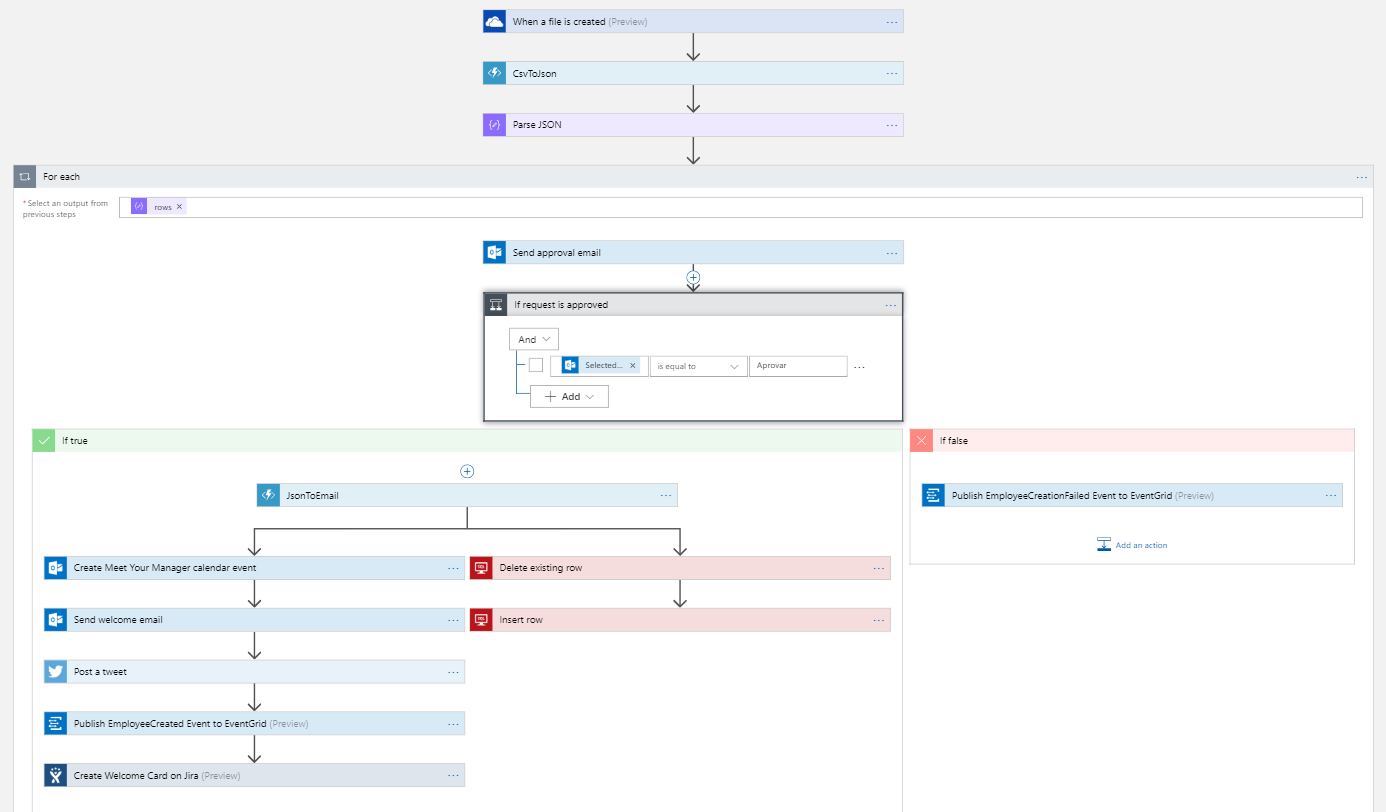
Quick Tutorial

This article shows how to create an employee onboarding workflow using Logic Apps and Azure Functions.

For that there are two ways to start, using the [Azure Portal](http://portal.azure.com) or [Visual Studio 2017 Logic Apps Tools](https://marketplace.visualstudio.com/items?itemName=VinaySinghMSFT.AzureLogicAppsToolsforVisualStudio-18551). Both approaches have the same visual appealing and features, but for this article the Azure Portal will be used.

The idea behind this quick tutorial is that a company wants a workflow to help it onboarding new employees, doing common and repetitive tasks for each of them.

