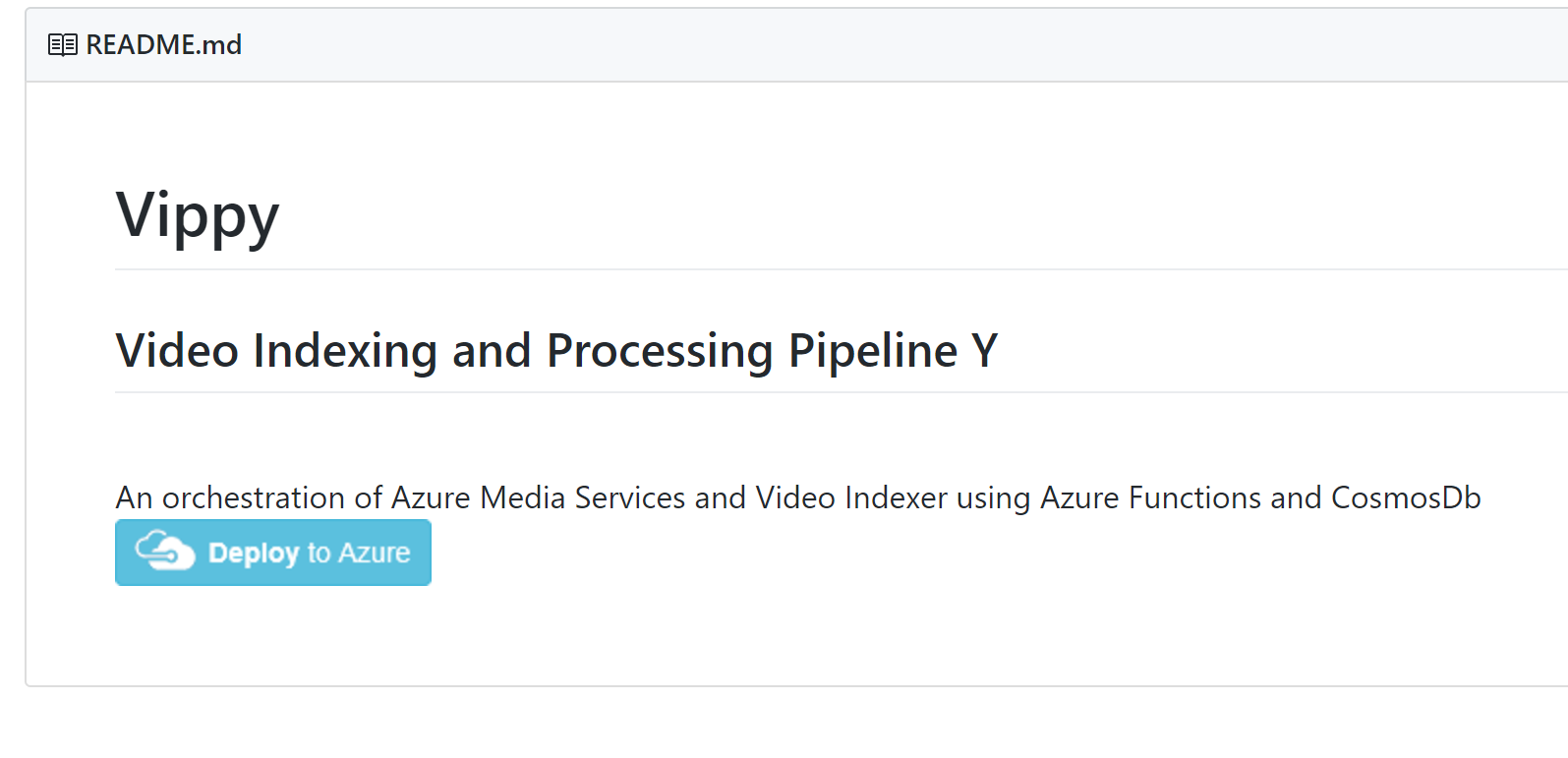
**Vippy Deployment Guide**

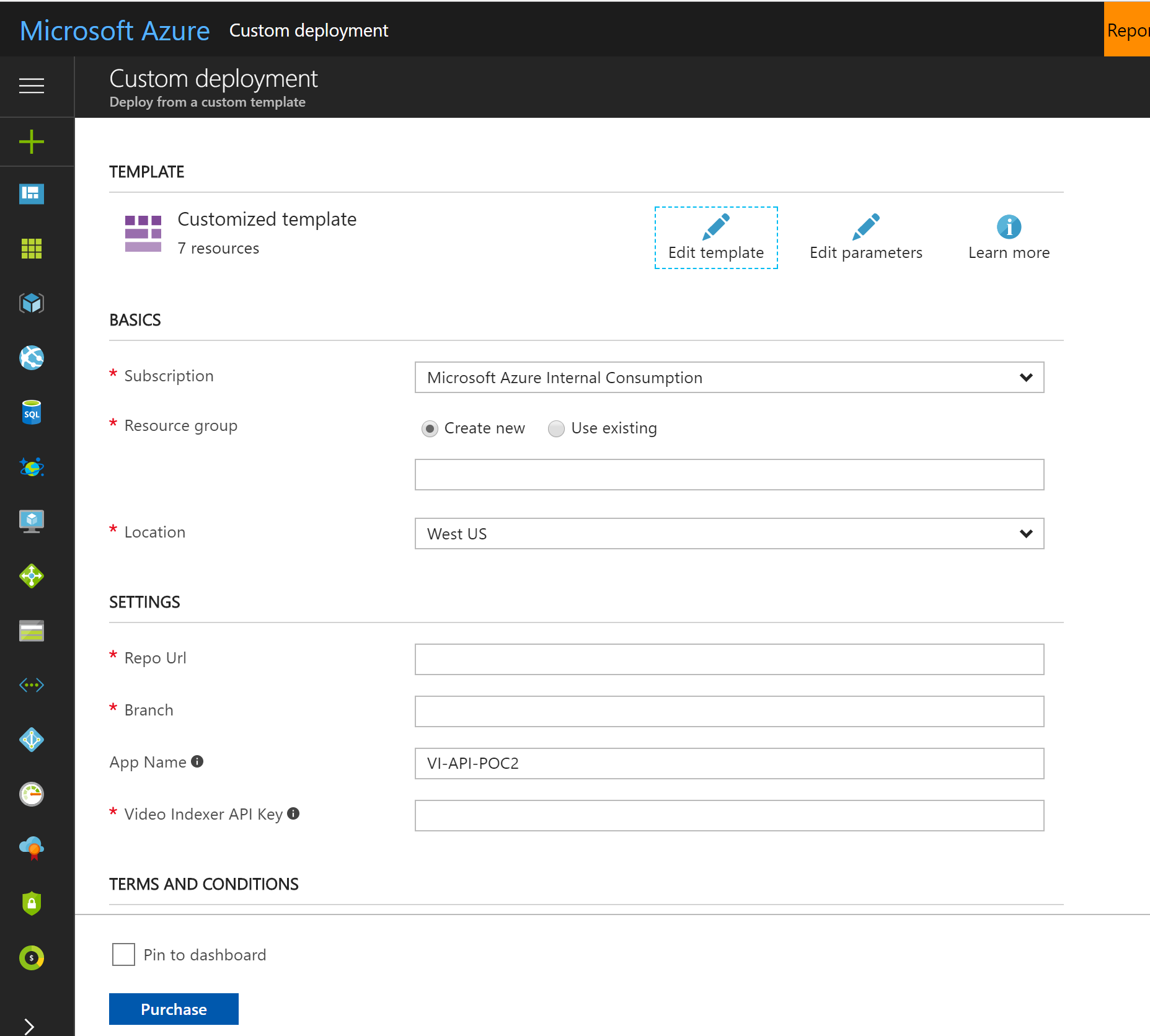
Follow the steps in this guide to deploy Vippy into your own Azure Subscription. The process includes deploying an ARM template to create a bunch of assets in a Resource Group in your Azure subscriptions, syncing source from your own fork of the Vippy repo, and updating some settings of the Function app in the Azure Portal.

