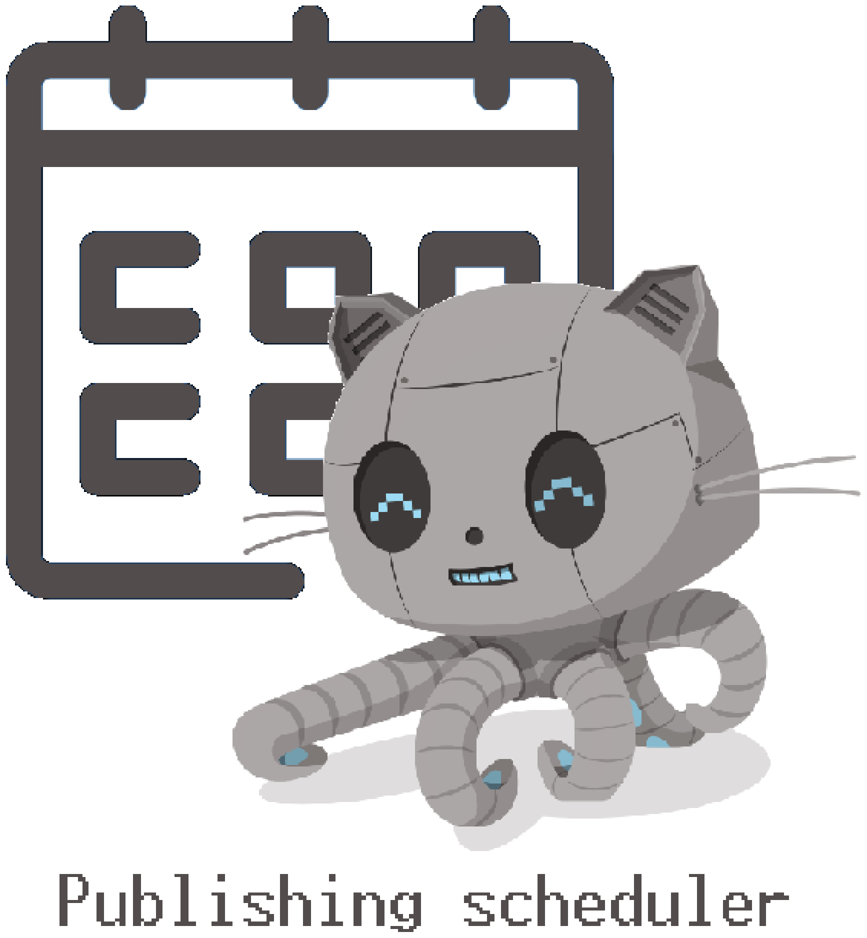


ANATOMY OF A SERVERLESS GITHUB BOT

How we built a serverless GitHub bot using Azure for the Microsoft hackathon.

https://github.com/Chris-Johnston/PublishScheduler

$ WHOAMI

I'm Chris Johnston. I'm a Software Engineer at Microsoft, working on Azure Networking.

I graduated UWB CSSE last year, and I've helped with the last two UWB Hacks hackathons.

CONTEXT

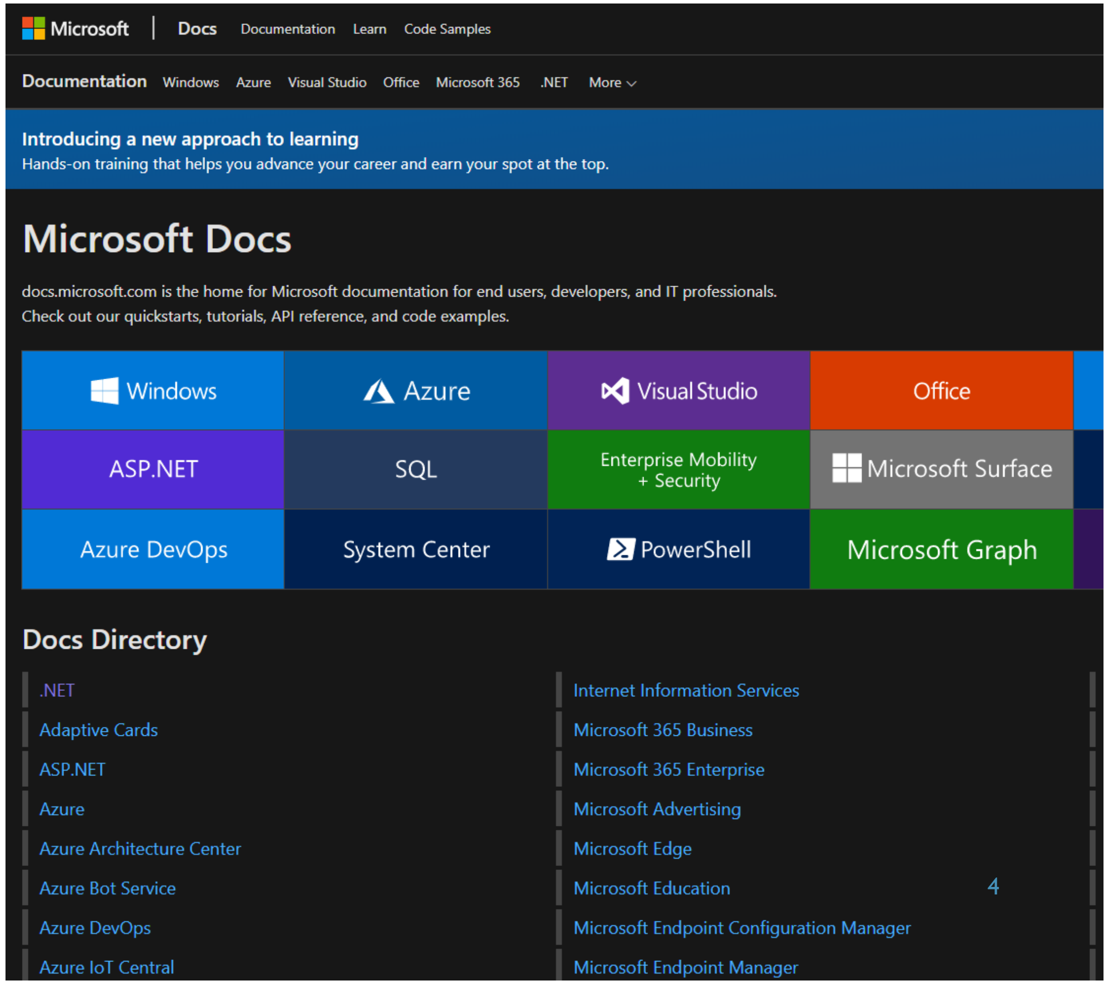
Microsoft hosts an annual week-long hackathon where employees are encouraged to work on whatever interests them.

Some of these projects turn into real products, the best example of this is the Xbox Adaptive Controller:

PROBLEM STATEMENT

I teamed up with some technical writers in the team behind docs.microsoft.com, which is the documentation site for all things Microsoft.

- They use GitHub repos to maintain their documentation.
  - GitHub Actions wasn't out yet.
- When changes are merged to master, they are automatically deployed.
- There's no way for them to "hold" content until a specific time/date.
  - A PR that's hours to months old could easily be forgotten.
  - Nobody wants to merge a PR outside of business hours.



Microsoft | Docs Documentation Learn Code Samples

Documentation Windows Azure Visual Studio Office Microsoft 365 .NET More

Introducing a new approach to learning
Hands-on training that helps you advance your career and earn your spot at the top.

Microsoft Docs

docs.microsoft.com is the home for Microsoft documentation for end users, developers, and IT professionals. Check out our quickstarts, tutorials, API reference, and code examples.

Windows	Azure	Visual Studio	Office
ASP.NET	SQL	Enterprise Mobility + Security	Microsoft Surface
Azure DevOps	System Center	PowerShell	Microsoft Graph

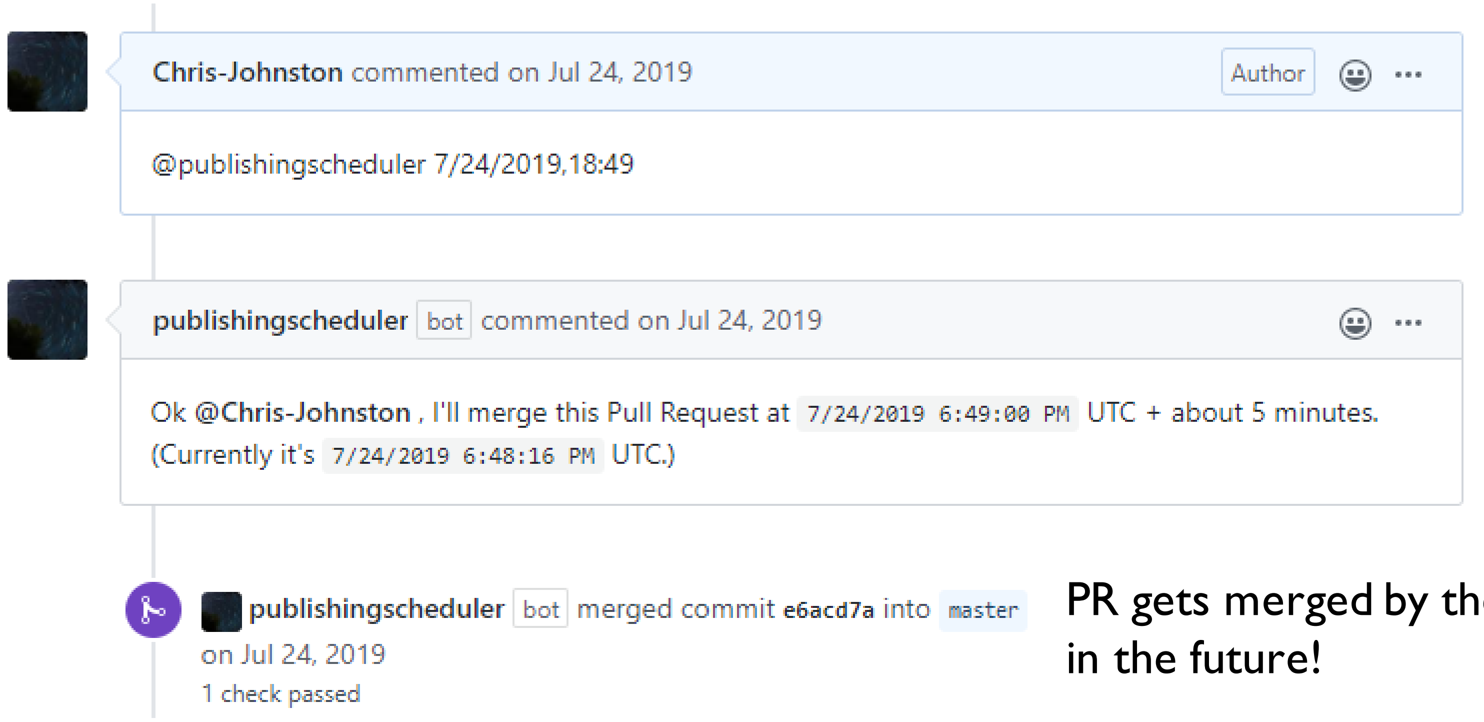
Docs Directory

.NET	Internet Information Services
Adaptive Cards	Microsoft 365 Business
ASP.NET	Microsoft 365 Enterprise
Azure	Microsoft Advertising
Azure Architecture Center	Microsoft Edge
Azure Bot Service	Microsoft Education
Azure DevOps	Microsoft Endpoint Configuration Manager
Azure IoT Central	Microsoft Endpoint Manager

MEET PUBLISHING SCHEDULER

Publishing Scheduler is a GitHub App that can merge pre-approved Pull Requests at any time.