To make things worse, picture the idea of that company being GFT, and it is buying another company with 1000 employees, and all these employees are being onboarded to GFT, following its standards and compliance tasks. Someone would be in charge of creating 1000 users in GFT’s Active Directory, creating 1000 email boxes, sending 1000 welcome emails, etc. Boring, repetitive and costly, right? With that in mind, our tutorial will start with a CSV file in OneDrive containing a list of employees to be onboarded.



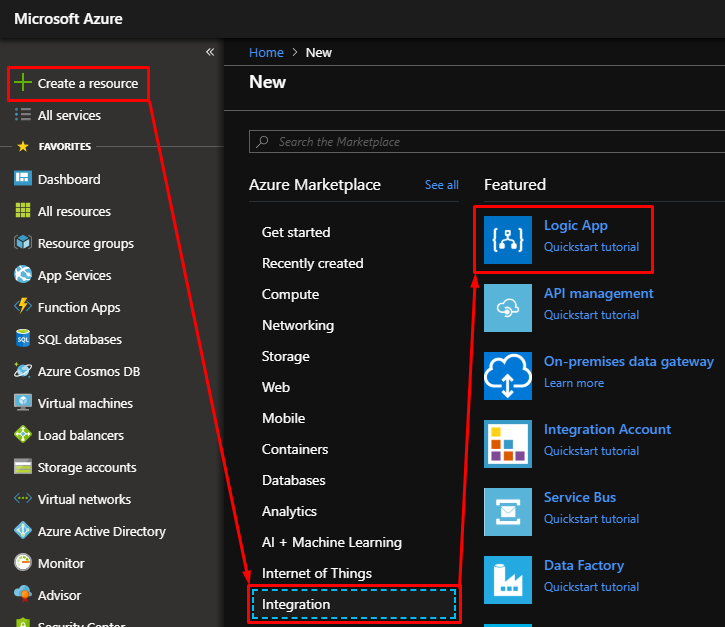
The workflow will follow the steps below:

* Trigger the Logic App when a file is added to OneDrive;
* Convert the CSV content to json;
* Parse the json, and for each employee:
* Send an approval email to its manager, asking if it’s OK to onboard that employee;
* If rejected, stop the workflow for that employee and publish an EmployeeCreationFailed to [Azure EventGrid](https://docs.microsoft.com/en-us/azure/event-grid/overview);
* If accepted, generate an email template with the employee data;
* Send a welcome email;
* Create a meeting in the employee’s and also in the manager’s Outlook Calendar;
* Add the employee to the company’s database;
* Post a welcome tweet in GFT’s twitter with the employee’s name;
* Publish an EmployeeCreated event to [Azure EventGrid](https://docs.microsoft.com/en-us/azure/event-grid/overview);
* Create a card on Jira asking everyone to go to the new employee’s desk for a hug.

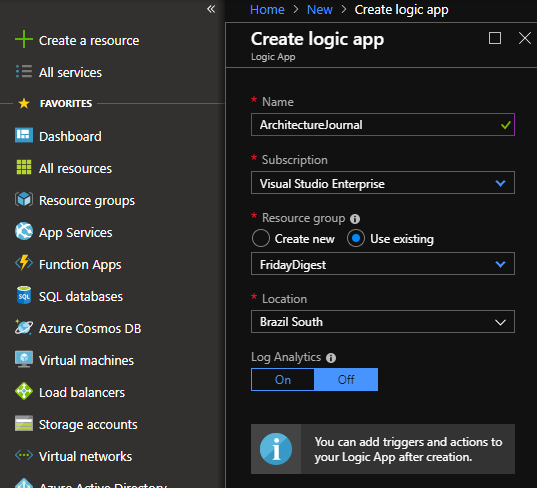
Figure 1 - Overall workflow

Creating a Logic App

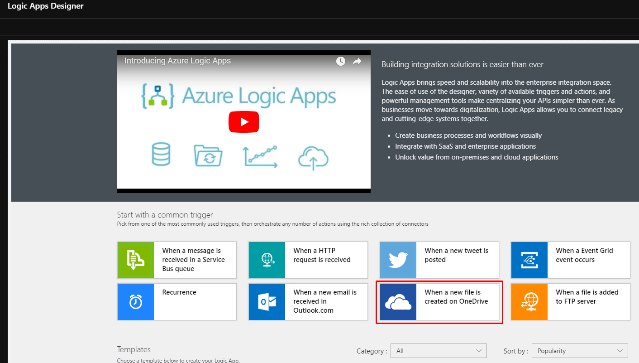
Go to Azure Portal and create a Logic App.



