Logo

Description automatically generated

ScriptPilot

Usage & Maintenance Manual

Version 1.0

Contents

[Overview 2](#_Toc84613525)

[Usage 3](#_Toc84613526)

[Authentication 3](#_Toc84613527)

[View Scripts 4](#_Toc84613528)

[Add Script 5](#_Toc84613529)

[Add Script Version 7](#_Toc84613530)

[Edit Script 7](#_Toc84613531)

[Delete Script 7](#_Toc84613532)

[Invoke Script 7](#_Toc84613533)

[Invoke Script (Externally) 7](#_Toc84613534)

[Enable Script API Key 7](#_Toc84613535)

[Download Script Code 7](#_Toc84613536)

[GitHub Backup 7](#_Toc84613537)

[Notable Limitations 7](#_Toc84613538)

[Maintenance 7](#_Toc84613539)

[AWS Lambda 7](#_Toc84613540)

[MongoDB 7](#_Toc84613541)

[Updating Tokens 7](#_Toc84613542)

[Heroku 7](#_Toc84613543)

# Overview

ScriptPilot is a zero-running-costs\* web application acting as a standalone platform to power user defined scripts. The scripts can be invoked locally or through external third-party services in order to perform computational work as defined by the script author. In the background, ScriptPilot relies on a service called [AWS Lambda](https://aws.amazon.com/lambda/), which is part of AWS (Amazon Web Services) to store and execute the scripts. This means that nearly all AWS Lambda concepts apply to ScriptPilot. Any scripts and changes made in ScriptPilot will be reflected in the AWS Lambda account linked to ScriptPilot. Scripts may even be modified through AWS Lambda with the exception of enabling API keys. Overall, ScriptPilot provides the following key functionality besides what AWS Lambda provides:

1. User-friendly UI to manage scripts. The AWS Lambda UI contains much more complex functionality, which is stripped down to the basics. Both may be used in tandem.
2. API Key generation. ScriptPilot is capable of randomly generating and storing encrypted API keys for each script and its particular versions. API keys can then be used for authentication when invoking a script externally.

Scripts operate in a serverless manner; meaning their code is only being executed by a computer in the cloud (hosted by AWS) whenever it is invoked.

\*Under reasonably high usage levels for a single organisation such as IndigoZest.

# Usage

## Authentication

Users arriving at scriptpilot.indigozest.co.uk will be greeted by a login page along with a login button. There is no sign up required. ScriptPilot integrates with IndigoZest’s Zoho CRM account and is able to identify its users. A user already logged in to Zoho in say, another tab, will automatically be recognised and logged into ScriptPilot upon clicking the login button. Otherwise, clicking the login button will redirect the user to a Zoho login page. Only certain users are allowed access on ScriptPilot, depending on their role in Zoho CRM: Administrator, Developer. The authorised roles may be adjusted by editing the application’s environment variables (see the Maintenance > Heroku section).

A picture containing text, nature, night sky

Description automatically generated

Figure 1 Login page

## View Scripts

Upon logging in, users are presented with a paginated list of all scripts. Hovering over the column names will bring up three dots – this is a context menu that allows for hiding columns, sorting, and search functionality. The same can be done for each script in order to interact with them, with the possibility to invoke them from this page.

Table

Description automatically generated

Figure 2 List of scripts

## Add Script

To start a script, click the New Script button after logging in.

A close-up of a logo

Description automatically generated with low confidence

Figure 3 New Script button

Adding a new script involves three parts: Configuration, Environment Variables, and Code.

Graphical user interface, application

Description automatically generated

Figure 4 New Script form

**Configuration:** these are key defining attributes of a script.

* Name: must be unique.
* Description: brief description of what the script does.
* Role (ARN): automatically filled in with the default role. This controls what permissions the script has within the AWS ecosystem (i.e. being able to execute other scripts). There should be no need to change this.
* Handler: this is the starting point of the script. When uploaded, a script’s folder may contain numerous files breaking up code (this is good), this just tells the computer where to start. The format is *filename*.*functionName*. In the case of Node.js scripts for example, this means the code folder should contain a file named index.js in the main directory, along with an exported function inside of it named handler.
* Runtime: scripts may be written in a variety of different languages and environment types.
* Memory Size: how much RAM memory is temporarily allocated towards invoking a script, up to 10 GB. The more memory that is allocated, the more powerful the CPU being used in the background. For regular scripts, 128 MB should be more than enough. More memory intensive scripts may benefit from a higher allocation of data (i.e. 256 MB or 1024 MB), resulting in faster execution times (testing at different memory sizes is recommended). Any memory increase that does not result in a considerable increase in performance should be avoided; higher memory allocation uses up more of the computational resources provided by the AWS Lambda free-tier (see Usage > Notable Limitations section).
* Timeout: should the script run for any longer than the specified timeframe in seconds, invocation will be halted. Note that this does not override any of the limitations listed (see Usage > Notable Limitations section).

**Environment Variables:** these are key-value pairs of data that are directly and globally accessible as variables within a script’s code (in the case of Node.js, these would be accessed by executing *process.env.VARIABLE\_NAME*).

**Code:** this should be a zipped file containing all of the code (including any module dependencies) necessary for the script to run. This code is directly uploaded to AWS for storage and there is a limitation to that storage size as well as each script’s file size (see Usage > Notable Limitations section). Code bundlers or minimizers such as [Webpack](webpack.js.org/) may be used to reduce the file size by up to 99% and even provide a minor performance boost as there is less code the computer needs to load. The name chosen for the zipped folder does not matter. Ensure only the contents of the script are zipped in order to maintain the file structure (i.e. the folder containing the script files should not be zipped as this adds an additional directory level).

Upon successfully adding a script, it will show up in the scripts’ list.

## 

## Add Script Version

Every script starts out with a version referred to as $LATEST. This version of the script can have its details edited at any time. It is recommended this version be reserved only for testing purposes. When deploying scripts in production, a separate version should be published; these have their configuration, code, and environment variables locked. This means that another service relying on the script won’t break if a new incompatible version is released. Each version represents a copy of that script frozen in time. An API key can be generated for each version individually (including the $LATEST version).

To add a version, click the Add Version button in the script’s details page.



Figure 5 Add Version button

Graphical user interface, text, application, email

Description automatically generated

Figure 6 Add Version form

Once published, the new script version will have a version number automatically assigned. All versions of a script can be viewed at the bottom of the script details page.

Graphical user interface, text, application, email

Description automatically generated

Figure 7 View of a script's versions

## 

## Edit Script

A scripts configuration and environment variables can be edited through the Edit button in the script’s details page. Editing a script’s details will only have the changes applied to the $LATEST version. Previously created script versions will be unaffected and are not editable.



Figure 8 Edit button

Graphical user interface, application, table

Description automatically generated

Figure 9 Edit Script form

## 

## Delete Script

A script and its associated code can be deleted through the Delete button in the script’s details page. Deleting the $LATEST version of a script will also delete all versions and their code while deleting a specific version will not affect any other versions.



Figure 8 Delete button

Attempting to delete a script will trigger a confirmation pop up. Upon confirming the deletion will be completed and the script will disappear from the scripts’ list.

Graphical user interface, text, application, email

Description automatically generated

## Invoke Script

Invoking a script

## 

## Invoke Script (Externally)

## Enable Script API Key

## Download Script Code

## GitHub Backup

## Notable Limitations

# Maintenance

## AWS Lambda

## MongoDB

### Updating Tokens

## Heroku