



Mick Mak

Follow

Feb 4 · 7 min read

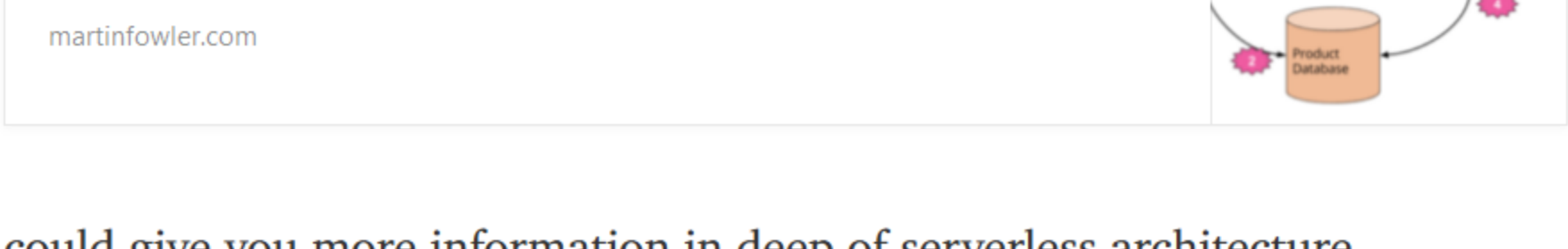
Save 80% of development time with Serverless architecture



Never miss a story from **MagicSketch Blog**, when you sign up for Medium. Learn more

GET UPDATES

I have been working for the backend of Magic Mirror for few months. From the beginning, the backend was a Ruby on Rails application hosted on heroku. Although Rails is a structural MVC framework, the project structure and program logic can be very different for different people. Finally, I used around two man-days to pickup with the project structure and then moved on to implement updates. This preparation process was really tedious and should be skipped in my thought. Fortunately, Magic Mirror website and the backend have to revamp and then we came up adopting serverless architecture in the new version. From learning to start developing, I used around 2 hours only! There is no complicated project structure, one service does one task. It is very simple and clear! For being serverless, I was really amazed by the advantages of it. Serverless is not a new concept but what is it about? This excellent article:



could give you more information in deep of serverless architecture.

...

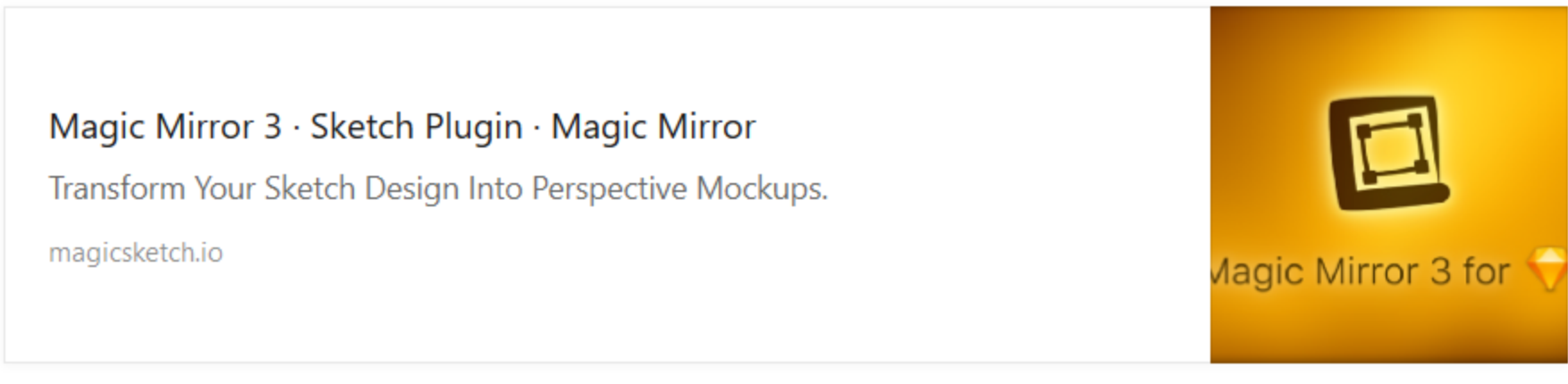
Why we adopt serverless?