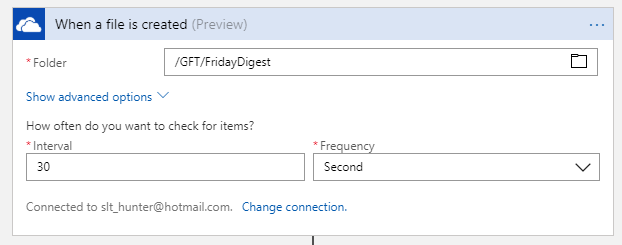
Specify name, subscription, location and resource group.



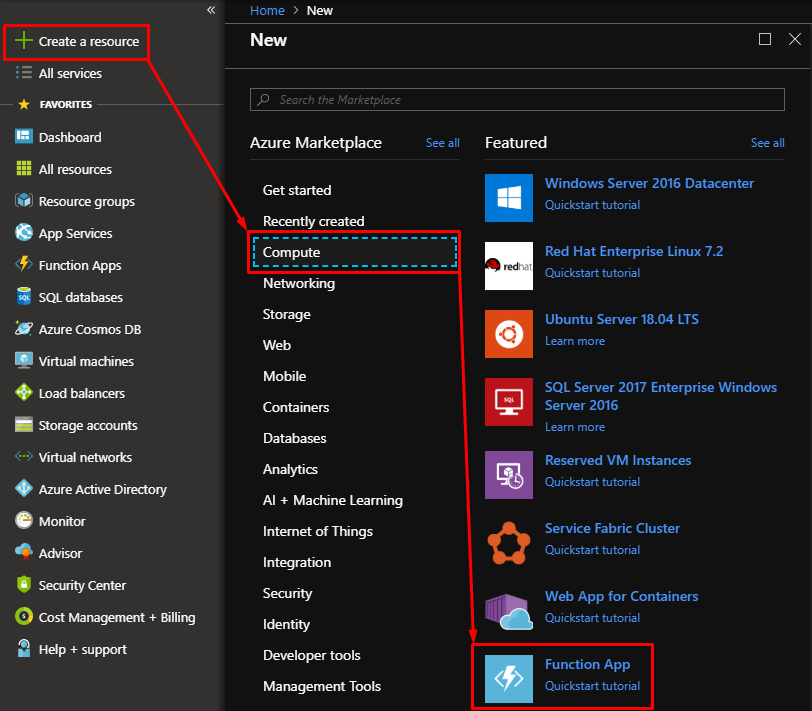
There are predefined templates for Logic Apps, so for this tutorial we’re using a “OneDrive file created” webhook.



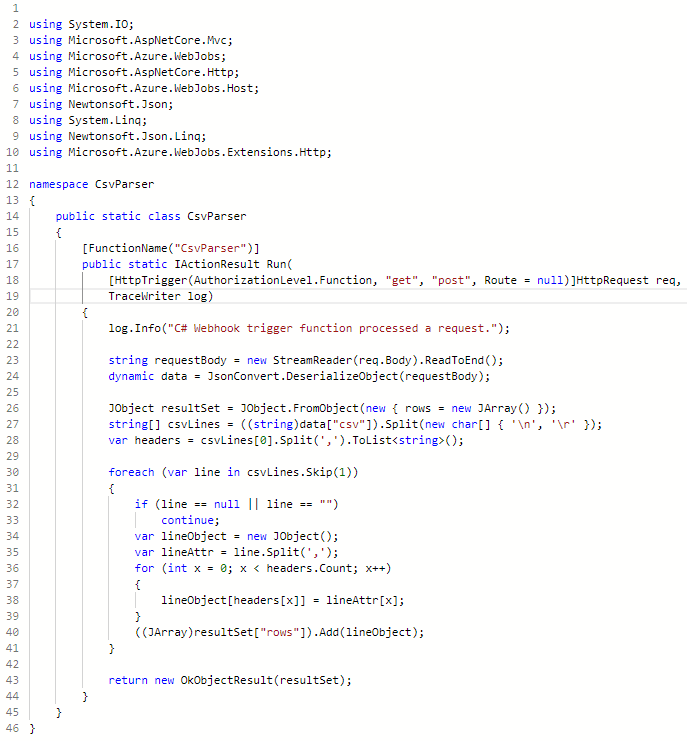
Sign in to your Microsoft Account and configure where the file is stored in OneDrive, and the pooling interval.



