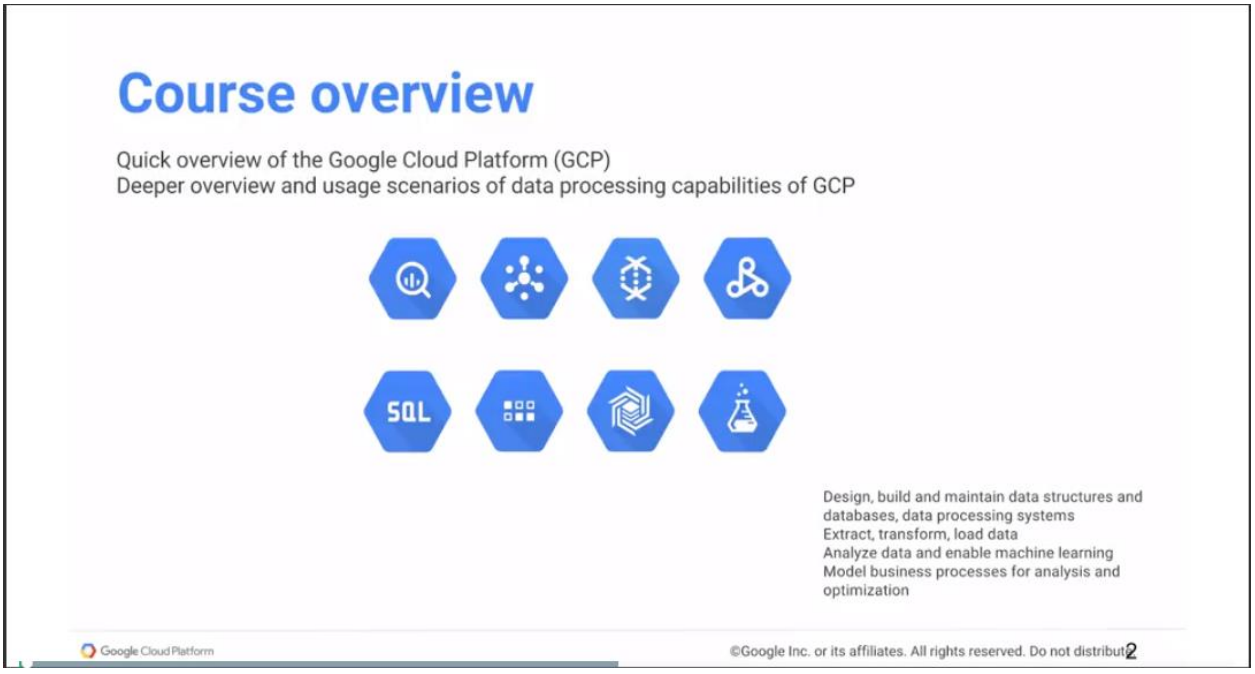
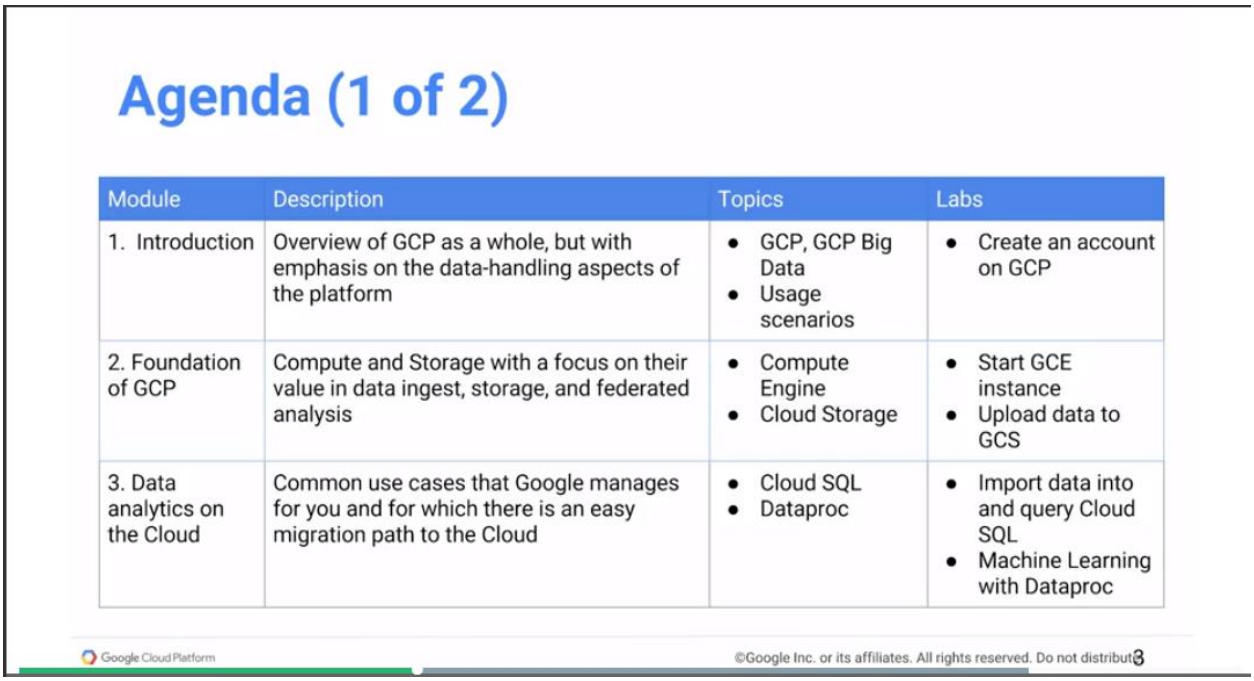
INTRODUCTION TO THE DATA AND MACHINE LEARNING SPECIALIZATION

In this first course of the data engineer track, you will gain an overview of the data and machine learning parts of Google Cloud platform. In each module, everything is going to be two pronged: at one level, we will look at some products that help you accomplish certain things on Google cloud, but at the same time we'll look at very specific use cases, very common use cases that involve machine learning, that involve data processing, that involve data analysis. The goal is to accomplish those use cases using these particular products, and we'll do this over and over again. And by the time we're done with this course, you have to basically gotten a pretty good overview of all of the moving parts of the data and ML parts of the platform.

In this course I'll be providing a very quick overview of GCP, in particular its foundational parts to compute and storage. Then we dive straight in to provide a deeper overview of the different ways you can process data with GCP: BitQuery, Dataflow, Dataproc, Cloud SQL, Datalab, etc. This class is mean for data engineers - people who design, build, maintain data structures, databases, data processing systems, data pipelines, who do extraction, transformation, loading of data, move data from one place to another. Engineers who may be data scientists who are analyzing data, enabling machine learning to happen, maybe even doing machine learning.

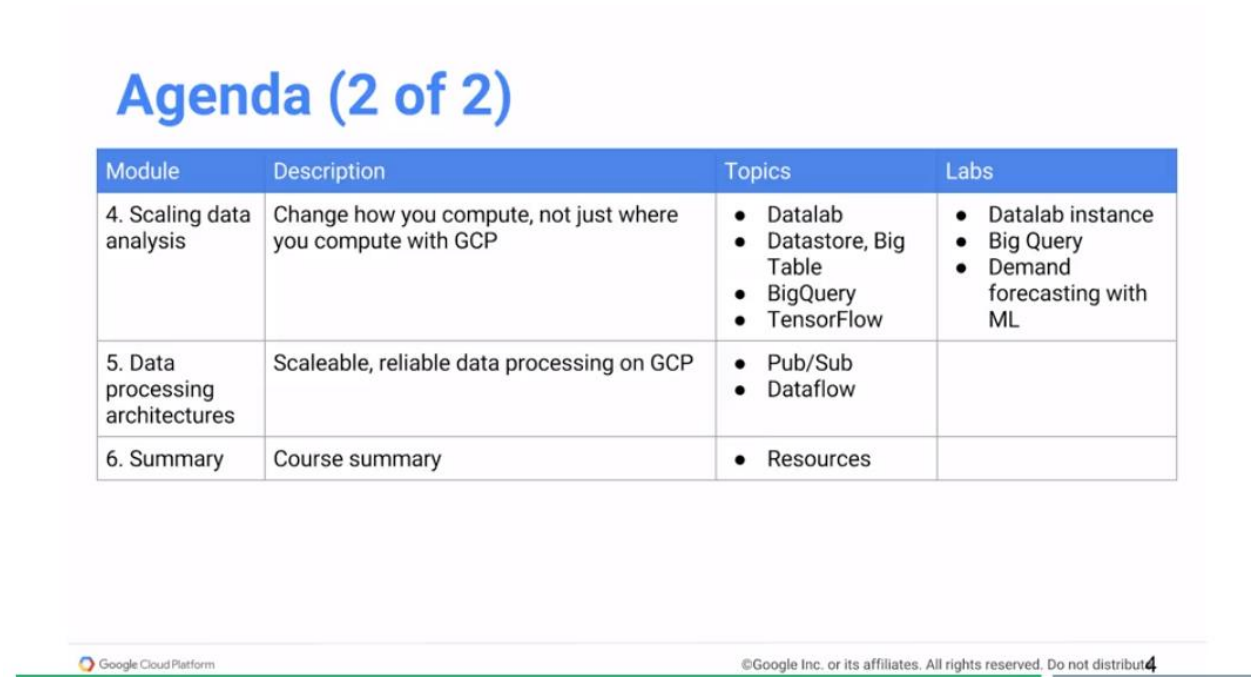


People who model business processes, who enable data driven decision making within your company. This class is meant for anyone who'll be working with data on Google Cloud Platform. It's also meant for any decision makers who are trying to decide whether your company should move to GCP and do your data processing on GCP. This gives you a good overview of all the capabilities that the Google Cloud provides so that you can make an informed decision.



In the course we will now start with an overview of Google Cloud Platform as a whole, but with particular emphasis on the ways you can handle data in the platform. The way you can ingest data, different ways of doing processing of data, whether with map produce, such as with Spark or whether with a streaming mechanism like with Dataflow. We look at BigQuery, which is our auto scaling date warehouse, etc.

We start with an overview of GCP, but with an emphasis on the data handling parts of it. Then we move on to talking about the foundation of GCP and this is computing and storage. Like any computer, and the Cloud is a computer, the two key parts of a computer are the computing units and the storage units of persistent data. And the way that happens on the Cloud is with Compute Engine and Cloud Storage, so we'll talk about both of those.



Then we move on to things that you're probably doing today, that you could easily move to the Cloud. We'll talk about the use cases that are quite common that Google provides a good managed environment for such that you can take things you're doing on premise, on your own hardware, and move it to the Cloud quite easily. The same software that you may be using Is also present on the Cloud. As examples of those use cases, we will talk about relational databases in the form of Cloud SQL, which is a MySQL database hosted on GCP. We will discuss Cloud Dataproc, which is a hosted version of big Spark hype of the Hadoop ecosystem software processes. In order to do that we'll look at how to import data and how to query MySQL running in Google Cloud. We'll also look at how to take a Spark program, submit it, and have it run on Cloud Dataproc. And the Spark program that we will look at will be a machine-learning program to carry out recommendations.

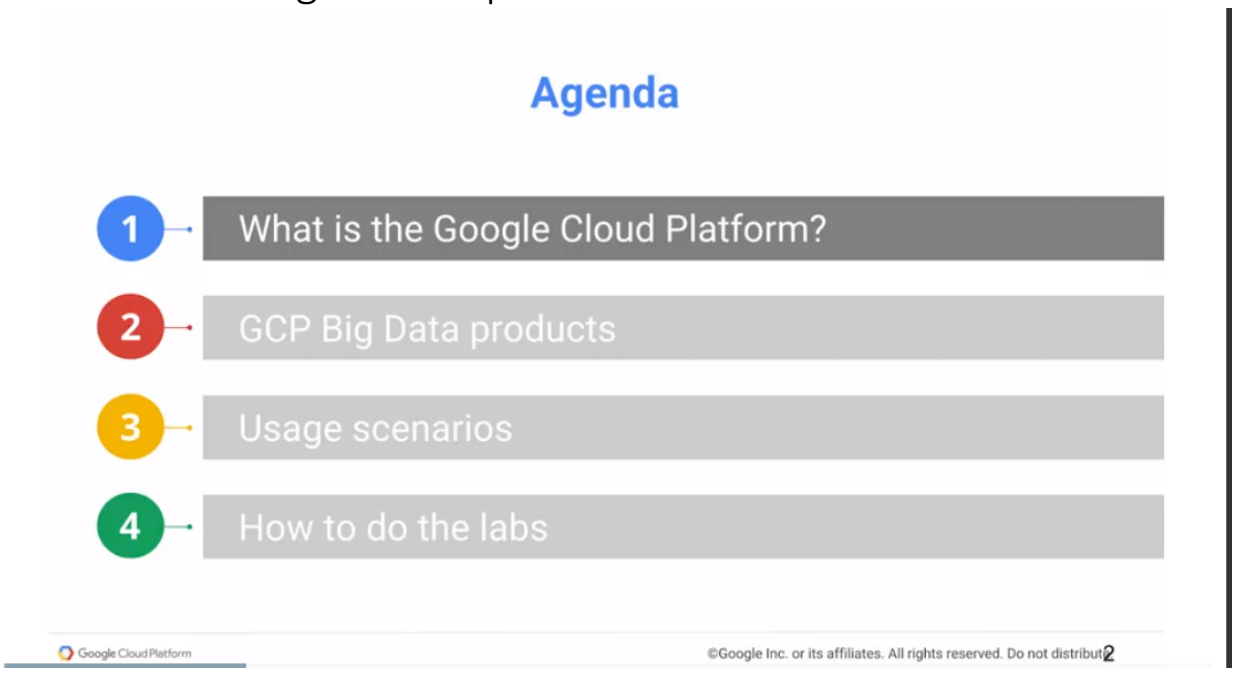
Once we have talked about the use cases that we're talking about, migrating use cases, we will then move on talking about more transformational use cases. These may be things that you may not be doing today, mainly because it may not be possible for you to do them today. So for example, we look at how you could query petabytes of data in a matter of seconds with Google BigQuery. We will talk about how to do fast random access, trading off global consistency versus low global availability. We'll also talk about machine learning. We'll talk about how to do TensorFlow. Maybe you run TensorFlow, but very commonly maybe running it on a single machine. We'll talk about how you'd run TensorFlow in a distributed fashion on the Cloud over extremely large data sets. And then we will move onto just providing a very quick overview of how you would do scalable reliable data processing on Google Cloud with Cloud Pub/Sub, which is a messaging architecture and with Cloud Dataflow, which is a way to execute a code that processes both streaming data and batch data in essentially the same way. And finally, we'll come to the conclusion of the course, and I'll leave you with some resources for further reading.

MODULE 1

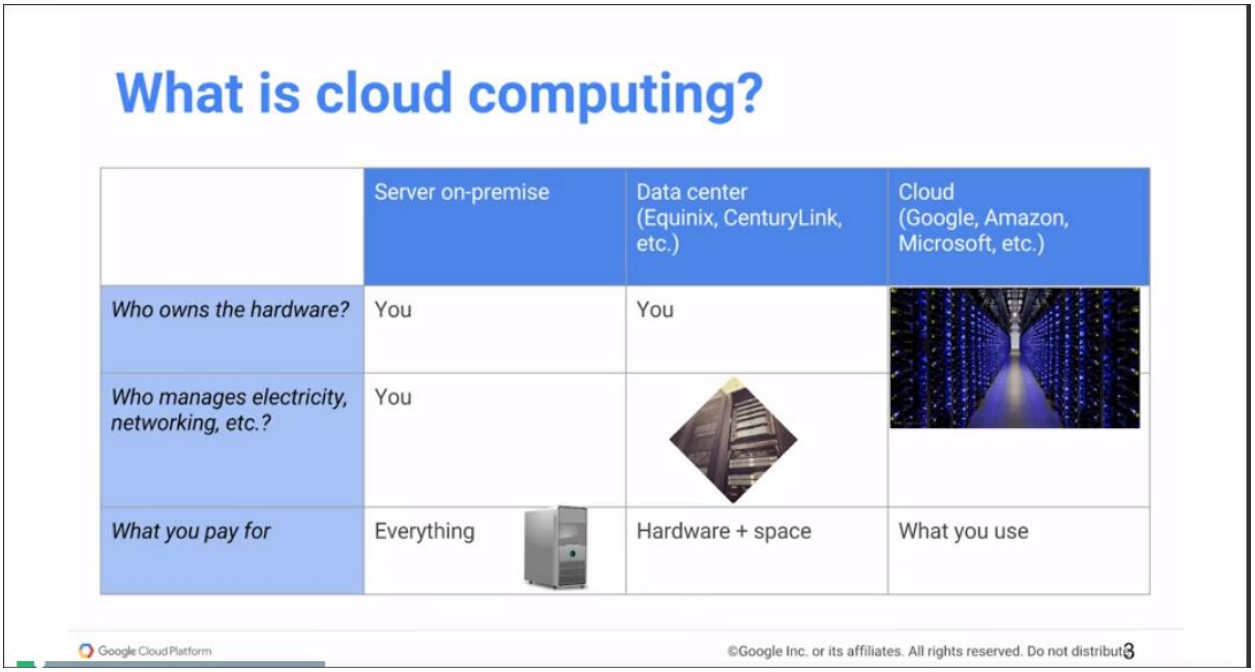
Learning Objectives

- Review Google Cloud Platform and the data handling aspects of the platform
- Identify the purpose and value of the key Big Data and Machine Learning products in the Google Cloud Platform

What is the Google Cloud platform?



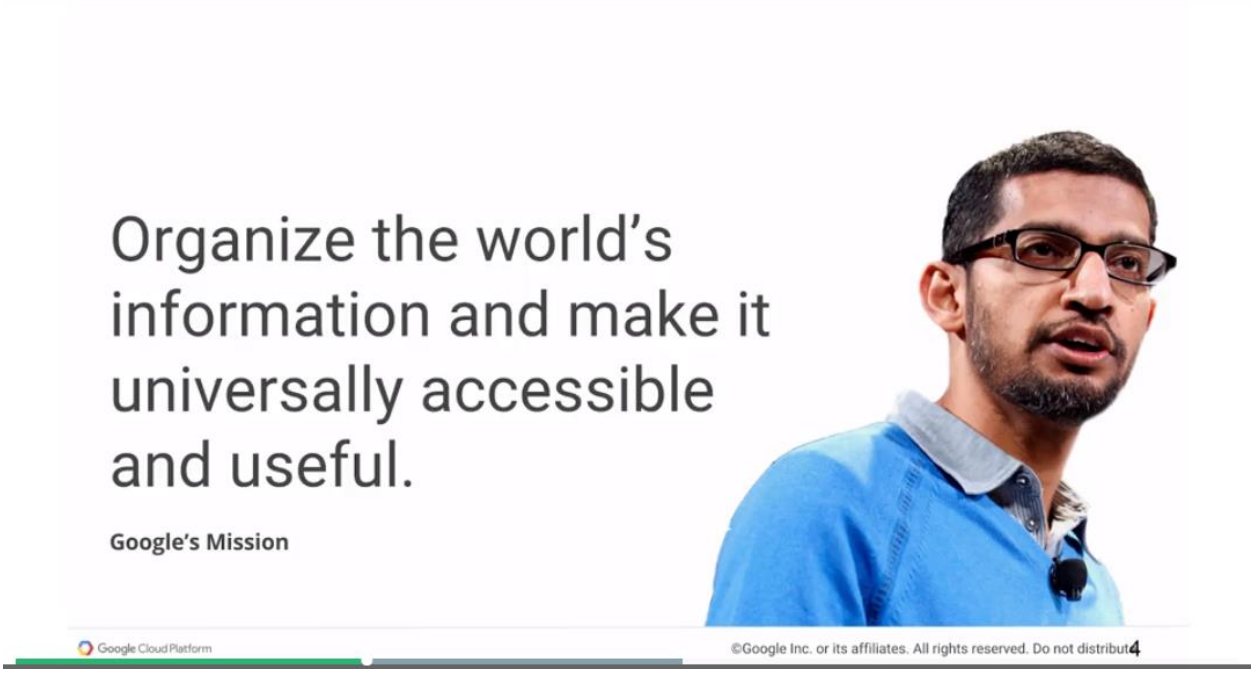
Start by defining cloud computing. When I started out in computing a couple decades ago, everything that we did was on-premise.



Desktop to Data Center. In the data center, we gave up was direct physical access to the computer. Now if you think about it, this evolution from things being completely under my desk to being slightly out of reach but something that I'm still controlling. The phone for data center it's just the, cloud computing is the next step of that. And with cloud computing the big cloud vendors, whether it's Google or Amazon or Microsoft, they own the hardware, they manage the electricity, they manage the networking, they manage the physical security into those data centers.

In the cloud, we ask for computing resource. The computing resource could be a virtual machine, but in many cases, you prefer it not to be virtual machines. You don't want to work at such a low level, where you're spinning up VMs and spinning them down, you want to think in terms of higher level constructs like here is a job that I want to run. Or here's a SQL query that I want to execute, right. But anyway you basically you have some kind of computing that you need to do. And when you need to that, you ask for resources to carry those computing out and you pay only only for those resources that you use. And the whole cloud is a shared resource and you get those resources for the time that you need and you give them them back, and you're not worried anymore about not using it or basically how the over-provisioning servers that you're not using or not having enough of a computing resource when you need it, because somebody else is using it, etc. Right, so the whole idea behind cloud computing is that you have an available resource whenever you need, and you're not paying for it when you don't.

So why is Google in the cloud business?

