

Missives about mostly GCP related things (<https://grumpygrace.dev/>)

◀ Home (<https://grumpygrace.dev/>)

GCP Flowcharts

Please note that I have no association with any training companies or third parties linking through to this post. This post is freely available to help folks understand Google Cloud!

For those of you who have been reading my GCP flowchart series over on medium the collection now lives here. This post contains all the ones that are still applicable at the time of writing that I posted on medium here (<https://medium.com/google-cloud/some-more-gcp-flowcharts-dc0bb6f7c94e>) together with a few brand new ones. I've now arranged them under the following headings:

Compute

Storage and Data

Security

Networking

Misc

So it's easy to find the one you want. This single post also allows me to maintain an up to date collection from one place.

Once I have more than 1 flowchart for a topic/ area I will create a new heading ,for now those singletons are under misc.

Attribution: All graphics & flowcharts apart from ones I drew myself & Sara's cheerfully copied from the Google Cloud platform (<http://cloud.google.com/>) or blog site (<https://cloudplatform.googleblog.com/>)

latest additions - Nov 30th 2019: Hybrid & multi-cloud logging & monitoring patterns (misc)

And this (<https://www.skimspace.com/Object/33/whats-the-right-choice-for-me-on-google-cloud-platform>) great interactive picker was created based on this collection of flowcharts. Go have a look it's really cool 😊

Compute

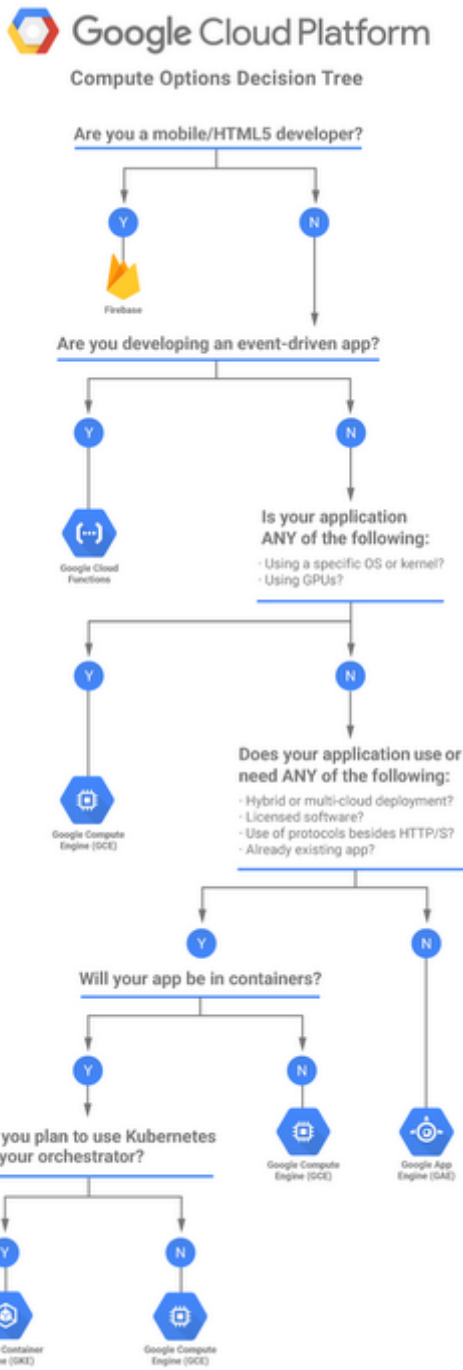
Which compute option ?

Even with the increasing popularity of serverless options traditional Compute options are very much in demand. I know I know I'm using traditional and including App engine & Kubernetes but even k8s is 5 years old now (at the time of writing June 2019) so I think I can get away with that :-) So choosing a traditional compute option flowchart is still very much valid

GCP has a continuum of compute options which can be graphically depicted as:



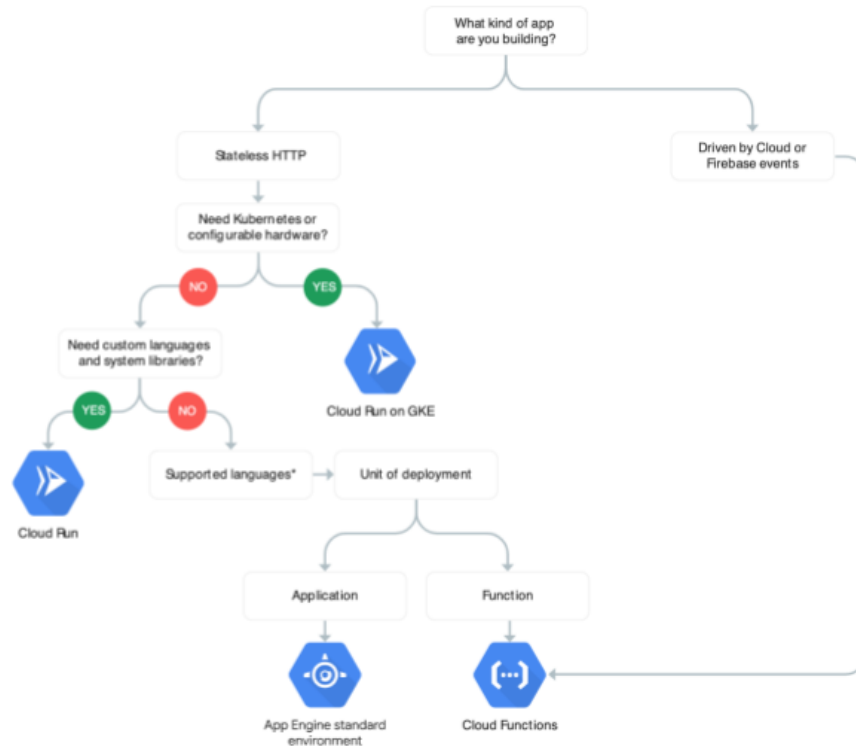
It may be obvious at either end of the continuum which option you choose but the decision becomes less straightforward in the middle so flowchart to the rescue :



The compute flowchart with accompanying words can be found here (<https://cloudplatform.googleblog.com/2017/07/choosing-the-right-compute-option-in-GCP-a-decision-tree.html?m=1>) and a nice table comparing the compute options is here (https://cloud.google.com/docs/choosing-a-compute-option#comparing_options).

Which Serverless (compute) Option?

If you want access to compute power where you just want to write the code and not have to worry about the underlying infrastructure then the serverless options are for you. Basically GCP takes care of the servers that are actually lurking way underneath the abstraction for you as well as the provisioning (scaling up & down) .



GKE by itself is not serverless as fits this description as you still have to define and configure way too much it's not just a here's my code and here you go through but it does provide the platform for a serverless platform as you can see in the flow chart. But the sharp eyed amongst you may have noticed that App Engine can be considered a serverless service although it's also included in the what I call traditional compute option

The flow chart and words about GCP serverless options can be found here

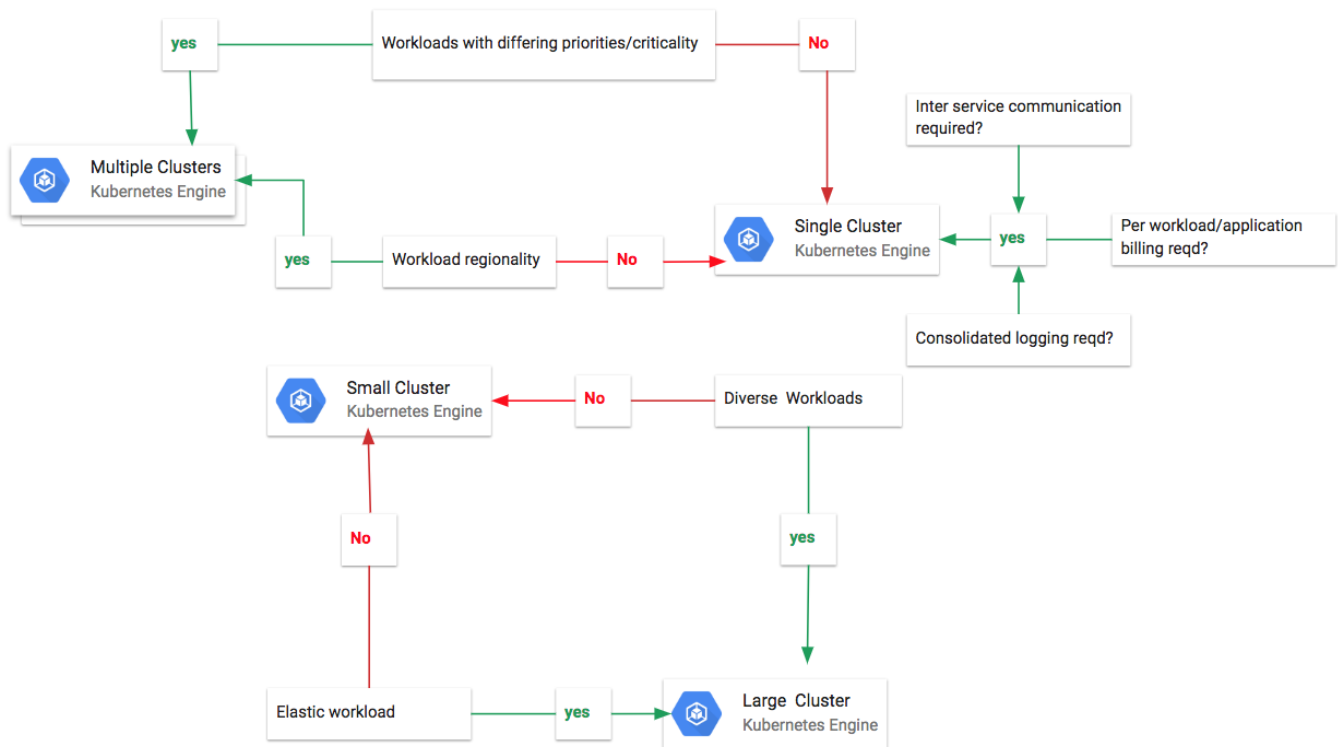
(<https://cloud.google.com/serverless-options/>) There's also a product comparison table

(https://cloud.google.com/serverless-options/#header_3)

Sizing & scoping GKE clusters to meet your use case

Determining the number of GKE (Google kubernetes engine) clusters and the size of the clusters required for your workloads requires looking at a number of factors. The article Choose size and scope of Kubernetes engine (<https://cloud.google.com/solutions/scope-and-size-kubernetes-engine->

clusters) discusses these factors. Alas it's sadly lacking a flowchart so I've addressed that for you (maybe at some point the article will include a flowchart). I know it seems I have created 2 mini charts but then it was a post about sizing & scoping your GKE clusters !