Add it to your repo, and leave this comment in any approved Pull Request:



Chris-Johnston commented on Jul 24, 2019 Author

@publishingscheduler 7/24/2019,18:49

publishingscheduler bot commented on Jul 24, 2019

Ok **@Chris-Johnston** , I'll merge this Pull Request at `7/24/2019 6:49:00 PM` UTC + about 5 minutes. (Currently it's `7/24/2019 6:48:16 PM` UTC.)

publishingscheduler bot merged commit `e6acd7a` into `master` on Jul 24, 2019

1 check passed

PR gets merged by the bot in the future!

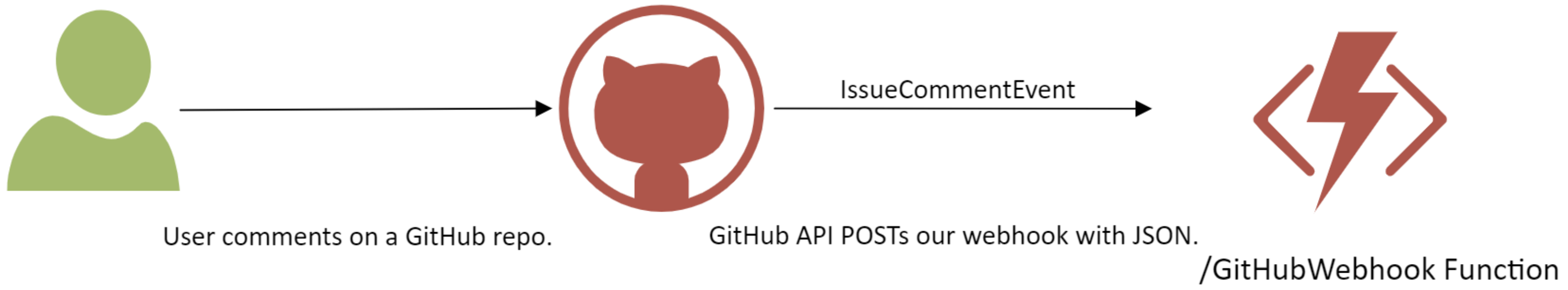
HOW DO WE BUILD GITHUB APPS?

GitHub offers a REST API that allows developers to build apps with it.

When your service talks to GitHub, you send it a HTTP request.
When GitHub talks to you, it sends you a HTTP request (these are called webhooks).

An API using webhooks is effective, since GitHub can directly notify your service about events when they happen.

- Pro: No polling required!
- Con: Requires that you have a server that GitHub can reach.


IssueCommentEvent
User comments on a GitHub repo.
GitHub API POSTs our webhook with JSON.
/GitHubWebhook Function

https://developer.github.com/webhooks/

CHOOSING OUR TOOLS

The GitHub API is over HTTP, so it's language/tooling agnostic.

Here's how my team decided on what tools to use:

- Most of my team uses C#
- GitHub has a pretty good library for C#/.NET: Octokit.Net
- We need some web server to accept GitHub webhooks, and nothing else
- We expect traffic to this server to be inconsistent and infrequent

CHOOSING OUR TOOLS

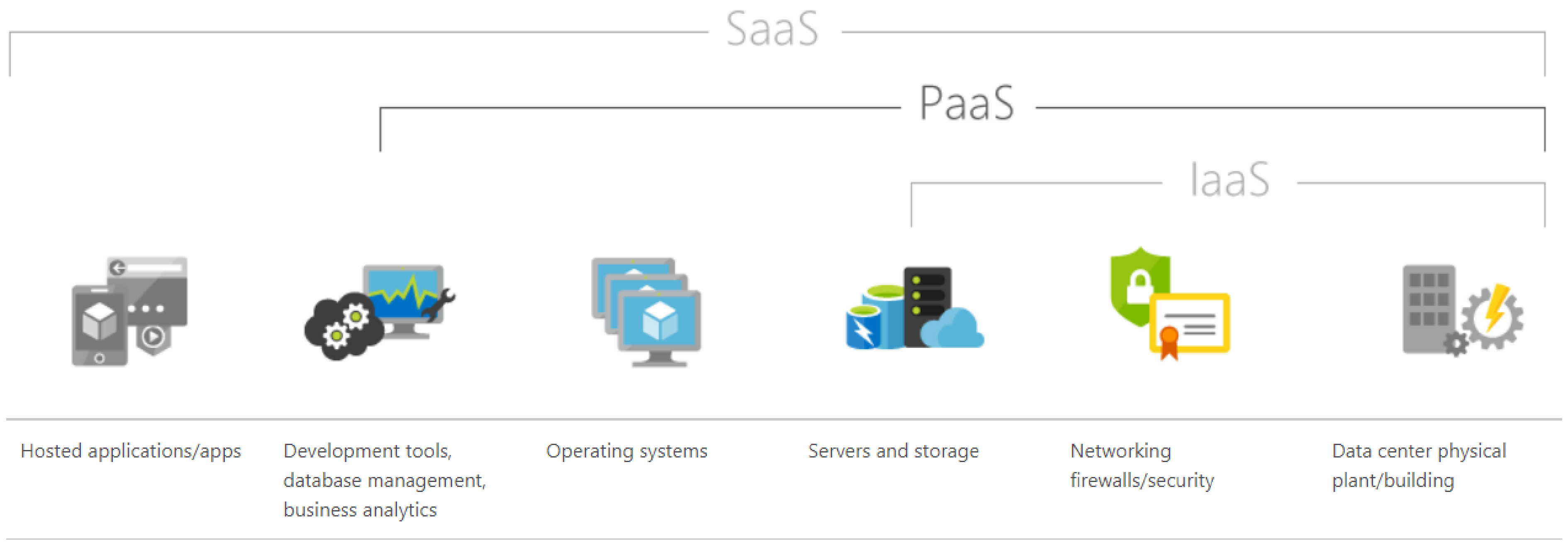
ASP.NET

- An entire open-source framework for building web apps and services
- Supports MVC pages, user authentication, etc.
- Runs behind a webserver, which is always on
- Comparable to Java Swing, Python Flask, etc.
- Free to host on Windows or Linux, but **pay for the VM that you use**

Azure Functions

- Azure's event-driven serverless compute platform
- Can be triggered by HTTP, storage, and more
- Only runs when we need it to, 99.95% SLA
- Comparable to AWS Lambda, GCP Functions
- **Pay for the execution you use**, only runs on Azure (or debug locally)

CHOOSING OUR TOOLS


SaaS
PaaS
IaaS
Hosted applications/apps
Development tools, database management, business analytics
Operating systems
Servers and storage
Networking firewalls/security
Data center physical plant/building

ASP.NET can run anywhere, but we'd still need to provision a dedicated virtual machine and storage to keep it running. With Azure Functions, the infrastructure is abstracted away for us. We only pay for the compute time we use.