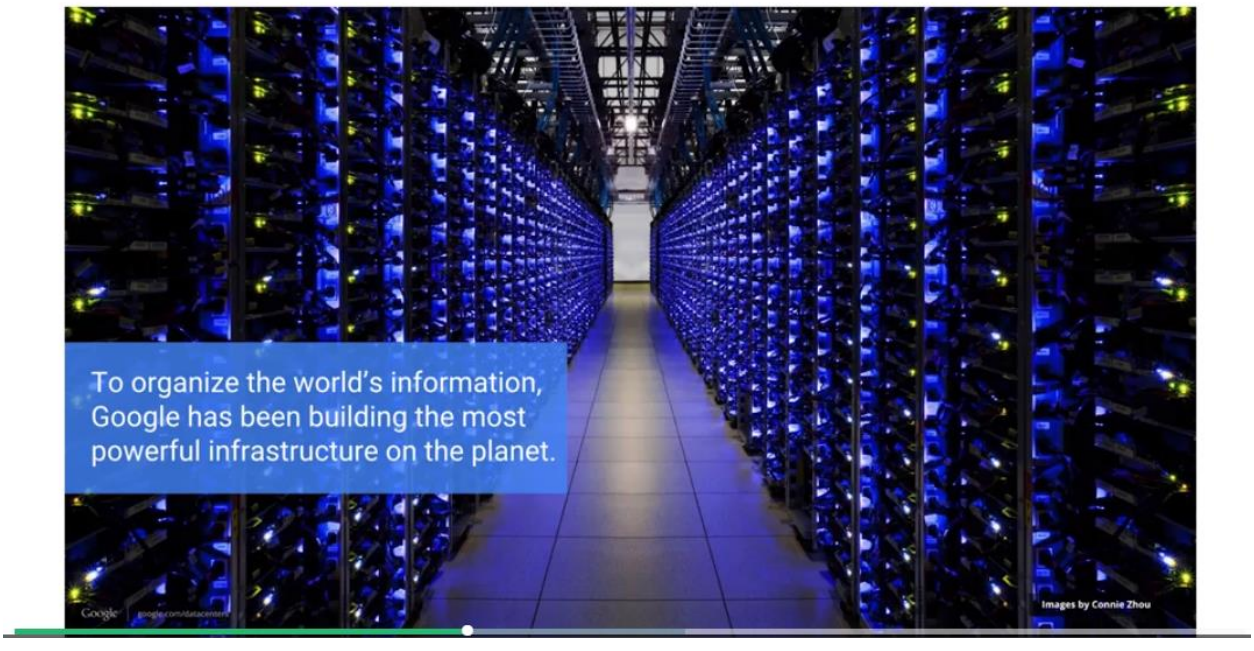


It turns out that if you need to organize the world's information and make it universally accessible and useful, there are some things that you have to do.

One of the things that you have to do if you are going to organize the world's information, because there is a lot of it and it's keeps on growing, is that you have to build extremely powerful infrastructure.

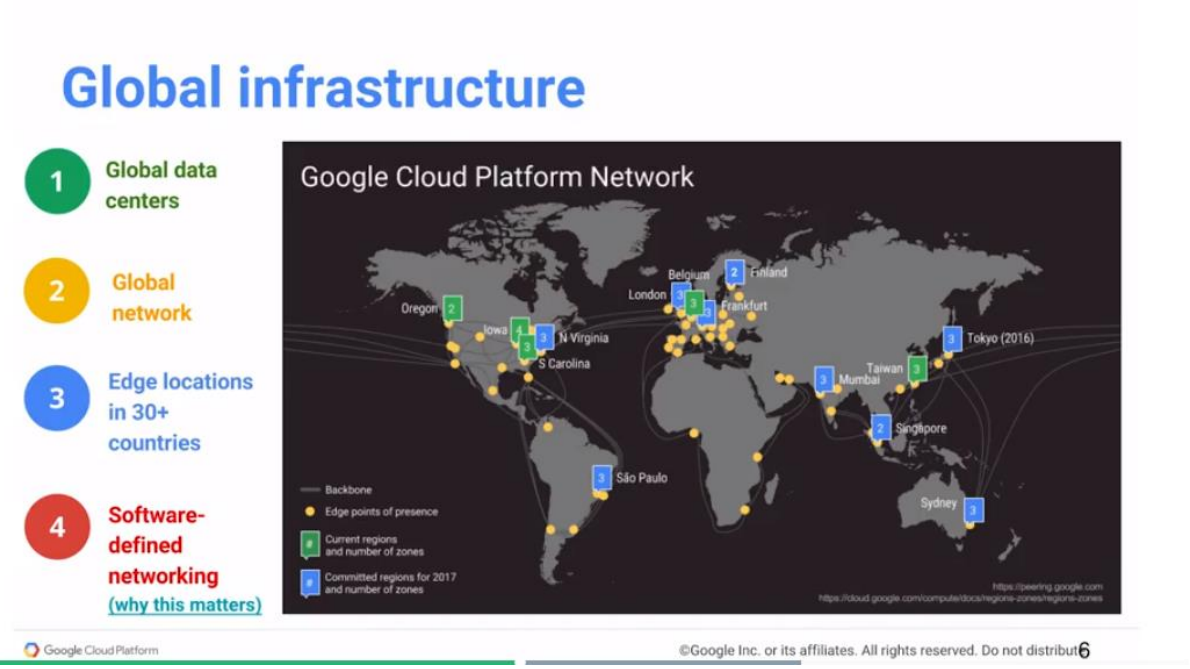
There is a statistic that blew my mind the first time I heard about it.

Of every five CPUs that are produced in the world today, year on year, Google buys one of them.

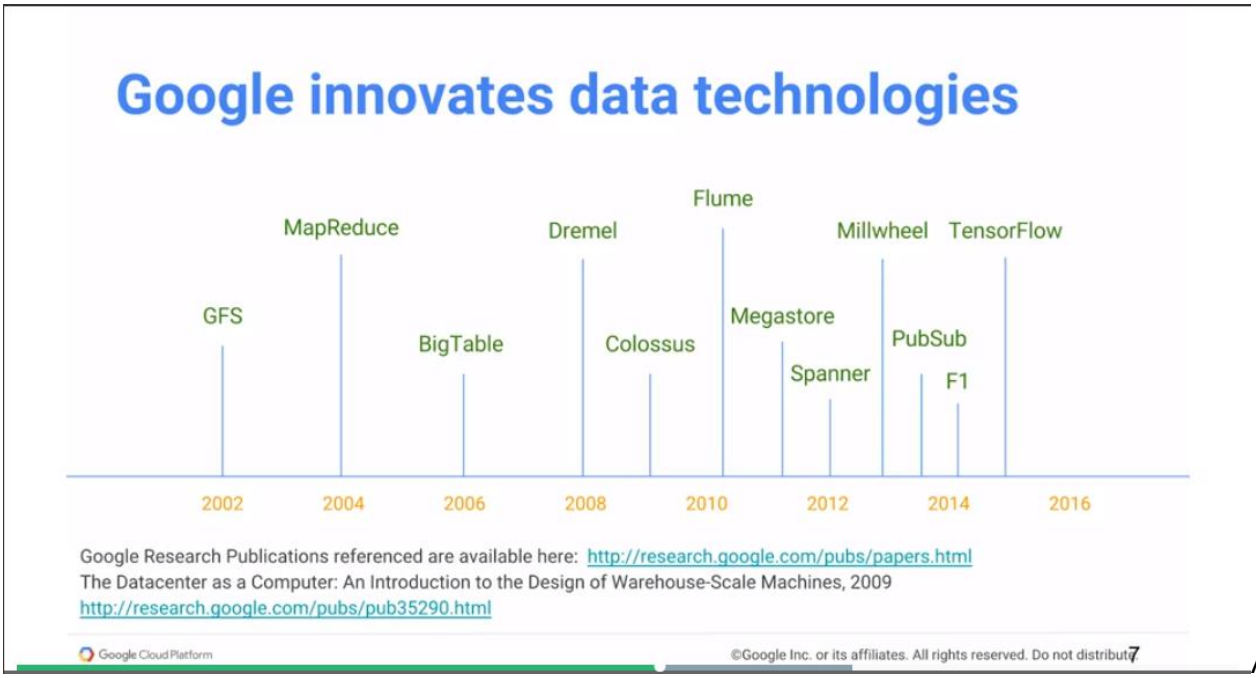


Think about that for a second. Google buys one in every five CPUs that's produced year on year. Goes into our infrastructure. You can imagine how powerful that infrastructure is that we have to build in order to organize the world's information. But that's the physical hardware, right? In addition, we have to make it universally

accessible. And if you want to make information universally accessible, you have to build global data centers, you have to build a global network so we have private fibers between all the continents, and you need to have edge locations in a lot of different countries. An edge location is essentially this idea that if somebody's accessing your resource, say from Africa for example, the second person who's trying to access that resource shouldn't have to go across the world to go get the resource again. They should be able to get a cached version of that resource from an edge location. You need to maintain edge locations, and edge locations are in a lot more places than the data centers. But within the data center also, the design of the data centers is such that any two machines in the data center are just a hop away. That you can basically have some. If you look at the East West cross-section versus, so if you look at the network bisectional bandwidth between machine and google data center it's more than a petabit a second. So now you're looking at an extremely fast networking and extremely global infrastructure that needed to get built in order to make that information that we had collected and organized universally accessible. That's the physical part of it.



In addition, we've tended to run into problems of large amount of data ahead of most other people, most other companies in the world. We have ended up innovating in data technologies, pioneering many of the alternative techniques that you're all familiar with now. So, for example, GFS, the Google File System, and Map Produce, those were two papers that came out of Google research.

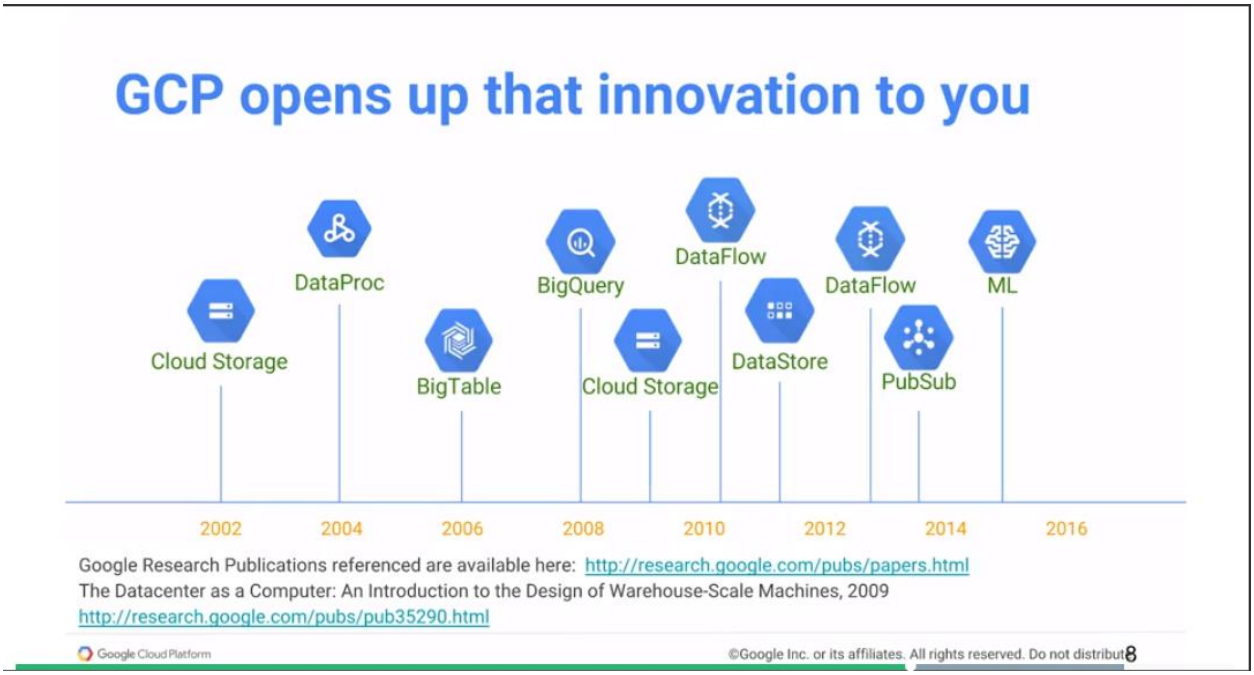


And they from HDFS, the Hadoop Distributed File System is based on the Google files system GFS. And Hadoop itself, the MapReduce frame work is based on the paper by Sanjay Ghemawat and Jeff Dean on MapReduce (2004).

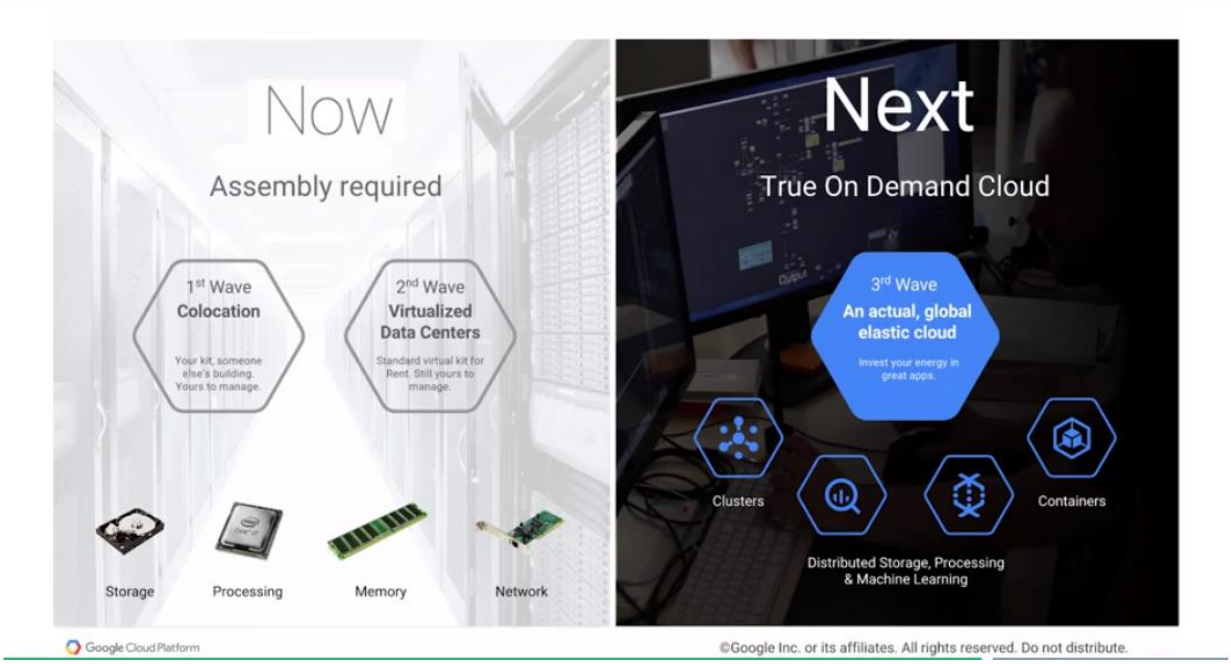
And, of course, HDFS and HADOOP basically led to the whole ecosystem of a bright and open source at DOOP Tools that are available in the world today. Something to kind of realize is that even though we published this paper in 2004 in MapReduce, by about 2006 we were no longer creating new MapReduce programs. We had moved on.

Why had we moved on? Well, some people realized that the whole MapReduce framework the way it works, is that if you have a very, very, very large dataset, you take that dataset and you chop it up into small pieces and you store those pieces on different compute notes. Close to the compute, and then you basically have each of the compute notes doing their little bit of processing on their local data. Those are the map operations. And then take those results. You combine them and then you basically do processing on it. But the key point is that you have to take your data set, and you have to shard it, or split it, across all of the computer notes, which means you are all decisive of your data sets and decisive of your computer notes are intimately tied together, and that kind of limits your scheme, because you are often wasting a whole bunch of computer nodes just because you need to store your data there. Or if you need to process some data you can only process it on the compute notes that already have that data.

This changed - this is part of the innovation and data technology that came about such that GFS, the Google file system, got replaced by Colossus, which is the current file system. Of course, there have been enhancements in Colossus all along the way. We don't do MapReduce at Google anymore. Instead, the data processing technology of choice are Dremel and Flume, and those are both externalized in Google Cloud as BigQuery and Dataflow, that we will look at.



The point is that with all of the innovation that's happening at Google to build our own infrastructure, with Google cloud we're opening up that innovation such that you can use it. So Dremel, which I talked about that In 2008 is the basis of Big Query. Flume, the thing that started in 2010, it started in 2010, but it has continually gotten enhanced. Flume and Millwheel, etc., they all form what is now externalized as data flow. And TensorFlow 2015 forms a basis of cloud ML. Cloud ML is a whole state and in flow solution. Similarly you have PubSub which has been released as itself. And spoonerism now in alpha etc. So you basically have all these innovations that continually get released into Google Cloud. So, one of the things is that if you're working on Google Cloud, you basically have access to all of Google's data processing capabilities.



Now, I talked about how the Map reduce that we used to use in 2004, we don't use anymore. One of the reasons that we don't use it is because not produced tends to be limited by the number of compute nodes that

you have, and that we have better solutions now in the form of big query and in the form of data flow. And that illustrates a key point.

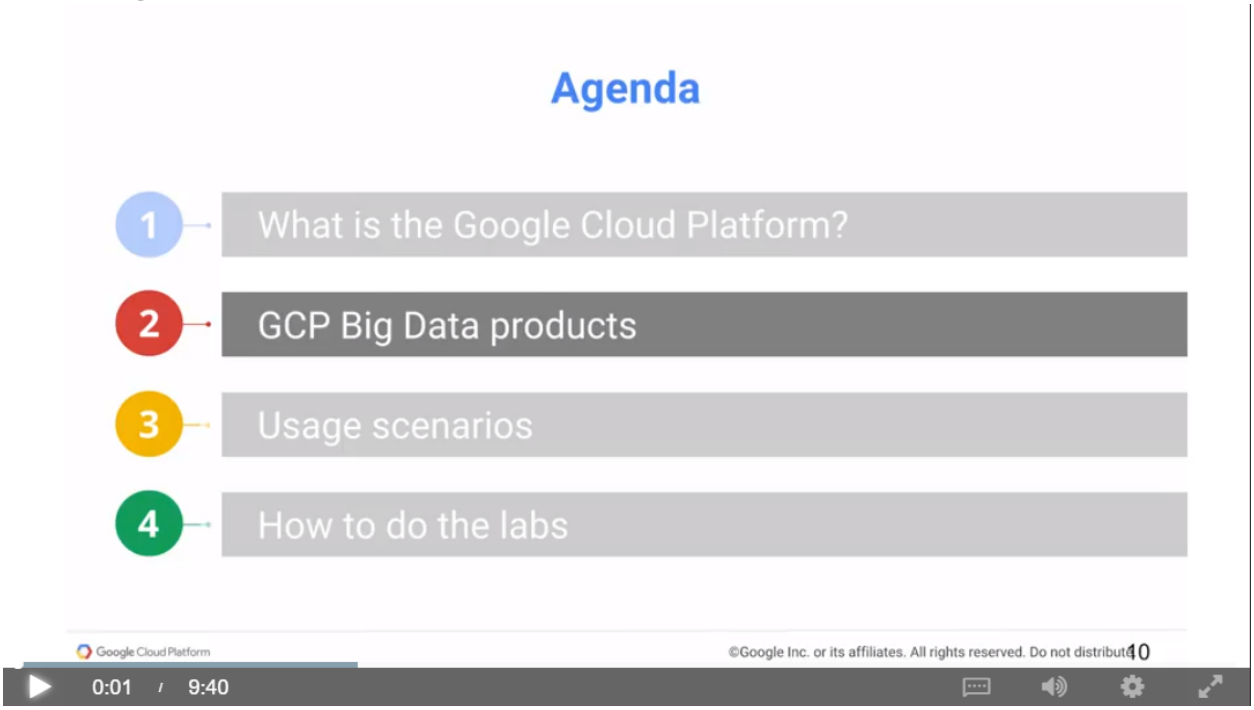
If you're doing cloud computing today, many times though what you may think of as cloud is essentially the change from core location to a vitalized data center. Remember the slide that I showed early on that said what is cloud computing? Cloud computing is evolution of you going from things that are being on premise to being on a data center, where you know you basically still own the hardware, but part of the management of that hardware has gone away and that with cloud computing you don't have to own the hardware you don't have to manage any of it. You just get to get the resources that you need. Right, that is the idea. But if you are working in the cloud at the level of virtual machines, if you're saying, I'm going to spin up a VM so I can run this job and I'm going to reserve this VM for months on end so I can keep running this job,

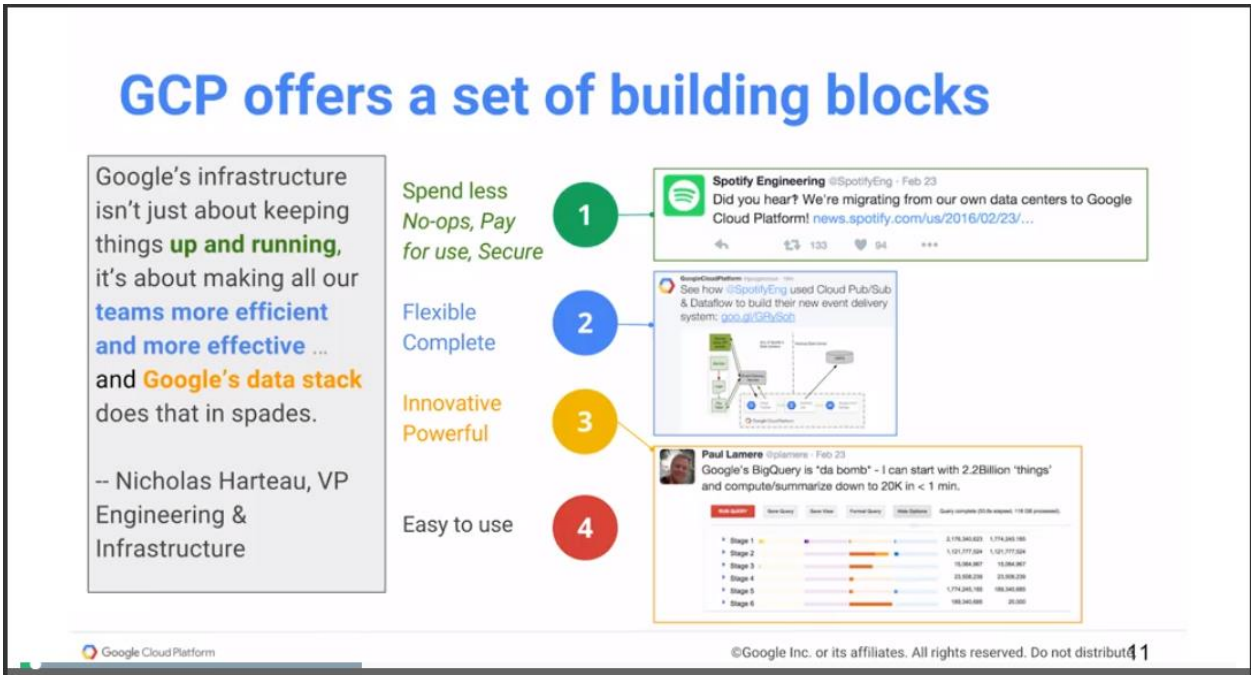
At that point, you've lost much of the benefits of the cloud. The cloud is hugely beneficial, if you are running things ephemerally, if you're running things just when you need them. So that's kind of what we're talking about when we talk about how a virtualized data center is your second wave of cloud. It's not yet getting the full benefit of the cloud. The true benefit of the cloud happens when you're using the cloud in an elastic completely global way. And what I mean by that is that you should be able to auto-scale your clusters, you should be able to do distributed storage, distributed data processing, distributed machine learning. If you have a job, that job can be done on a thousand machines

in seconds, you should be able to do it on a thousand machines for ten seconds and just pay for that. You shouldn't have to go ahead and create a cluster of 10 machines, and certain process that job on those ten machines for 10 hours or twelve hours or whatever it takes you.