1. One important advantage is about no maintenance. In our case, we are using hook.io(1) to store our backend logic and Airtable(2) to be the database. They are both providing very stable service and we do not need to care about the server stuff. There is also no server downtime during we deploy changes to production. It is almost nothing to do with the server if we want to scale up later.
2. Every endpoint is a small code snippet that simply does one job like register an user, handle a payment request, check license key valid etc.. There is no complicated project structure. Even if we hire virtual assistant to follow up the project, they can easily pickup. We are more available to outsource the backend development and are able to use the saved time to focus on other tasks such as business development. Imagine every freelance developer can save up to 80% time to pickup your project. It is really worth to spend some time to try serverless architecture.

...

How do we start with serverless?

Let's start from example. You could take a look on our running website on <https://magicsketch.io/mirror/>. It mainly consists of three parts.



1. **Github page(3) for the website front-end.** Github page provides a way to implement website with html templates + environmental configurations using Jekyll. You can find introduction below(3). All data communicate with database will go through API requests.
2. **Hook.io to be backend controller.** APIs on hook do communication among all other services we are using such as Stripe, Airtable(Our database), Zendesk etc.. Each API is a task in a hook service then you will be able to run the service with an URL. It can be an API call returning a JSON to let your front-end application handle display or can be a page displaying the result in HTML format after the task is done. It depends on the task going to be completed within the small service. If you are interested in what hook.io can be done. [Documentation on hook](#) is a great source to pickup.
3. **Airtable to be our database.** We treat it as a database that provides RESTful API for us to handle records. (Node module of Airtable is available on hook.io, even better!). It is an incredible database choice although we may be using it in a tricky way.

As you may aware, we are choosing services that provide an easy-to-use interface to manage the data. It is because one of our concerns is to lower the barrier of picking up the whole project. It is why we choose these services over some traditional choices. With similar setup, you may now able to launch a product with stable and scalable servers in relatively low cost. There are other advantages worth to say about choosing Airtable to be database. It is providing forms for us to create forms to let user submit their sketch template and views for us to setup a staging environment of the website. It helps a lot on both development and administration work as we no need to build the same feature by our own. The UX is good and UI is also easy to understand also!

...

Findings of being serverless approach

Our service is running under this setup near half of a year, there are some interesting findings.

1. **It should be suitable for Agile development approach.** Serverless focuses on the function tasks. We need not to care about tedious file structure, complicated code life cycle provided by framework. It saves much development time which is not necessary to spend.
2. **API approach enables more choices on development.** Since all tasks are written as APIs at the beginning. We can run tasks with anything that can initiate HTTP request. Sometimes we can find some existing services to help us finish the job in a convenience way. In our case, we have setup Zapier for some tasks to do task automation.
3. **Less command line control.** It is because we mostly choose third-party to be part of our system. We can free a lot from controlling the server with command line interface. The graphic user interface provided by third-party service is a more friendly way for the team to control everything. But some people may love using CLI over GUI. It can be both good point and bad point. At least we do not need to worry about our virtual assistants doing rm -rf on our production server.

“Anything can possibly go wrong, it does.” We encountered some challenges for sure. Let me summarise them.

1. **We cannot control maintenance period of third-party services.** As we are using third-party service in our serverless structure. They will possible have maintenance period. The maintenance possibility is especially high on the day that you are going to launch new features. No figure for this but it's real! For the sake of providing good user experience even if third-party service is down, we have to think of setting up extra service to make every transaction can be continued. For example, we will log important actions like payment, subscription to Slack channel in order to trace back the actions of an user and then provide corresponding support for them. We will also use Zapier(4) to do automation tasks for key generation or logging. It brings us details of each step about how services interact with each other and the action performed can also be replayed. The extra setup for this problem can be different but we should think of it when we are relying on third-party services.
2. **Data backup.** Unlike traditional database choice, we are using Airtable as database. We have to find a way to automatically perform backup periodically. Our solution so far is using Zapier automation.
3. **Unit testing.** Thanks to API development approach, we can easily create unit testing for every API endpoint.
4. **Data migration.** This can be a quite challenging task as we are not using traditional database server. We need to due with the data on Airtable if we would like to migrate the data to elsewhere in the future. For now, the data backups are controlled in excel format.
5. **Source version control.** Source version control is a must for development. In hook.io, it provides a way to build a task with hook hosting the source but it is not recommended. The editor does not provide good version control function. That means we have to keep track of version control somewhere else. So the Github repo integration is really useful on hook.io. This problem may possibly appear on other serverless approach.
6. **Possibly slow API response time.** With more third party service involving in a task, the response time will be longer. It needs to be aware of the timeout limit of each API. The performance of each API also needs to be optimized.