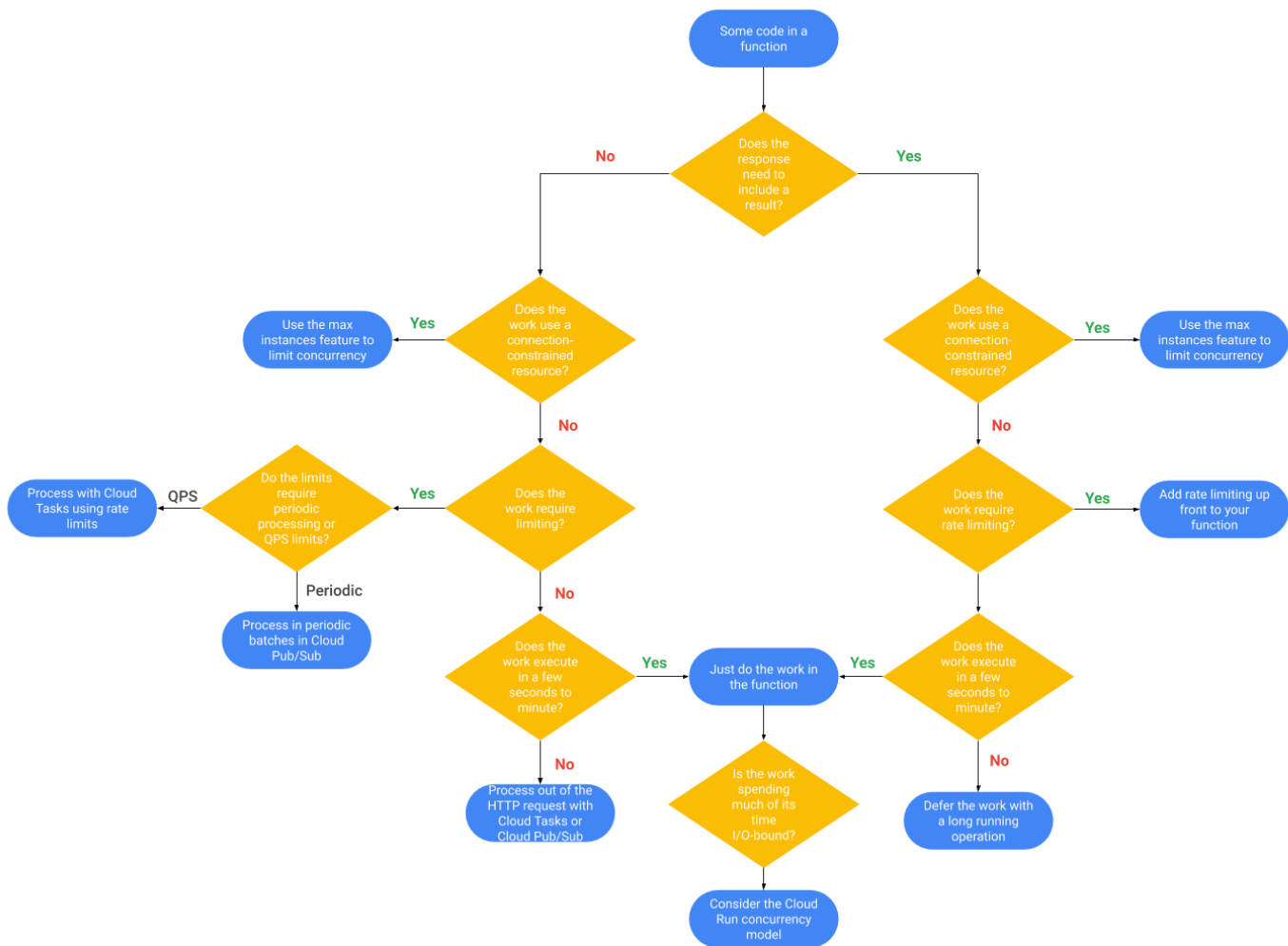


The words discussing the decision points are all in the article
<https://cloud.google.com/solutions/scope-and-size-kubernetes-engine-clusters>

Serverless Scaling Strategies

Write code, deploy it and the scaling will happen automatically for you that's the use of "serverless". That may be mostly true if your full stack auto scales but in a lot of cases that isn't the case and suddenly you do need to start worrying about backend services such as a database for example that has rate and connection limits. To help you with architecting your serverless applications built with GCP so they scale effectively my colleague @ptone wrote about 6 strategies you can adopt here <https://cloud.google.com/blog/products/serverless/6-strategies-for-scaling-your-serverless->

applications) . And yes he included a flowchart for your delectation to help you figure out which strategy is the right one for your use case :



If after admiring that flowchart you want to dive deeper into rate limiting techniques using GCP there's this (<https://cloud.google.com/solutions/rate-limiting-strategies-techniques>)

Storage and Data

What Storage type?

Data data data data data! (Sung to the 60's Batman theme music (<https://www.youtube.com/watch?v=1qP-NglUeZU>)) . I struggle to think of any application where data isn't a thing . The myriad ways you can store your data is probably after considering the security controls needed the most

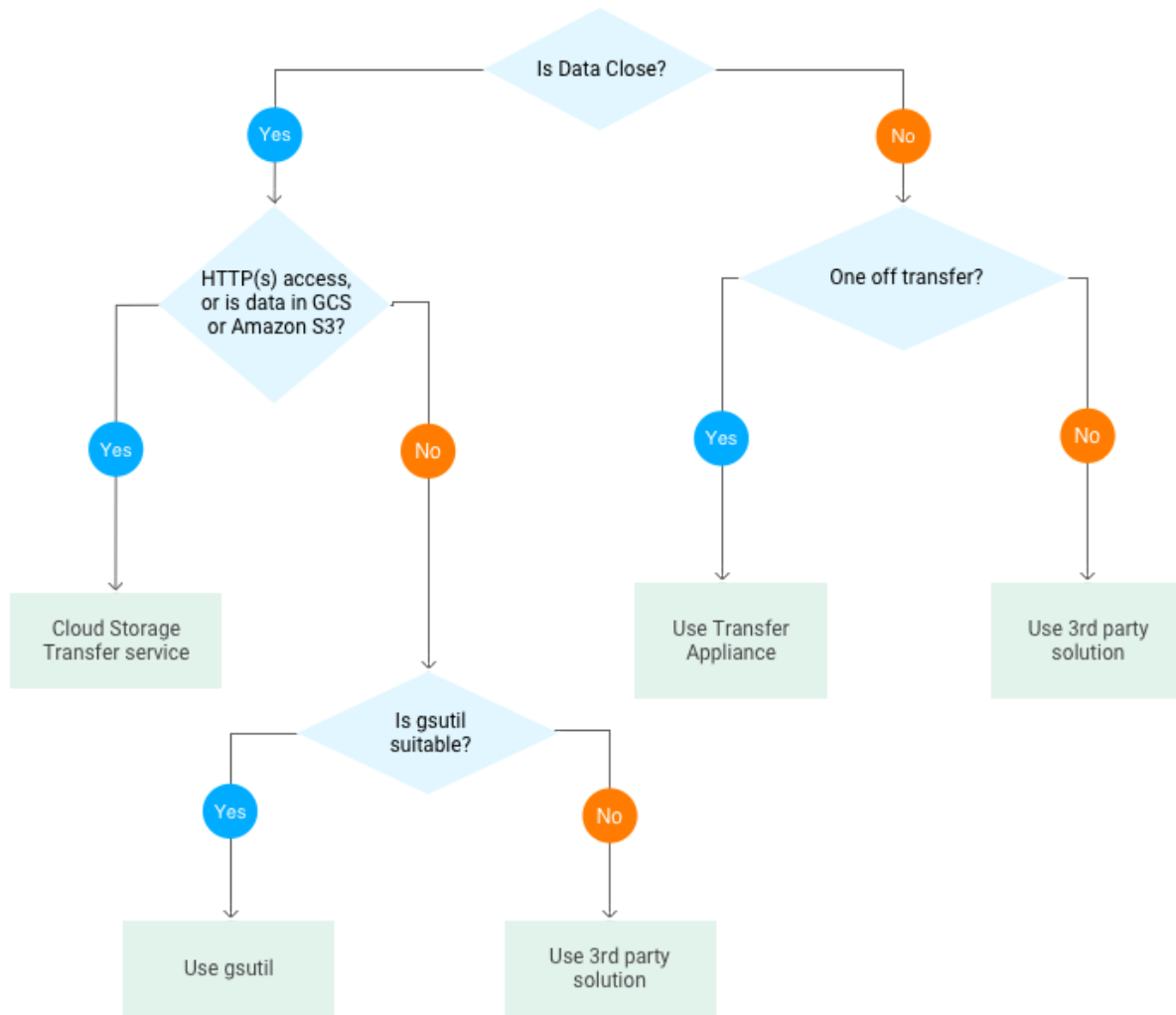
```

graph TD
    Q1{Is your data structured?} -- NO --> Q2{Do you need Mobile SDKs?}
    Q1 -- YES --> Q3{Is your workload analytics?}
    Q2 -- YES --> A1[Cloud Storage for Firebase]
    Q2 -- NO --> A2[Cloud Storage]
    Q3 -- YES --> Q4{Do you need updates or low-latency?}
    Q3 -- NO --> Q5{Is your data relational?}
    Q4 -- YES --> A3[Cloud Bigtable]
    Q4 -- NO --> A4[BigQuery]
    Q5 -- YES --> Q6{Do you need horizontal scalability?}
    Q5 -- NO --> Q7{Do you need Mobile SDKs?}
    Q6 -- YES --> A5[Cloud Spanner]
    Q6 -- NO --> A6[Cloud SQL]
    Q7 -- YES --> A7[Cloud Firestore for Firebase]
    Q7 -- NO --> A8[Cloud Datastore]
  
```

The flowchart 'How to choose Google Cloud storage services' provides a systematic approach to selecting the right storage solution based on specific requirements. It begins by checking if the data is structured. If not, it evaluates the need for mobile SDKs, leading to either Cloud Storage for Firebase or standard Cloud Storage. If the data is structured, it checks for analytics workload. If there are analytics, it assesses the need for updates or low latency, leading to Cloud Bigtable or BigQuery. If there are no analytics, it checks if the data is relational. For relational data, it evaluates the need for horizontal scalability (leading to Cloud Spanner or Cloud SQL) or mobile SDKs (leading to Cloud Firestore for Firebase or Cloud Datastore). This decision tree ensures that the chosen storage service aligns with the application's data characteristics and performance needs.