The whole idea behind an elastic cloud is this idea that you get to use your machines just for the time that you need them, and then when you don't need them, it's not on your bill anymore.

GCP Big Data Products





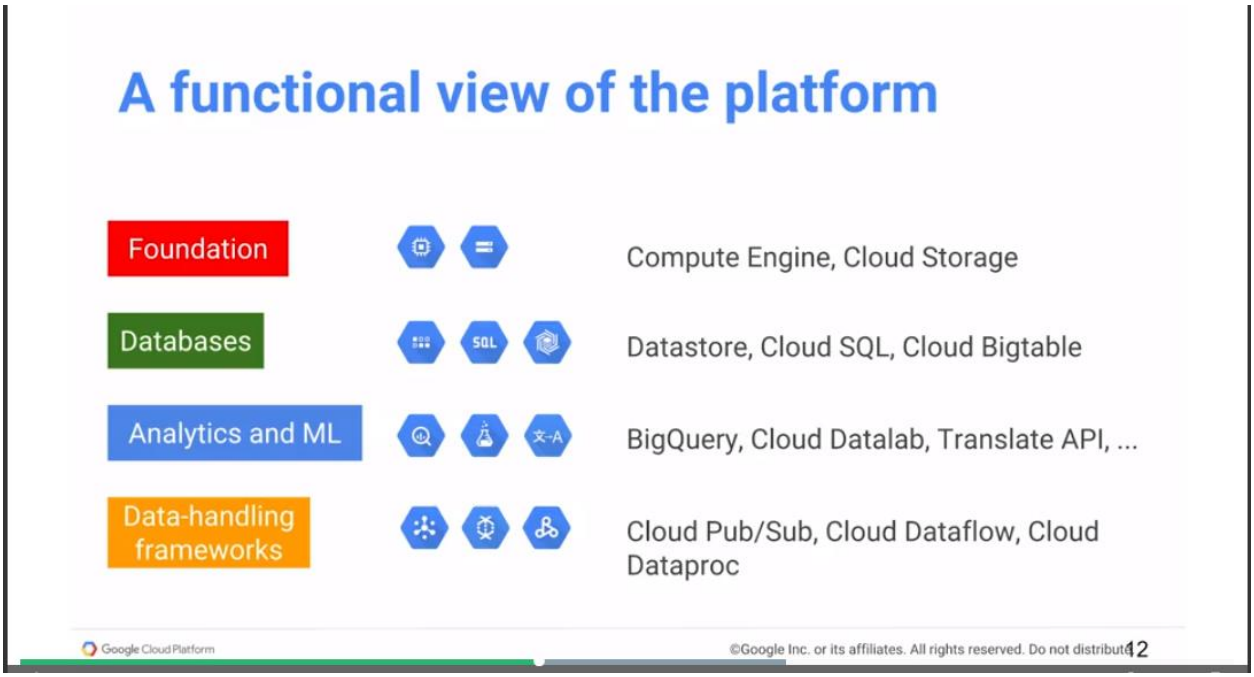
The best way to think about the Google Cloud Platform is that it's a set of building blocks. It is a set of things that are well designed such that they are well designed, that they are well integrated, such that you can put together your own solutions to solve your problems in a very convenient and very intuitive way. Spotify illustrates the typical customer journey. Spotify had a lot of tweets so they came to Google Cloud, the reason that most people come to the cloud. It's not to spend less, but to just pay for the use to basically get more security, now because things can be safer if they're secured on the cloud with Google's security engineering team. It can be no-ops, no-ops is, again, a word that we'll use quite a bit. And what we mean by no-ops in this context, a no-op is essentially no systems operation required. In other words, you don't need to spin up a cluster and do a job on it. You just have a job, and you submit it to the cloud. Google cloud figures out which machines to run it on, runs it, and you don't even interact with those machines at all, for example. So those are all advantages. So that's the reason why many companies come to the cloud. And that's why Spotify came to the cloud. This was one of their tweets, did you hear we're migrating from our own data centers to Google Cloud Platform?

They realized that there are two nice building blocks in Google cloud: Pub/Sub with some messaging system and data flow with a data pipeline execution environment. Integrating these two things together, we can build our own event delivery system. So that's our second tweet. See how we use Google cloud, Pub/Sub, and data flow to build our own event delivery system. In other words, there is this whole flexibility to put together your own solutions.

That's great, but in addition to building your own solutions, there is also the extreme power that already comes built in. And that was that third tweet where there's a Google's Big Query is the bomb. You can start with 2.2 billion things and summarize down to 20,000 things in less than a minute. And the reason that he's so excited is that before Big Query, it probably took them, I don't know, maybe an hour, maybe a couple of hours.

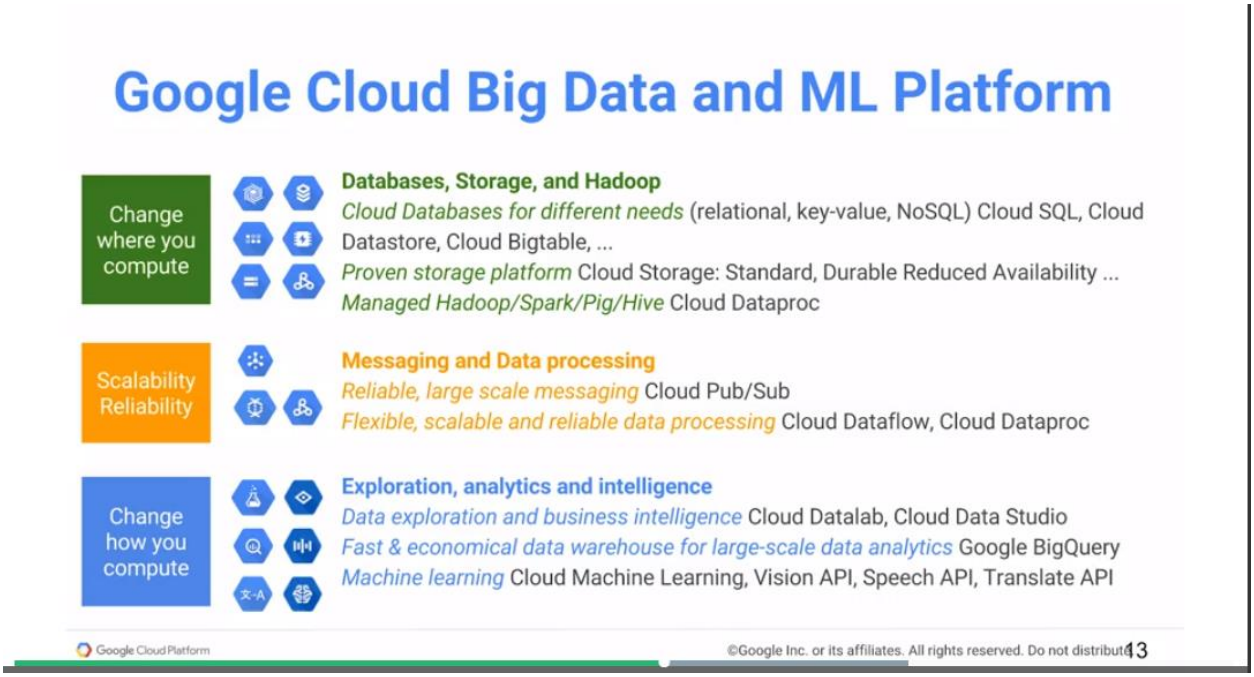
There's a huge benefit in terms of business advantage, to take things that used to take you hours and do them in seconds. And that's kind of where that's a promise, that transformational aspect of doing things on the cloud. You may never be able to convince your boss to give you 3,000 machines if you're not going to use those

3,000 machines all the time. But on the cloud, you can get 3,000 machines for a few minutes, get your job done. And that time savings is worth a lot.



Another way to look at the Google cloud is functionally, what does it do? What do these products do? Each of these blue hexagons is a Google cloud platform product. And they all do different things. You can group them. For example, there is the foundational pieces, there is the computing, there is storage, Compute Engine and Cloud Storage, which are a foundation. You have a number of types of databases, Cloud Datastore, Cloud SQL, Bigtable, etc. These all have different target use cases, and we'll talk about them, why you might want to use Datastore versus Cloud SQL versus Bigtable. But these are all databases. And then you have some things that are about analytics, like BigQuery, or about machine learning, like Cloud ML, or the Translate API, or the Vision API, etc.

Then there are data-handling frameworks, things that help you deal with streaming data, that help you move data from one place to another. This is things like Pub/Sub, Dataflow, Dataproc, etc. This is the way that we are going to look at the cloud platform, we look at them in terms of these things. We look at foundational stuff, we look at databases, we look at analytics and ML. And then we look at data-handling frameworks.



The reason that there are all of these big hexagons is that Google is trying to solve a particular issue related to how people get onto the cloud. Many times, people want to migrate the code that they're already running. It's just about changing where they're computing, not actually changing the code itself. Because you don't want to do too many things at the same time. You may take software services that you're running on-premise, and you may might want to move it to Google cloud.

in that sense, all that you want to do is to change where you compute, but you want to do exactly the same things. If today, in your own data center, you're running a Hadoop cluster - you just want to move that code, that project, over to the cloud platform,

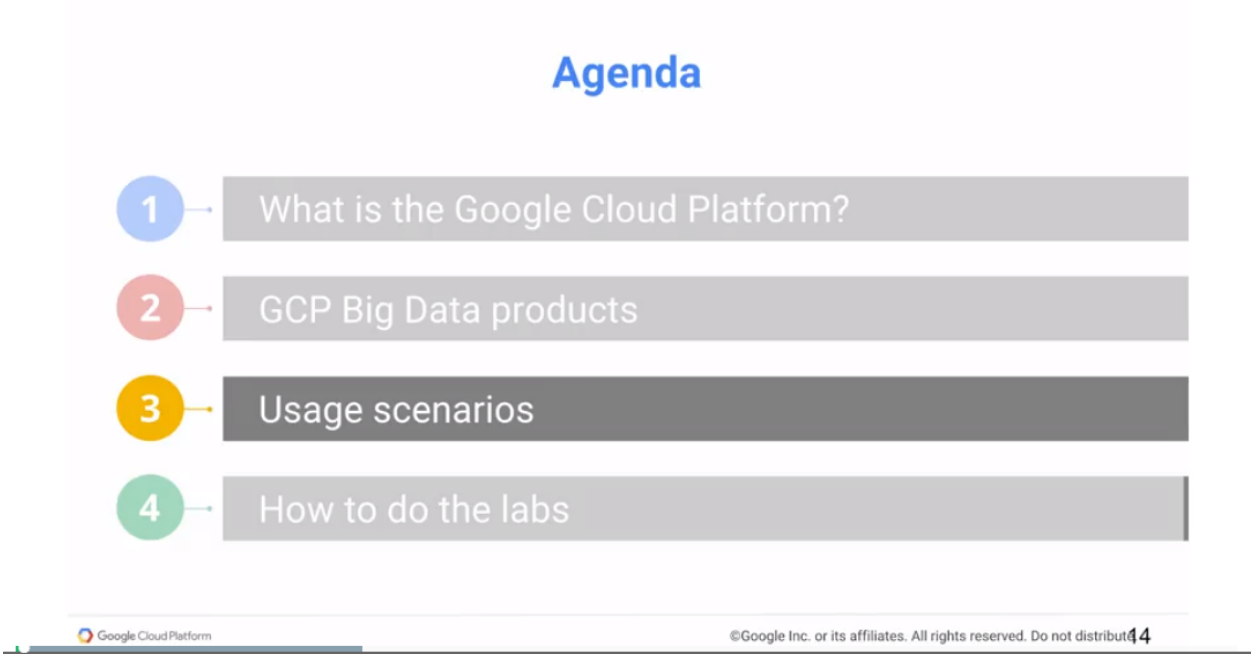
You can take Cloud Dataproc and migrate things over. You can migrate your Hadoop, Spark, or big jobs over to Google cloud, and run it on Cloud Dataproc. Similarly, if you have a MySQL database, you could run it on Google Cloud using Cloud SQL. These are all different ways of taking things that you're already doing and just changing where you're doing the computation.

At another level, the reason that you maybe move in to the cloud is because the cloud gives you greater scalability and reliability. If you need to do very large scale, very reliable messaging, you might want to use Cloud Pub/Sub. If you want to do very scalable, very flexible, very reliable data processing, you may want to do Dataflow. But you might also do it in Spark with Dataproc and take advantage of the fact that with Dataproc, you'll get to resize clusters very quickly, right? You may be looking at using the DataFlow, Dataproc, Pub/Sub for the scalability and reliability that the cloud provides. This is the concept that I was talking about where if you are running things on a cluster of, say, not 20 machines or 30 machines, then the reason you're doing it is because you have to justify the cost on an annual basis. You may be able to go ahead and do that exact same thing in a much more scalable fashion on the cloud because you can easily justify using hundreds of machines for now 20, 30 minutes, rather than having to have those things crunched through your data and take now days or months.

The third reason why people move to the cloud is all of the innovation, right, the things that the cloud, Google's environment makes possible. So, whether it's data exploration, business intelligence, and economic data warehouse for petabytes of data, whether distributed machine learning, those are all things that change how

you do your computing. These are not things that you may be doing today. You may not be analyzing petabytes of data because it may not be the kind of thing that you can do on a timely basis. But you can do that on Google cloud, and that basically means that now rather than just moving code that you're doing over, you may be taking and creating new business concepts, new capabilities. New lines of business that open up because you can now do something that you didn't used to be able to do. Right, maybe you're able to analyze your factory floor in real time, which you weren't able to do. Or maybe you're able to analyze your customer's behavior, give them recommendations on what to buy in real time, which you may not have been able to do. Those are the kinds of transformational use cases that you may be able to do. If you're looking at Google cloud, then there are three possible situations that people come to the cloud that now, again, this is stuff that we see because I'm part of the Professional Services Org at Google. We can see these different use cases play out all the time. There are people who are doing migrations, changing where to compute, people trying to scale up their data processing or make it more reliable, and people looking at transforming their businesses. All three of these things are supported by the BigData and Machine Learning platform, and we will look at all of them.

Usage Scenarios



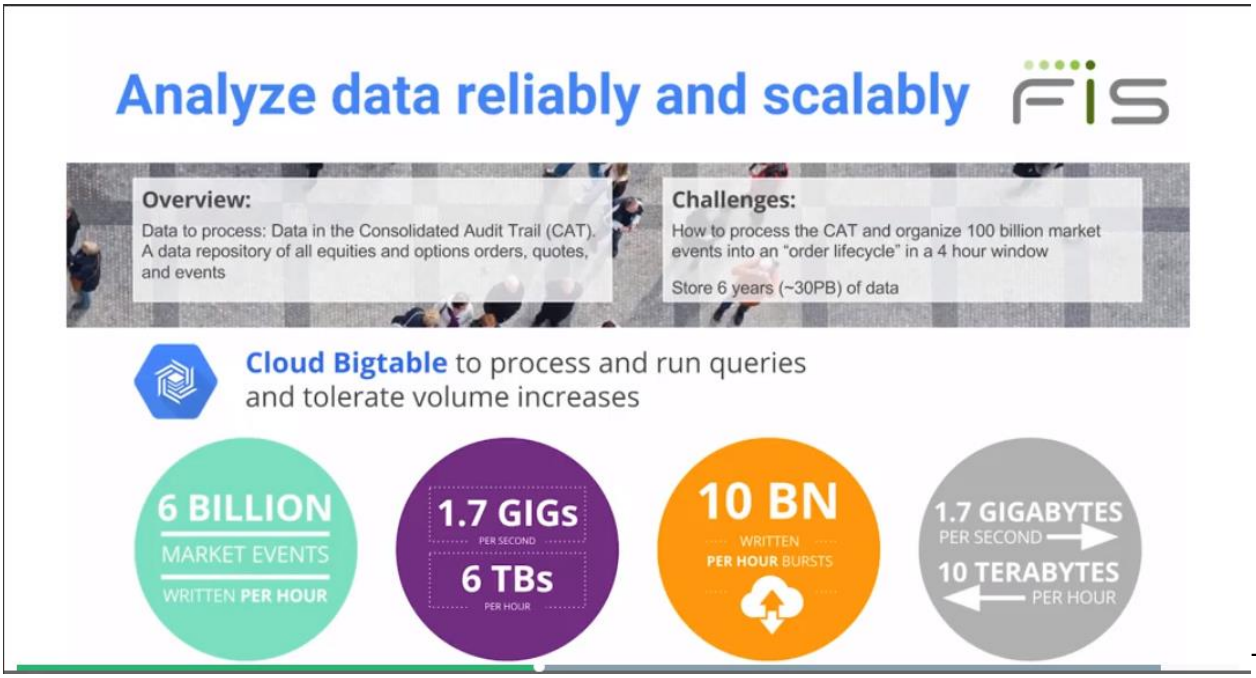
In 2010, we created Atomic Fiction, which is a visual effects and animation studio. One of the most challenging parts of doing visual effects is called rendering. That process can take anywhere between two minutes and 40 hours, for every single frame, every 24th of second of a film.

Rendering takes an immense amount of computing horsepower. Using Google Cloud Platform to do our rendering, we have massive infrastructure offsite when we need it, and then nothing when we don't. It allows us to use computers for minutes at a time and only get billed for what we use. We took that as an opportunity to create our own toolset, more specific to our industry, called Conductor, which runs on top of Google's Cloud Platform. It helped us put the resources on where it was important, the artist, and making sure that every penny that our client spent made it up on the screen. These movies, that might have otherwise been bottled up in a filmmaker's mind, are now actually seeing the light of day on the movie screen. There's no greater satisfaction than creating a visual effect, that people just believe that they're somewhere else. That's what we

do, is we create fiction. Hopefully, it's just the most believable fiction that you can imagine. That's an example of someone who's changing where they compute. It is the same rendering software that needs to run to create these movies. They could have done it on premise, but instead they did it on Google Cloud, because it allowed them to get a lot more machines, get it done faster, and get it done cheaper.

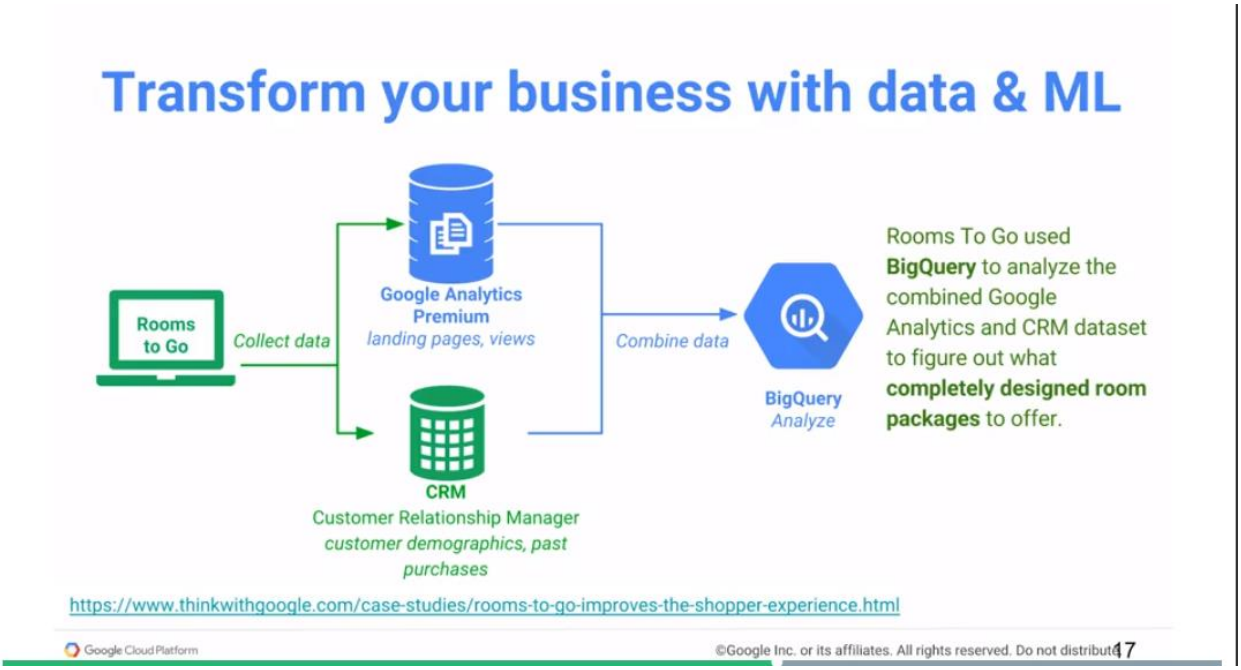
https://www.youtube.com/watch?v=mBY-RjE15WA

As an example of the second thing, which is of how Google Cloud gives you the ability to do reliable and scalable processing of data, take this company called FIS, that needed to basically process the consolidated audit trail. This is every security trade that happens on a US stock exchange. They needed to find events on an entire life cycle. And the kind of data that we're talking about here, is six billion market events per hour, on



That's on average, ten billion per hour at a burst, 1.7 gigabytes a second, 10 terabytes an hour. This is humongous amounts of data that need to get processed. And how much of it can be lost, discarded? None, this is financial data, it's literally money.

If you need to process this kind of data at scale, very, very, very reliably, Google Cloud is a place to do it. And in this case, FIS used Bigtable to process and run queries.



The third use case that we talked about is a transformational use case. And of course, when we take transformation, we tend to think of all of the digital natives, which is why I love this use case. This is a company called Rooms To Go, a furniture retailer.