...

We are fresh to this development approach so we are still finding the best solution for existing problems. Up till now, we have prepared solutions to handle some of them. For example, we have built a replayable URL API gateway to mask third-party API link and provide replay function for every transaction. We are using Zapier automation tasks for logging, operations, backup. We have set up staging environment for our backend. Let's discuss it in future blog post.

...

Links:

- (1). Hook.io: <http://hook.io/>—Hosting nano-services.
- (2). Airtable: <https://airtable.com/>—Airtable works like a spreadsheet but gives you the power of a database to organize anything.
- (3). Github page: <https://pages.github.com/>—Websites for you and your projects, hosted directly from your GitHub repository. Just edit, push, and your changes are live.
- (4). Zapier: <https://zapier.com/>—Zapier makes it easy to automate tasks between web apps.


Serverless Backend Development Web Nodejs

4










Mick Mak

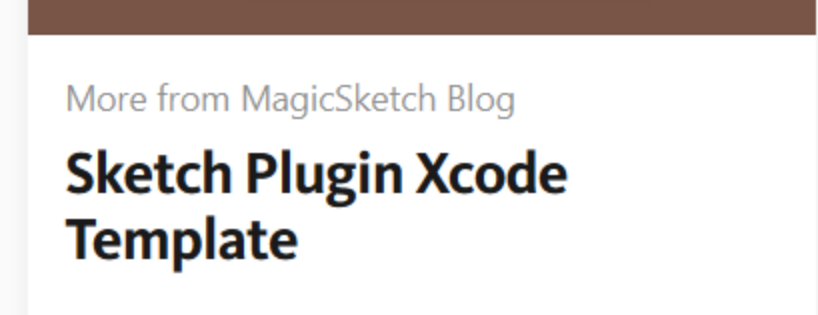
Follow



MagicSketch Blog


Sketch Plugins—Magic Mirror, Magic Presenter Updates, Tips & Tricks, etc.

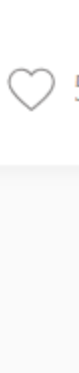
Follow

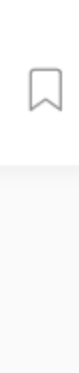


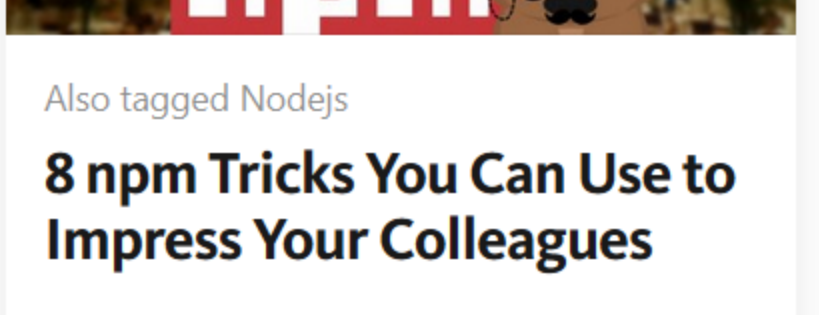
More from MagicSketch Blog

Sketch Plugin Xcode Template

James Tang
4 min read


51

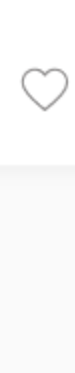


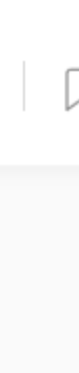



Also tagged Nodejs

8 npm Tricks You Can Use to Impress Your Colleagues

Adir Amsalem
6 min read

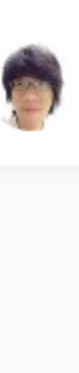
583

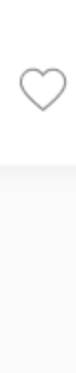


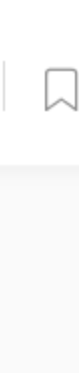


More from MagicSketch Blog


Debugging Sketch Plugins

James Tang
3 min read

22



Responses

Write a response...