How to select the appropriate way to transfer data sets to GCP for your use case

Transferring large data sets to GCP (or indeed any cloud) means that you have to consider two initial questions How much data do you need to transfer? and how long have you got to get that data to GCP? In this case we are really focusing on getting large volumes of data to Cloud Storage. This then leads onto the other questions that you need to consider to allow you to determine what transfer method may meet your use case . How are you connected to GCP? How much bandwidth is actually available between your source and GCP? The article on Transferring big data sets to GCP (<https://cloud.google.com/solutions/transferring-big-data-sets-to-gcp>) discusses the information you need to determine the connectivity required and what methods to choose. It has a flowchart and the one below is a slightly modified version of the one found in the article.



Choosing a Cloud Storage class for your use case

Cloud Storage (GCS) is a fantastic service which is suitable for a variety of use cases. The thing is it has different classes and each class is optimised to address different use cases.

(https://cloud.google.com/storage/docs/storage-classes#comparison_of_storage_classes) All the storage classes offer low latency (time to first byte typically tens of milliseconds) and high durability. You can use the same APIs, lifecycle rules etc. Basically the classes differ by their availability, minimum storage durations, and charges for storage and access.

There are 4 classes that you need to care about.

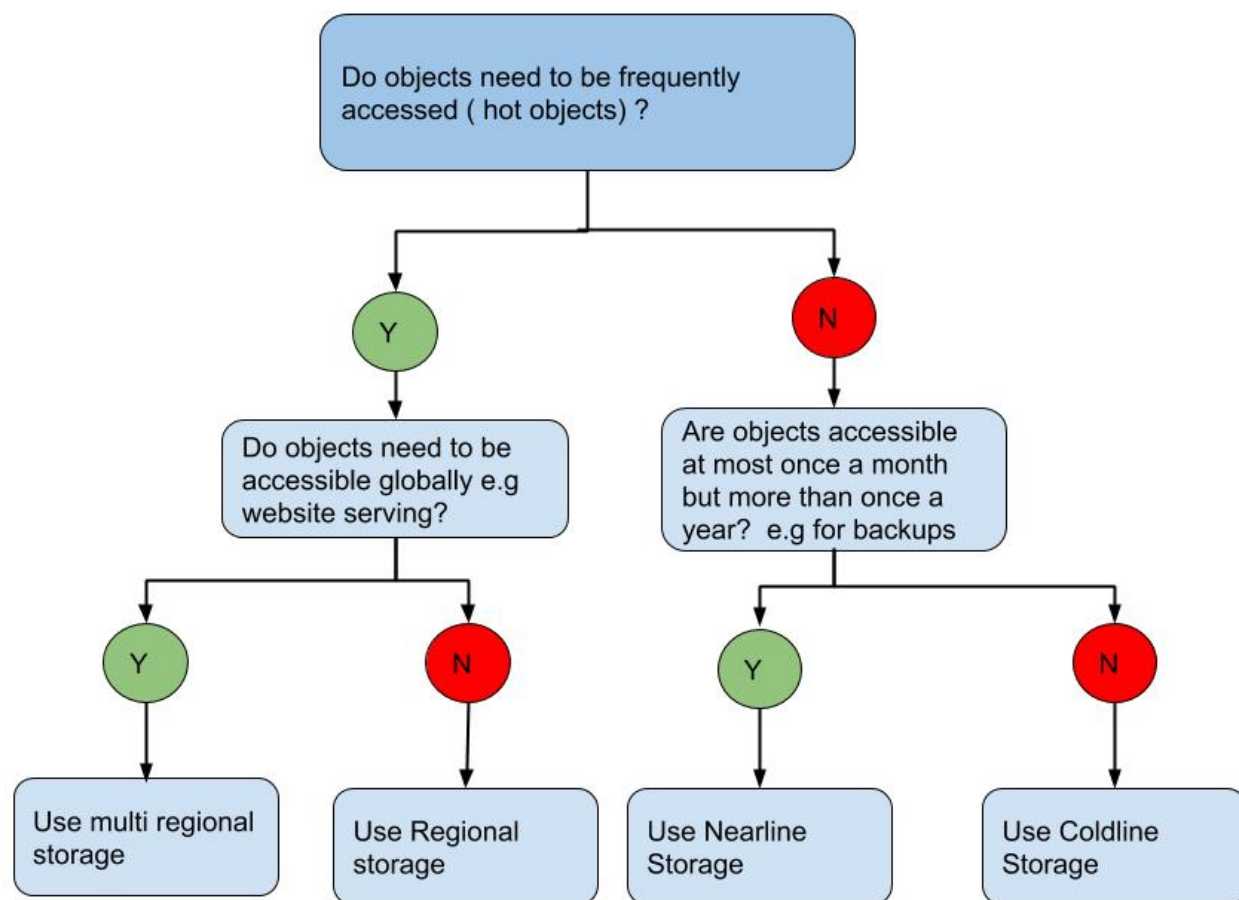
Multi regional—geo redundant (<https://cloud.google.com/storage/docs/key-terms#geo-redundant>) storage optimised for storing data that is frequently accessed (“hot” objects) for example web site serving and multi media streaming.

Regional—Data can be stored at lower cost, with the trade-off of data being stored in a specific regional location, instead of having redundancy distributed over a large geographic area. This is ideal for when you need the data to be close to the computing resources that process the data say for when using Dataproc.

Nearline—Nearline Storage is ideal for data you plan to read or modify on average once a month or less. Nearline Storage data stored in multi-regional locations is redundant across multiple regions, providing higher availability than Nearline Storage data stored in regional locations. This is great for backups . You should be carrying out regular DR fire drills at least once a month which includes recovering your data from your backups !

Coldline- a very low cost, highly durable storage service. It is the best choice for data that you plan to access at most once a year, due to its slightly lower availability, 90-day minimum storage duration, costs for data access, and higher per-operation costs. This is ideal for long term archiving use cases

Here’s a flow chart that helps you decide which storage class is appropriate for your use case when you don’t feel like reading too many words to figure out your choices (which after all is what flowcharts are for) .

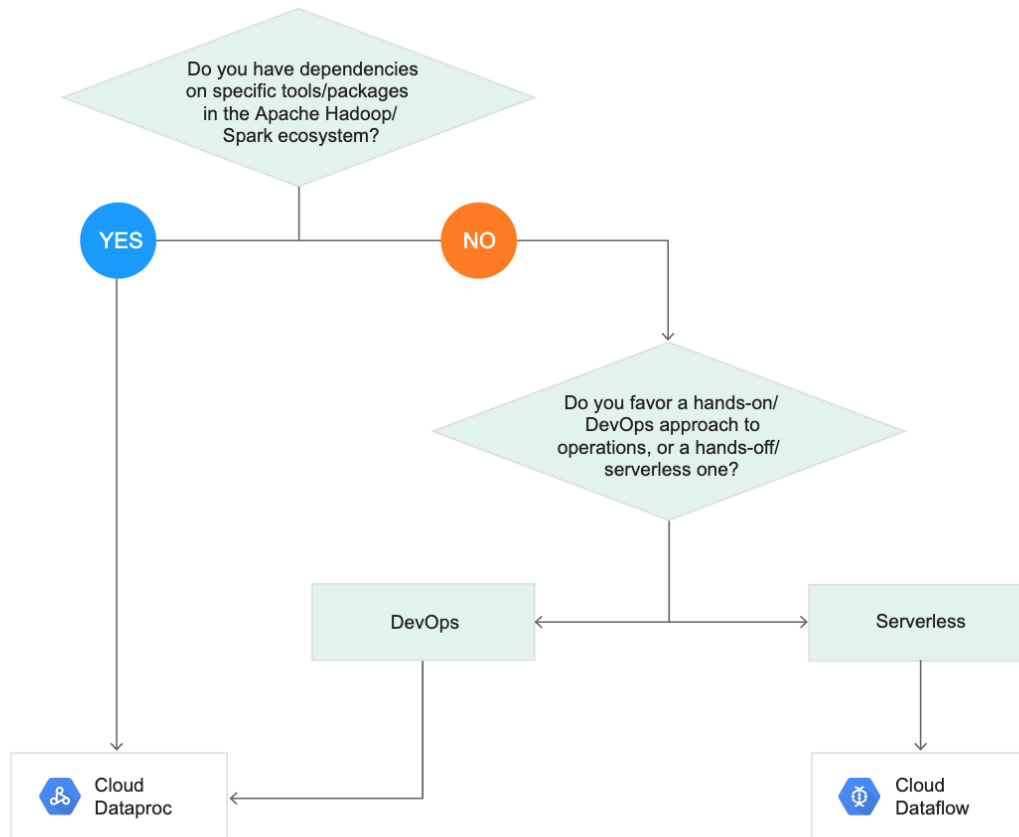


For an overview of the GCS storage classes see here (<https://cloud.google.com/storage/docs/storage-classes>)

Data processing - Cloud Dataflow versus Cloud Dataproc

If you have lots of files that need processing you may already be familiar with the Hadoop /Spark ecosystem and you would probably use GCP's Cloud Dataproc

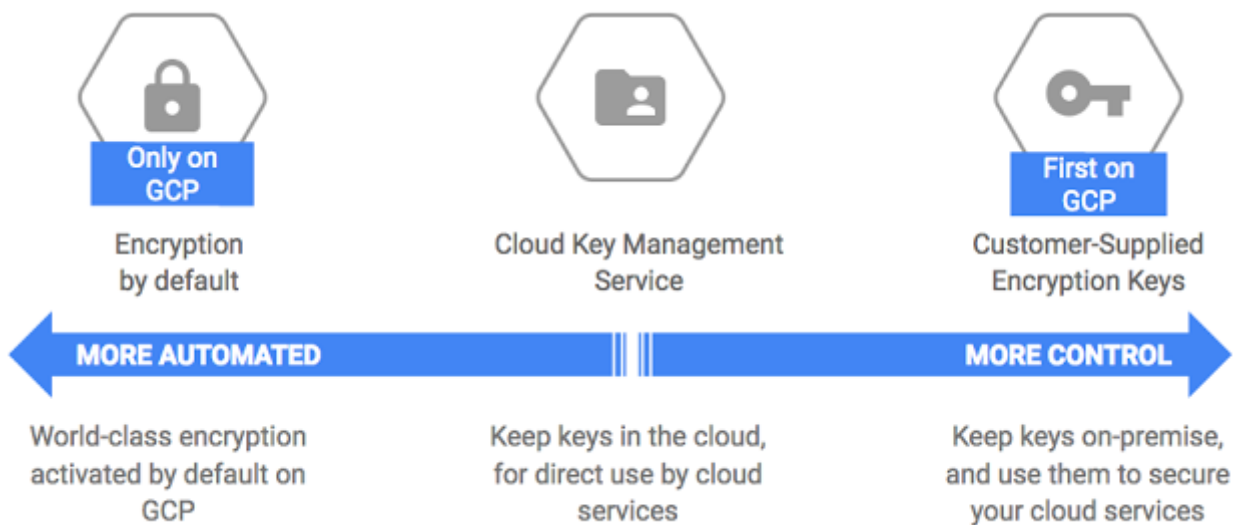
(<https://cloud.google.com/dataproc/>) as the path of least resistance. But GCP also has a unified batch & stream service Cloud Dataflow (<https://cloud.google.com/dataflow/>) which is their managed Apache beam (<https://beam.apache.org/>). Cloud Dataflow is a service unlike Dataproc where you don't need to worry about the compute so it's a "serverless" service because GCP takes care of provisioning and managing the compute on your behalf. GCP have created a handy flowchart for you which can be found on both the Cloud Dataflow & Cloud Dataproc landing pages with more words than I have here.



