medium.com

Cloud Functions vs Kubernetes Engine - Statuscode - Medium

James Wilson

6-7 minutes

Updated Aug 2019.

Google's compute line-up offers a lot of great choices. Two of the best technologies on GCP are Kubernetes Engine and Cloud Functions. Both are powerful and as a developer it is easy to default to Cloud Functions because it takes a lot of administrative work out of my hands. This administrative work, albeit declarative yaml files is an important aspect to configuring Kubernetes Engine.

The original premise for this article was a developer simply asking me, "when would you choose Kubernetes over Cloud Functions?". It's a question I have considered a lot myself since working with

both of these, and so I think the best way to handle this is to default to Cloud Functions and explain the instances where Kubernetes Engine would be a better choice. While I do not catalog every reason, here are the top several that play into choosing Kubernetes.

3 languages are not going to cut it

When using Cloud Functions at the moment you have only three development environments to choose from and that is Node.js, Python, and Go. This is an incredible technology and it is powerful.

Kubernetes Engine offers you freedom because the pods you are creating are isolated environments that can run any language and runtime. You might be a .NET shop and need to leverage C#. I have entertained the idea of using the Core Foundation libraries from Apple in a service. That service will need to be written in Swift. These are just some of the cases that might involve using a different language and set of frameworks. In the future, Cloud Functions will support more of these technologies, but that will take quite a few years before it is in production.

Speed

Speed, and I mean right now, not in 200 ms. This is a fundamental difference. There are instances when you just want to get data and push it somewhere. If a Cloud Function is not used for sometime then all instances of that function are shut down. This is a good thing, you are not paying anything if you are not using it. However, there might be instances where you just don't want to wait for that cold start time. If that isn't an option then Kubernetes will have your back, you are already a cluster and you can have a couple pods running for that particular service ready to jump into action when the need arises.

Heavy processing and large workloads

Functions are great for connecting various services together. However, they are not meant for heavy or long running tasks. They are short on CPU and Memory compared to Compute, Kubernetes, and App Engine. They have a much quicker timeout time at 5 minutes which means the work needs to be done quickly and you need to return from the function quickly.

Plus, it doesn't handle heavy loads well. If you're going to do a lot of image processing or big data analysis on a cloud function you'll run into trouble. With Kubernetes Engine you have the ability to scale pods based on various parameters such as high CPU, memory, etc. With cloud functions you don't have the ability to select CPU and the memory allocation is pretty low compared to the other compute services.

Invocation Madness

How many times are you invoking the function? Is it one hundred or one hundred thousand times in a day? It's different if your cloud function is relying on a firebase database trigger, at that point it's worth settling for the Cloud Function. However, if you are trying to build a service that is going to be validating emails or simply ingesting a massive amount of data then it's worth considering Kubernetes. First, you can always have a few pods running so you're reducing any latency on the service. Second, with Cloud Functions part of the price is how many times a function is invoked. With Kubernetes you are able to scale up to more instances during peak times than scale back down. It doesn't matter if your service is invoked ten thousand times, at this point you are paying for the number of nodes and pods that you are running which will reduce your overall costs.

Microservice Communication

Finally, we have service-to-service communication. Cloud functions have some really powerful triggers for both Firebase and GCP. For

instance, you can setup a Cloud Pub/Sub trigger or trigger another function by uploading files to Cloud Storage. In addition, we have HTTP triggers which are helpful for creating web hooks.

However, what if you have several services that don't need to be triggered, but you just want them to talk to each other? At the moment talking and communicating between services isn't really an effective approach when using Cloud Functions. Think of the deployment process alone. With Cloud Functions this is cumbersome as you start updating a bunch of services on Google Cloud. Meanwhile, with Kubernetes you are simply uploading your configurations and Kubernetes is ensuring that the environment is running properly for you. Yes there is a lot more work upfront to create the yaml files, but then it's super simple to keep that infrastructure up and running.

Ultimately, it depends on what you are building and what your needs are, but if you are juggling the idea of calling multiple HTTP function triggers based on one call into your Cloud Functions then you should step back and ask yourself whether Kubernetes is a better option for the 90% intercommunication that is occurring.

This list is not exhaustive and it is definitely open for interpretation. Again, it's based on your needs as a company and organization. Kubernetes is still growing at a rapid pace and not everyone has a developer/team on hand that can manage a Kubernetes Engine project for them. On the other hand, maybe you have a lot of .NET Core services that you want to deploy and Cloud Functions just won't cut it because you need to use Node.js, Python, or Go. It's always worth taking a step back and thinking about the different technologies at play and how we can leverage them to be as productive as possible.

James Wilson is a developer building distributed systems using Go and Google Cloud. He is also a Pluralsight author and you can view

his courses here.