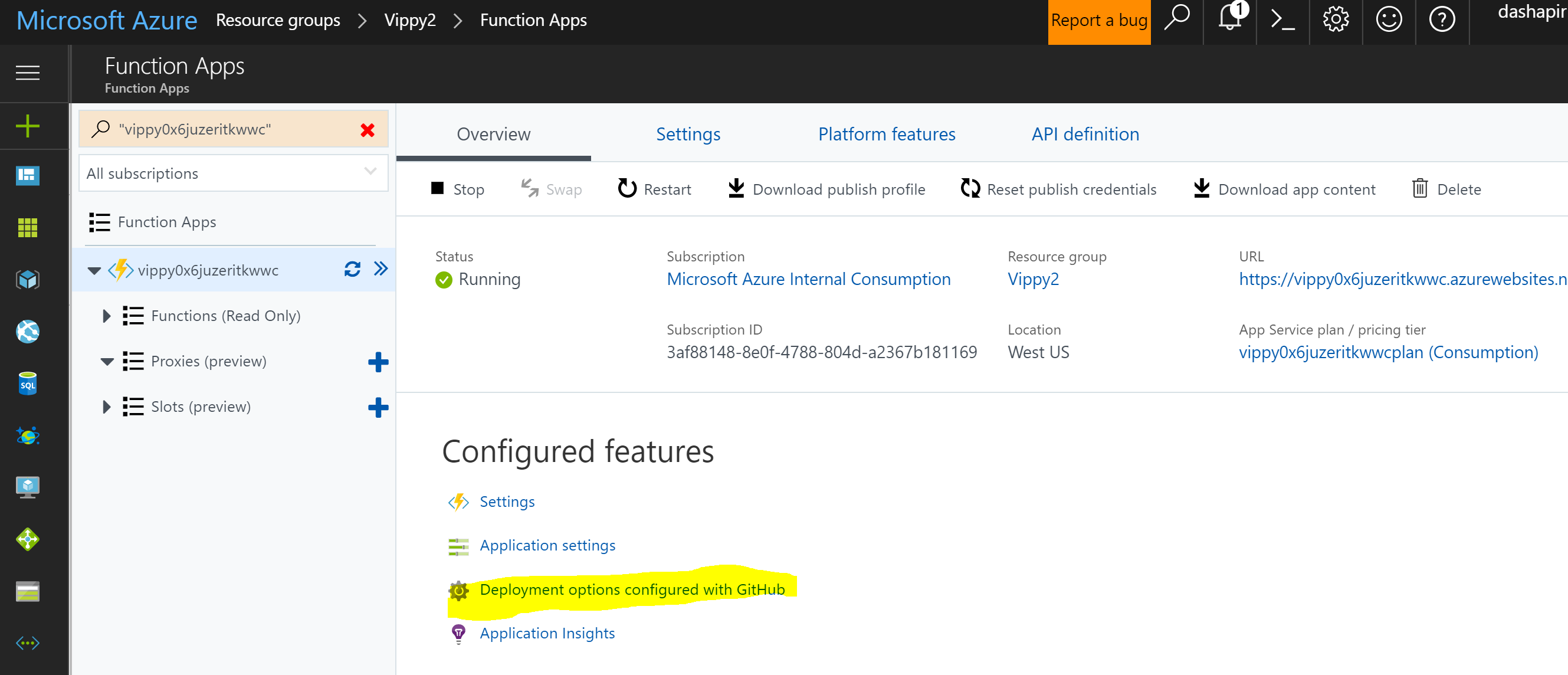
**Steps**

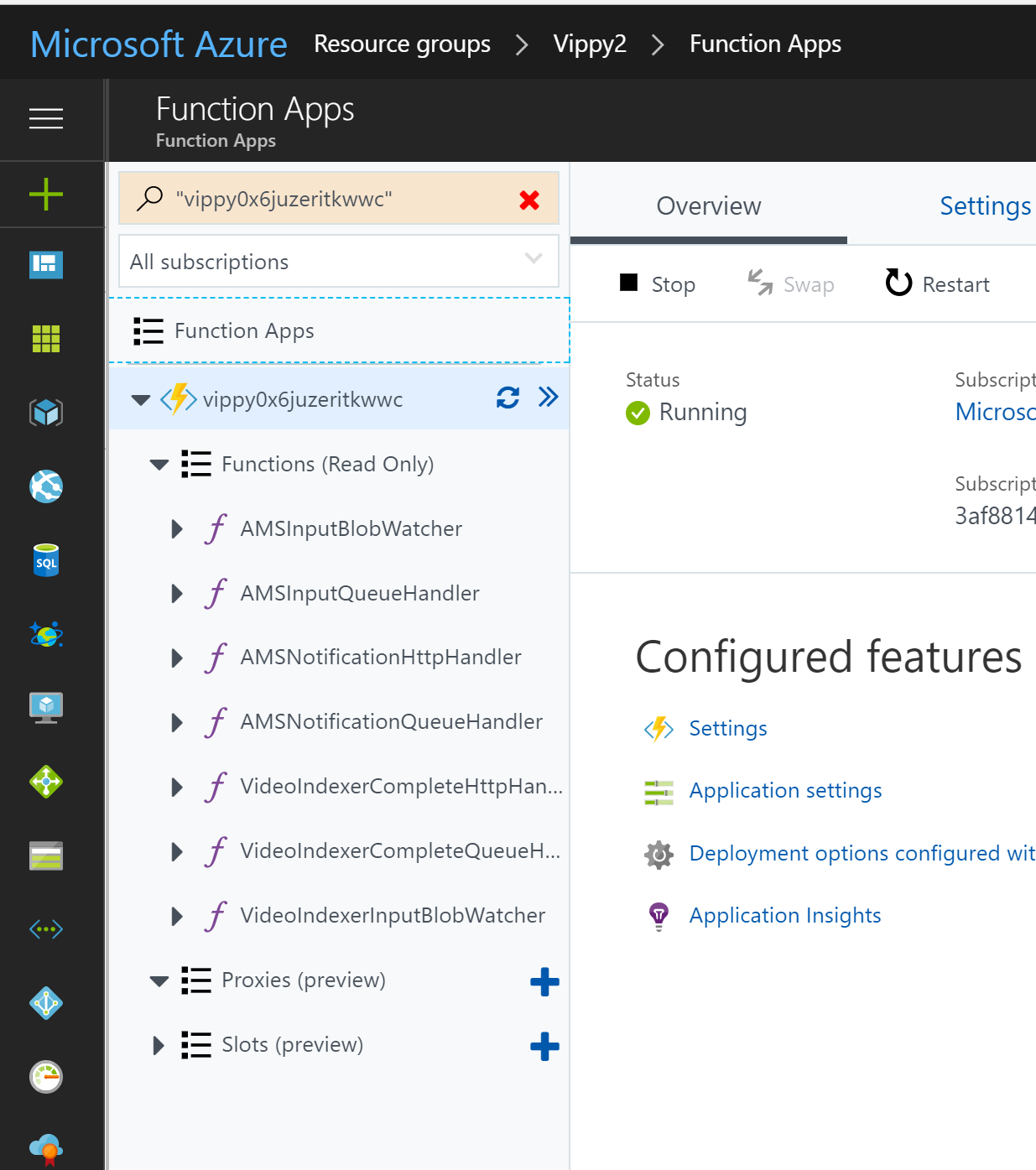
1. Fork the Vippy repository from <https://github.com/DarinShapiroMS/Vippy>. It’s important not to sync your Vippy deployment with the main Vippy repository to give you control over the source code.
2. Click the ‘Deploy To Azure’ button on the README.md of the Vippy repository, as shown below.



1. Once the template loads in the Azure Portal, fill out the form using the repository url of your fork of Vippy instead of the main repository’s url. You’ll also need to give the app a name and enter the API Key for your Video Indexer Account. Your api key can be retrieved from the Video Indexer API management site at <https://videobreakdown.portal.azure-api.net/products>.



1. When the deployment is finished, navigate the Azure Portal to the Function App that was created in the resource group you specified for the deployment. You should see something like the following screen. 
2. Click the ‘Deployment options configured with GiHub’ button, then click ‘Sync’.
3. When the source code sync completes, you should see the list of Azure Functions now on the Functions page.



1. Click on the ‘Application Settings’ button so we can change some default settings.
   1. You need to change ‘MediaServicesNotificationWebhookUrl’ to the actual url of your 'AMSNOtificationHttpHandler' function. If you go back and click on that function, there is a button to display the function url that includes the key necessary to call the function. Paste the value of that url into the App Setting for MediaServicesNotificationWebhookUrl’ and click save.
2. Create blob containers based on app settings values for videoIndxerBlobInputContainer and amsBlobInputContainer keys - set container security to private.
3. Restart the Function App service then you’re ready to test.

To test if the app is fully functional, upload a video to one of the watch folders from step 8. The end result after processing will be a data inserted into Cosmos Db. Status monitoring of current executing jobs will be coming soon.