(1) Should we move to GKE? Why? ***mohammed/shahd***

Google Kubernetes Engine is a management system for Docker containers and container clusters within Google's public cloud services. It serves as a way to deploy, scale and manage kubernetes. It allows for easy cluster creation, load balancing, auto upgrades, logging monitoring and most importantly auto-scaling. Based on the demands of your workloads, GKE will autoscale the cluster's node pools. This is extremely helpful especially when clients are increasing in your company, as it will reduce the manual workload of scaling, and can increase the availability of your workloads when needed. GKE's cluster autoscaler automatically resizes, so there is no need for manual work. All you need to do is specify the minimum and maximum node pool size. Additionally, Cloud Functions service runs code written in Node.js, Java, Python, and C++ which are all APIs that are used in the company. In terms of API use, GKE has two primary API integrations, the google kubernetes engine API and simply the kubernetes engine API. Both integrations have unique features that can be indulged by the user. The google kubernetes engine API is primarily used to configure the users cluster on their google cloud, therefore this api allows the user to create or delete a cluster, configure cluster-level networking, update kubernetes, add or remove or modify node pools in the cluster, setting machine types/node images, configure the location on where cluster is deployed. However, the Kubernetes API allows a heavier approach towards containerized applications and workloads. This includes features such as deploying an application, scaling an application, configuring intra-cluster networking, configuring containers and pods and controlling when pods are evicted or restarted.

Azure features also allow use of docker containers containing Node.js. Java. Python, and C++ code. Scalability in Azure features is slightly different than GKE, as it has a maximum number of instances unlike GKE which autoscales based off of minimum and maximum nodes. The greatest possible number of instances in Azure functions is 200 in the consumption plan which automatically scales based off of the number of incoming trigger events.

References:

https://cloud.google.com/kubernetes-engine

https://cloud.google.com/kubernetes-engine/docs/quickstart

[Conclusion: What is the better one to use in the client's situation, and why is that one better than the other option?] ***Mohammed***

In conclusion, it would be highly recommended that the client should transfer to GKE. Azure has extremely limited scaling and API use, whereas GKE has a broad variety of features that can help execute the clients project more efficiently. Specifically, GKE has a complex internal system which allows for the system to handle a much larger load than what Azure can. This is because GKE uses a variety of factors such as the type and number of node pools, number of pods available and much more to increase load handling. However Azure uses an autoscale system to determine load size, however it can only hold a maximum of 200 instances which creates a limited foundation to the clients project. In terms of API use, GKE integrates the use of API through two different functions that essentially work hand in hand. Those functions

are GKE API and Kubernetes API, these two functions help in creating the structure and the usages of the API. It is also recommended to deploy the API with ISTIO as ISTIO runs on kubernetes, it allows for diverse functions of deployment needs. This essentially allows ISTIO to run much more efficiently, allowing for enhanced secure communication, TLS encryption, authentication functions and much more. Although Azure does have beneficial usages in regards to web API's, our client is not working with web API's making Azure functions essentially meaningless.