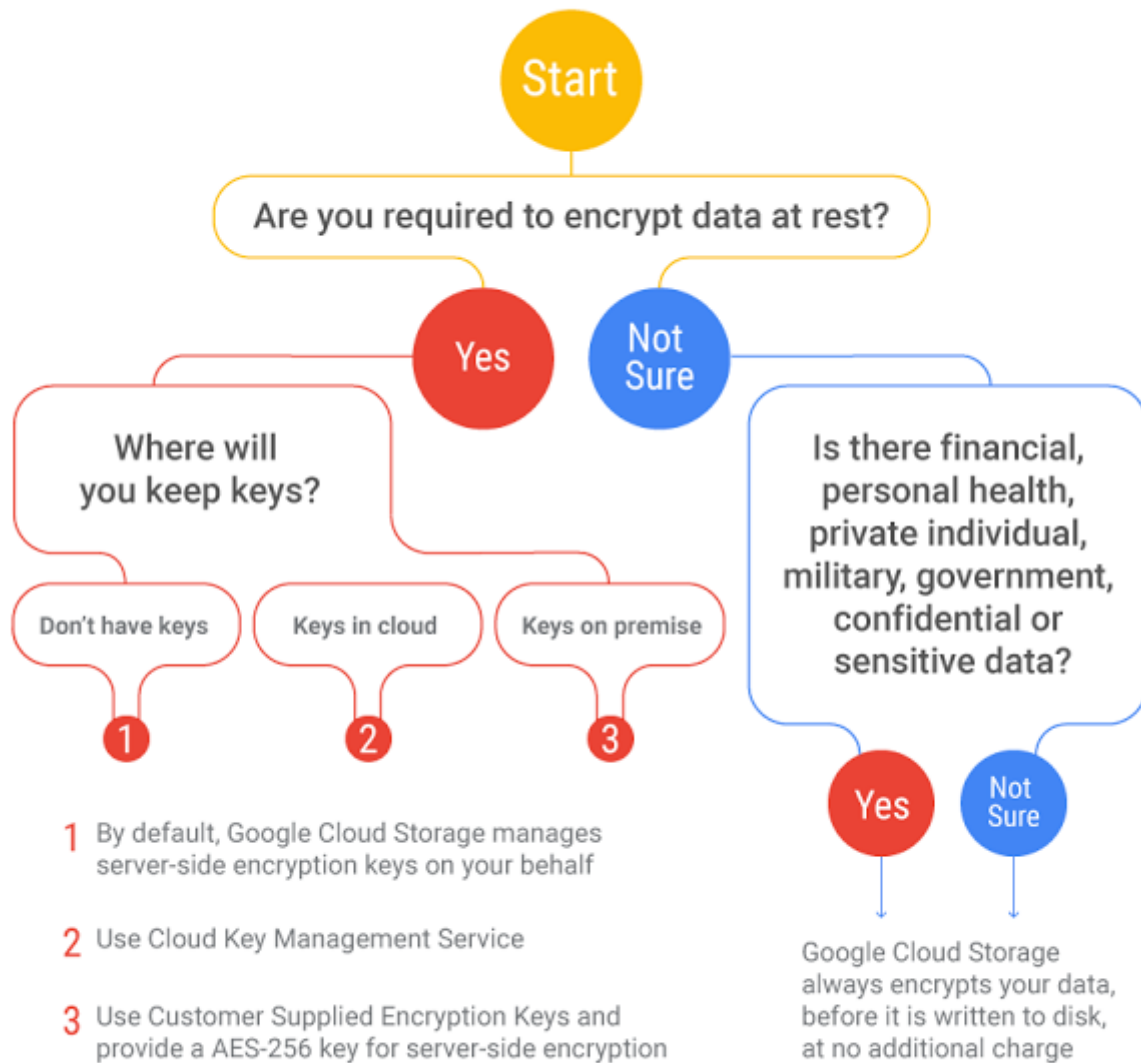
Security

How to manage encryption keys

GCP has a continuum of ways for you to manage your encryption keys graphically depicted as



Yes I know that the continuum graphic alone is probably all you need but when the announcement for the KMS service was made they produced a flow chart and I Just had to include it here



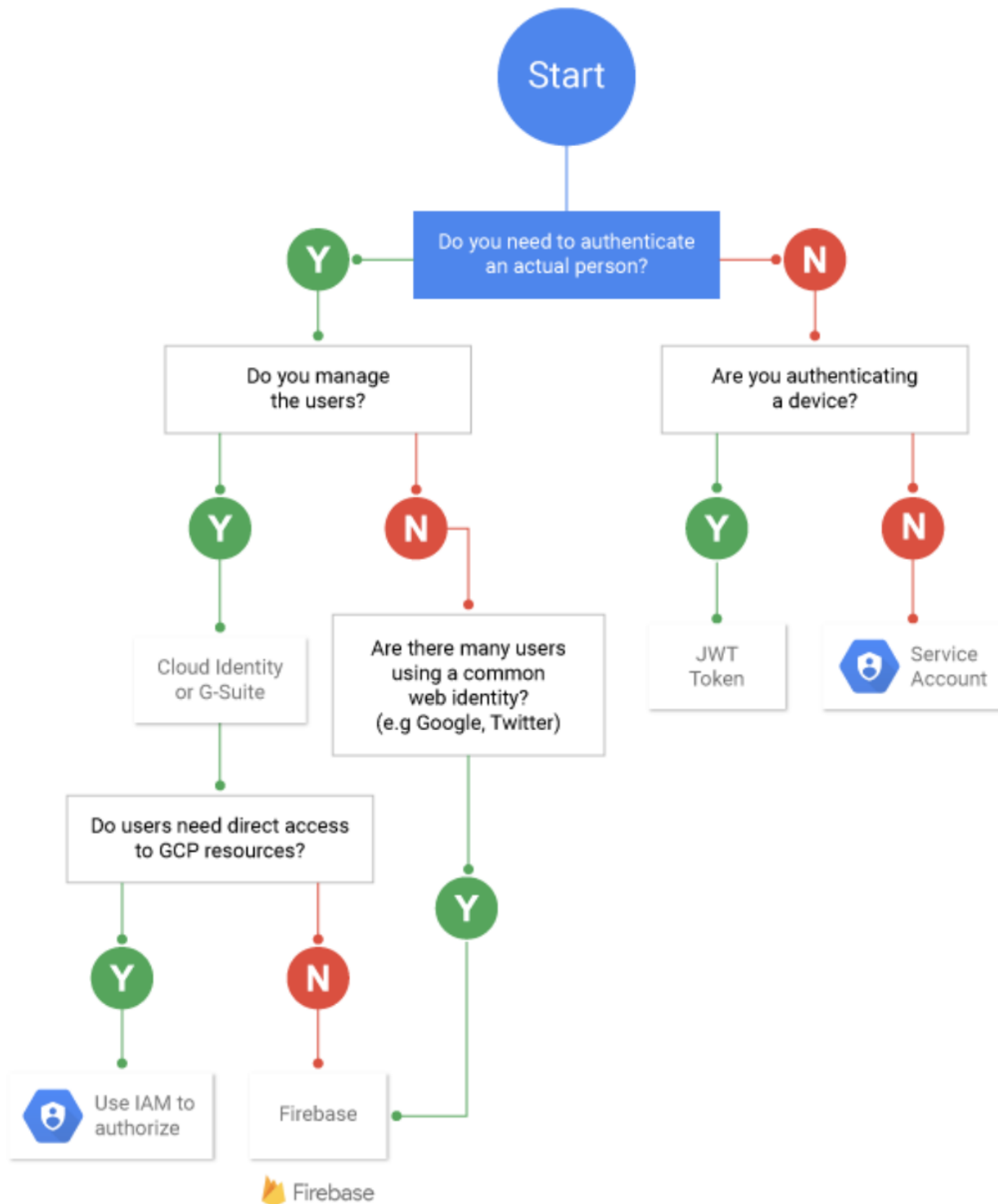
The words that go with the above can be found here

(<https://cloudplatform.googleblog.com/2017/01/managing-encryption-keys-in-the-cloud-introducing-Google-Cloud-Key-Management-Service.html?m=1>) and a nice table that compliments the flow chart can be found here (<https://cloud.google.com/security/encryption-at-rest/>) at the Encryption at rest landing page . (Everything you ever wanted to know about Encryption at rest on GCP and more !)

Which Authentication option ?

I was torn about keeping this one in this list but in the end I decided to keep it as it was still valid and the flowchart below it on using GCP's Identity platform complemented rather than replaced it. This is one of my own flowcharts as at the time I wrote the original medium post GCP didn't have one for this yet!! Then in Dec 2nd 2017: Neal Mueller (<https://medium.com/u/bb122fd3fe7a>) responded to my hint about wanting a GCP flowchart for Authentication and it's so much prettier than my version 😊 so I updated the flowchart below with the prettier version! Thanks Neal.