CHOOSING OUR TOOLS

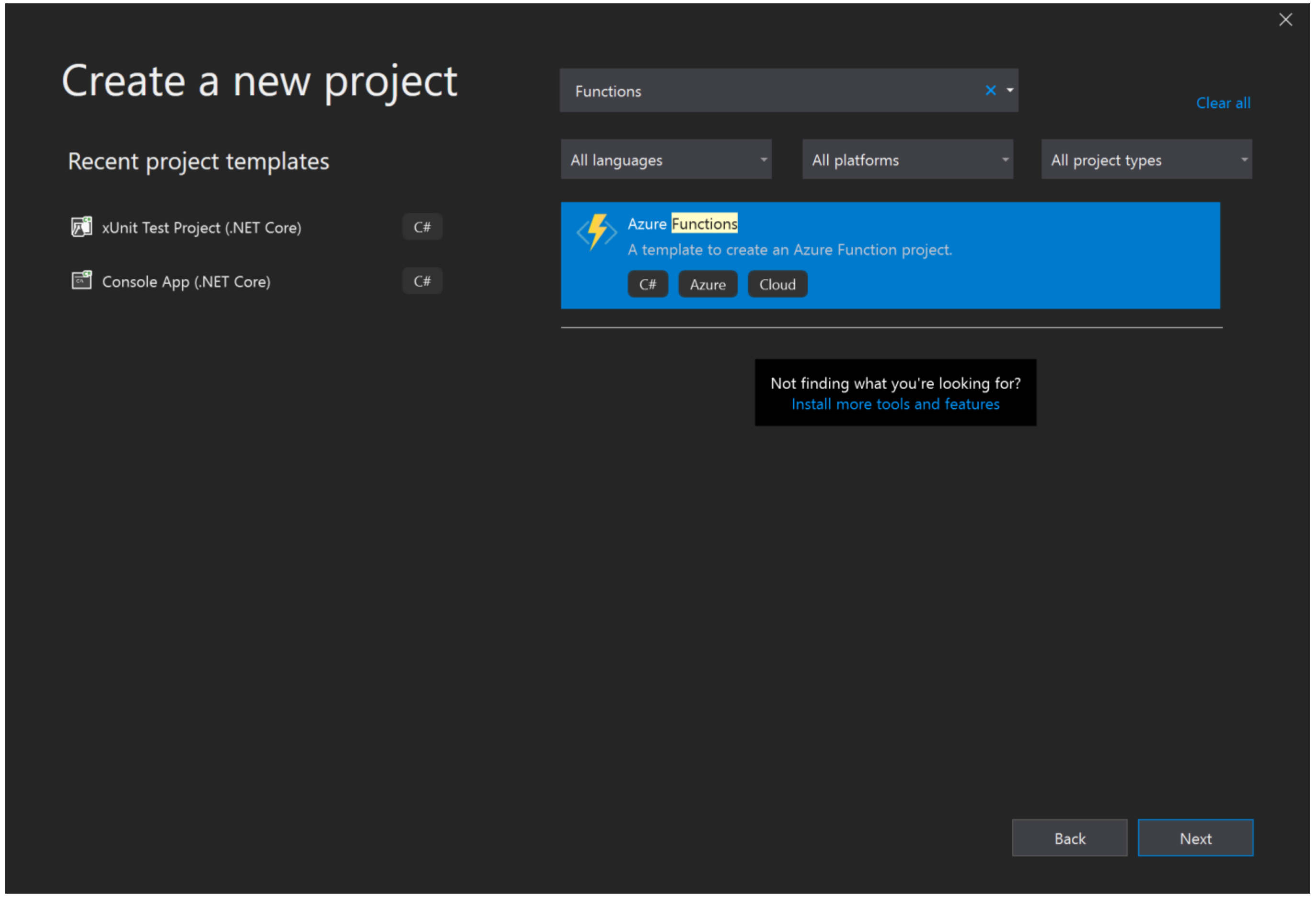
In the end, we chose to use **Azure Functions** over ASP.NET for the following reasons:

- We only need an API, not an entire web app w/ a front-end
- We expect infrequent use, so Functions are **cheaper** than a dedicated VM
- HTTP and Queue storage triggers will prove to be useful for us in just a moment
- I wanted to learn more about how to use Azure Functions

We really could have used either one here, with some differences.

WHERE TO START WITH AZURE FUNCTIONS?

Functions start with your code first. Visual Studio provides templates which make it easy to get started.


Create a new project

Functions

Clear all

Recent project templates

xUnit Test Project (.NET Core)	C#
Console App (.NET Core)	C#

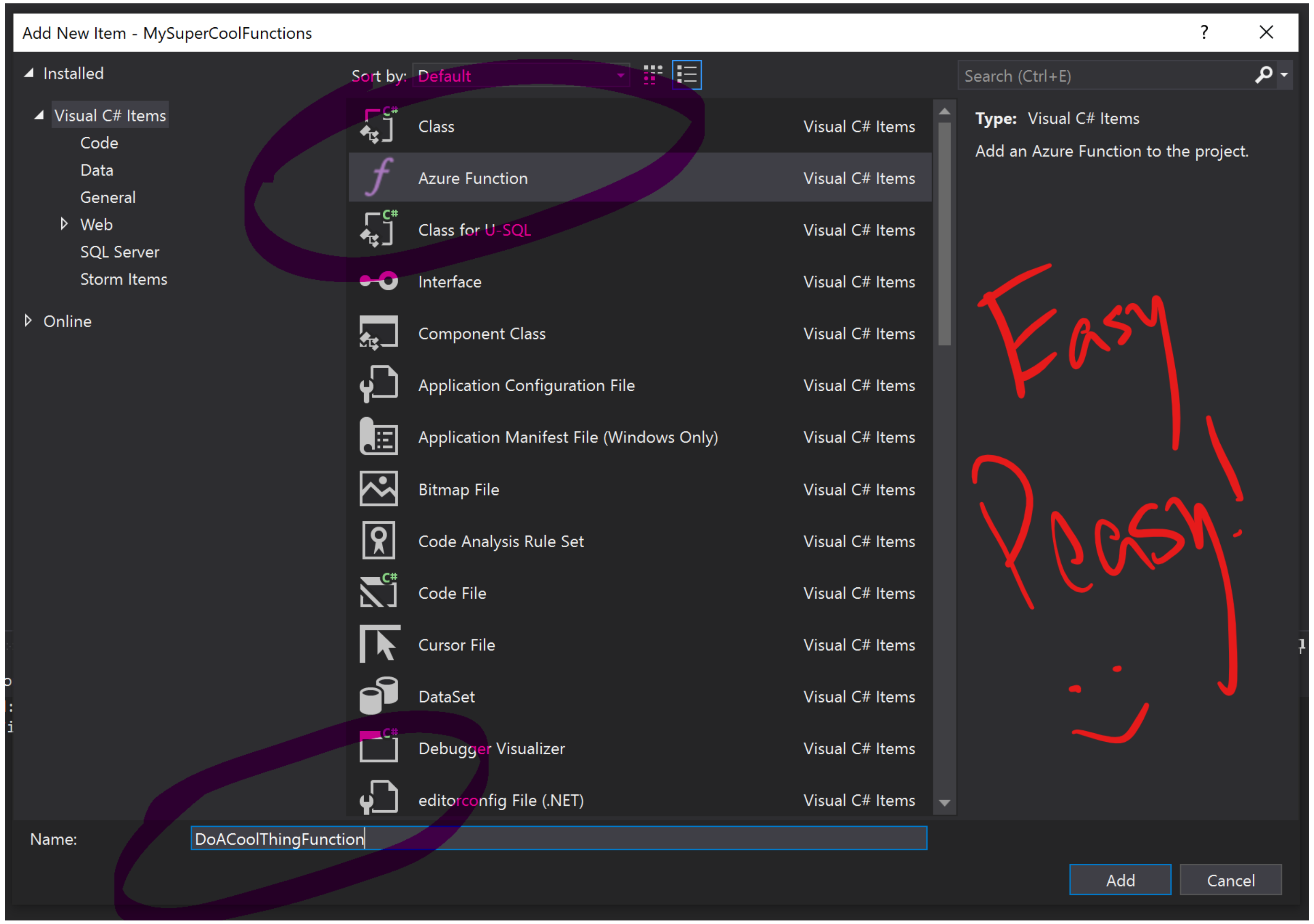
All languages | All platforms | All project types

Azure Functions
A template to create an Azure Function project.
C# Azure Cloud

Not finding what you're looking for?
Install more tools and features

WHERE TO START WITH AZURE FUNCTIONS?

Create a project, and add some Functions to it.


Add New Item - MySuperCoolFunctions

Installed
- Visual C# Items
  - Code
  - Data
  - General
  - Web
  - SQL Server
  - Storm Items
- Online

Sort by: Default

Search (Ctrl+E)

Class	Visual C# Items
Azure Function	Visual C# Items
Class for U-SQL	Visual C# Items
Interface	Visual C# Items
Component Class	Visual C# Items
Application Configuration File	Visual C# Items
Application Manifest File (Windows Only)	Visual C# Items
Bitmap File	Visual C# Items
Code Analysis Rule Set	Visual C# Items
Code File	Visual C# Items
Cursor File	Visual C# Items
DataSet	Visual C# Items
Debugger Visualizer	Visual C# Items
editorconfig File (.NET)	Visual C# Items

Type: Visual C# Items

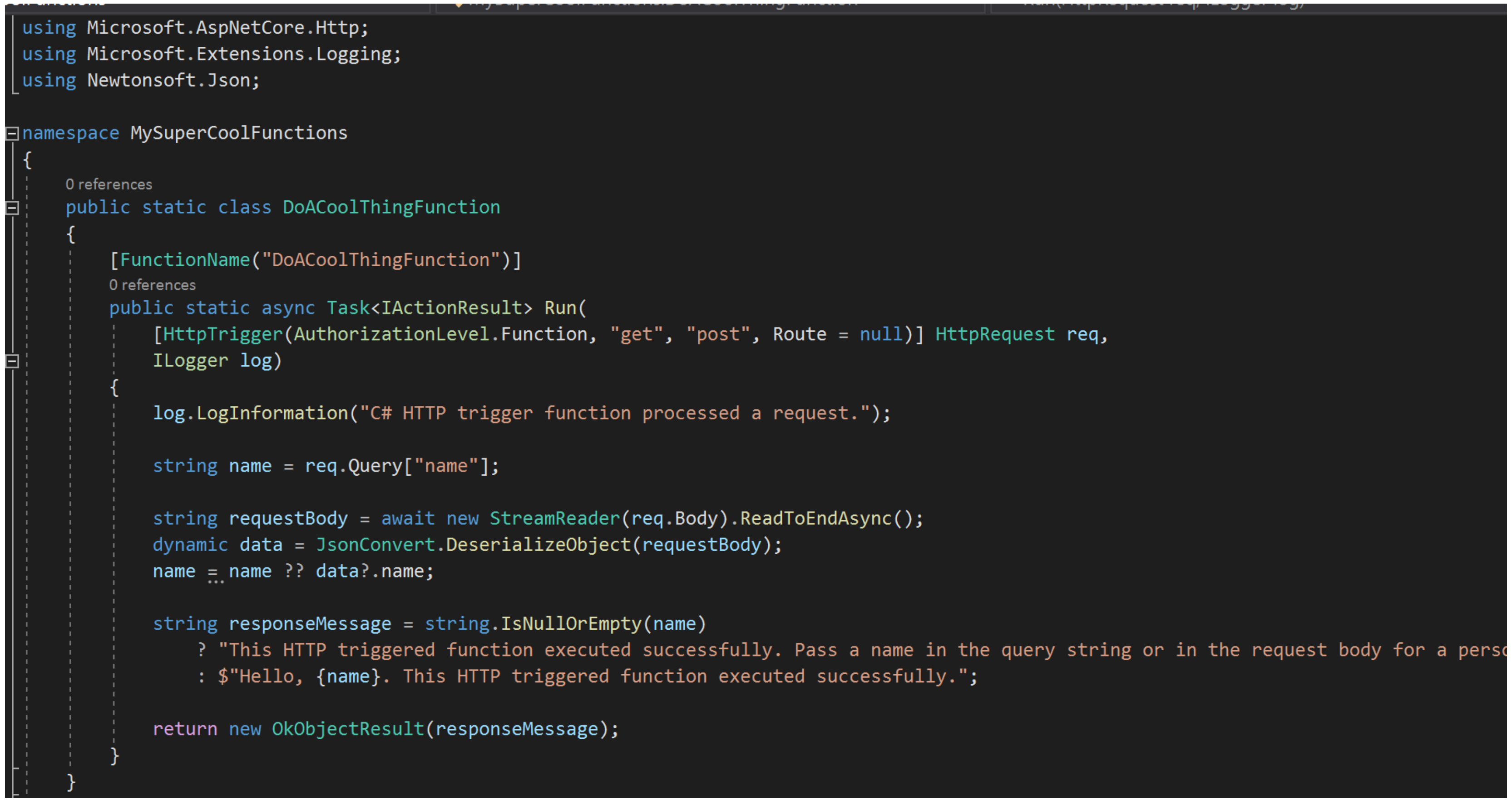
Add an Azure Function to the project.