Not maybe the first company that comes to mind, when you think about machine learning. But what they decided, was that they would basically go ahead and mash up their customer relationship data, based on who had bought what, and combine it with their website, basically looking at what new pieces are being looked at, etc. And using that combination, they could basically go ahead and create completely redesigned room packages and offer it to their customers

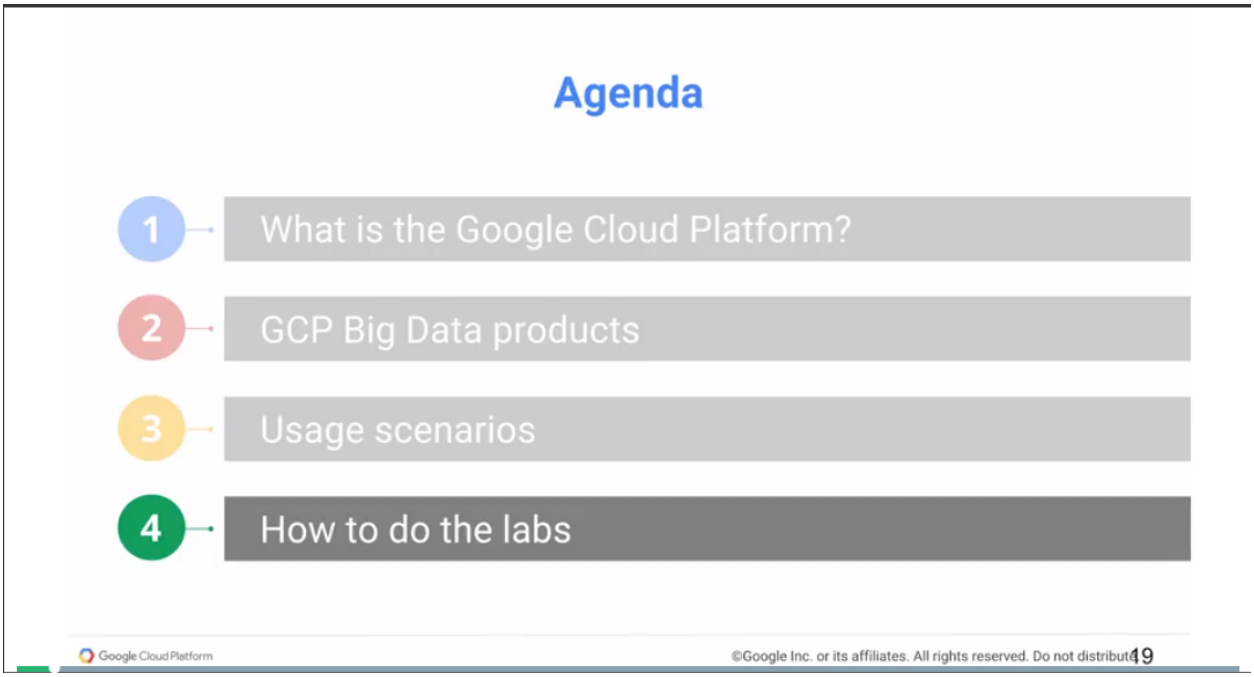
In summary, then, GCP gives you ways to spend less on ops and administration, because we've automated out a lot of the complexity that's involved, if you're building and rolling out your own infrastructure. The second

thing that Google gives you, is the ability to do real-time apps and architectures and to do things faster, get to market faster.

The third thing is the transformational aspect of machine learning.

We give you a lot of machine learning capabilities, to be able to apply ML broadly and easily. And finally, the goal of all of this is such that everybody in your company is capable of deriving insight from data. The goal here is to transform from your organization into a data-driven company, such that if someone needs to make a data-driven decision, they are not waiting for this data to become available. That data is there at their fingertip, for them to be able to make the decision to move your company forward, for your company to become more agile, and GCP is a good way for you to do that.

How to do the Labs



https://codelabs.developers.google.com/cpb100

--

Lab 1

https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/supplement/qYXYF/sign-up-for-the-free-trial-and-create-a-project

https://codelabs.developers.google.com/codelabs/cpb100-free-trial/

Resources

Google Cloud Platform: https://cloud.google.com

Datacenters: https://www.google.com/about/datacenters

Google IT Security: https://cloud.google.com/files/Google-CommonSecurity-WhitePaper-v1.4.pdf

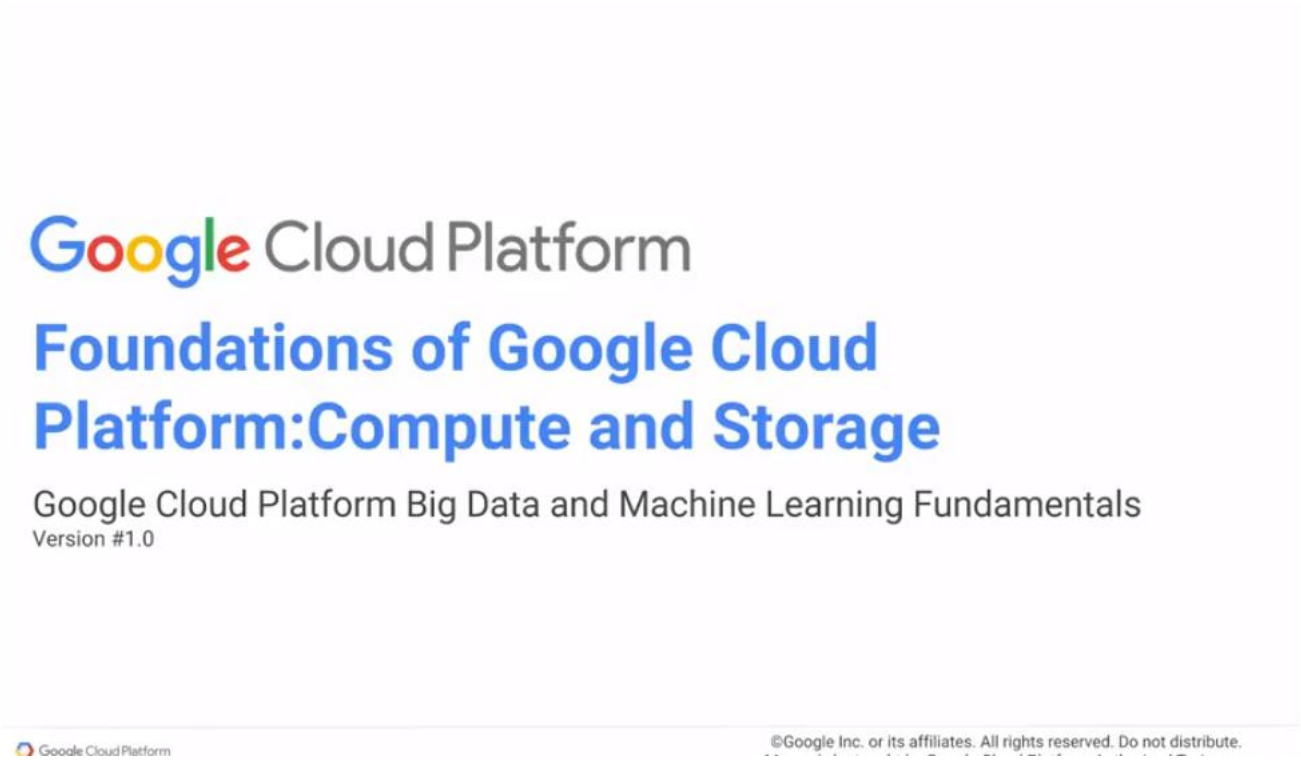
Why Google Cloud Platform: https://cloud.google.com/why-google/

Pricing Philosophy: https://cloud.google.com/pricing/philosophy/

--

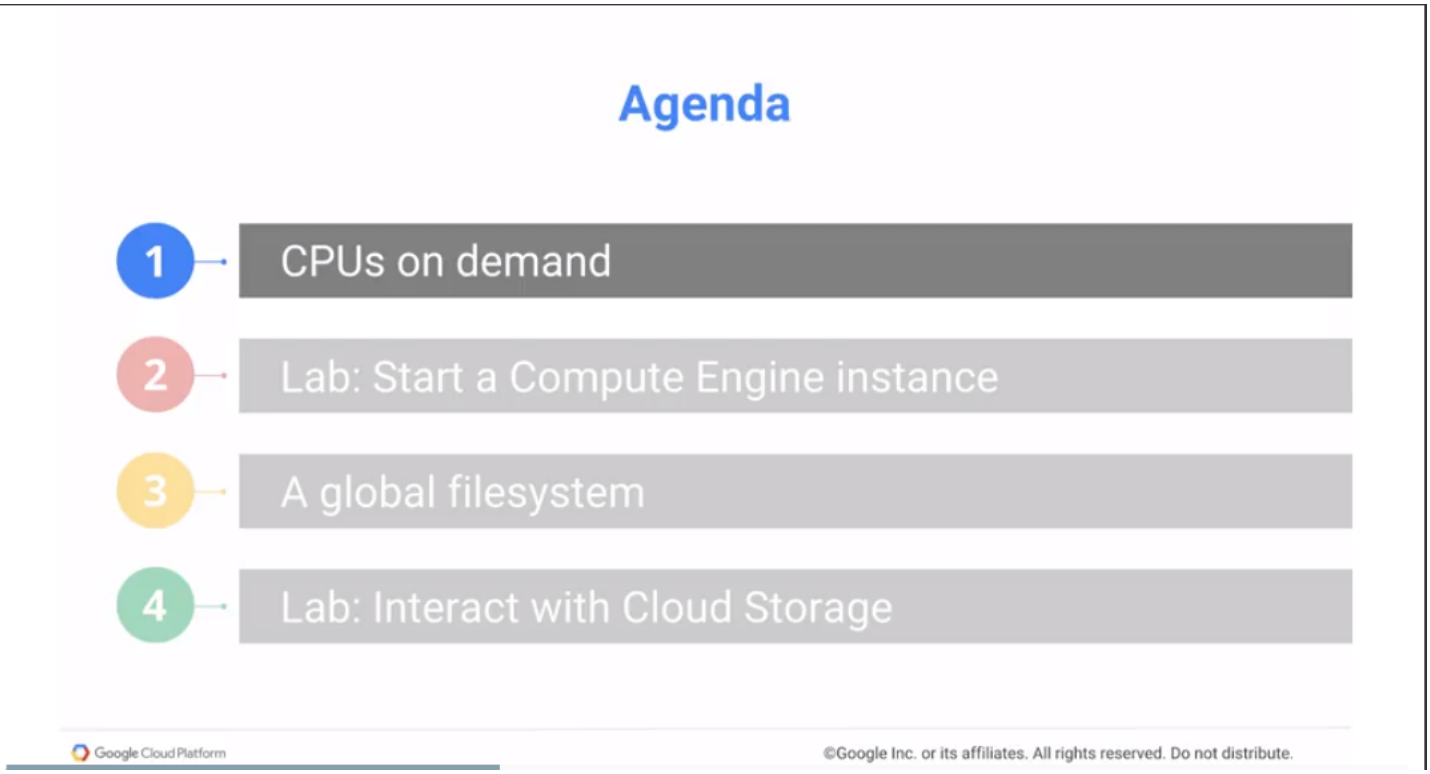
MODULE 2: FOUNDATIONS OF GCP COMPUTE AND STORAGE

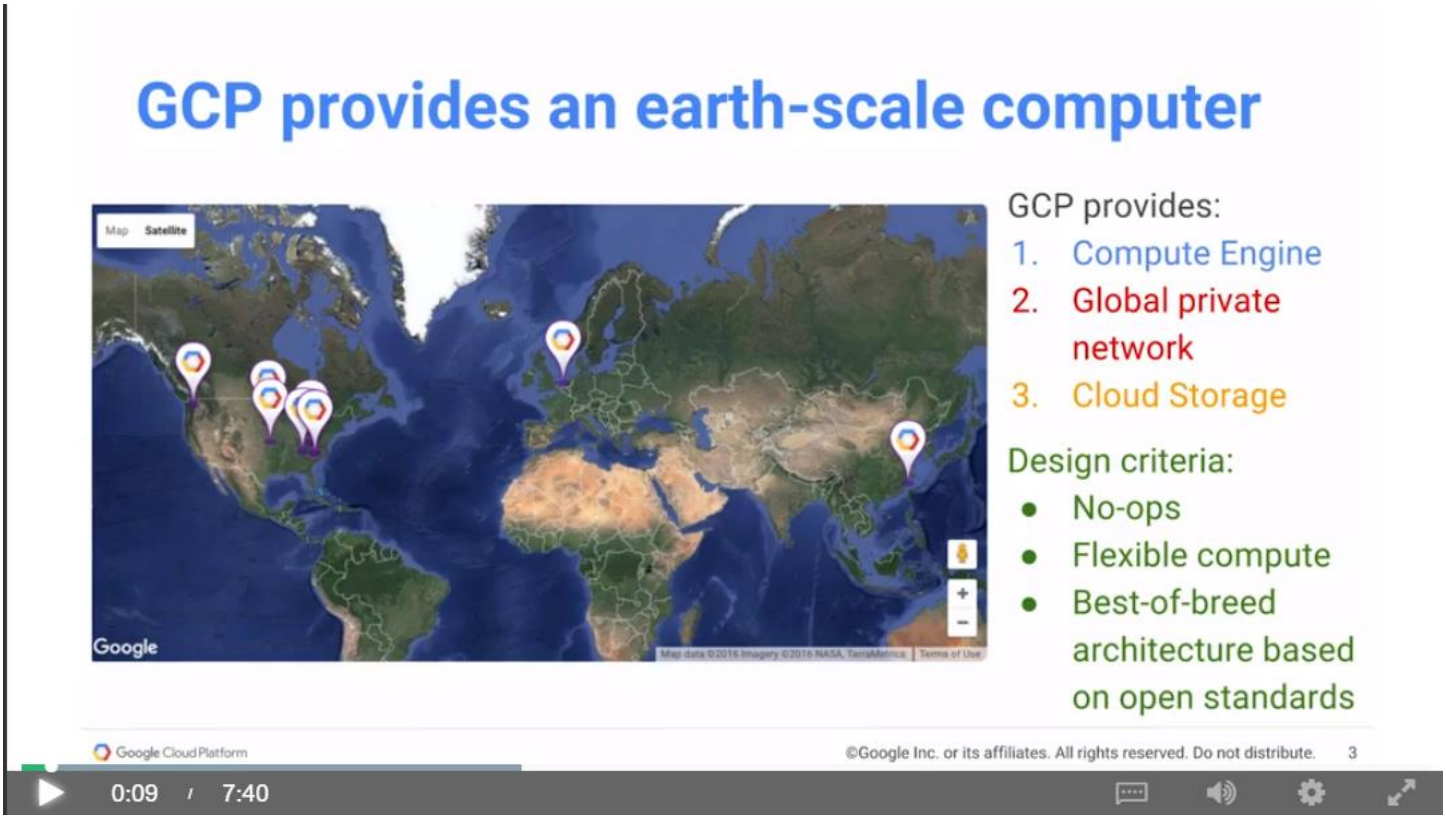
In this module, we introduce the foundations of the Google Cloud Platform: compute and storage and introduce how they work to provide data ingest, storage, and federated analysis. Choose between Cloud SQL, BigTable and Datastore.



The foundations of GCP lie with its computing and storage infrastructure. Any computer consists of computing, and storage, and networking to connect the computing in storage. The Cloud computer, is also a computer. It's a global computer, but it also contains a compute engine. It contains storage and networking that you don't directly interact with, but networking that's there in any case, for you to connect the computing that you are doing with your data that you have stored. In this module we look at the foundations of Google Cloud Platform, the compute engine and cloud storage.

CPUs on Demand





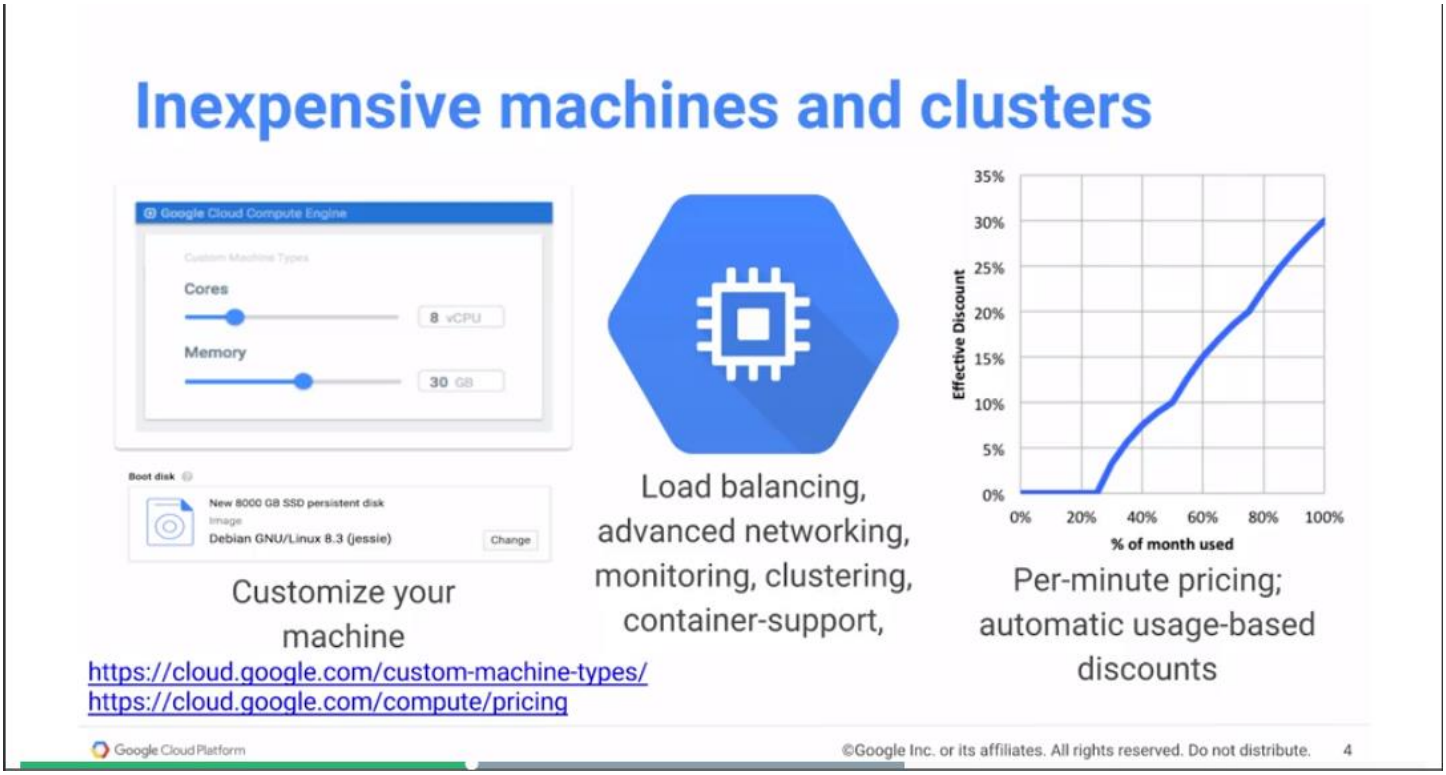
Like any computer, and the cloud computer is a computer, you need compute processing units, CPUs. And the CPUs on the cloud are provided by a compute engine of virtual machines.

And you need a place to store your input data, to store your output data, to store your intermediate data, things that are persistent, things that are temporary. And that storage on GCP is provided by cloud storage. And

connecting the two, connecting the virtual machines or compute engine instances with these storage units or cloud storage is a private network. You think not to directly interact with this network but it's there and it is what allows you to have a global scale data and compute infrastructure.

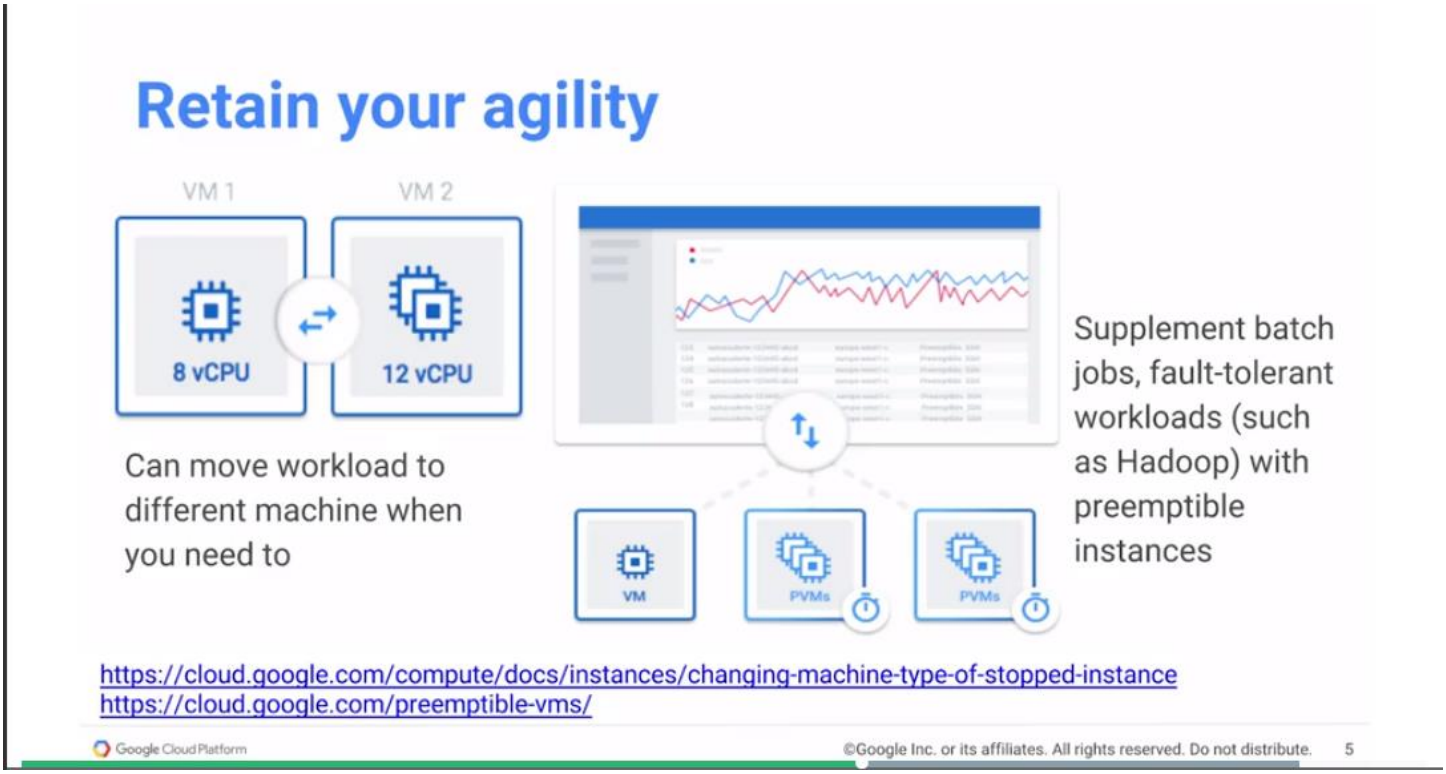
For the most part, when you work with GCP, you will not be working at the level at which we're going to be talking about in this chapter. We will not be working at the level of individual virtual machines. No, you're not going to be spinning up VMs in order to do a job, we'll be working with things that are much higher level than that. But even if you need to work at this low level, in terms of infrastructure, the design goals of GCP remain the same. And the design goal is for working with cloud infrastructure to be as no-ops as possible. And no-ops here essentially means that we want to minimize a system administration overhead. And because we're talking about computing and storage, we want to basically also mention that we want this to be as flexible as possible. In such a way that you can change the type of virtual machine that you're running without paying any penalties, for example. So you're not reserving instances for long periods of time. In fact, we want to make it as flexible and easy for you to get your compute jobs done as possible.