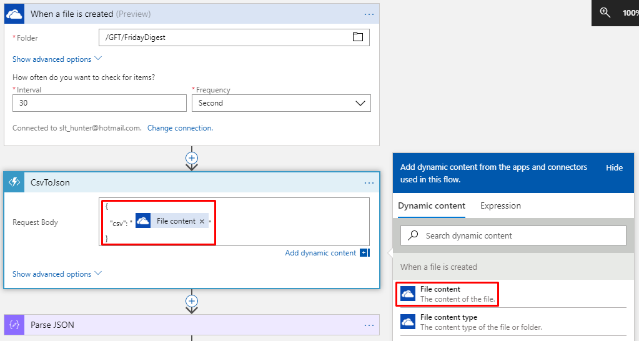
For parsing the CSV, we’re using an Azure Function. To create it, add a Function App and create a function in it.



Use the code below to parse a CSV in a nice json object.

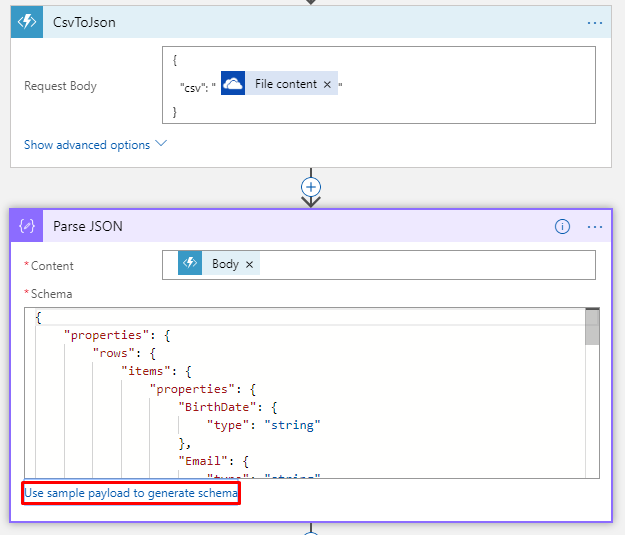


Add the Azure Function as the next step after retrieving the file from OneDrive. In Logic Apps, each connector makes available a set of dynamic contents to be used in the next steps of the workflow. For OneDrive connector, we’re using the File Content to build a json containing a “csv” property, to be used by the Azure Function (HttpTrigger), that parses its content into a proper formatted json to be used by the rest of the workflow.

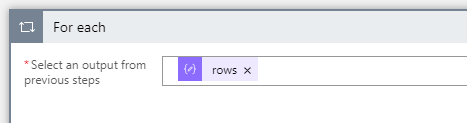


Add a Parse JSON step to be able to handle the output of the Azure Function, converting the json to a format that allow Logic Apps to iterate through each employee in the json.

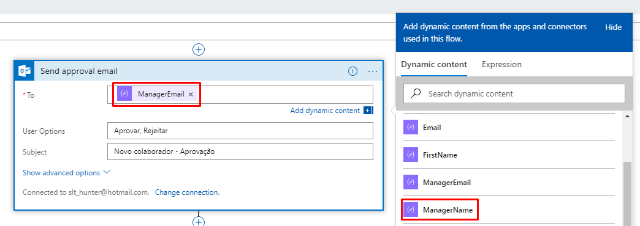
To use the Parse JSON step, you don’t need to specify a json Schema manually, you can just paste a json in the format you expect the Azure Function to return using the “Use sample payload to generate schema” option”.