Name: DoACoolThingFunction

Add Cancel

WHERE TO START WITH AZURE FUNCTIONS?

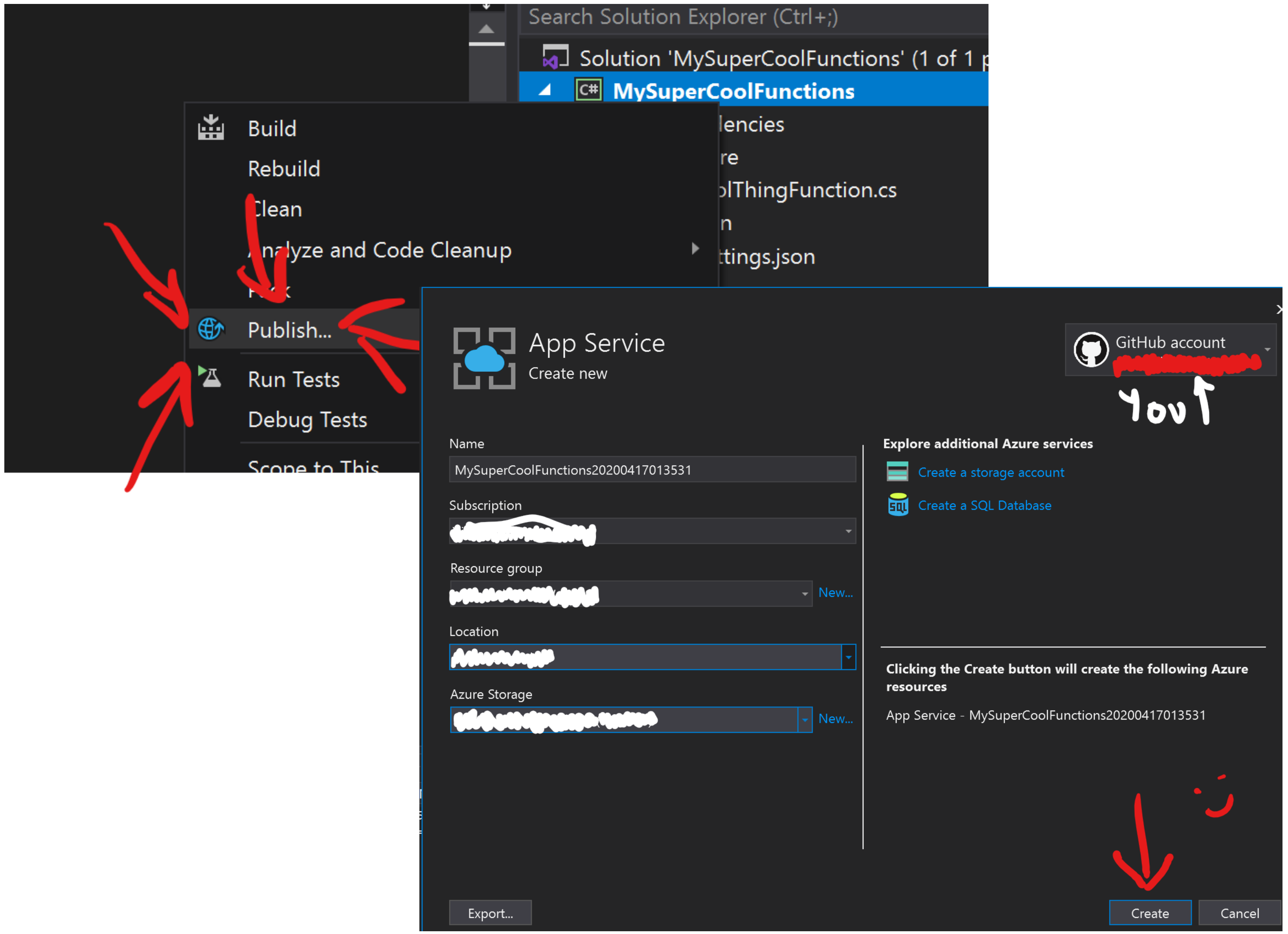
And suddenly we have a whole bunch of code!



```
using Microsoft.AspNetCore.Http;
using Microsoft.Extensions.Logging;
using Newtonsoft.Json;

namespace MySuperCoolFunctions
{
    0 references
    public static class DoACoolThingFunction
    {
        [FunctionName("DoACoolThingFunction")]
        0 references
        public static async Task<IActionResult> Run(
            [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
            ILogger log)
        {
            log.LogInformation("C# HTTP trigger function processed a request.");

            string name = req.Query["name"];

            string requestBody = await new StreamReader(req.Body).ReadToEndAsync();
            dynamic data = JsonConvert.DeserializeObject(requestBody);
            name = name ?? data?.name;

            string responseMessage = string.IsNullOrEmpty(name)
                ? "This HTTP triggered function executed successfully. Pass a name in the query string or in the request body for a perso
                : $"Hello, {name}. This HTTP triggered function executed successfully.";

            return new OkObjectResult(responseMessage);
        }
    }
```

WHERE TO START WITH AZURE FUNCTIONS?

If we want to deploy this code, we can just click "Publish". After we sign in, we can provision our resources and upload our code.



GitHub account [redacted]

Name: MySuperCoolFunctions20200417013531

Subscription: [redacted]

Resource group: [redacted] New...

Location: [redacted]

Azure Storage: [redacted] New...

Clicking the Create button will create the following Azure resources

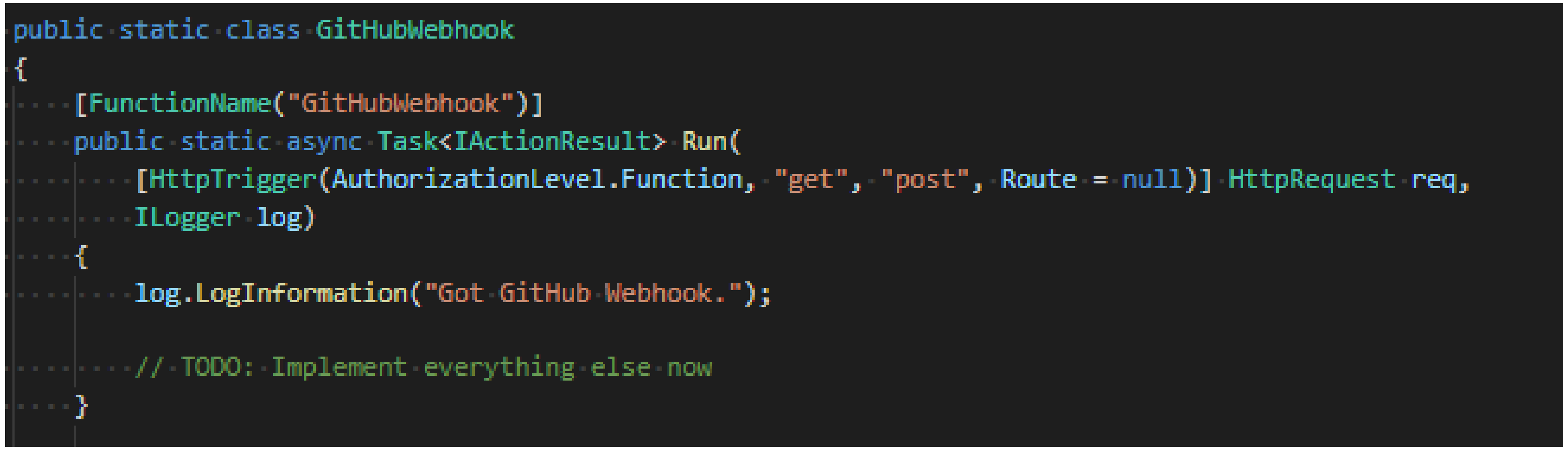
App Service - MySuperCoolFunctions20200417013531

WHERE TO START WITH AZURE FUNCTIONS?

Azure Functions uses attributes on your methods to define when and how it’s called, instead of being configured elsewhere. This snippet defines the webhook that GitHub uses to talk to us.

The attribute [FunctionName(“GitHubWebhook”)] defines this method as a new Function (named GithubWebhook).

The [HttpTrigger(…)] attribute defines that we can send GET or POST HTTP requests to this Function to trigger it.