When we talk about compute engine, the idea is in terms of flexibility.



You can go ahead and get a compute engine that is say, N1 standard four and that's a very specific configuration of machines that you can have. But we would like you to stop thinking about it in terms of this very specific physical infrastructure, and instead start thinking in terms of more abstract concepts. For example you might say, I want a virtual machine that has 8 CPUs and 30 gig of RAM.

And it's the job of the Google cloud infrastructure to go ahead and fetch you a virtual machine that has 8 CPUs and 30 gigs of RAM.



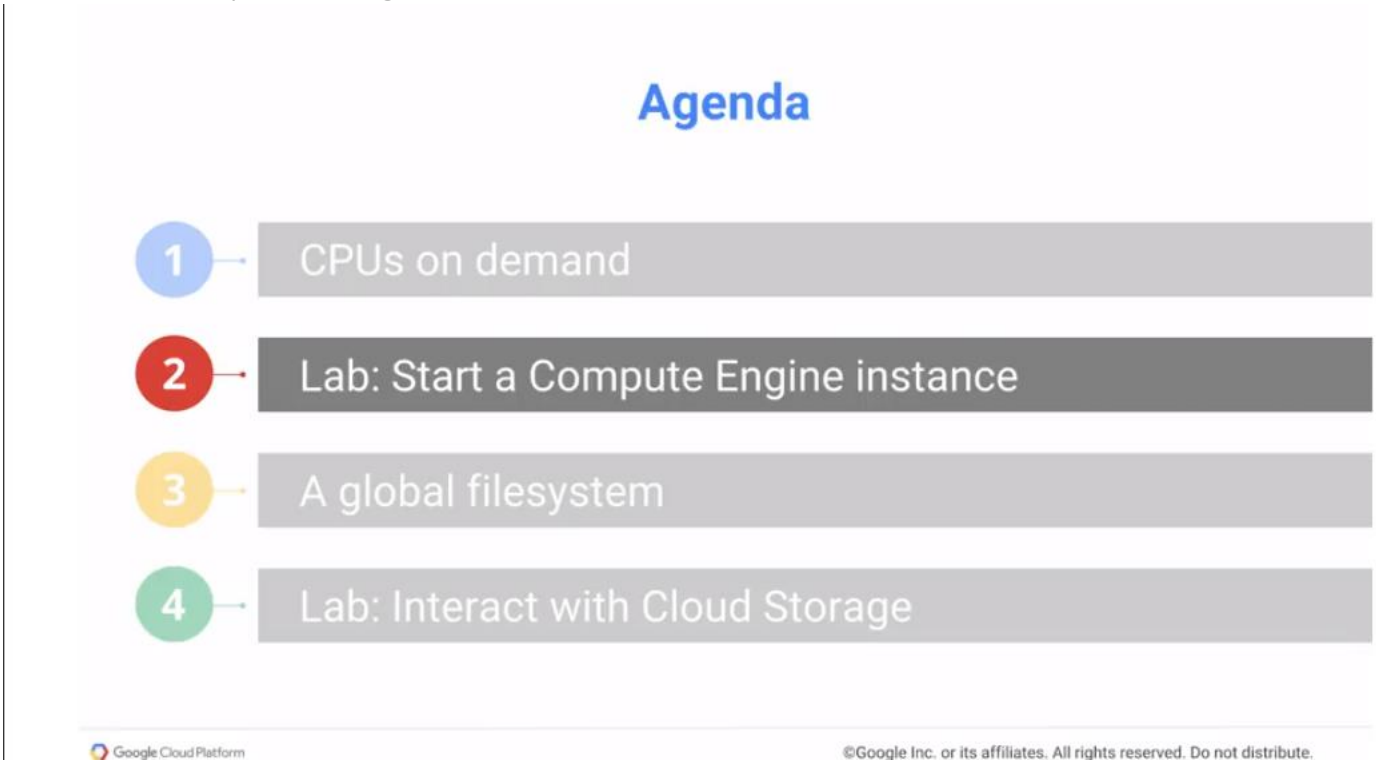
Regardless of the type of machine that you get, you will always get load balancing, advanced networking, monitoring, clustering, container support, etc. There are no second-class machines here. Every machine that you have, has all of these capabilities built in. At the same time, we want to give you flexible compute. And you lose flexibility whenever you say that I have to go ahead and get a machine and keep it running for months on end. Because face it, if you're running a machine for months on end, you have essentially bought the machine. And what we want is for you to work with machines on the order of minutes. However, there are always going to be workloads where you might find yourself having a machine and using it fully tilt for long periods of time. Rather than ask you to try to determine which of your workloads you are going to be running for long periods of time, GCP gives you a discount after the fact. At the end of the month, if it turns out that you've used a machine for 60% of the month, you will automatically get a 15% discount. And this is something that happens on your bill after we've found that you've used it. What this means is that you always get to retain your agility. For example, if you have a workload that's currently running on 8 CPUs and you decide that you need to increase it to 12 CPUs, for a few hours, well, go ahead and do that, right? You can move your workload to a different machine when you need to and move it back to a smaller machine when the peak loads go away. In addition to this whole idea of being able to change the machine type of stopped instances, you have another concept that's very, very, very useful, especially when it comes to jobs like Hadoop jobs. And this is the idea for pre-emptible virtual machine. The reason that GCP, one of the reasons that GCP can say, well, if you want an 8 CPU machine, 30 gigabytes of RAM, we'll find it for you and we'll give to you, is because some of those machines that are currently being used are what are called pre-emptible. Whoever is using those machines has agreed that in return for a hefty 80% discount on the machine charge, they agree to give it up if someone comes along and is willing to pay full price for those machines. So that's what a pre-emptible machine is. A pre-emptible machine is a machine that you get a great discount on

in return for your flexibility, you agree to let go of it when you don't need it. But why would you do that? Why would have a machine that you're willing to give up?

If you're running a workload like Hadoop, which is fault-tolerant, if a machine goes away, well, whatever that machine was doing, those jobs get basically distributed among the other workers. Then pre-emptible machines are a great strategy to reduce your overall cost. You might say, for example, that you're creating a data proc cluster, a data proc is a Hadoop cluster on GCP, but we look at it in the next chapter. You may say I'm going to create a data proc cluster, and in my data proc cluster, I'm going to have 10 standard VMs and 30 pre-emptible VMs. So now your job is going to get done four times faster.

And at the same time, those extra 30 machines that you're using, are actually at 80% of the normal cost. So not only are you getting it done faster, you're also getting it done cheaper. So pre-emptible machines are good thing to incorporate into your strategy. With the idea that even if you don't get pre-emptible machines, those standard machines are enough for you to get the job done in a timely manner. You don't want to bank on a pre-emptible machine being available when you need it, but if it is available and you happen to get it, you automatically gotten a huge discount on the total cost of your job.

Start a Compute Engine Instance



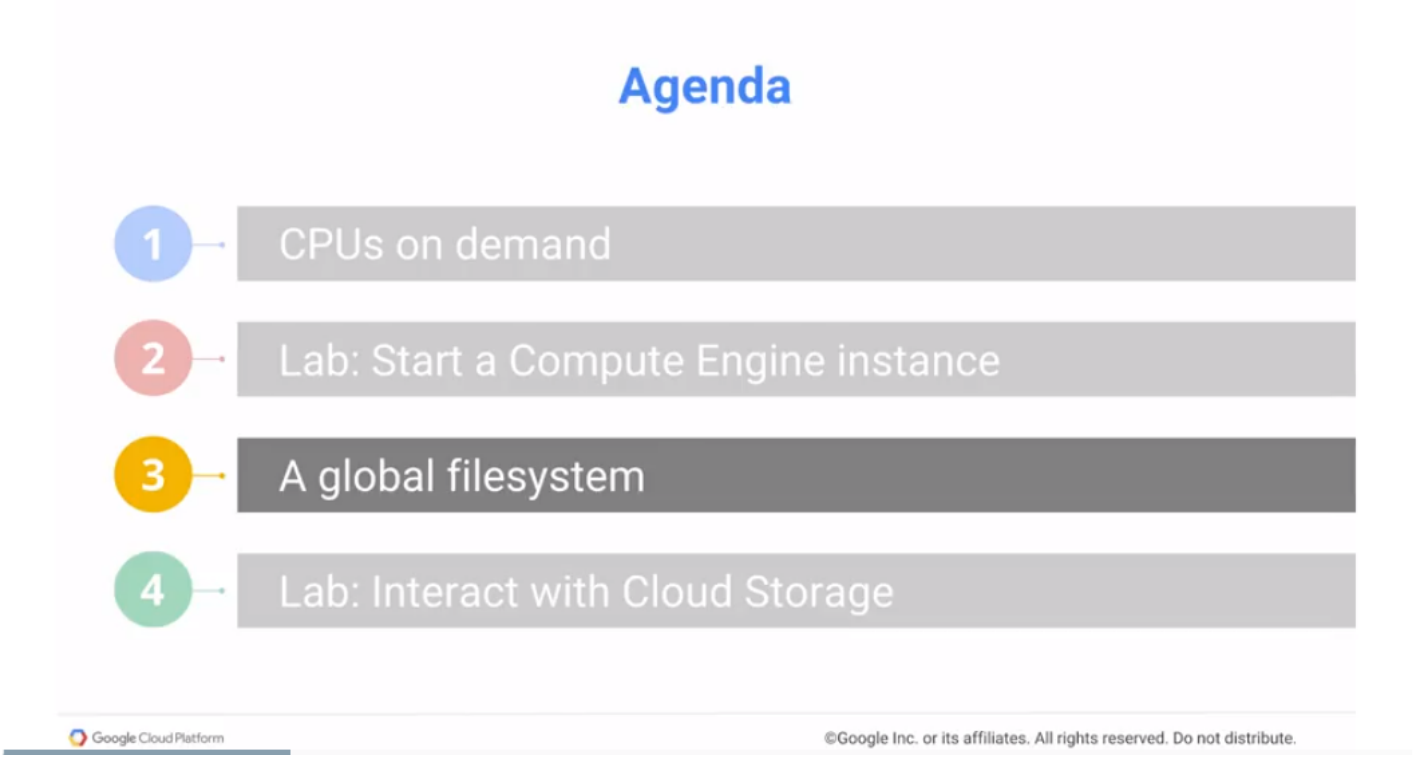
Let's go ahead and try to start a compute engine instance, so we'll start a lab. We'll go ahead and do or start a compute engine instance. And in this compute engine instance, what we are going to do is that we're going to create a compute engine instance using the GCP console. We will add SSH to this instance. And then just to show you that you have root access into this machine, you will install a software package kit, which is basically used for source code version control. Let's go ahead, try out this lab, and come back and join me.

Start Compute Engine instance

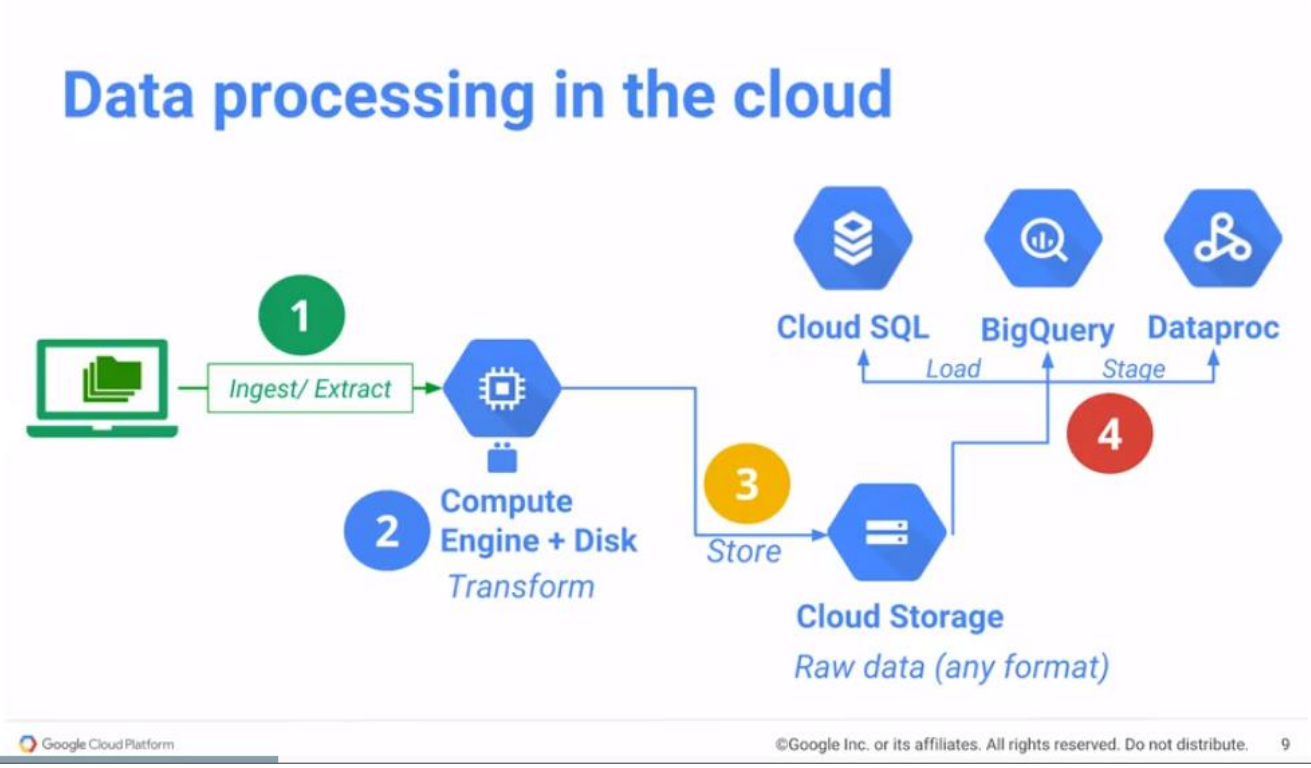
https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/supplement/Mrs7g/start-compute-engine-instance-lab-2a

Review of lab - https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/OxvkJ/lab-2a-review

A Global Filesystem



In the previous section we looked at how to create a compute engine. Let's continue our journey into the low-level infrastructure of GCP by now looking at storage.

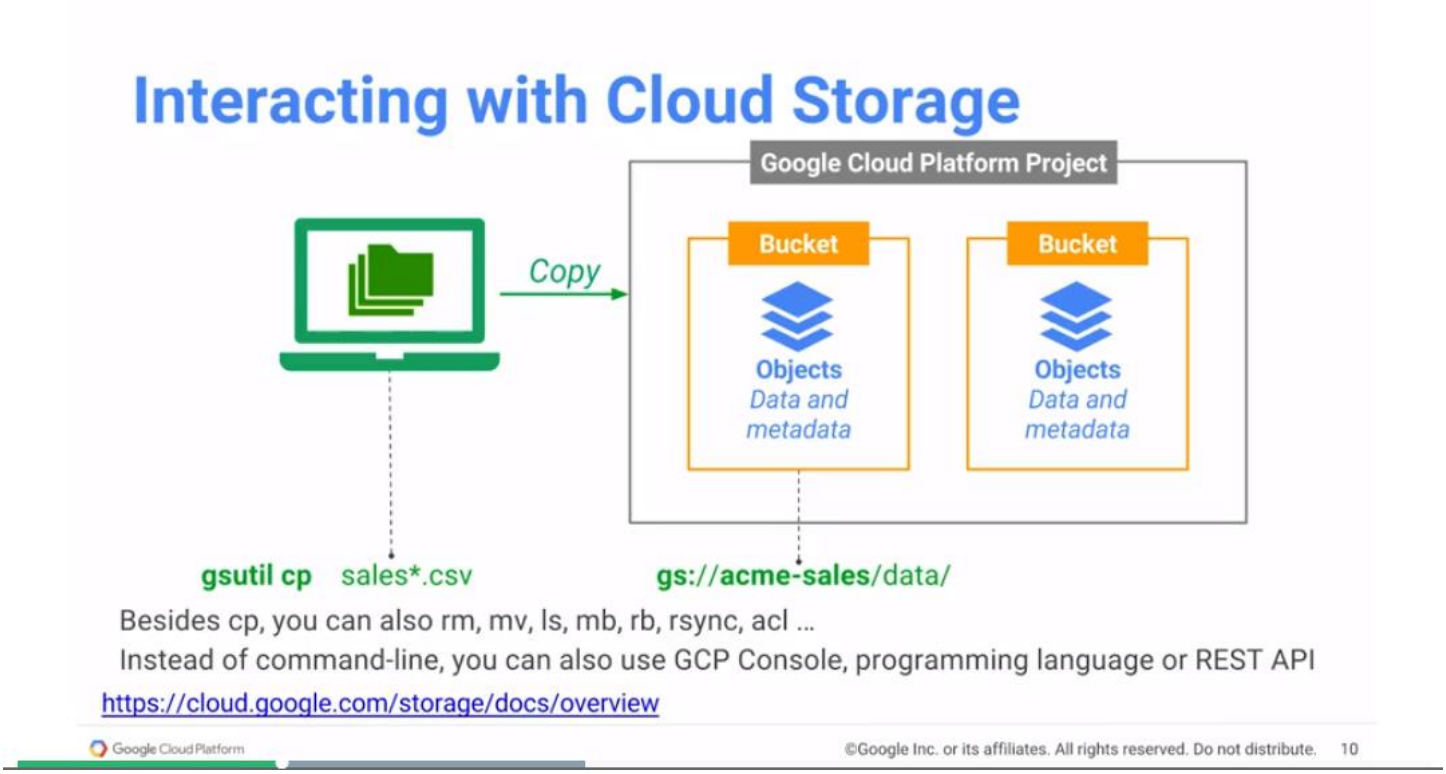


Why are we talking about cloud storage? Well, one of the reasons that we are talking about cloud storage is that this is the way that you stage any input into a relational database which is Cloud SQL, into BigQuery which is a data warehouse, or into Dataproc which is a Hadoop cluster. If you want to stage the data into GCP, you

first have to get the data into cloud storage. Cloud storage is blob storage - storing raw data in any format, directly onto cloud storage. In order to get there though, normally what you might want to do is to get this data from somewhere else. It may be in your data center, it could be on instruments out in the field, it could be logs that are being created. Tends to be that you're basically ingesting extracting this data. You're doing some processing, and that processing could happen on a compute engine. You could have a compute engine VM and you're basically doing a whole bunch of processing. And if you need to do processing, you need very fast seeks, reads and writes of the data. And a good way to do that is to basically store that data on disk. Typical disks are associated with the compute engine that they're attached to. When the compute engine goes away, the disk also goes away.

In order to keep data persistent, the standard practice is to take this data and not store it on a persistent disk, not store it on a disk, because those tend to be expensive, but instead to store it on cloud storage. Cloud storage is persistent storage, it's durable, it's replicated. It can be made globally available if you need to. And you can use it to stage the data onto other GCP products. So often the very first step of the data life cycle is to get your data onto cloud storage.

So how do you get your data onto cloud storage?



The simplest way is to use a command line tool called gsutil. It comes with the G cloud SDK, so you can install the G cloud SDK and once you have G Cloud SDK installed, you will have the gsutil command line. So whichever machine you're going to be uploading the data from, install G Cloud, get gsutil, and then say gsutil copy, that's a cp, gsutil cp sales*.csv. That's a file that I'm copying. Where am I copying to actually files that I'm copying, where am I copying these files to?

Well, in this case I'm copying them to Google Storage, that's GS, and I'm giving them a full destructure /data, and I'm putting all of the sales files in /data. But /dataware, and that's where the concept of a bucket comes in. Loosely, you can think of a bucket as like a domain name, right. Or if you're on the Internet you have different machines, have different machine addresses, and then each of them has their own file system which is your

web pages. Well, buckets are very similar to that. Acme sales in this case, plays the part of a domain name. It's this thing that makes it unique, that's your bucket. And you can create any number of buckets that you want and the bucket name that you provide has to be unique.

Most commonly, what people do is to make their bucket name have some relationship with their corporate domain name.

So you may have addresses that match the company name. But if you're basically going to be using addresses that match a company name, well GCS is going to basically say, well are you really this company, right? And so you need to prove that you own it, typically by changing a D name field etc. Similar to the way that you prove that you own a domain. After you prove that you own the domain, you get a bucket name that matches the name of the domain. In the case of a classroom like this, we want a unique bucket name. But we're not going to go through the bother of proving our identity, etc. We'll just try to come up with a bucket name that happens to be unique.