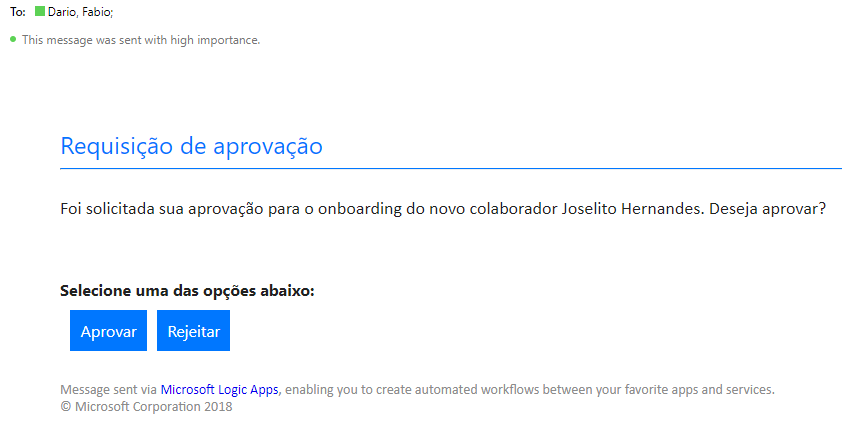
Logic Apps accept common clauses like If, While, ForEach, so in the example, for each employee in the json, we’re doing the rest of the steps. We’re iterating through the “rows” content of the Parse JSON step.



Every employee will need approval of its manager to be onboarded, so we’re adding a “Send Approval Email” step, using the ManagerName property of each row from the Parse JSON step. We can specify the actions to be shown in the approval buttons of the email. In this example we’re overriding the default “Approve” and “Reject” with “Aprovar” and “Rejeitar”, to follow current company’s language (pt-br)

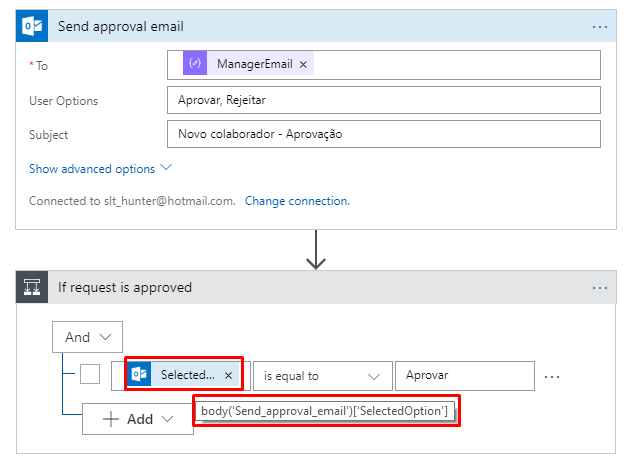


The workflow only continues when the manager receives an email and clicks on the “Aprovar” option.



As said before, Logic Apps accept control clauses, so we’re specifying that if the request is approved, the workflow continues, if not, just publish a failure event and continue the foreach loop.

For that, we need to check if the return of the approval email is equals to “Aprovar”, using the “SelectedOption” dynamic content. Note that using a dynamic content basically is the same as using an expression based on the output of a step. In this case, the “SelectedOption” content resolves to an expression retrieving data from the body of the approval email step, as seen in the hint in the image below.

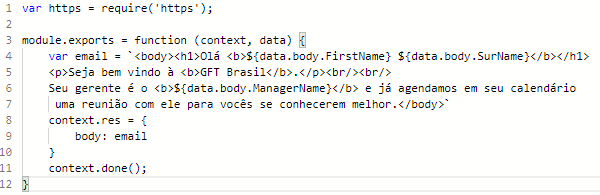


In case of rejection, the “false” condition will just publish a failure event to [Azure EventGrid](https://docs.microsoft.com/en-us/azure/event-grid/overview), so systems that may want to be aware of an employee being rejected can subscribe to these events and be informed whenever that occurs.

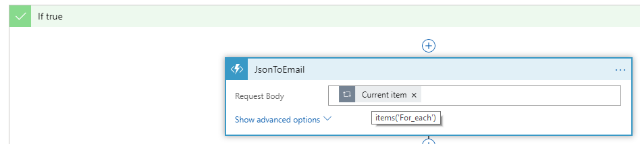
If the employee is accepted, the workflow continues.

To be able to send a pretty formatted welcome email, we’re using another Azure Function, written in JavaScript this time, to convert the employee’s personal data into an HTML formatted string, sent as the body of the email.

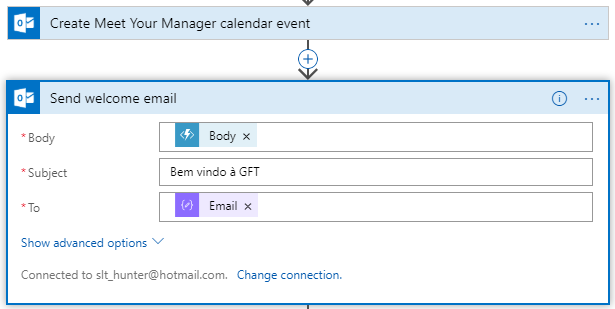
For that, create another Azure Function with the code below.