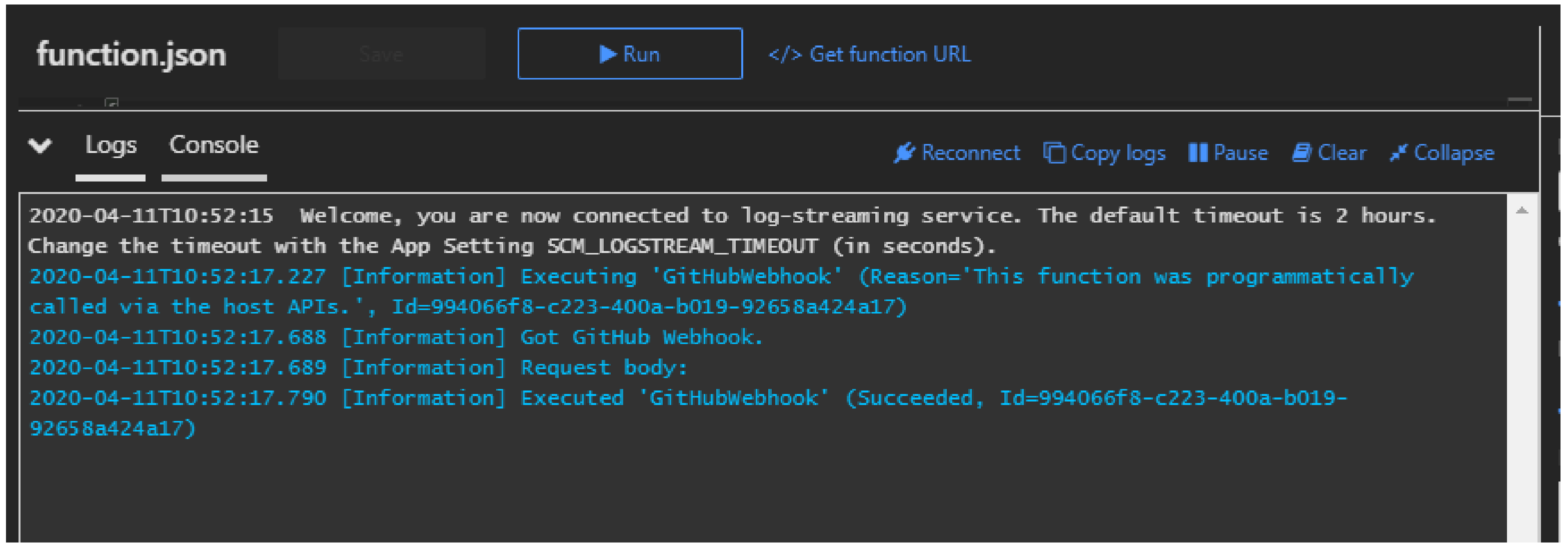


```
public static class GitHubWebhook
{
    [FunctionName("GitHubWebhook")]
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req,
        ILogger log)
    {
        log.LogInformation("Got GitHub Webhook.");

        // TODO: Implement everything else now
    }
```

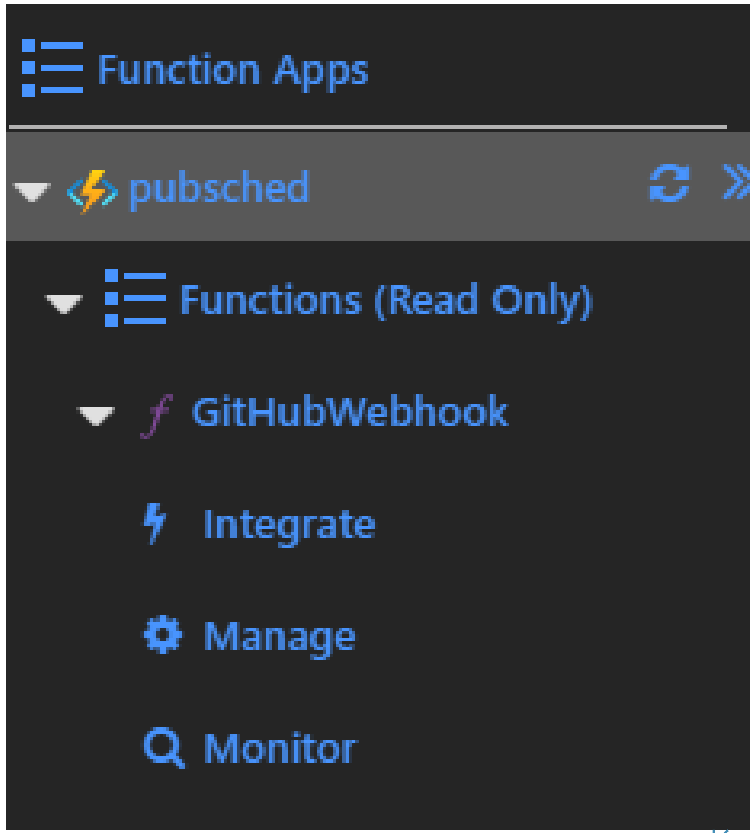
WHERE TO START WITH AZURE FUNCTIONS?

Once it's deployed, our Functions should show up in the list.

We can test this from within the Azure Portal itself:



```
2020-04-11T10:52:15  Welcome, you are now connected to log-streaming service. The default timeout is 2 hours.
Change the timeout with the App Setting SCM_LOGSTREAM_TIMEOUT (in seconds).
2020-04-11T10:52:17.227 [Information] Executing 'GitHubWebhook' (Reason='This function was programmatically
called via the host APIs.', Id=994066f8-c223-400a-b019-92658a424a17)
2020-04-11T10:52:17.688 [Information] Got GitHub Webhook.
2020-04-11T10:52:17.689 [Information] Request body:
2020-04-11T10:52:17.790 [Information] Executed 'GitHubWebhook' (Succeeded, Id=994066f8-c223-400a-b019-
92658a424a17)
```



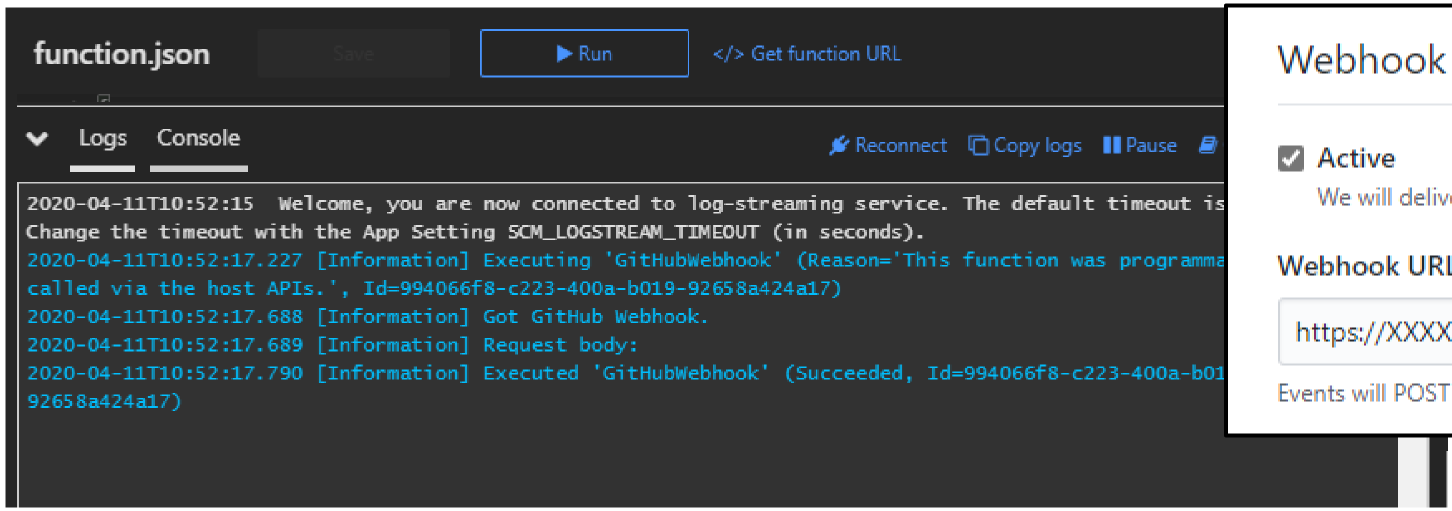
HOOKING IT UP TO GITHUB

Once our function works, we can use the “Get function URL” button on this page to copy the URL we need to access this endpoint.

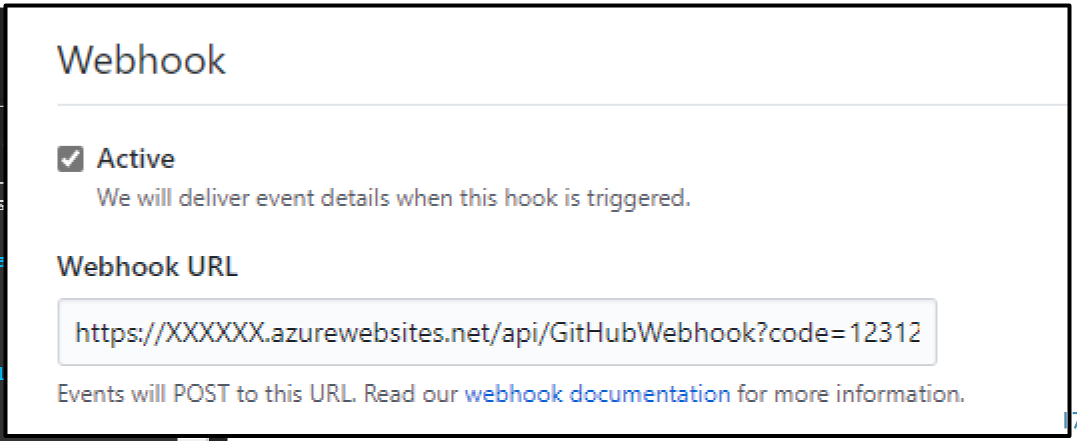
The URL will look something like this: **https://myapp.azurewebsites.net/api/GitHubWebhook?code=123abc**

This includes a private token, so that random 3rd parties that aren’t GitHub can’t call this function.

We can share this URL with GitHub to allow our services to talk to each other.



```
2020-04-11T10:52:15  Welcome, you are now connected to log-streaming service. The default timeout is
Change the timeout with the App Setting SCM_LOGSTREAM_TIMEOUT (in seconds).
2020-04-11T10:52:17.227 [Information] Executing 'GitHubWebhook' (Reason='This function was programma
called via the host APIs.', Id=994066f8-c223-400a-b019-92658a424a17)
2020-04-11T10:52:17.688 [Information] Got GitHub Webhook.
2020-04-11T10:52:17.689 [Information] Request body:
2020-04-11T10:52:17.790 [Information] Executed 'GitHubWebhook' (Succeeded, Id=994066f8-c223-400a-b01
92658a424a17)
```



Webhook

☑ Active

We will deliver event details when this hook is triggered.