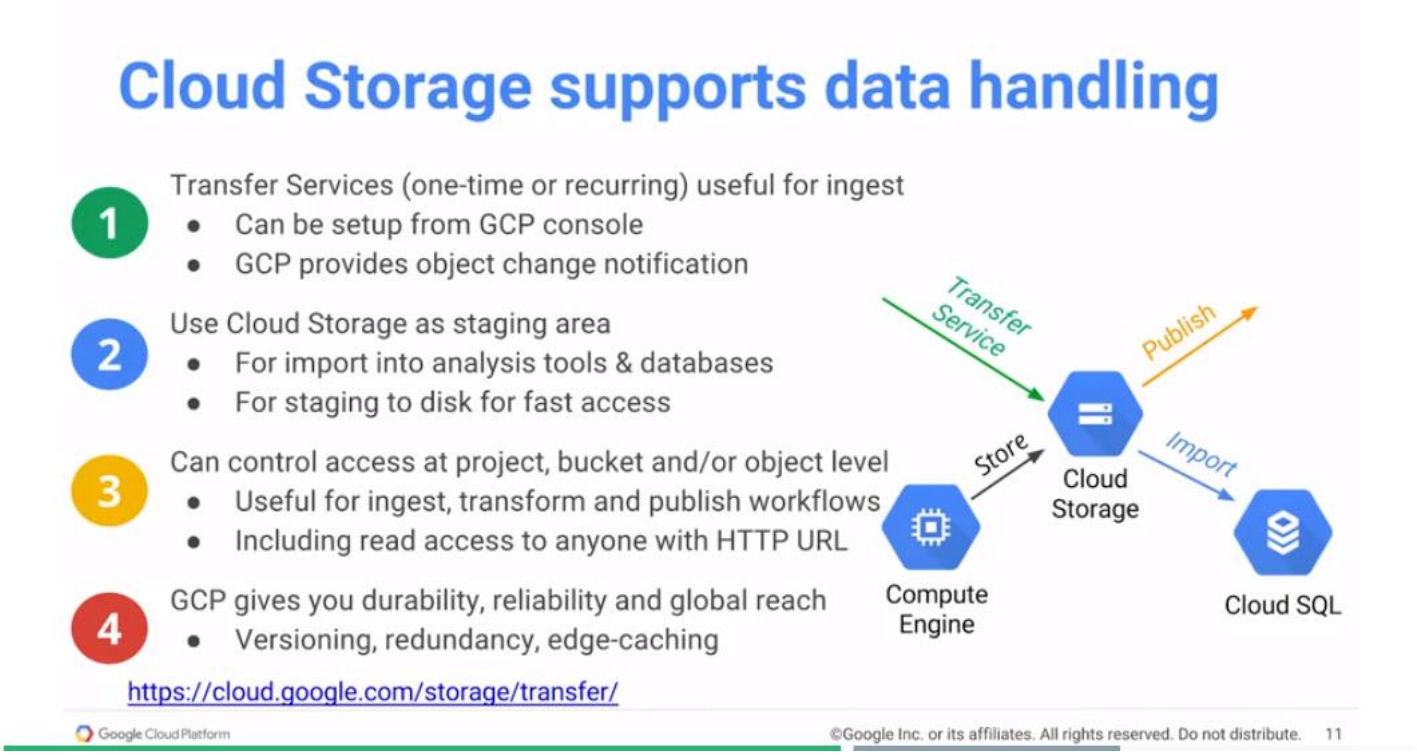
You can basically copy files, gsutil cp. Besides that, you can also do remove, rm. mv is move, so that is essentially copy it and then delete it locally. You can do ls to list, so you can do listing of things on the cloud. The thing to realize is that this URL here, gs://acme-sales/data/ even though I explained it as like a folder structure, that's purely convenience, right. All that it really is, here's a name to blob. And this name is just a string. However, people tend to think of file systems as being hierarchical, and so you might tend to think of GCS as also being hierarchical. And that's kind of where ls comes in. You can basically go ahead and look at the hierarchy. But remember that this hierarchy is additional semantics that your placing on top of a pure key value store. You can also do mb to make a bucket, rb to remove a bucket, rsync which is a Unix utility, so this is an emulation of that Unix utility. Where you can basically have a mirror of something that's local up on the cloud, and then whenever you run rsync again, it's just going to look at the files that have changed and upload only those files. ACL is an access control list, that's the way you change permissions.

Even though I'm talking about using gsutil as a command line tool, you don't have to use it as a command line. Because all that that command line tool actually does is that it invokes a web service, a REST API. So you can make that exact same REST API call yourself, so that's one way to do it.

The other way is that you could go to web console, the same way that we've created the compute engine. We could basically go to the, instead of going to the compute engine part of the user interface, we can go to the storage part of the user interface and you could use that. But because it's a REST API you can also use Python or you can use Java or you can use your favorite language, any language that can talk HTTP, which is pretty much any language. You can basically interact with cloud storage using the REST API.

And incidentally this is a very common theme.

Things that you can do in the command line you can do with the REST API, and because you can do it with the REST API, you can also do them from any language that you want. And anything that you do from a graphical user interface, also uses the same REST API. In the previous lab, we created a compute engine and we created it by using the web user interface. But we could have also done it using a command line, and the way you do it with a command line is that you would have said G Cloud instants create, right, and that would have allowed you to create a computer Ged instance.



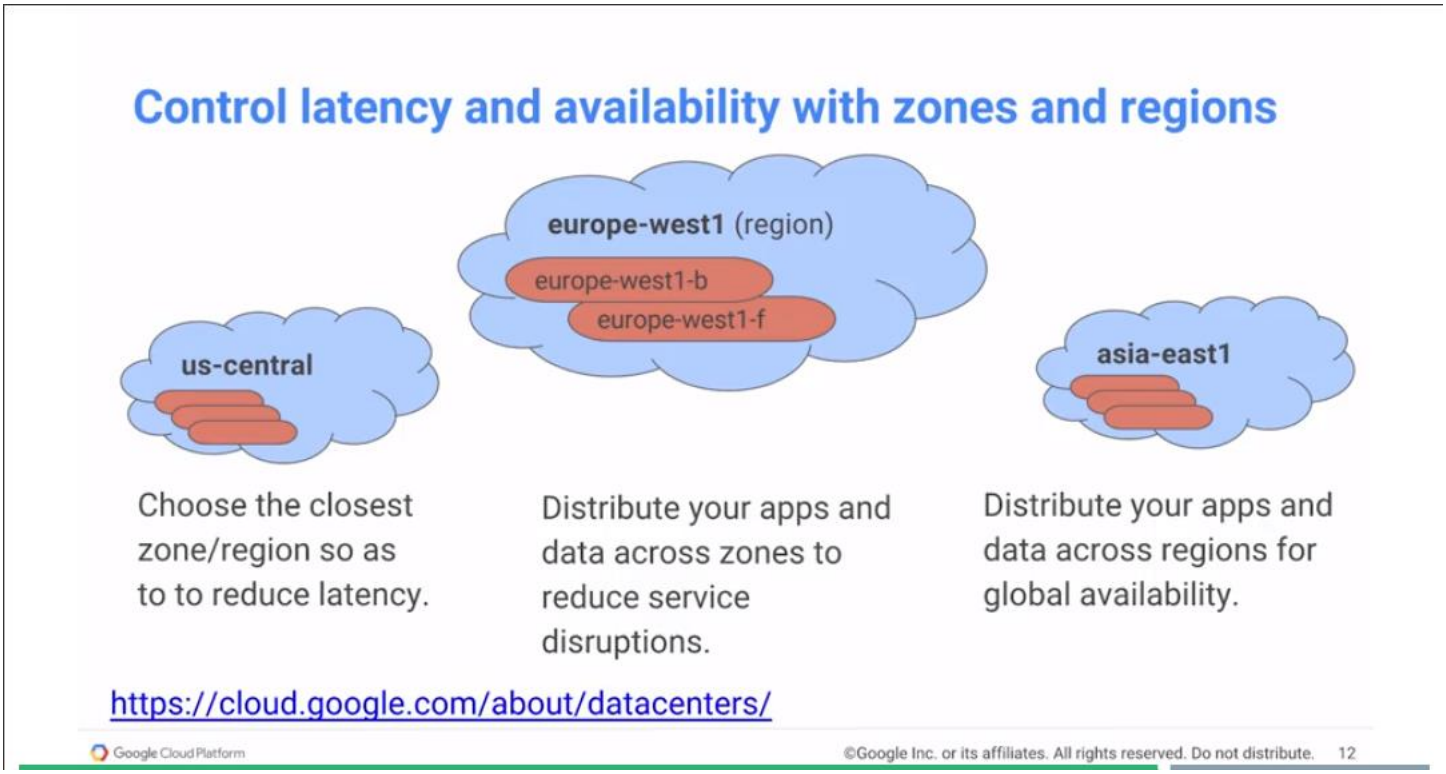
You can also set up at transfer service. The transfer service could be one time or could be recurring. Take all the data from here and transfer it over, and as new data shows up I'm going to keep transferring it. And the source of your transfer could be your local machine, it could be a local data center, something that's on premise, it could also be AWS with S3 brackets, you can transfer them and you could keep this transfer service going.

Now as we mentioned, the whole idea of cloud storage is that you use it as a staging area. You can input it into Cloud SQL, into BigQuery, into Dataproc, into variety of the different analysis tools and data bases. You could also use that to take the data from cloud storage and move it to a local SSD disk on a compute engine VM, so that if you're going to be reading some data routinely all the time, you might want to move it from cloud storage into local.

Once you have your data on cloud storage you can control access to it at that object level.

But lots of times you will have objects that are related and they will be in the same “folder” structure or in a bucket level. You can control access at that bucket level but every bucket belongs to a project, and a project is essentially the way you do billing, etc, in GCP. So essentially, when you create a bucket in a project, you are basically saying, which billing account is going to be responsible for paying for the storage, right.

You can control access at that project level. You can say people who are editors on this project can also add and remove files from this particular bucket. But crucially one of the access control things that you can do is that you can actually make access control to all authenticated users. In other words, anybody who's logged in with a Google account, you can provide access to them.



The other way that you could do it is you could provide access to all users. By providing access to all users, what you're essentially saying is that they don't even need to be logged in, they can just come and get the data that you have. And they could get that data using just an HTTP URL. Why would you want to do that? Well, remember what I talked to you about GCP. I said that once you put something on cloud storage, that thing is durable, it's persistent, it's also edge cached, it has multiple copies made of it. In other words, this is an easy way to get a content delivery network going for your data. Just take your data, your static data, put it on GCS, and give people the URL to that GCS location. And Google Cloud takes care of doing the edge-caching, and replication, and reliability, and durability, and all of those kinds of things.

If you're going to be putting a data on the cloud, even if you are creating instances, computing the instances to process data on the cloud, you should have a good idea about what zone, what region, you want to be doing the event. What is a zone? What is a region? Well, it's a geographic construct. You can think of a zone as like a data center. You want to choose the closest zone region to where all of your users are. And the reason you do that is that you want to reduce latency. If almost all of your users are in the central US, you want to use it, you want to choose the data center in Iowa, and you would use us-central to close the zone, to close the region just to reduce our latency.

But the problem with having everything in one zone is that what happens if that zone goes down, okay? If a tornado hits Iowa and the data center in Iowa is not able to be accessed, your application cannot be accessed either. In order to limit service disruptions, you might want to have multiple zones within the same region. For example, you might run your application in both zone b and zone f with zone f acting as a back up to zone b, for example. Distributing your absent data across zones is a way to reduce service disruptions.

This is good if all of your users are in Europe, or if all of your users are in the US. You would use us-central if they're all in the US, you will use europe-west if they're all in the EU.

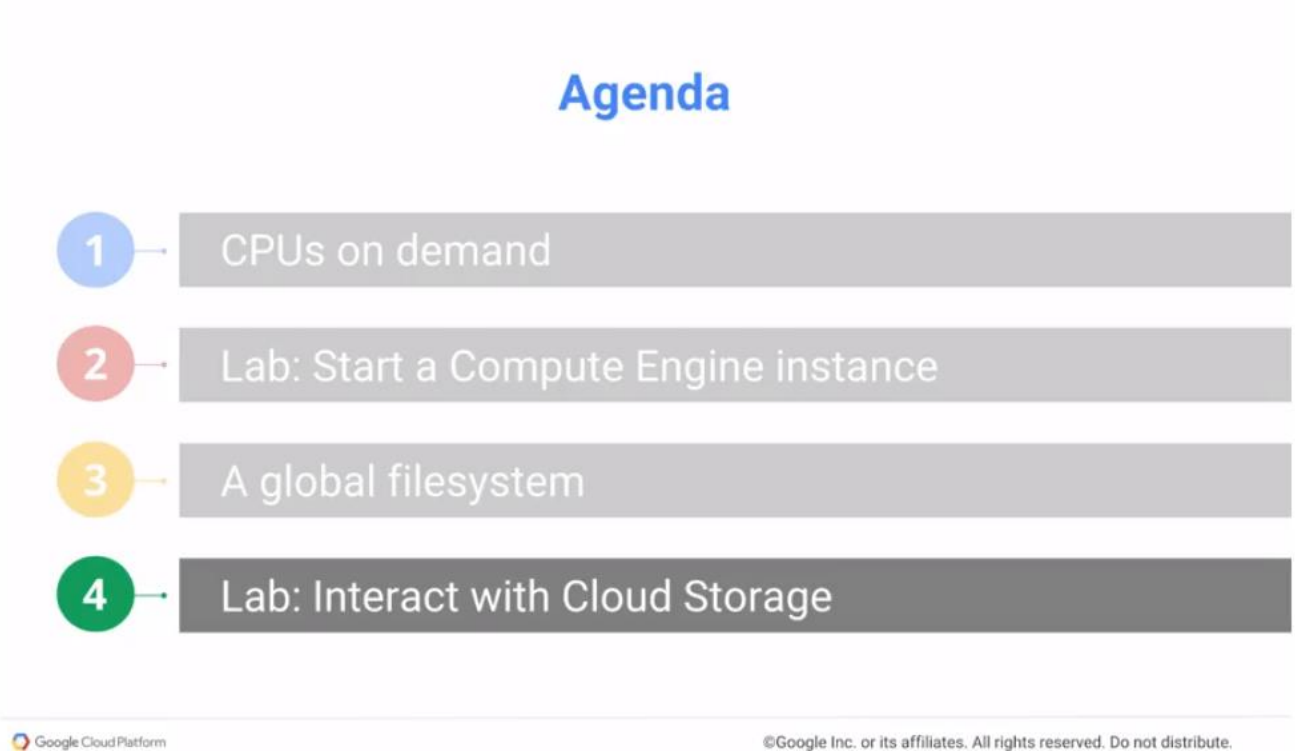
But what if you have a global application?

You have an application, we have some users in Japan, and some users in Europe, and some users in the US. If that's the case, then you need to distribute your apps and data not just within a region but across regions. And the reason you do that is to basically make your applications globally available. Bottom line, control your latency by choosing the closest zone or region.

Use multiple zones in a region to minimize the impact of service disruptions.

And use multiple regions to provide global access to your application.

Interact with Cloud Storage



Let's go ahead and try out Cloud Storage. What we're going to do in this lab is a continuation of the previous one where we created a compute engine, so we're now going to go ahead and use that Compute Engine instance. So what we'll do is that we'll ingest earthquake data into this Compute Engine instance.

And then we will take that earthquake data which is just going to be raw CSB files and we'll transform it into a map of earthquake activity. And we'll take the transformed data along with some webpage information and we'll store it onto Cloud Storage.

And having put it on Cloud Storage, we'll flip a bit that will make it publicly accessible.

And having done that, we'll go ahead and access it publicly just to see how they work.

Go ahead and do this lab, and then come back and join me.

https://codelabs.developers.google.com/codelabs/cpb100-cloud-storage/

Review of lab - https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/BfKIr/lab-2b-review

--

Module 2 Review: https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/swQns/module-2-review

Resources

Compute Engine: https://cloud.google.com/compute/

Storage: https://www.google.com/storage/

Pricing: https://cloud.google.com/pricing/

Cloud Launcher: https://cloud.google.com/launcher/

Pricing Philosophy: https://cloud.google.com/pricing/philosophy/

Calculator

And let's say that I have a workload that requires 15 instances for doing something that's a Linux thing, it's a regular VM. And the VM type is going to be in one standard for and it's going to use four SSDs and it's going to be running. I'll put that again. It's going to be running three hours a day for three days a week, and plan this is hard to estimate.

And this is going to cost us $381 a month at the time that it will be recording again. Now go back and check the pricing, pricing keeps changing. Usually, keeps dropping but just keep changing. And let say in addition to that we want on cloud storage, we want to store within a region and we want to store, let's say we want to store 100 terabytes of data. And we can add that to the estimate and that's basically what that's going to cost us.

In addition to the pricing, the other thing that I'm going to talk about here is something called the Cloud Launcher.

Remember that we decided that we're going to create a compute engine VM and be able to install software on it. Well, lots of times, the software that you want to install on a machine has already been installed by someone else. Let's say, for example, that we want to create Cloud launcher that has WordPress.

Okay, there it is, right? So here is a WordPress, click to Deploy. And we can say, I want to basically launch WordPress on Compute Engine. And that's basically going to give us single VM with WordPress already installed on it, and this is basically typically done by partners of Google that basically take a compute into VM and provide what's called a deployment manager script. This is just a configuration file, so you can use this to even customize you own virtual machines. And that's basically what my deployment script is. What's being used for the Cloud Launcher can also be extremely helpful if you're doing your own.

MODULE 3: DATA ANALYSIS ON THE CLOUD

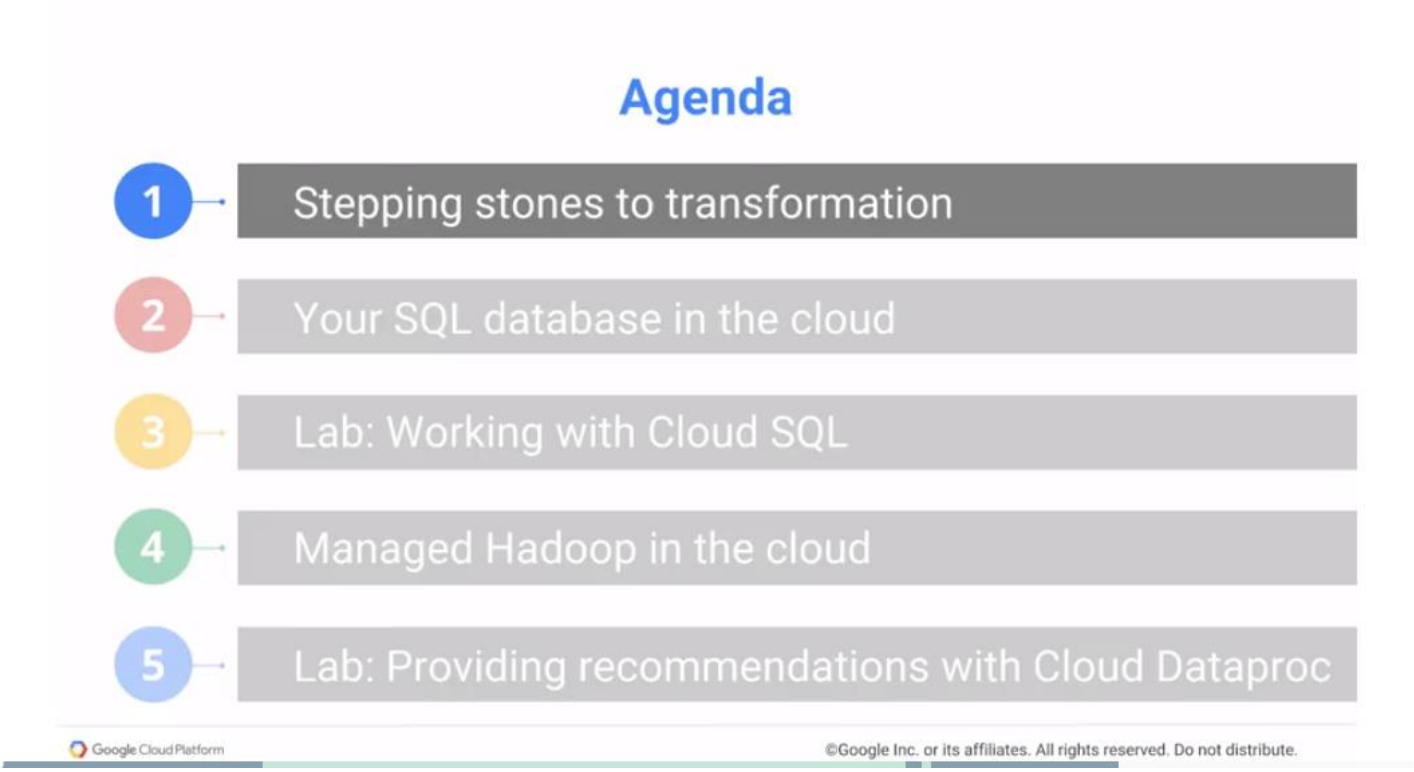
In this module we introduce the common Big Data use cases that Google will manage for you. These are the things that are widely done in industry today and for which we provide easy migration to the cloud.

Learning Objectives

Use CloudSQL and Cloud Dataproc to migrate existing MySQL and Hadoop/Pig/Spark/Hive workloads to Google Cloud Platform.

In this module, we'll look at how to do data analysis on the Cloud using tools that you're probably familiar with, using a relational database and using the big data tools that are part of the Hadoop ecosystem. What are we looking at is how you can take existing programs that work with the relational database, in particular the MySQL database, and process them with Hadoop. In particular, a Spark program and migrate it so that you can run those programs on Google Cloud. Essentially what Google Cloud provides you in those cases are managed services. So, we will look at managed services for common use cases.

Stepping Stones to Transformation

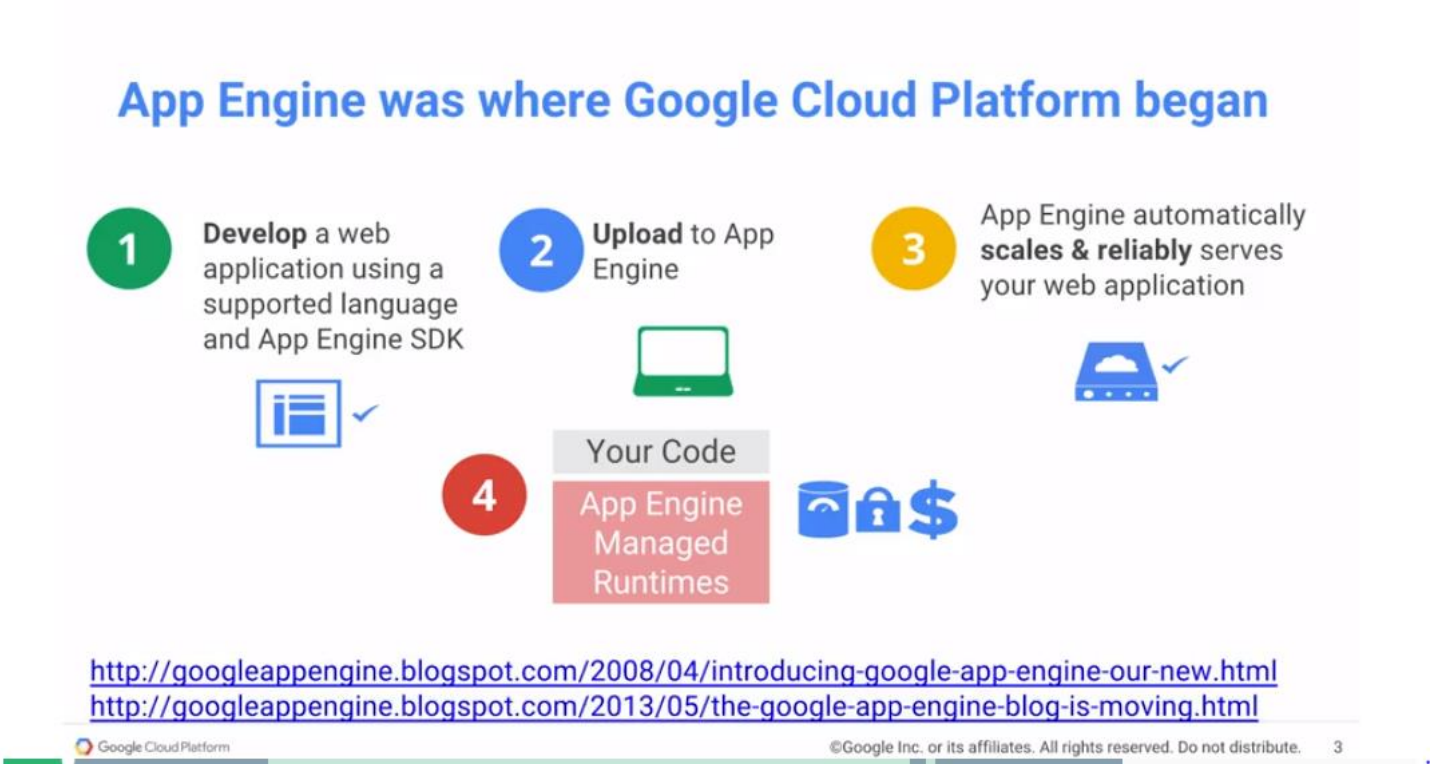


Why are we talking about this in the first place? Why are we talking about migration to the Cloud and products that help you do migration? To do that, it's probably good to look at this, a little, from a historical perspective. The first product, the first cloud product that Google had was App Engine. And the way App Engine worked was, you would develop a web application, write it in Java, upload this application to App Engine, and that was it. App Engine would, essentially, scale your code and server to users. If you have 20 users it may run it on one machine. And as the number of users ramped up, maybe at lunchtime you had 200 users, it'd automatically scale out to three machines. And then at one point your app goes viral, you have millions of users, no problem. App Engines just scales. So in 2008, Google was already doing a server-less, fully managed web application framework. So, this is great, but people found it very hard to get started, why?