So just to make sure we are on the same page authentication identifies who you are ! This flowchart is focused on whether its identity—> application (deployed on GCP) or identity—> direct access to GCP



and as I haven't written the words to go with this flowchart I've left you a few links instead:

Firebase Authentication (<https://firebase.google.com/docs/auth/>)

Service Accounts (<https://cloud.google.com/iam/docs/service-accounts>)

GAE User authentication options

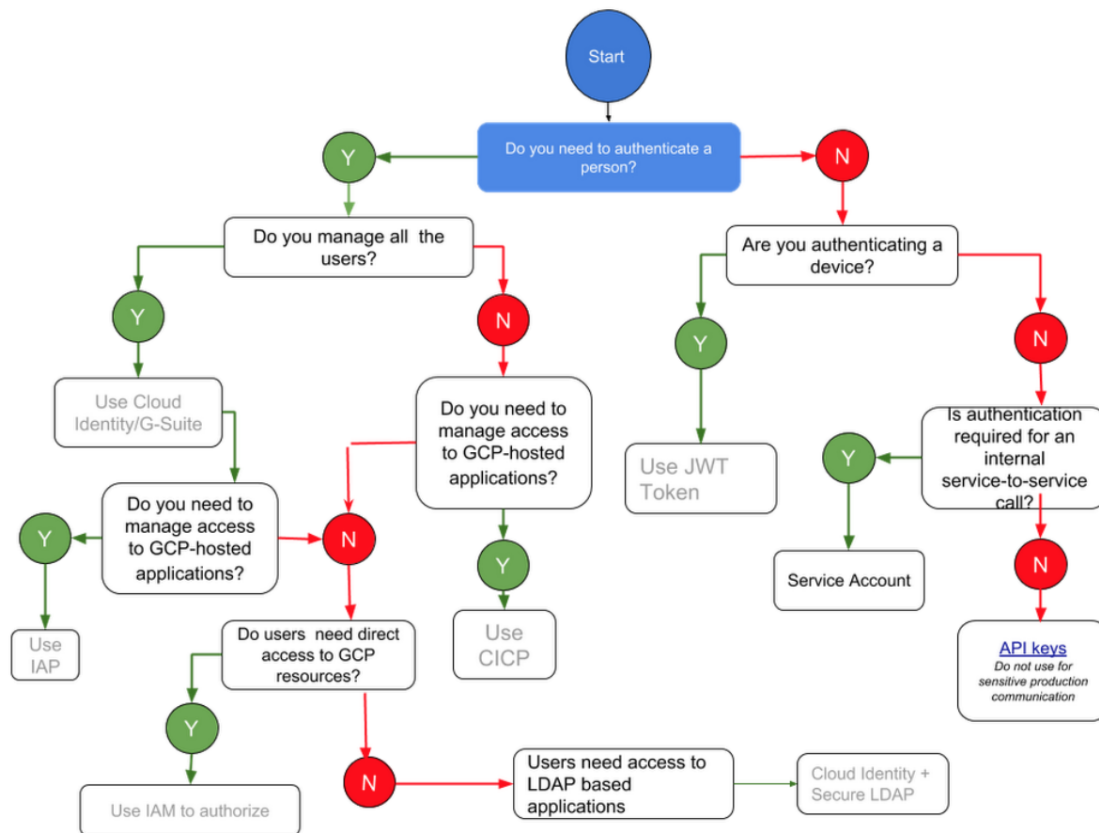
(<https://cloud.google.com/appengine/docs/standard/python/oauth/>)

Cloud IoT using JSON Web Tokens (<https://cloud.google.com/iot/docs/how-tos/credentials/jwts>)

Cloud Identity (<https://support.google.com/a/answer/7319251?hl=en>)

Need an identity mgt product?

How you manage your identities depends on the use case. Need to manage users who will have direct access to GCP resources versus users who need access to an application that you're hosting on GCP? Different requirements and thus different solutions required. Here's a flowchart to help you figure out the right solution for your use case.

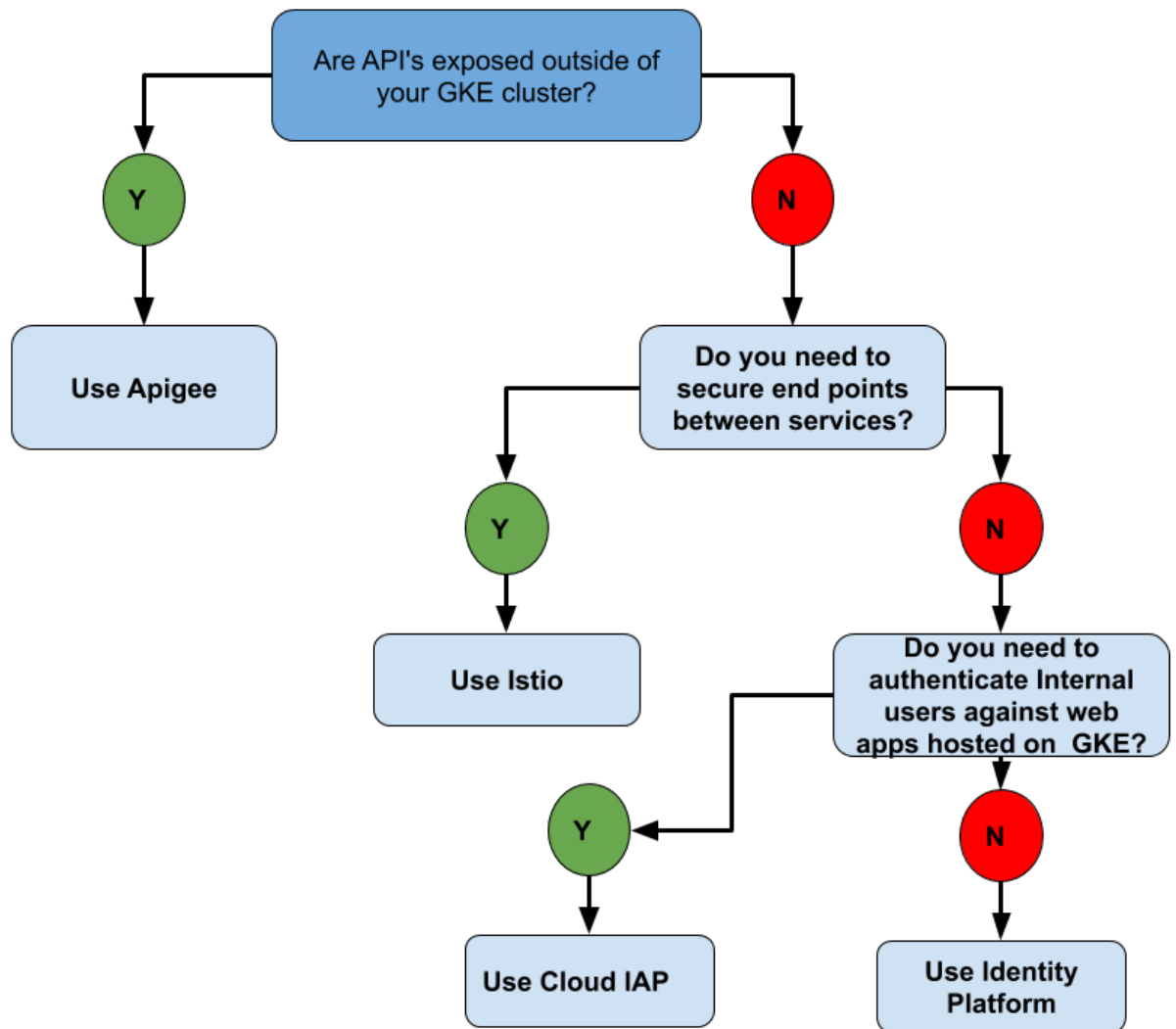


I will get round to updating this flowchart one day to reflect the name change from CACP to identity platform (<https://cloud.google.com/blog/products/identity-security/simplifying-identity-and-access-management-of-your-employees-partners-and-customers>). The words that go with the flowchart can be found here (<https://cloud.google.com/blog/products/identity-security/identity-and-authentication-the-google-cloud-way>).

Securing your GKE end points

Arguably this flowchart could be catalogued under Compute but as it's about **securing** end points under security it goes. The idea for this flowchart arose after my team had the discussion re what option would be appropriate for what use case when you want to secure your end points using GKE. So thanks team for the inspiration for this one

When a GKE operator wants to serve content from GKE and secure it they have a number of ways of addressing this depending on the use case as shown in this flow chart:



API's exposed outside of your GKE cluster then use Apigee edge (<https://docs.apigee.com/api-platform/get-started/what-apigee-edge>) which provides a way to manage your API's acting as a proxy to them. It can provide services such as security e.g is that call to your API authorized.

If you are looking at service to service security within the cluster then Istio is the mesh (<https://istio.io/docs/concepts/security/>) for you

if you are wanting to authenticate access to your web apps it depends on whether they are internal users or external. For internal users then Cloud IAP (<https://cloud.google.com/iap/docs/enabling-kubernetes-howto>) is where you need to stop and have a look while for end users Identity platform (<https://cloud.google.com/blog/products/identity-security/getting-started-with-identity-platform>) is the stop you need.

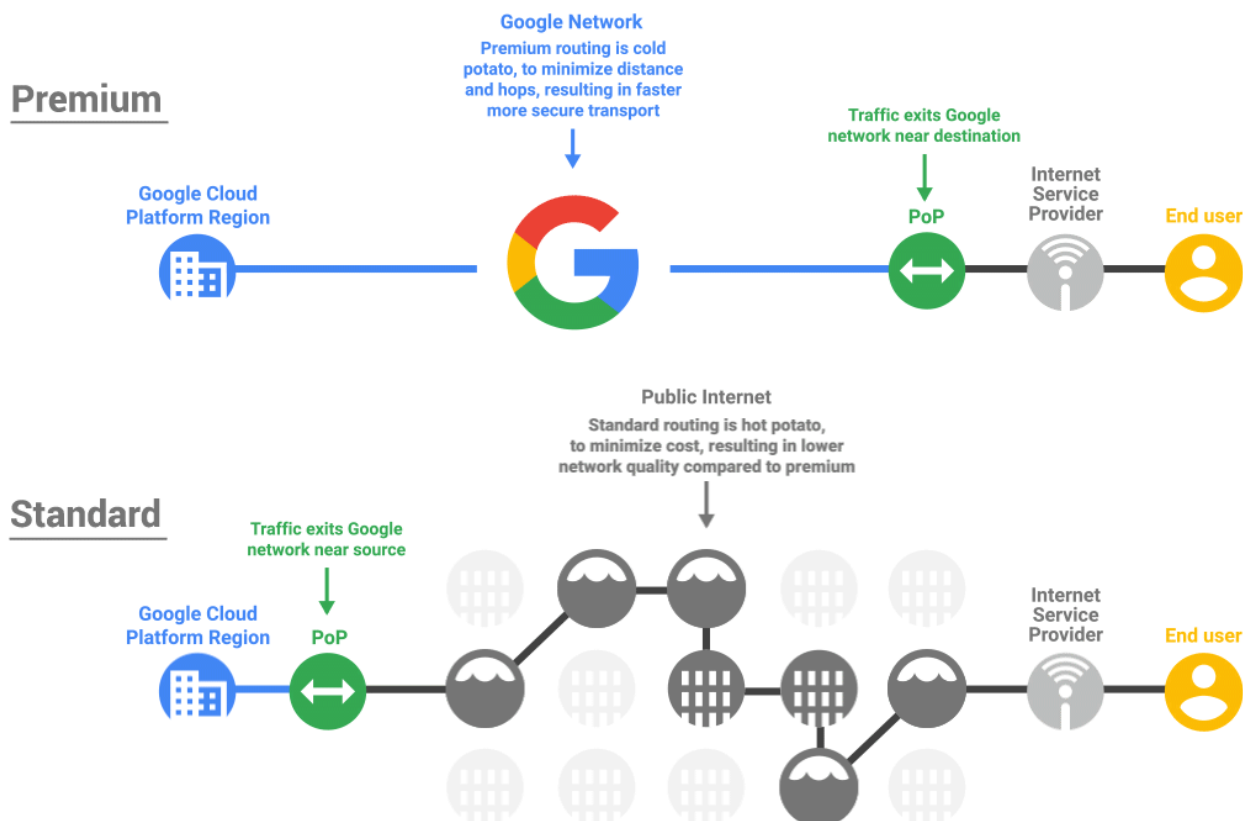
You can also use Istio and Apigee together. Istio can secure the communication between services, provide observability, etc while Apigee can provide external authentication, quotas and overall API policy management.