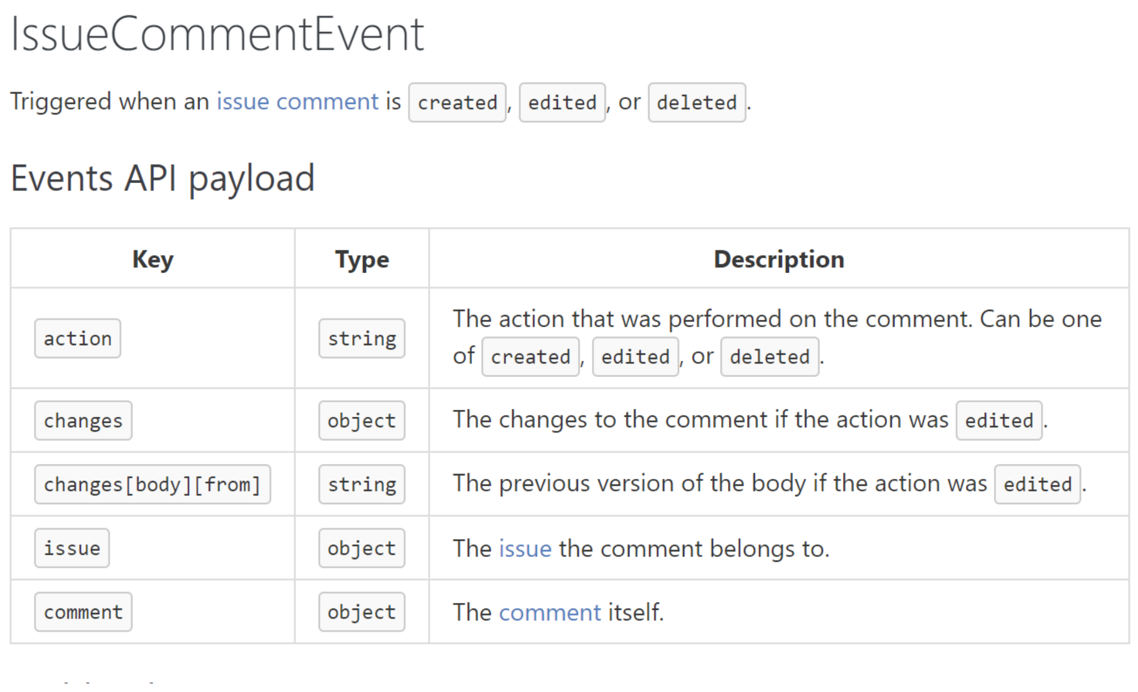
Webhook URL

https://XXXXXX.azurewebsites.net/api/GitHubWebhook?code=12312

Events will POST to this URL. Read our webhook documentation for more information.

HOOKING IT UP TO GITHUB

The GitHub webhook will then send POST us data for each of the events that we care about:

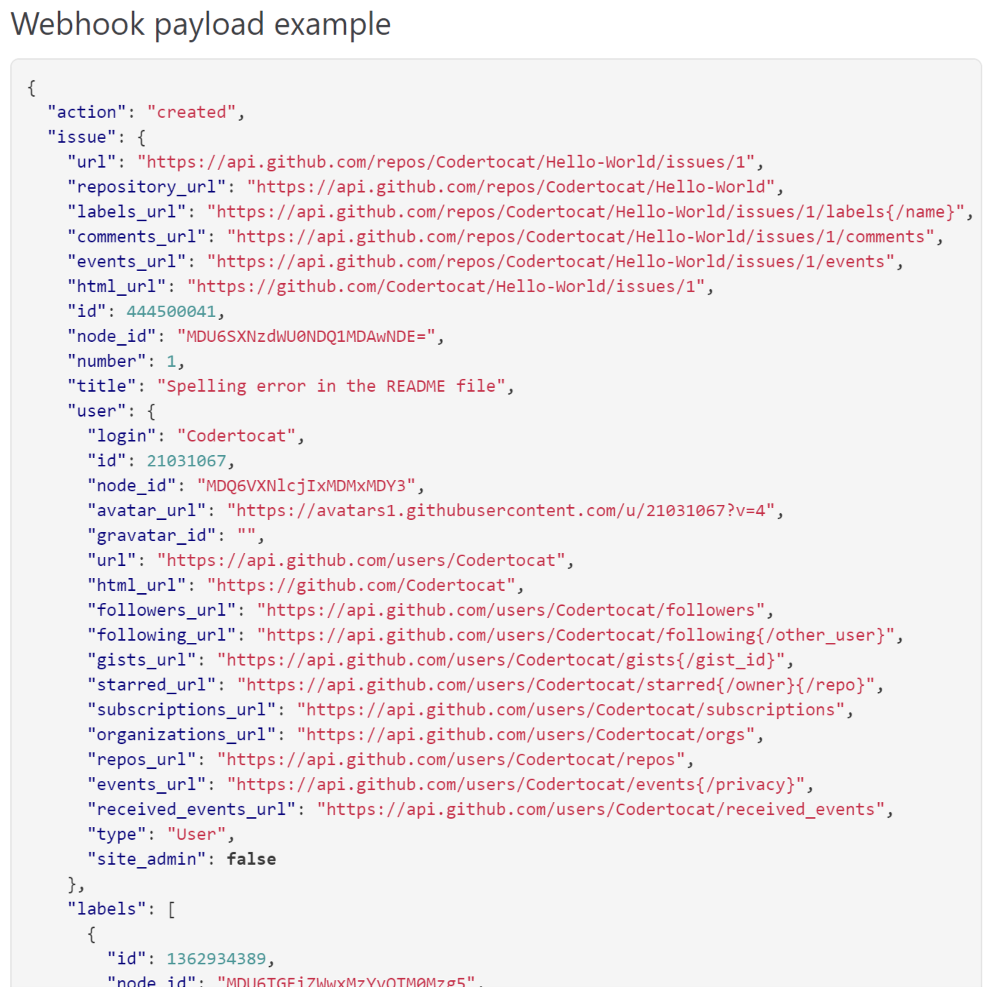


IssueCommentEvent

Triggered when an issue comment is `created`, `edited`, or `deleted`.

Events API payload

Key	Type	Description
`action`	`string`	The action that was performed on the comment. Can be one of `created`, `edited`, or `deleted`.
`changes`	`object`	The changes to the comment if the action was `edited`.
`changes[body][from]`	`string`	The previous version of the body if the action was `edited`.
`issue`	`object`	The issue the comment belongs to.
`comment`	`object`	The comment itself.



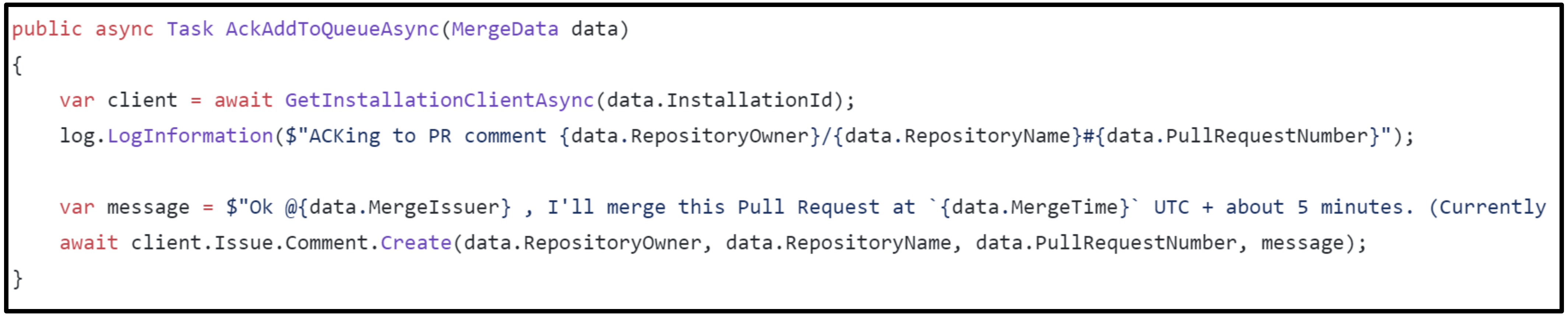
Webhook payload example

```
{
  "action": "created",
  "issue": {
    "url": "https://api.github.com/repos/Codertocat/Hello-World/issues/1",
    "repository_url": "https://api.github.com/repos/Codertocat/Hello-World",
    "labels_url": "https://api.github.com/repos/Codertocat/Hello-World/issues/1/labels{/name}",
    "comments_url": "https://api.github.com/repos/Codertocat/Hello-World/issues/1/comments",
    "events_url": "https://api.github.com/repos/Codertocat/Hello-World/issues/1/events",
    "html_url": "https://github.com/Codertocat/Hello-World/issues/1",
    "id": 444500041,
    "node_id": "MDU6SXNzdWU0NDQ1MDAwNDE=",
    "number": 1,
    "title": "Spelling error in the README file",
    "user": {
      "login": "Codertocat",
      "id": 21031067,
      "node_id": "MDQ6VXNlcjIxMDMxMDY3",
      "avatar_url": "https://avatars1.githubusercontent.com/u/21031067?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/Codertocat",
      "html_url": "https://github.com/Codertocat",
      "followers_url": "https://api.github.com/users/Codertocat/followers",
      "following_url": "https://api.github.com/users/Codertocat/following{/other_user}",
      "gists_url": "https://api.github.com/users/Codertocat/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/Codertocat/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/Codertocat/subscriptions",
      "organizations_url": "https://api.github.com/users/Codertocat/orgs",
      "repos_url": "https://api.github.com/users/Codertocat/repos",
      "events_url": "https://api.github.com/users/Codertocat/events{/privacy}",
      "received_events_url": "https://api.github.com/users/Codertocat/received_events",
      "type": "User",
      "site_admin": false
    },
    "labels": [
      {
        "id": 1362934389,
        "node_id": "MDU6TGFiZWwxMzYyOTM0Mzg5",
```

HOW DOES OUR APP CONTROL GITHUB?

The Octokit.Net library we use simplifies how we interact with GitHub.