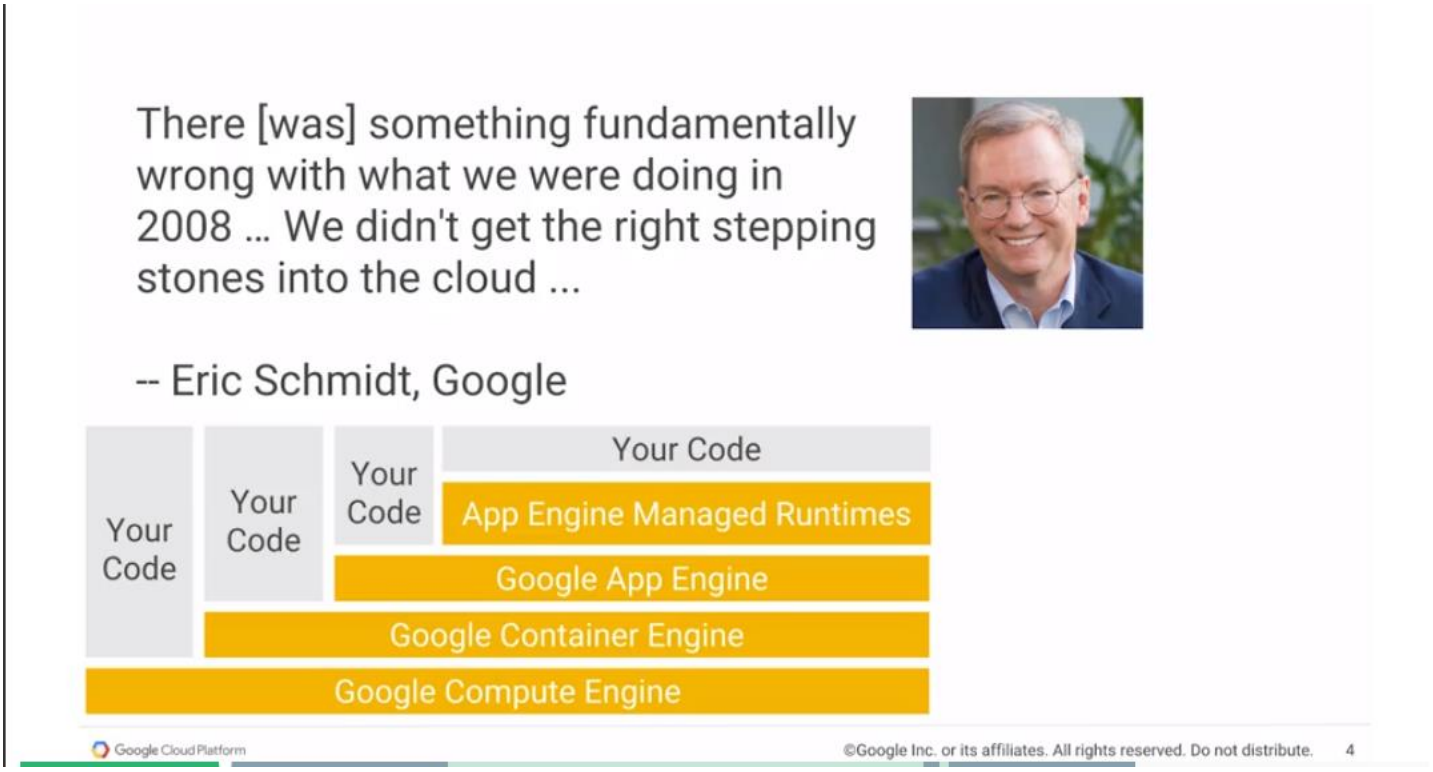
A Note from Norton: Watch it with Serverless. Serverless isn't really set up for big compute. It is really just set up for very small jobs. And watch how you code it and run it. I ran up a fairly big bill on AWS and am still not even sure how I got there. Therefore. you have to be very careful with serverless because as much as everybody touts it, I just don't think it is ready for big time jobs.

One thing I did find out about AWS Serverless (Lambda) – log, log, log! On AWS, the logs get sent up into Cloud Watch so you can track what is going on with your code.

Another thing, GCP is really restrictive on Serverless – they apparently only support JavaScript. Seems odd to me, but that is yet another reason why I believe that Serverless is ready for the big leagues yet.



Well, because we required people to write their code in Java and that was it. We required that you use the App Engine framework. And people said, well, no, I have web applications already, I write them in Tomcat, what do you mean I cant' run Tomcat in App Engine? Or people would say, well I would rather write things in PHP, thank you very much, and App Engine was a no-go at that point.

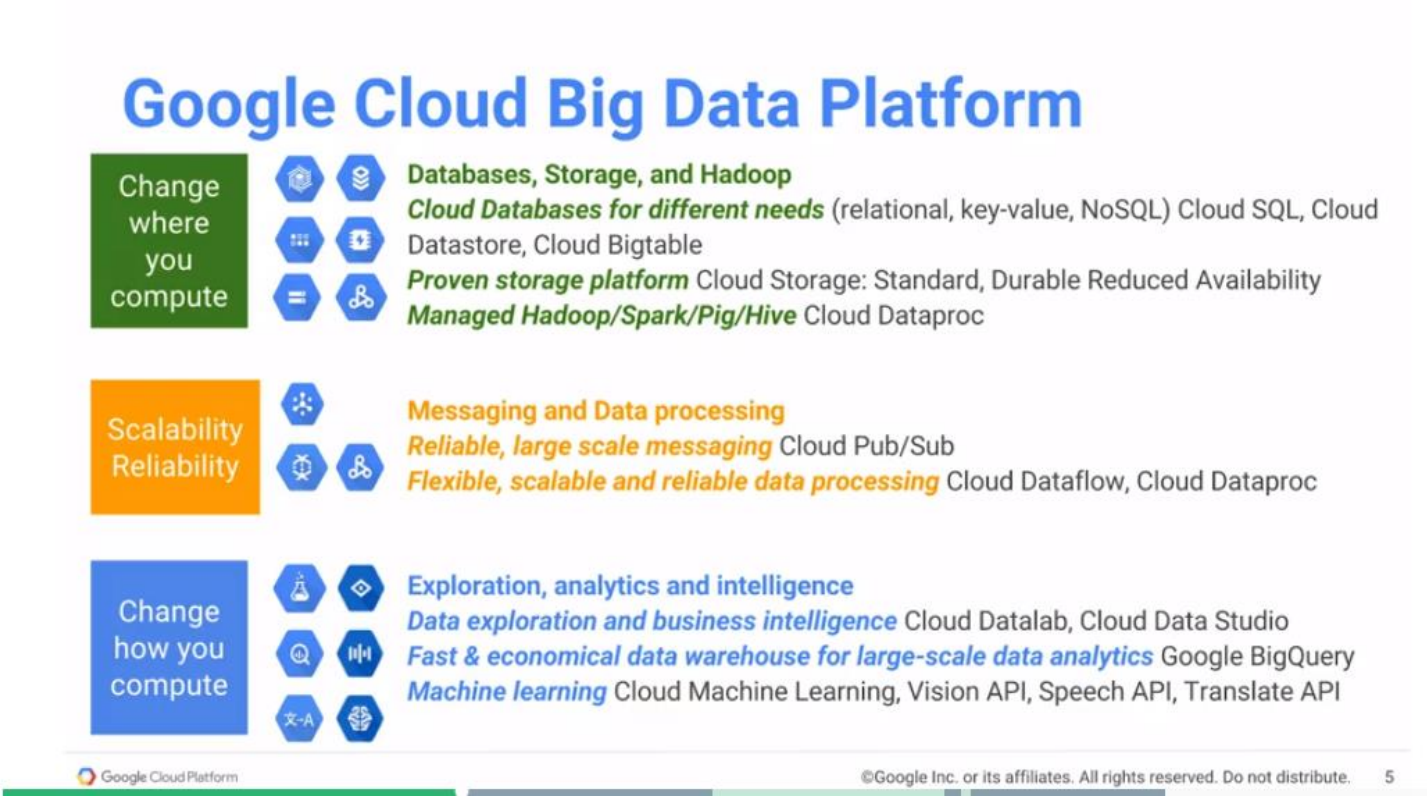


It took us a while to basically recognize this issue. That while a Google engineer was perfectly happy to give up control and say well take care of this for me. Here's my code, just run it, scale it, deploy it, manage it. I'm quite happy to just write the code and have the service take care of all of the auto scaling and reliability considerations. This was not where the rest of the industry was. And so, this is what Eric Smith, who's the executive chairman of the board at Google. This is what he was referring to, when he said that there was something fundamentally wrong with what we were doing in 2008. He's talking about App Engine here. And what was wrong? What was wrong was that we didn't get the right stepping stones into the cloud. We weren't meeting our customers where they were. We were, instead, giving them a solution and saying look, this is what we use, come use it too. And, that works for people who have greenfield development needs. They don't have any legacy systems. They don't have systems that already work. They just want to have them continue working. That's good for start-up companies, starting new. So, we do have very large customers like, for example, Snapchat uses App Engine. It's one of the largest web traffic companies in the world and they essentially run completely on App Engine. And that allows them to basically get all kinds of scaling. Any issues just call Google, Google fixes it, that's great.

However, if you're not a new company, doing greenfield development, there was no way to get onto App Engine. There was no nice, easy ramp up. And so, that's what Eric Schmidt was talking about when he said that we are doing something fundamentally wrong in 2008 that we need to fix. And we're fixing it now. So, the idea is that initially, we would just have your code, Java code. It would basically run in App Engine, and that was all you needed. And App Engine would provide all of the infrastructure. But now, we basically give you Google App Engine, which lets you run things that are not Java. So, you can run in a flex environment, you can run Python Flask apps, you can use other web frameworks. It's still your code, it's still going to get auto scaled. It's still going to take advantage of all of the capabilities that App Engine provides. But you get a lot more flexibility in terms of, taking code that you already have, and running it in App Engine. But let's say you don't have a web application, you actually want to take the application that you have and not run it in App Engine. You want to run it as is in, Tomcat. Well, containerize it and put it into a dock or container. And we will basically orchestrate those containers and manage them for you with Google Container Engine.

You don't want to even containerize it. It's running on bare metal, on-premise, and you'd like to move it to the cloud, and have it continue running in bare metal? No problem, we'll give you Compute Engine. The idea behind Compute Engine is to run your workload as is in the cloud. These are all of the stepping stones to take something, like a web application. Ideally, we hope that over time, you will not have need to run all of this core infrastructure yourself. And instead allow Google Cloud to manage those infrastructure needs for you, so that you can concentrate on your business needs. But, we give you the stepping stones so that, over time, you can do this migration on your own schedule, rather than it being all or nothing.

Norton note: For AWS geeks – Google App engine is a different application than Google functions (which is the google serverless). It is designed for highly scalable mobile and web applications (https://cloud.google.com/appengine/docs/) and has a separate pricing model (https://cloud.google.com/appengine/pricing).



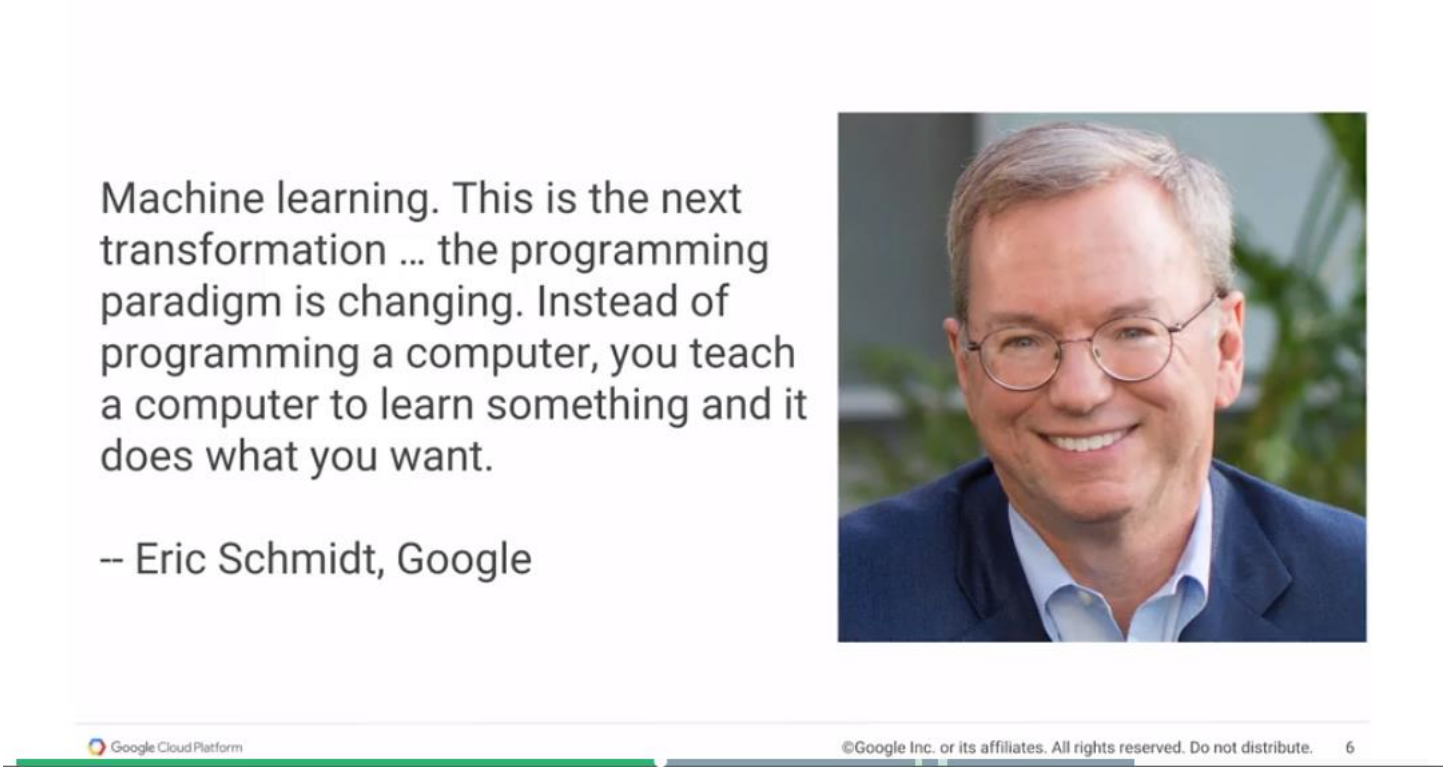
What is true of App Engine is also true of everything else. When we look at the Google Cloud, there are all of these hexagons. In many cases, these are different entry points into the cloud. If you're running a certain type of database on-premise and you want to move those workloads to the cloud, we want to give you an opportunity to do that. And that's essentially what many of these hexagons are about. When you are looking at running things on Google Cloud, part of it might be simply taking the same workload that you have, and simply changing where you do the computation. The reason that we are not going to change the where is because Cloud can be cheaper. Cloud can be a little more secure, and that's basically what you want.

But on the other hand, there could be other instances where the reason that you're moving to the cloud is because you want additional scale and additional reliability.

Maybe you want very reliable large-scale messaging and you're very interested then in Cloud Pub/Sub, which is a serverless. You don't have to actually even launch a server to get a message system going. A messaging service that you can, basically, publish to and subscribe to without actually having a cluster of POP subservers running, for example.

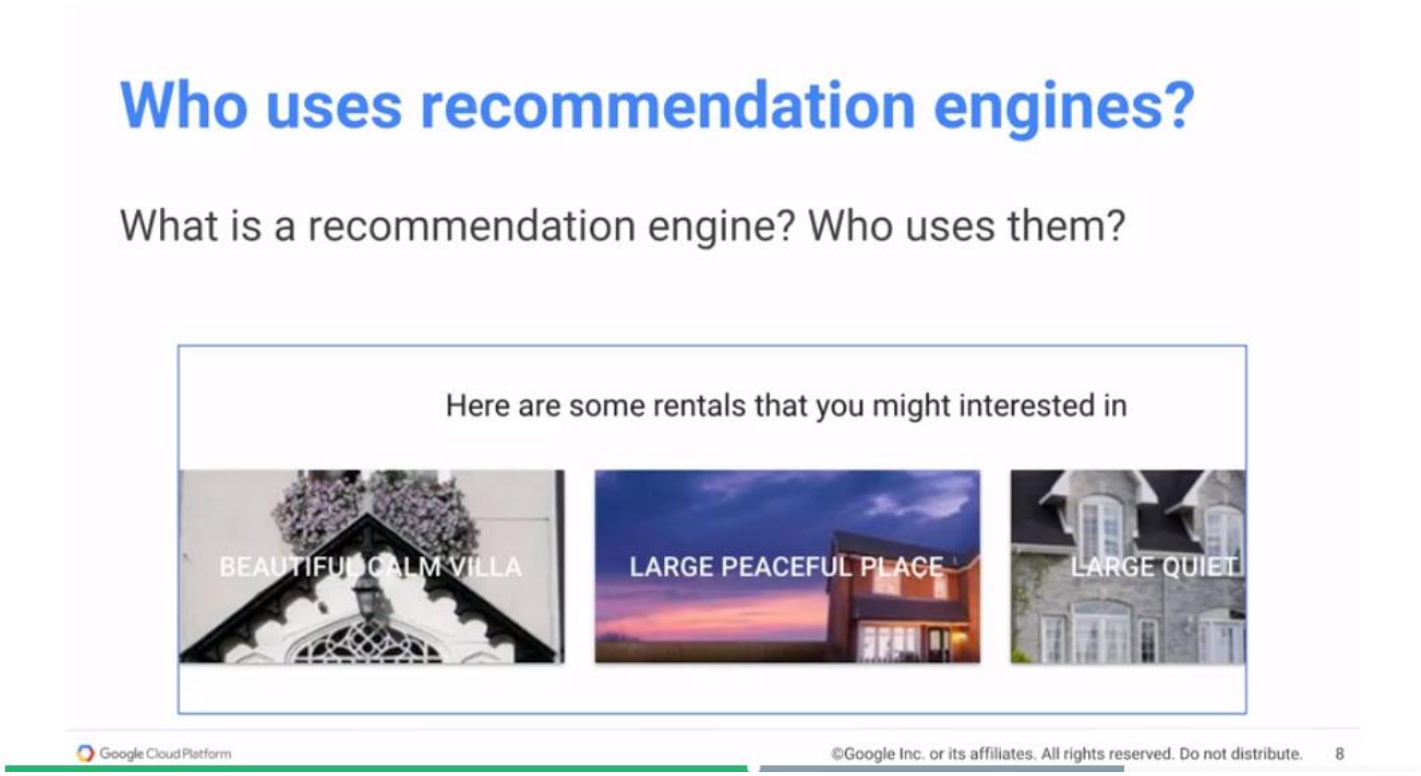
Similarly, you have needs around scalable, reliable data processing, data flow - Dataproc. Those are all different ways to do that. But the idea is that these are all stepping stones to the transformation the cloud can provide. And that’s where we get to the changing how you do your computation. In terms of data expiration, in terms of business intelligence, in terms of machine learning, and all of those kinds of things. So, for example, for data warehousing, we believe that Google BigQuery is probably the best solution for a whole host of companies. But you may not be ready to move into a so completely serverless, fully managed data warehouse solution yet. And so, a stepping stone to that could be, take the workloads that you have. And run them on Cloud Dataproc. And over time, maybe migrate some of your data sets, some of your workloads into BigQuery so you can forget about the infrastructure management that you're probably doing today. The idea behind this module is to talk about those migrations.

The other thing that we want to talk about, and this is something that's throughout this course, is this transformation that's happening in our industry around machine learning. Eric Schmidt again, the Executive Chairman of the Board at Google on machine learning - machine learning is the next transformation.