There are nuances particularly with istio which and I quote my team mate James “the lines blur a bit when looking at Istio” but starting from here isn’t a bad place to start from

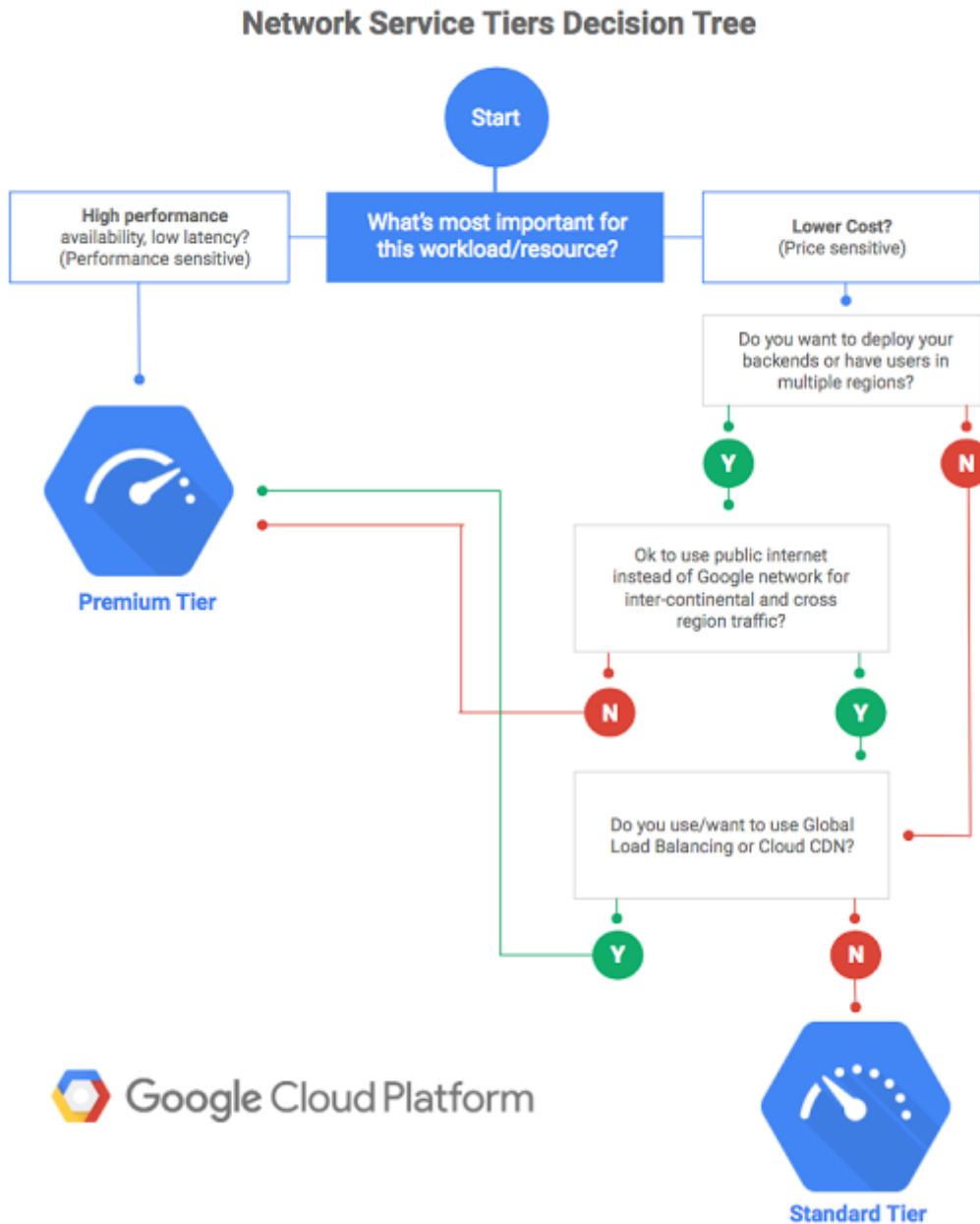
Networking

Which Network Tier?

GCP’s network even if I say so myself is fantastic but it’s recognised that not every use case needs to optimize for performance and cost may be the driver. So welcome to Network tiers.



You can see the funky animated gif for the above image here (https://2.bp.blogspot.com/-Za3HWtGbQK8/WZ3TuWoVxzI/AAAAAAAAETc/bkqmGj9TBXYGTMO6naL3t_pRh_LIz7XtACK4BGAYYCw/s1600/image2.gif)

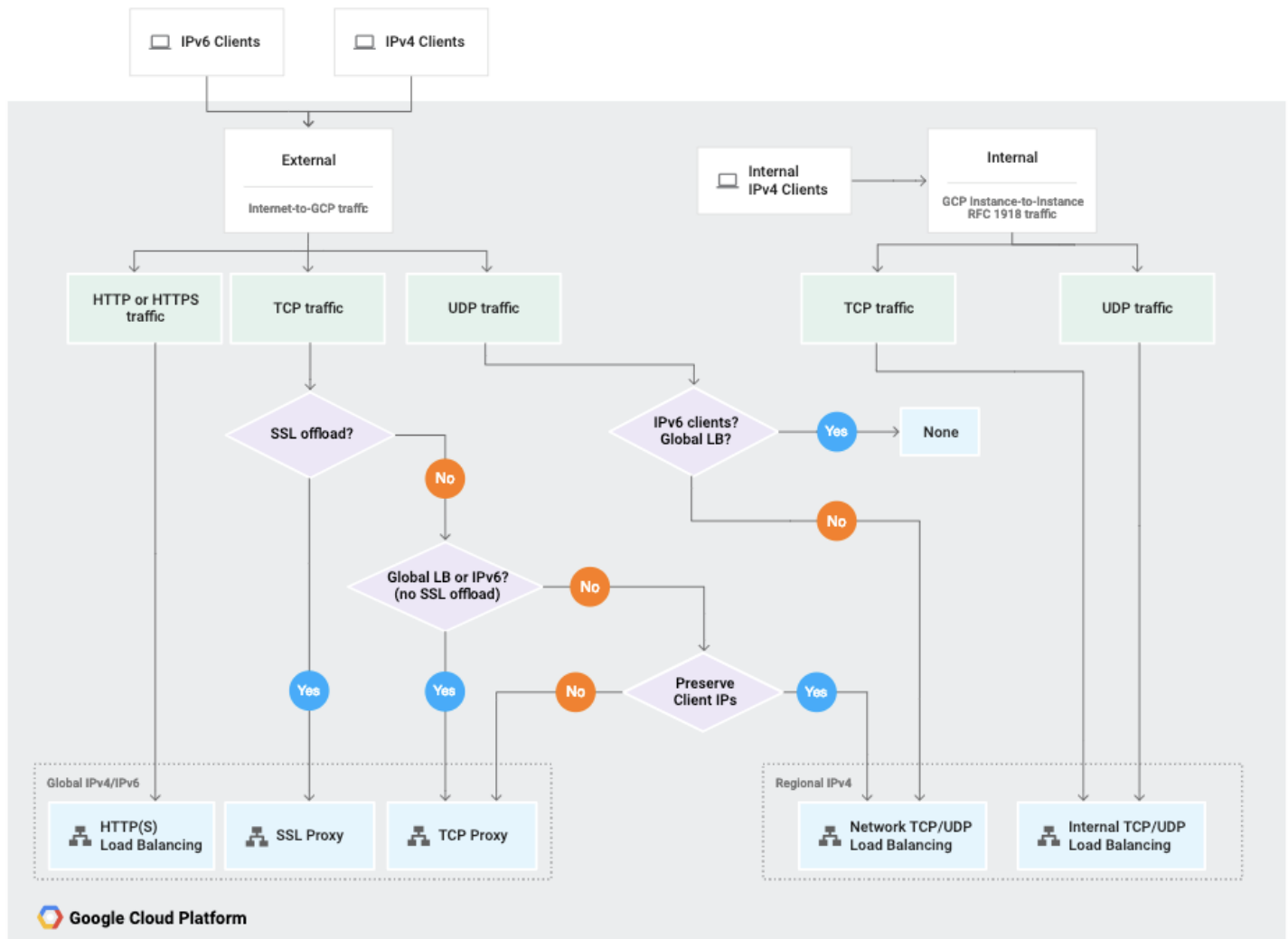


The words that go with the above can be found here (<https://cloud.google.com/network-tiers/>) . There are some useful tables there too.