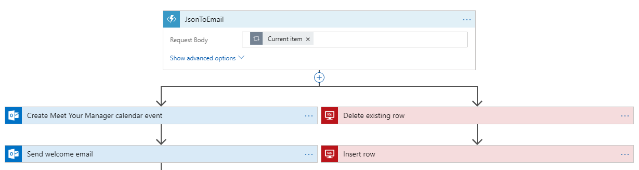
Note that we’re using the “Current item” content, what means we’re providing to the Azure Function the whole row from the ForEach step.



With the Azure Function output, we’re able to send a welcome email. We can also schedule a meeting with the employee and its manager for a quick chat, ensuring they get to know each other.



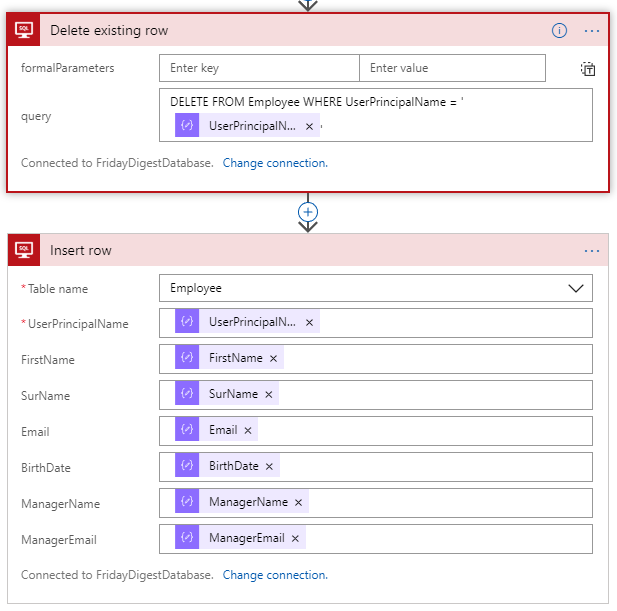
Logic Apps also allows the developer to create branches, splitting the execution into parallel steps, like seen below. In the same time the email is being sent and the meeting is being scheduled, the employee is added to the company’s database (in this example, a SQL Azure).



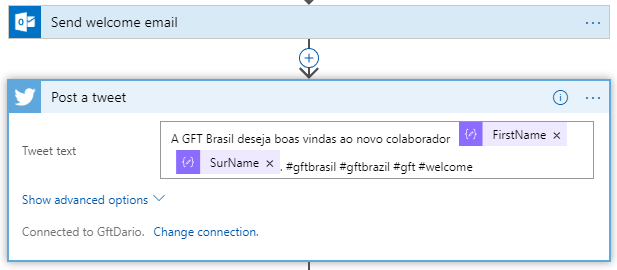
In this example, for the sake of simplicity, we’re just setting up a quick “Delete before Add” pattern, ensuring the same UPN (User Principal Name) is not added more than once.

To connect to a SQL Azure, just provide the connection data you grab from the SQL Azure Database Connection Strings section.

To delete the desired records, you can specify a delete command, and to insert a record, just pick the values from the Dynamic Content tooltip, like seen below. The connector is smart enough to understand which columns you have in the table you chose and ask for the values for each column.

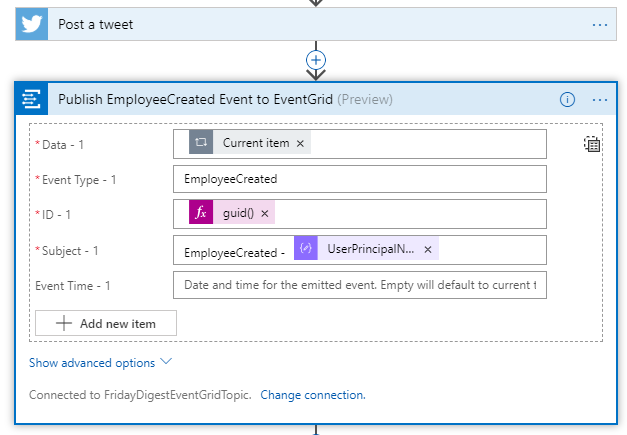


Next step is to write a nice welcome tweet in GFT’s Twitter account, telling everyone that we have a happy new employee joining the company. Logic Apps has a Twitter connector, and to configure it, just sign in to the Twitter account you want through the connector’s sign in screen. After signed in, just provide the tweet text using the dynamic contents that come from the Parse JSON step, iterated in the ForEach control action.