- We store a private token in environment variables, which we use to generate another token, which we use to authenticate to GitHub.
- Once the Octokit API client has this token, we can issue requests (like creating comments, merging PRs, etc.).



```
public async Task AckAddToQueueAsync(MergeData data)
{
    var client = await GetInstallationClientAsync(data.InstallationId);
    log.LogInformation($"ACKing to PR comment {data.RepositoryOwner}/{data.RepositoryName}#{data.PullRequestNumber}");

    var message = $"Ok @{data.MergeIssuer} , I'll merge this Pull Request at `{data.MergeTime}` UTC + about 5 minutes. (Currently
    await client.Issue.Comment.Create(data.RepositoryOwner, data.RepositoryName, data.PullRequestNumber, message);
}
```

QUEUES: PUBLISH SCHEDULER’S “SECRET SAUCE”

Great, now our app is hooked up with GitHub and we can hack on reading the webhook payload to do something with it.

From a user’s POV, our app does the following:

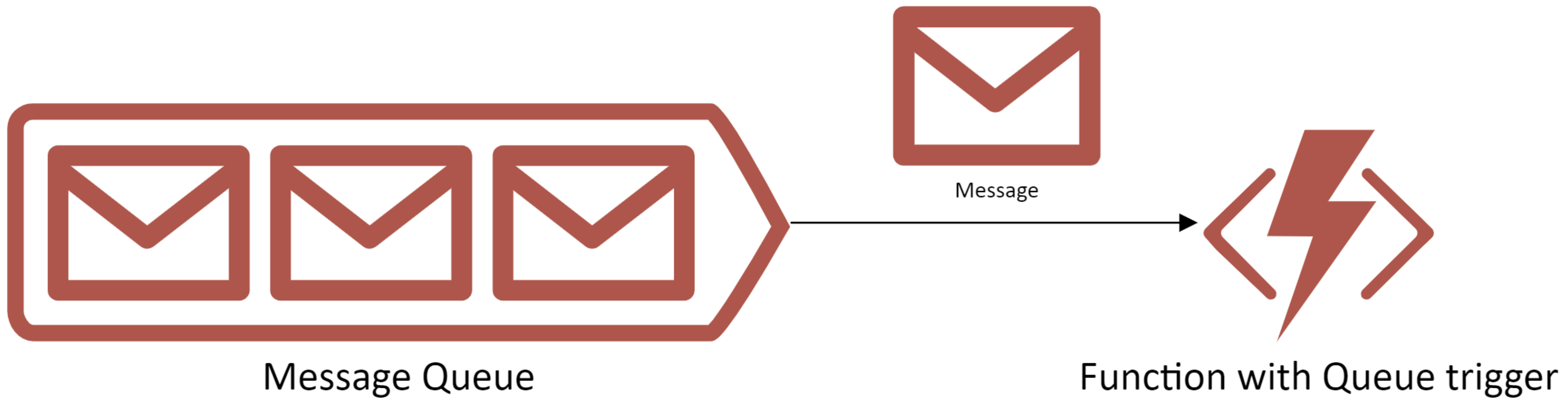
1. We leave a comment for it to go merge the current PR in some time in the future (can be months).
2. The app acknowledges this comment.
3. The app waits for the amount of time requested.
4. The app merges the current PR.

How do we delay for months in a way that is effective, while still being accurate within a few minutes?

AZURE QUEUE STORAGE

Azure Queue Storage is a message queue system that works great with Functions. It's just a queue, First In First Out.

When items are added to Queues, they can trigger Functions which consume the message provided.

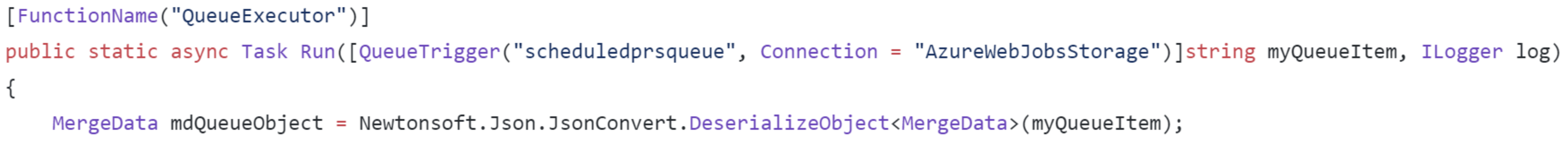


AZURE QUEUE STORAGE

Azure Queue Storage is a message queue system that works great with Functions. It's just a queue, First In First Out.

When items are added to Queues, they can trigger Functions which consume the message provided.

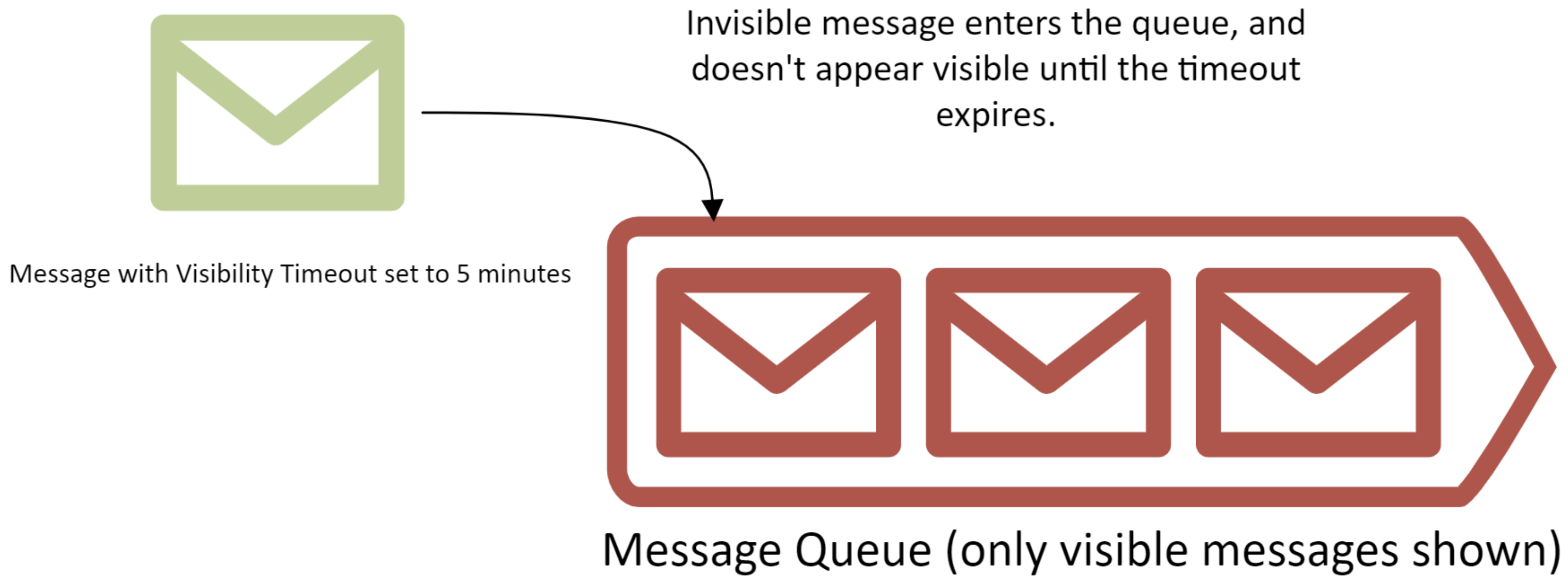
In this example, this "QueueExecutor" function listens to the "scheduledprsqueue" for messages, and converts them from JSON.



```
[FunctionName("QueueExecutor")]
public static async Task Run([QueueTrigger("scheduledprsqueue", Connection = "AzureWebJobsStorage")]string myQueueItem, ILogger log)
{
    MergeData mdQueueObject = Newtonsoft.Json.JsonConvert.DeserializeObject<MergeData>(myQueueItem);
```

QUEUES: PUBLISH SCHEDULER'S "SECRET SAUCE"

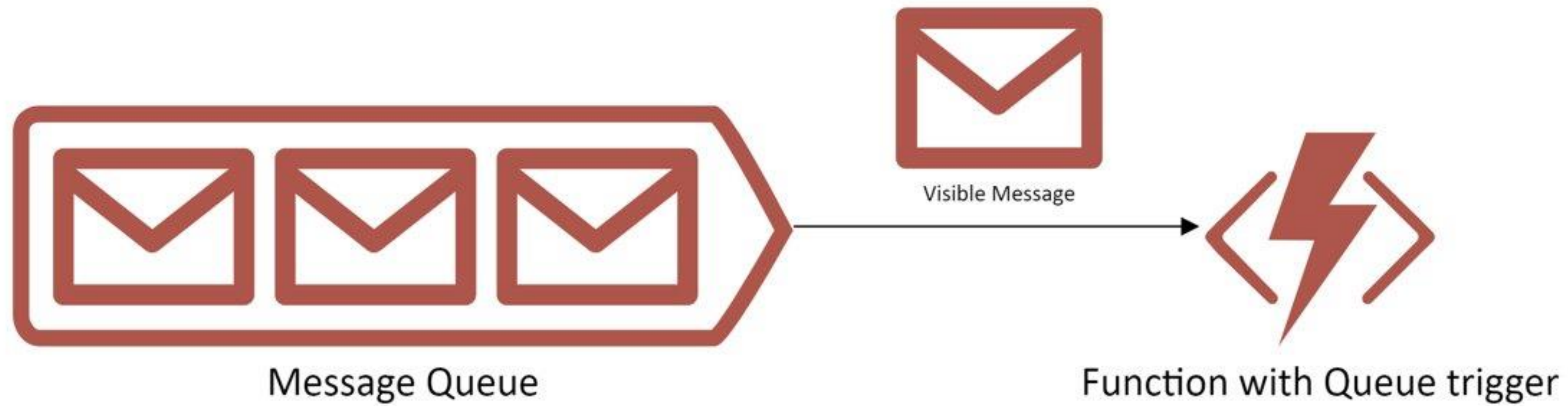
When inserting messages into the queue, we can specify a "visibility timeout". Messages with a visibility timeout will not appear in the queue until this timeout expires, which can be up to a week.