Choosing a Load balancer

Load balancing is great it allows you to treat a group of compute resources as a single entity providing an entry point that has in the case of GCP load balancing services a single anycast IP address. Combining GCP Load balancers with autoscaling you can scale the resources up and down

according to metrics you configure. There are loads more cool features but you get the idea. So what type of load balancing service do you need? Layer 7, layer 4, global , regional? Maybe you need an internal load balancer well there's a flowchart for helping you decide (Okay you knew that was coming didn't you? 😊)

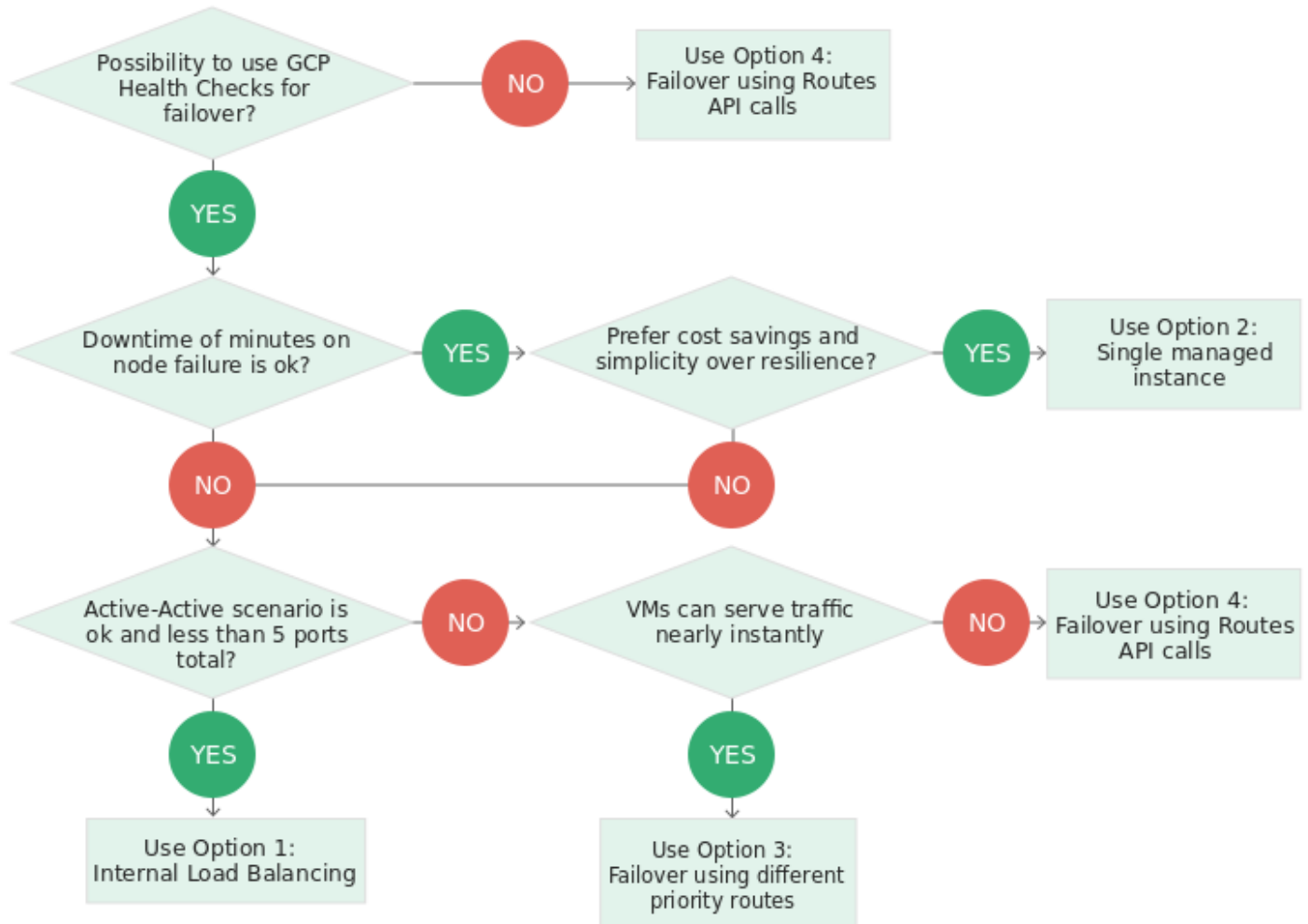


Here (<https://cloud.google.com/load-balancing/docs/choosing-load-balancer>) are the words to go with the flowchart. Once you have figured out what load balancing option is likely to address your needs have a look at the load balancing overview (<https://cloud.google.com/load-balancing/docs/load-balancing-overview>) page as a first stop before diving in.

Choosing the floating IP address pattern that maps to your use case

Floating IP's are a way to allow you to move an IP address from one server to another . Typically this pattern is usually required for HA deployments or for disaster recovery scenarios. For example where you have one active server or appliance such as databases with a non serving replica /hot standby .

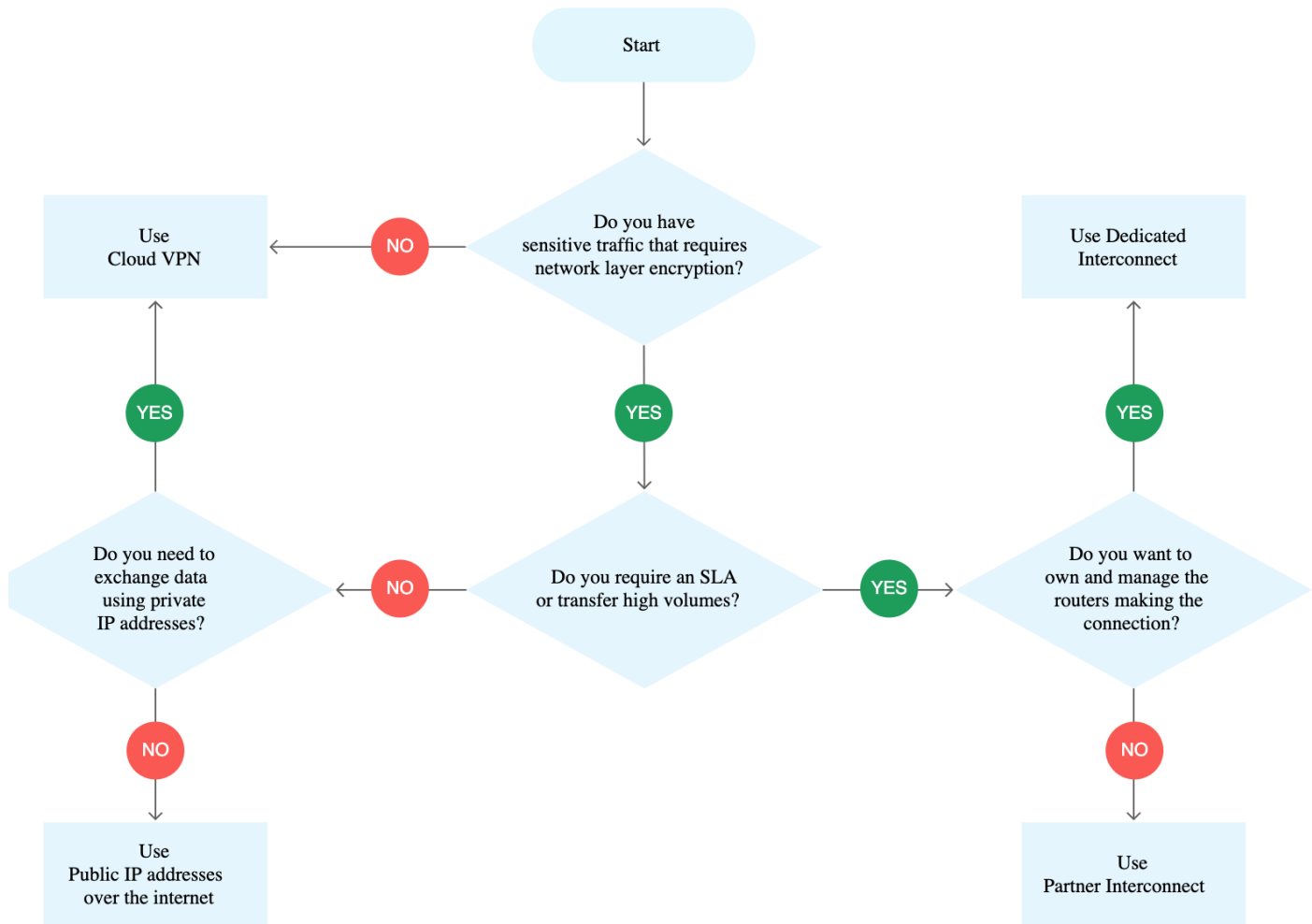
When you have to swap to the secondary server you point the floating IP to it. This negates the need to update clients to use an alternative IP to point to the alternative server. The article [On best practices for floating IP addresses](https://cloud.google.com/solutions/best-practices-floating-ip-addresses) (<https://cloud.google.com/solutions/best-practices-floating-ip-addresses>) has a list of uses cases for on premises (https://cloud.google.com/solutions/best-practices-floating-ip-addresses#floating_ips_in_on-premises_environments) and provides a number of options for implementing the pattern for Compute engine instances and yes has a flowchart to help you choose the solution for your use case . Here's the flow chart



Options for connecting to other clouds from GCP

Whatever the reasons (They range from having processing in one place and data somewhere else, to distributing processing across clouds, through to DR etc) people want to be able to connect to other clouds from GCP.

GCP have written a great article describing the various patterns that can be employed and yes they have a flowchart to help you decide which pattern is the right one for your use case which I share here for your delectation:



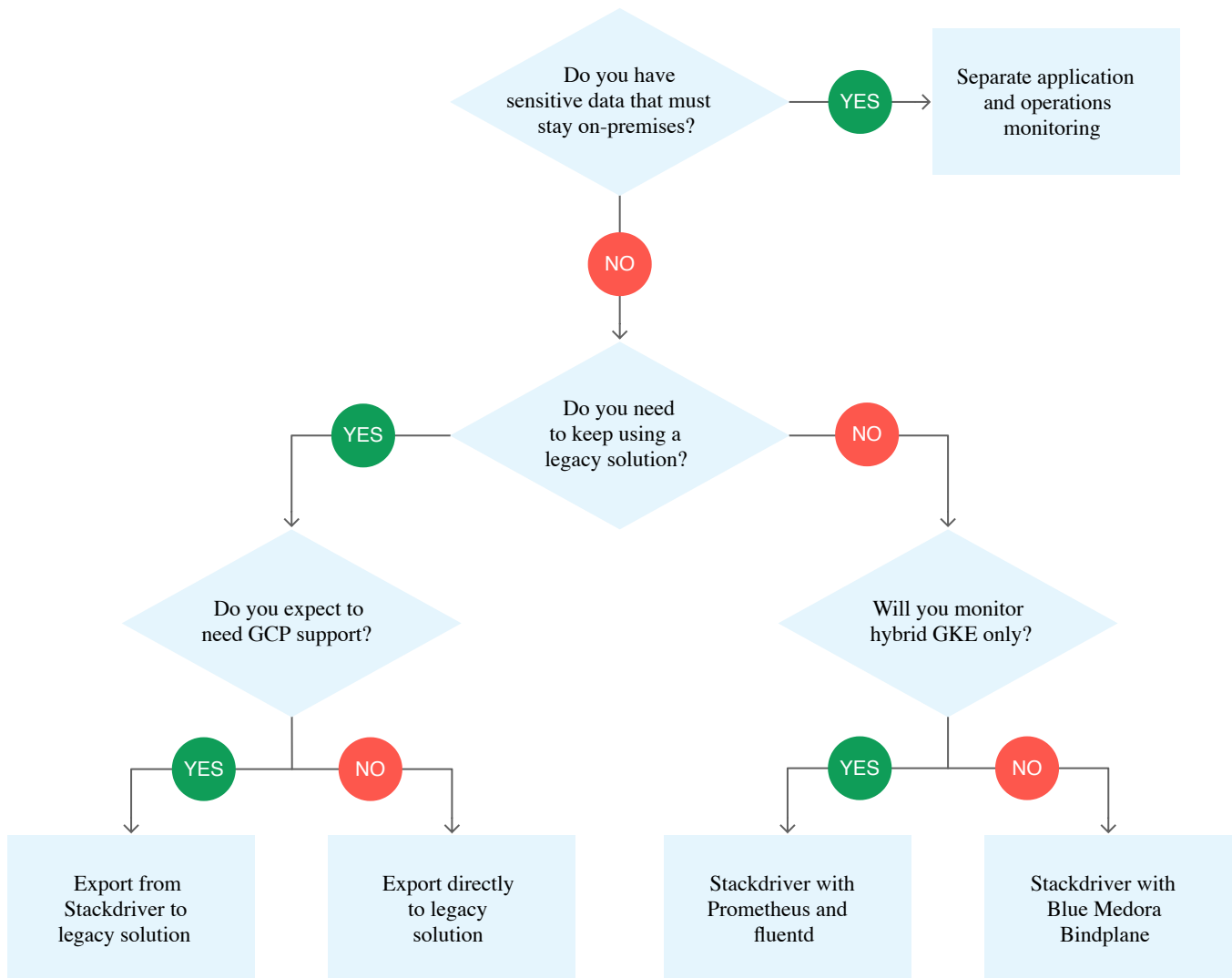
The article with this flowchart and a walk through of the different patterns can be found here (<https://cloud.google.com/solutions/patterns-for-connecting-other-csps-with-gcp>)