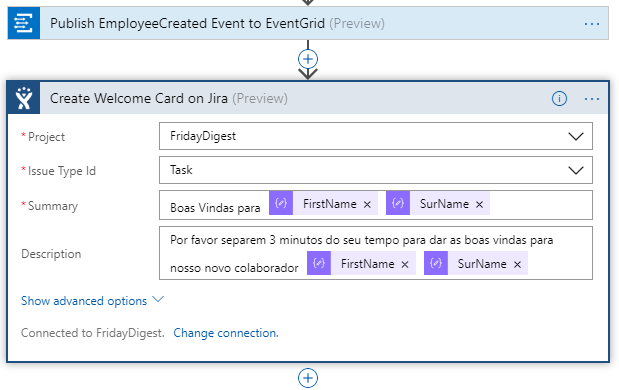




The same way we’ve done for the failure action, it’s nice that we publish an event once the employee is fully onboarded, so any systems that want to be aware of a new employee being created, it’s just a matter of subscribing to an [EventGrid](https://docs.microsoft.com/en-us/azure/event-grid/overview) topic. Specify the Data, the Event Type, an ID, a subject just for the sake of categorization, and have it published to anybody to listen to it.



Last step of the workflow is to create a card in Jira, to make sure everybody knows there is a new employee in the company, and a 1 min trip to the new colleague’s desk for a hug is encouraged by the company. As with other connectors, there is a sign in screen from Jira where you get authenticated and authorized, and the connector has all the information from your Jira account. Select the project you want to create the card into, the type, a summary and the description of the card.



That completes our Employee Onboarding Workflow.

Stay tuned for the next article that will show you how to automate the deployment of Logic Apps and Azure Functions, with the help of ARM Templates, Visual Studio and Azure DevOps (former VSTS).

Conclusion

Try to think about how long you, as a full-stack developer, would take to create a workflow this size, with all the integrations with Outlook, Twitter, Jira, Databases, all the approval requests and conditional response actions, event publishing, as well as creating the interface for managing all this craziness.

Think know about what you would do if you needed to add a step where you create a user in the company’s AD, or integrate with an existing BizTalk workflow, maybe connect your app to a CosmosDB instance and retrieve some data, listen to a queue for a message to be delivered, or even listen to an Application Insights telemetry event coming from another system. That could take days of development, a lot of Stack Overflow researching and a huge amount of manual code. Any changes in the overall workflow could be a pain, with code rewriting and refactoring, tedious and exhaustive tests, etc.

Last but not least, think about all the infrastructure changes you might end up needing, provisioning more resources and installing and configuring a lot of features in your servers.

With the help of Azure Logic Apps and Azure Functions, adding another is step is as simple as, well, adding another step to a really intuitive diagram interface, that visually represents your workflow, allowing even business users to make changes and adequate the workflow to its needs, no infrastructure changes required.

That is the true power of serverless offers from Azure, delivering more value in significantly less time, in an intuitive and straightforward way.

Just for the record, the workflow shown in this article took less than two hours to be created, cheers Azure Logic Apps!!!

References

[Azure Logic Apps Documentation]

<https://docs.microsoft.com/en-us/azure/logic-apps/logic-apps-overview>

[Azure Event Grid Documentation]

<https://docs.microsoft.com/en-us/azure/event-grid/overview>

[Azure Serverless Computing]

<https://azure.microsoft.com/en-us/overview/serverless-computing/>

[Serverless Concepts]

<https://serverless.com/>

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***Fábio Dario*** *is a Solutions/Software Architect and Team Leader at GFT. Microsoft Certified Technology Specialist and Microsoft Certified Professional Developer, with 12+ years of experience with cutting-edge Microsoft technologies, always looking for innovative solutions and new things to learn. Passionate developer and software engineer, family guy and dog lover. –* ***fabio.dario@gft.com***