Invisible message enters the queue, and doesn't appear visible until the timeout expires.
Message with Visibility Timeout set to 5 minutes
Message Queue (only visible messages shown)

QUEUES: PUBLISH SCHEDULER'S "SECRET SAUCE"

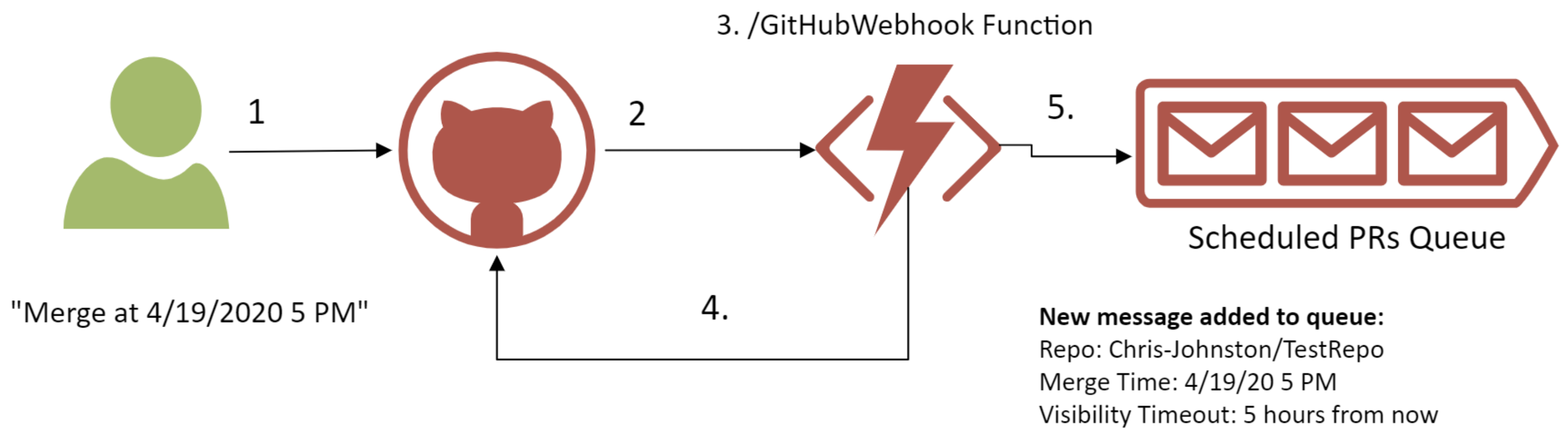
Once a message becomes visible, it can be picked up by a Function. This way, we can delay processing this message for up to a week at a time, without any compute from us.

But how do we wait for longer? We can re-add to the queue!



PUTTING IT ALL TOGETHER

How Commands Are Issued

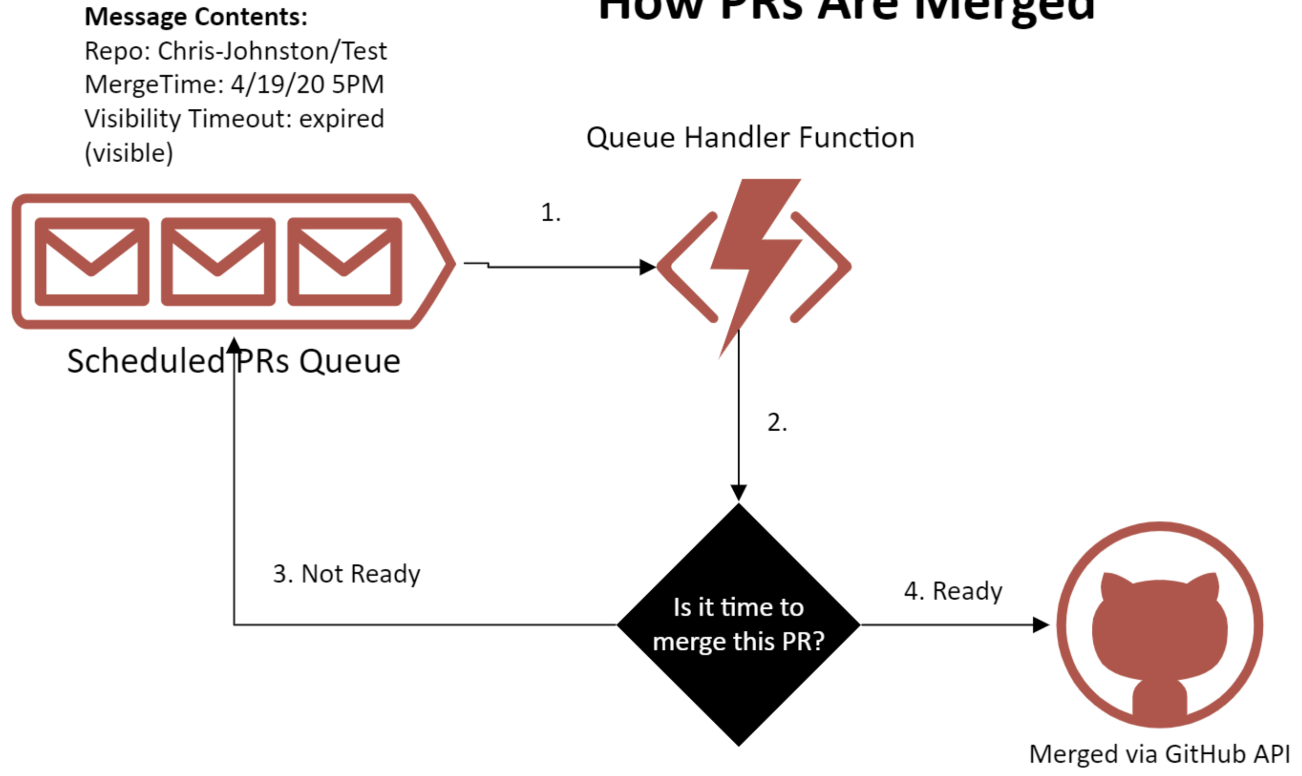


New message added to queue:
Repo: Chris-Johnston/TestRepo
Merge Time: 4/19/20 5 PM
Visibility Timeout: 5 hours from now

1. User comments on a GitHub repo.
2. GitHub POSTs our webhook with JSON.
3. GitHubWebhook Function executed. Reads the payload from GitHub and checks user permissions, parses message content, etc.
4. Function posts a comment to GitHub which acknowledges the request.
5. Function adds a message on to the queue, which contains the details of the request with the "visibility timeout" option set.

PUTTING IT ALL TOGETHER

How PRs Are Merged



1. Queue message visibility timeout passes, and triggers the Queue Handler function.
2. Queue Handler Function checks if PR is ready to merge.
3. If not ready, re-calculates the new visibilty timeout, and re-queue's the message.
4. If ready, merges PR in GitHub.

ACKNOWLEDGEMENTS

This project was built by:

- Chris Johnston: https://github.com/Chris-Johnston
- Dani Halfin: https://github.com/DaniHalfin
- KC Cross: https://github.com/KCCross

Resources:

GitHub API: https://developer.github.com/v3/

Azure Functions Docs: https://docs.microsoft.com/en-us/azure/azure-functions/

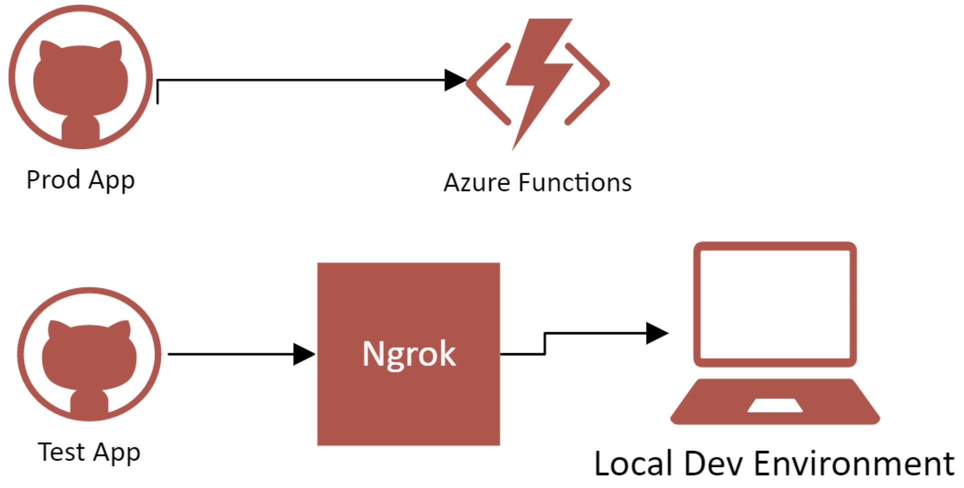
Azure Queues Docs: https://docs.microsoft.com/en-us/azure/storage/queues/storage-queues-introduction

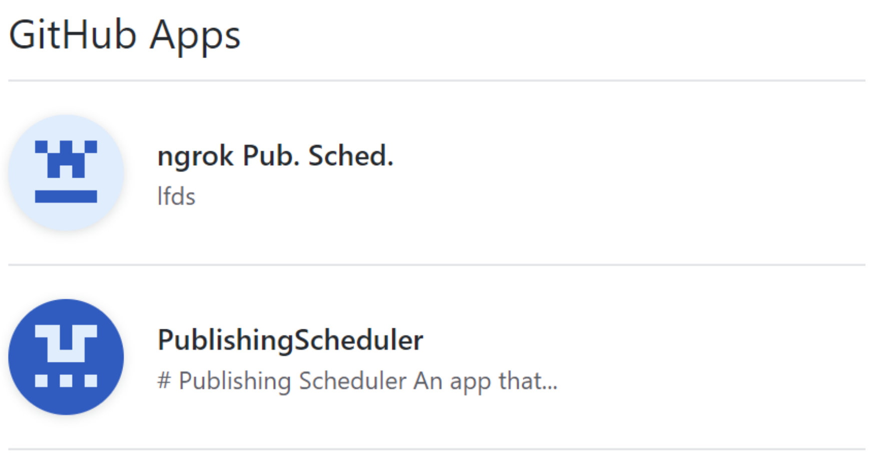
EXTRA TIDBIT: NGROK

Ngrok (https://ngrok.com/) is a fantastic tool for hackathons. It can expose your server running on localhost to a public endpoint which can be reached by other computers, phones, and even services like GitHub!

Here's how we used it with Publish Scheduler:

- "Prod" webhook goes to Azure Functions
- "Test" webhook goes to Ngrok, which goes to our dev machine for debugging locally




GitHub Apps

ngrok Pub. Sched.
lfds

PublishingScheduler
Publishing Scheduler An app that...