Misc

Hybrid & multi-cloud logging & monitoring patterns

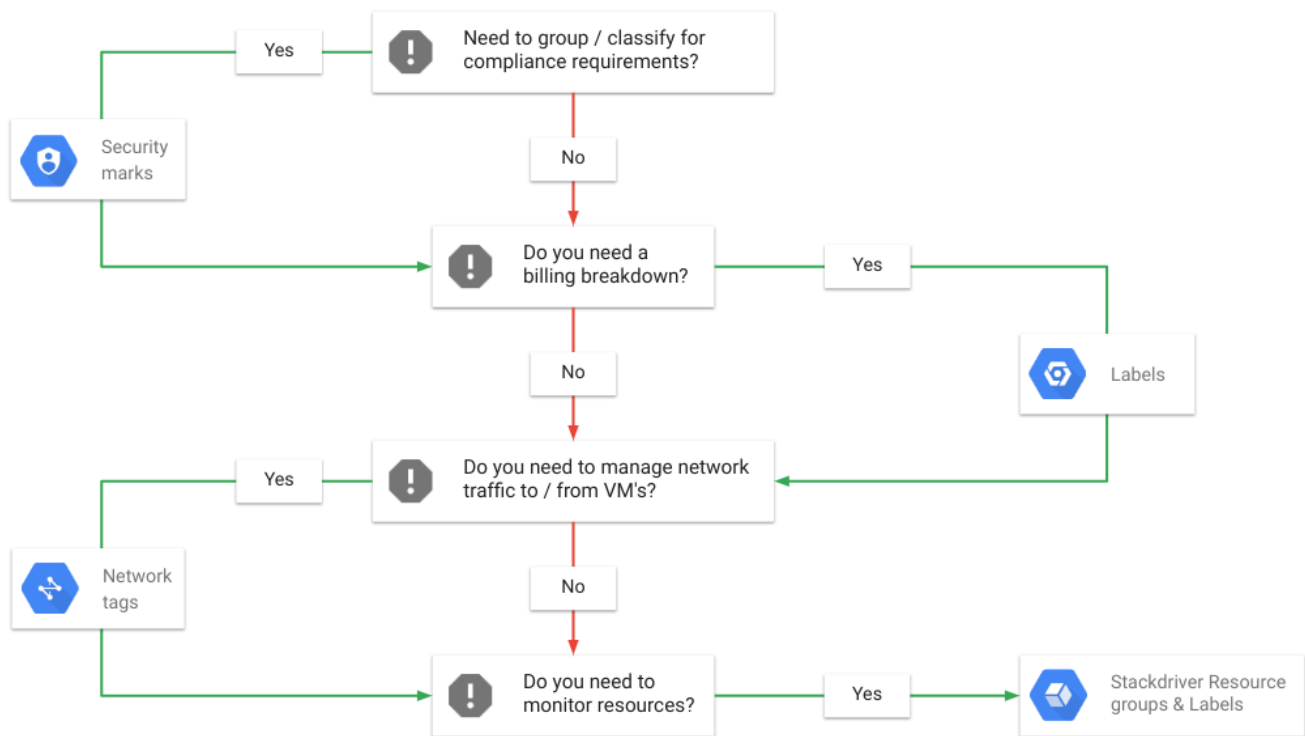
Hybrid and multi-cloud architectures are here to stay and looking at ways to manage those is key to not having to wipe the tears of ops/ sysadmin staff dealing with the operational overhead. It's important to have a consistent logging and monitoring approach not only to give a single pane of glass but to simplify the admin of managing applications in two environments. This guide (<https://cloud.google.com/solutions/hybrid-and-multi-cloud-monitoring-and-logging-patterns>)

discusses architectural patterns for logging & monitoring in hybrid or multi cloud environments and it's flow chart helps navigate your choices between a centralised logging approach no matter where your apps are deployed versus a segregated approach.



What annotations(labels) should you use for which use case

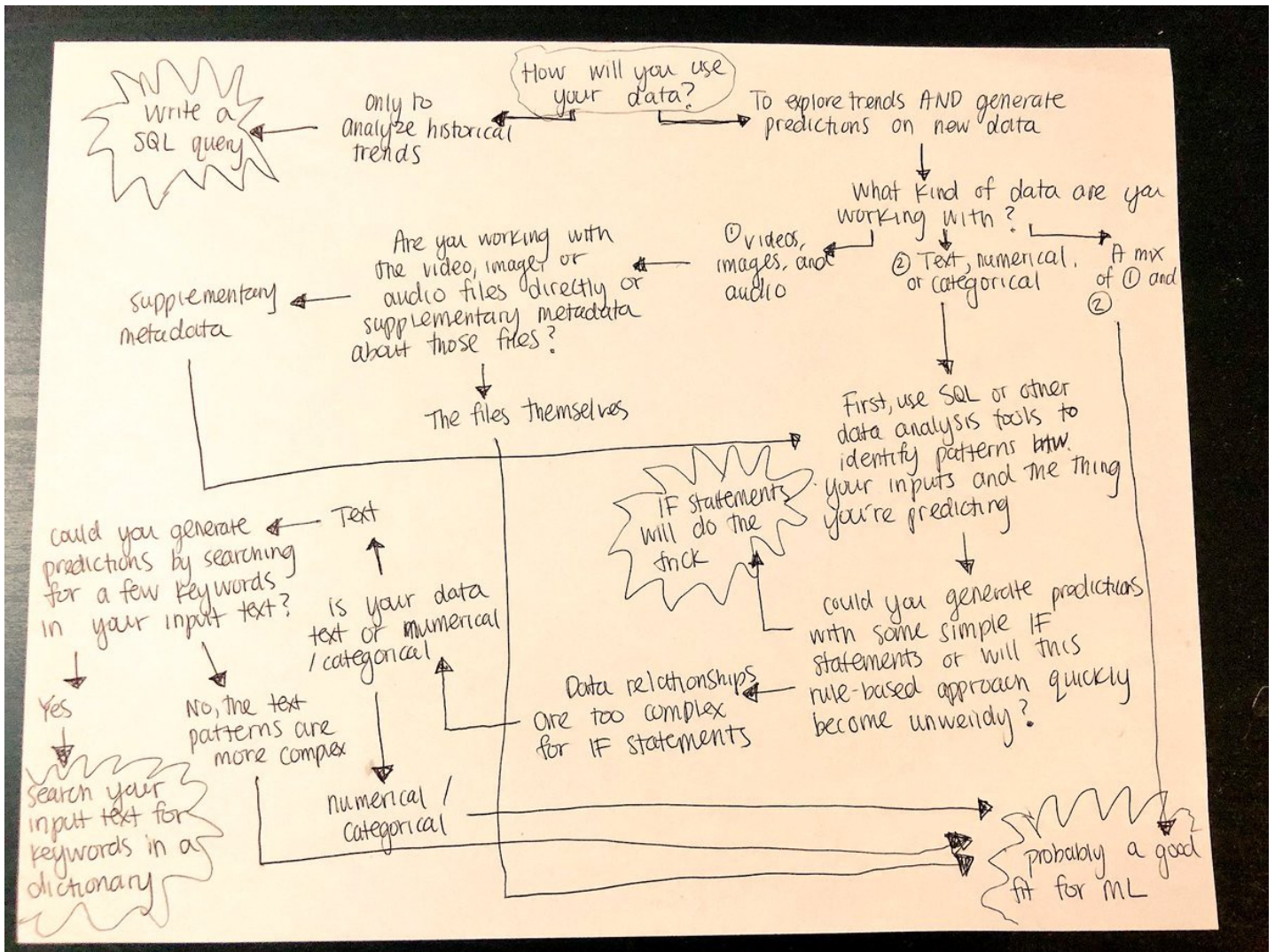
GCP has a number of ways of annotating or labelling(this can get slightly over loaded hence the use of the word annotation) resources. Each annotation has different functionality and scope, they are not mutually exclusive and you will often use a combination of them to meet your requirements so I wrote a post (<https://cloudplatform.googleblog.com/2018/06/Labelling-and-grouping-your-Google-Cloud-Platform-resources.html>) with added flow chart to help you navigate which annotation(s) to use for what use case. Here's the flow chart :



Read the post (<https://cloudplatform.googleblog.com/2018/06/Labelling-and-grouping-your-Google-Cloud-Platform-resources.html>) for the words .

ML or SQL ?

Always wanted to know whether you really need to use ML or whether a SQL query will suffice well
Sara Robinson (<https://medium.com/@srobtweets>) tweeted this flow chart



From <https://twitter.com/SRobTweets/status/1053273512079699968>
 (https://twitter.com/SRobTweets/status/1053273512079699968)

She then wrote some words to augment the flowchart here

(<https://medium.com/@srobtweets/when-should-you-use-machine-learning-part-1-b0028603ac3a>)
 and then wrote some more words (<https://medium.com/@srobtweets/sql-or-ml-the-same-dataset-two-ways-233e611619bd>) walking you through figuring out if ML is a good fit for your prediction task. A SQL query may be all you need. Use the right tool for the job. I love these two posts well I do get to look at the flowchart twice!

Posted on Jun 16, 2019 at 13:36

← Newer Posts

(<https://grumpygrace.dev/posts/top5-gcp-shoutout/>)

Older Posts →

(<https://grumpygrace.dev/posts/back-logging/>)