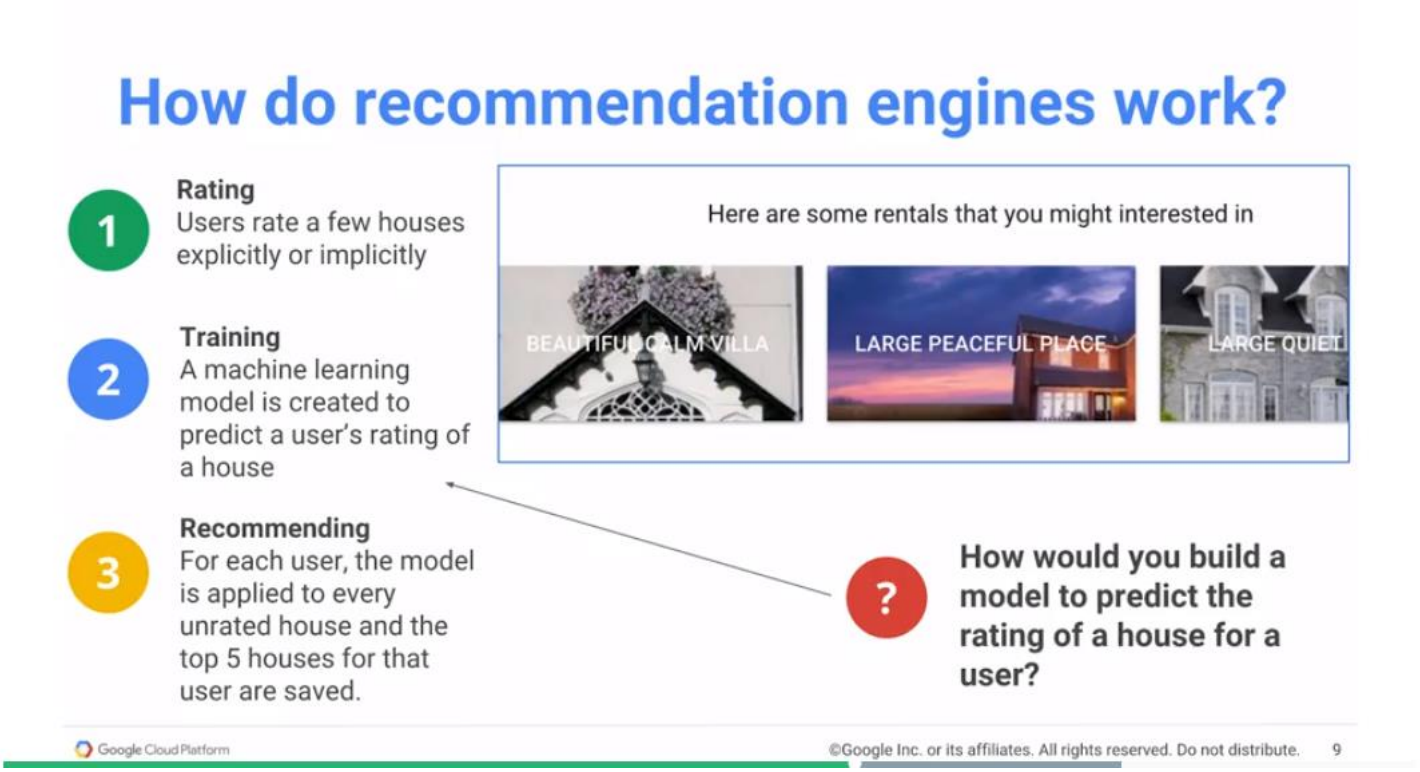
It's the new thing that we're going to be doing. And what's new about it? Well, the thing that you're changing is a programming paradigm. Instead of you programming a computer, Eric says, you're going to teach a computer to do something. And it's going to do what you want.

Notice here that the way Eric's talking about machine learning is that he's not talking about it from the point of view of data. We know that machine learning is about learning from data. Data is very important, but Eric is actually having us think a little bit more about machine learning and saying this is about logic. It's about replacing the way you do things and, of course, wired saw this and the headline is, soon we won't program computers, we'll train them like dogs.



That gets to the point that Eric Schmidt's also making. The way that dogs learn is not in terms of rules saying do this, do this, do this, if this do that. But instead by being exposed to a number of scenarios and figuring out what's correct and what's incorrect behavior in those scenarios. And over time, the dog learns, and the idea is that over time our computer programs can also learn. Now, machine learning has been around for a very long time. Neural networks have been around since the 70s, for example. But, I think machine learning basically came into people's consciousness, only with recommendation engines. This is where you go to an e-commerce site and then you basically see a message that says, people who bought this also bought this. Or you may have gone to a music service and the music service is basically giving you a playing list of songs that you might like: a daily song list that's personally curated for you, except that it cannot have been really personally curated by a person, because that doesn't scale. Instead, it's that computer algorithm that's somehow figuring out the kind of songs that you would like and putting together a playlist for you. Or movies, where you basically get movie recommendations. What is a recommendation engine? It's all of these kinds of things where it says, you would like this product because you've liked products like this in the past. Who uses them? E-commerce sites, movie sites, etc.

The example that we're going to pick up is that we're going to try to basically talk about rental houses and say as our example that we want to be able to recommend to people. Here are some houses for you to go look at. How would they work? How do recommendation agents work?



The way it works often is that you start with ratings. Either explicitly or implicitly, users go through a catalog and over time they've rated some houses. You could rate a house by actually going and visiting the house. A user comes back and says that was not a house they liked. I'm going to give that a two or a five. And they give you ratings over time; now you have ratings from that user for a few houses and you can use that as part of your training dataset. But you could also have implicit ratings, and implicit ratings could be the user clicked on a link for that house. That basically indicates that they are interested in the house. Or if you have a mobile app, you can actually look at how long somebody looked at the house before they scrolled past it - that could be part of the rating. You could have explicit ratings in the form of stars or implicit ratings in the form of links or time that you've looked at it, etc. The thing to realize is, how many of the products actually get rated? Let's say you have a catalog of 10,000 houses. How many of those houses per individual user have rated? Maybe five, maybe ten. That's pretty much it.

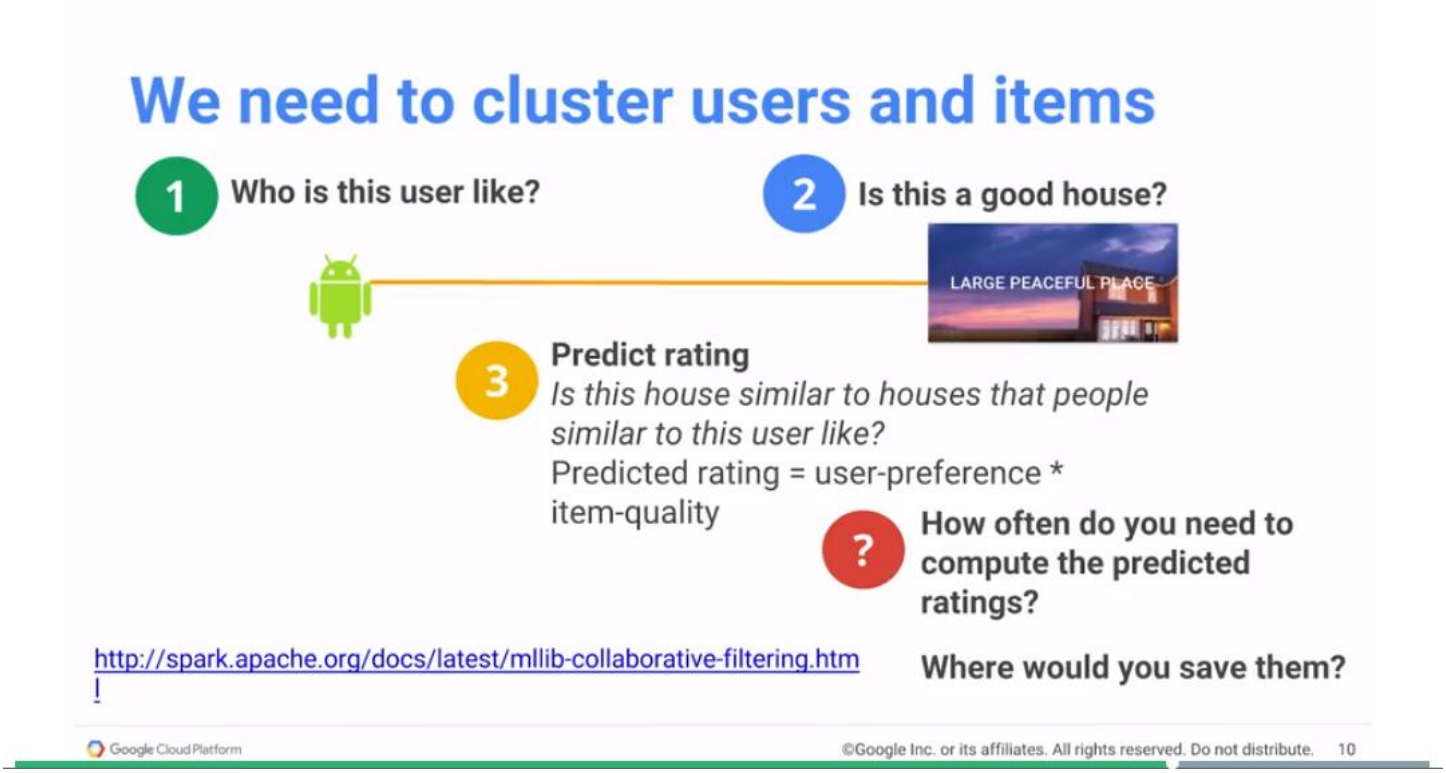
Suppose every user has rated 5 houses. If we have a million users, and we have 100,000 objects, we essentially have a matrix that's got 1 million rows and 100,000 columns. And that matrix is extremely sparse because every user has rated only maybe 5 or 10 of those items and the remaining 995 objects are unrated. But we need to come up with the rating for each of them. So that's basically what the ML model needs to do. It needs to predict what a user would rate a house that they haven't yet visited.

In order to do that, we need to build a ML model.

Let's suppose that we have a machine learning module that allows a user to rate a house 2.1, another house they would rate it 3.4, this house they may rate it 1.7, etc., etc. Essentially the recommended part of it is simply to go through that catalog of 100,000 houses. Take for every user, and then take the ratings, find the top five ratings and basically suggest those to them. That's essentially all they're recommending. Their recommending is a very cheap operation, it's just a sort find the top five, return the top five. The interesting thing is how you build the ML model. So how would you build this model to predict the rating of a house for a user?

Intuitively, we kind of know how this ought to work.

We will basically say that if we have two users that happen to have rated house A, the same, because they happened to have visited and the first user rates it a four, second user also rates it a four. We could kind of say intuitively that let's go to the first user, find out all the other houses that they rated, and say these two guys now if they rated this house both a four, maybe they will share the rating for everything else. In other words, you could basically say who is this user like. Go through all of those users for those houses and say, has this user rated the same house as somebody else, and let's kind of propagate that rating that way. Remember the whole idea is to fill out the metrics of every user, let's say a million users, and every item, let's say 100,000 items - this 1 million by 100,000 matrix needs to get filled out. It could get filled out starting by looking at whether two rows, which each row corresponds to a user, are not similar to each other. The other way that you can do this is to look at the houses themselves and seek popular houses that tend to be alike. Unpopular houses will tend not to be alike. You might have a house that everyone who's looked at has agreed that it's a dump. They're basically rating it one or two out of five.



You could infer that everyone is going to rate it at one or two. It's a dump. The house is a bad house, everyone's going to rate it badly. In the absence of other information about a particular user, and the kinds of things that they like, go with the majority vote, and that's the second way that you do it. Essentially we need to cluster users, we need to cluster items, and combine the two of them to produce a rating.

So how often do we need to compute this predictive rating? Remember that computing the predicted rating is filling out that huge matrix. It takes a long time. It may take hours. But when do you need to recompute the rating?

You will need to recompute the rating whenever the user comes in and rates a new house, or something else happens. Maybe some other user rates a house that causes your rating to change. Now this is not a very common occurrence. It's not something that you may have to do up to the minute. This is the kind of thing that

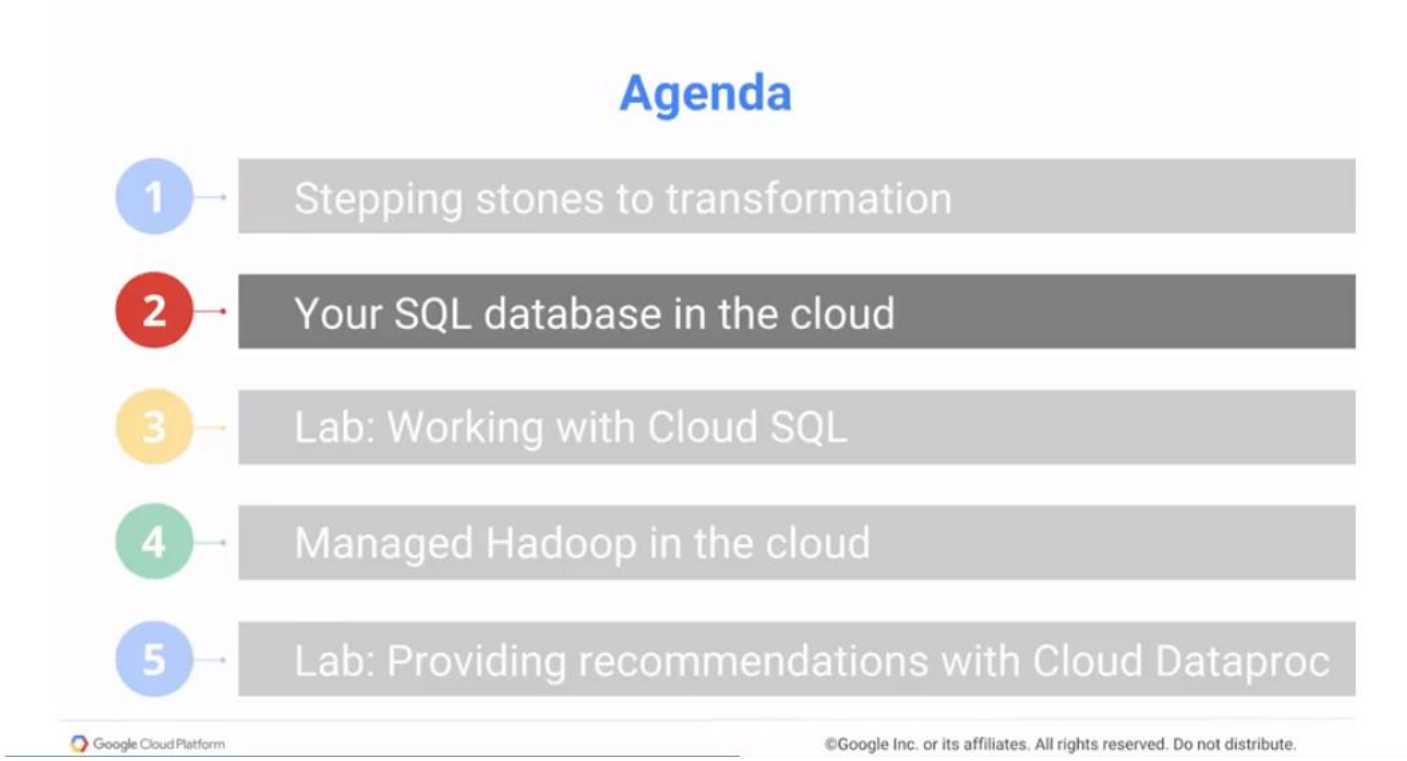
many people do as a batch job. They may say well, I'm going to do this once a week. Based on all the ratings that I have, I'm going to create my new recommendation model. And that is the recommendation model that I'm going to basically use for doing product recommendations.

Because this is the batch job that runs occasionally, a good place to run it would be Hadoop. We will look at running this using PySpark, that's Spark in Python, on a Hadoop cluster. On GCP, the Hadoop cluster is called Dataproc. We are going to use the Dataproc cluster to run a Python Spark job. And that's how we're going to compute the predicted rating.

The second part of the question is where do you see the results? And if you're going to talk about where to save it, let's think about what to save. What are we saving?

What we are saving are the top five houses for every user. If you have a million users, you have five houses per user, that's 5 million houses. This is small data. This is the kind of thing that you could store in a relational database. It's transactional, because every day you may want to keep updating these recommendations. You may want to update the database in the context of a transaction for example - therefore a relation database makes perfect sense. The second part of this use case is to store our results in a MySQL database which in the Google Cloud is Cloud SQL. It's a managed MySQL offering. In order to solve this recommendation problem, we're going to look at MySQL on Cloud SQL, and we're going to look at Python Spark on Dataproc.

Your SQL Database in the Cloud



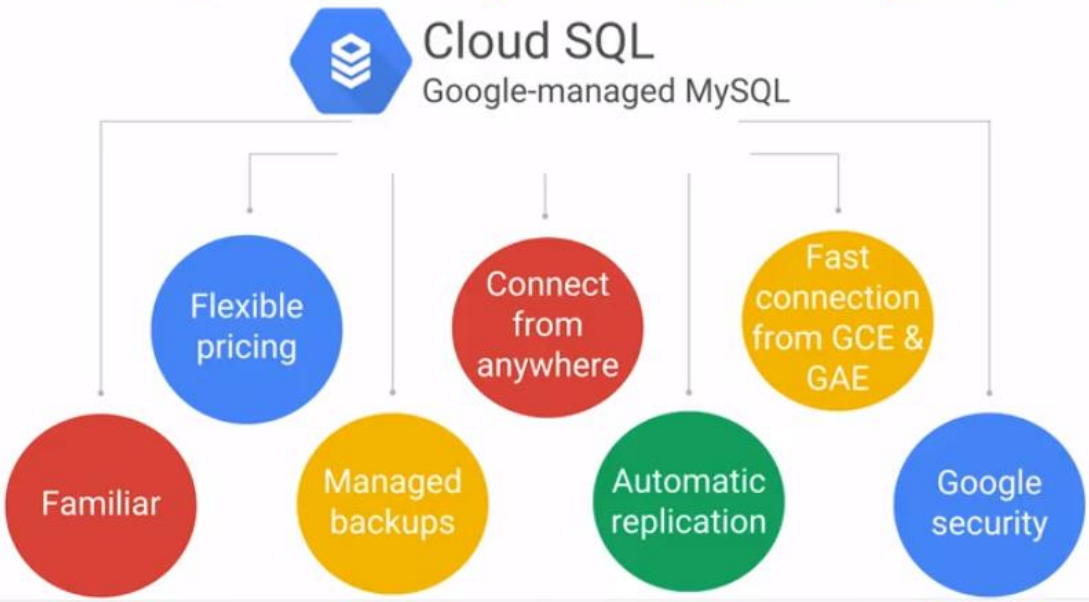
Let us start off by looking at how to do a relational database in the cloud. As we mentioned, the data that we have is relatively small, a few hundred gigabytes perhaps, maximum. So that's the kind of thing that Cloud SQL handles perfectly well – we would use Cloud SQL. The capacity of a gigabyte is fine for a relational offering. You use it like a relational database. You do selects, you do inserts. You do updates, you use deletes. And you can update an individual field. So that's what Cloud SQL is.

Choose based on access pattern

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Relational
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Later, we will look at other access patterns that you could use if you needed to go at larger scale or we want to basically deal with objects rather than relational data. What if we need to deal with high throughput? What if we need to deal with petabytes of data and still get very quick, timely responses? Right now based on our access pattern of relatively small data that we want to use in a very familiar way, Cloud SQL is great service.

Cloud SQL is a fully managed MySQL



Cloud SQL is essentially MySQL. MySQL is an open source database and Cloud SQL is Google-managed MySQL. In the last chapter we looked at Cloud Launcher. And if I wanted to run WordPress on Google, I could just go to Cloud Launcher and get a Compute Engine VM that's pre-installed with WordPress ready to go.

Why can't I do that with MySQL? Why does Google have to manage MySQL?

Isn't it just getting a Compute Engine VM and installing MySQL on it? Well, not quite. By having Google Cloud manage MySQL, we get a few extra benefits. First benefit, flexible pricing. One of the things that you could do is to say, well, I have this database and it's going to be used only between 9 and 5. So between 5 and 9 just passivate the database. Then so you don't have to pay for it when it's passivated, so that's great.

That is something that you can just have Cloud SQL do for you. You don't have to write any extra infrastructure handling to passivate the database when you're not using it. This is particularly useful for things like test & dev databases. A database that needs to be run only when a unit test is running. If a unit test isn't running, we don't need the database, we don't have to use it.

Second advantage, it's familiar. It's MySQL and so every workload that you have that uses MySQL is a prime candidate to run on Cloud SQL.

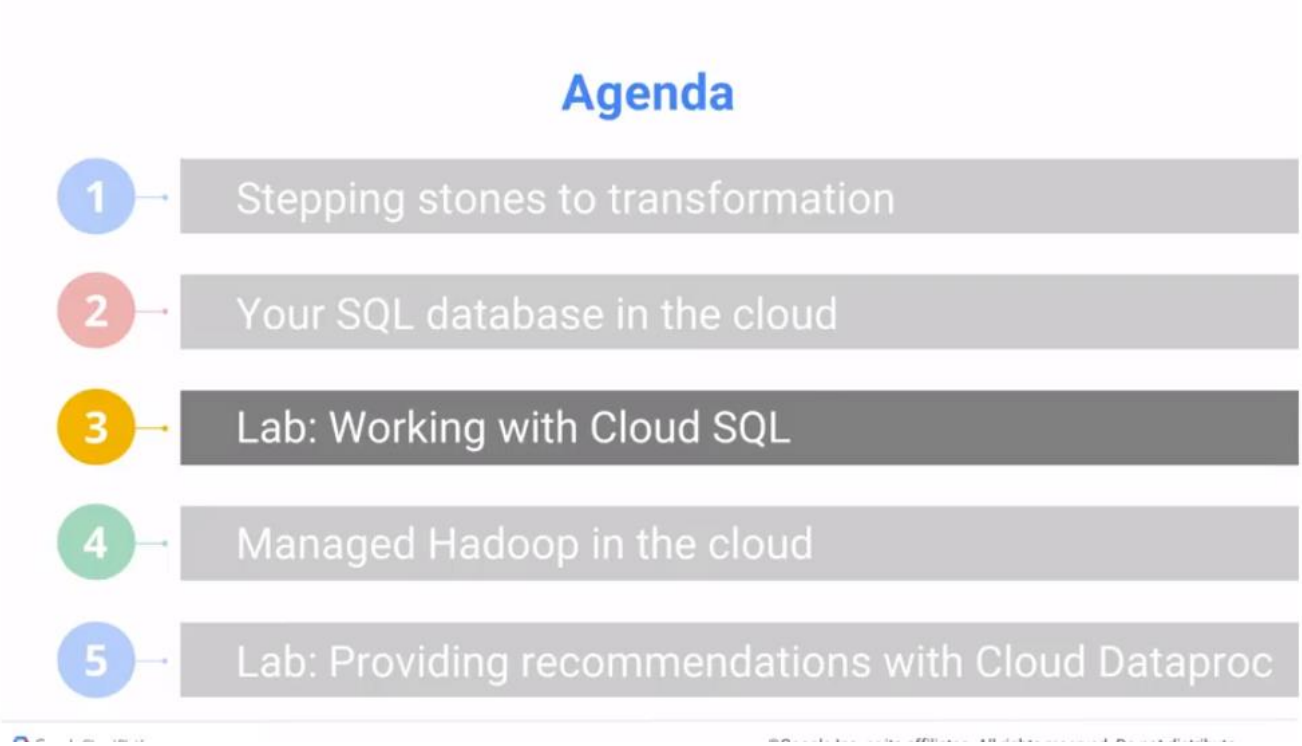
This is the advantages of running MySql as Cloud SQL on Google for relational database workloads. First of all - flexible pricing. You don't have to pay for a machine if you aren't using it. And this is particularly useful for machines that aren't used 24/7. Secondly, Google manages the backups so you can just go to Cloud SQL and establish a policy to back up this database every Thursday at 2:00 AM and it will happen. Third, you can connect to it from anywhere. It's on the Cloud so you don't have to worry about your company firewalls and things like that. You have a database that needs to be connectable from anywhere.

You get automatic replication if you need it.

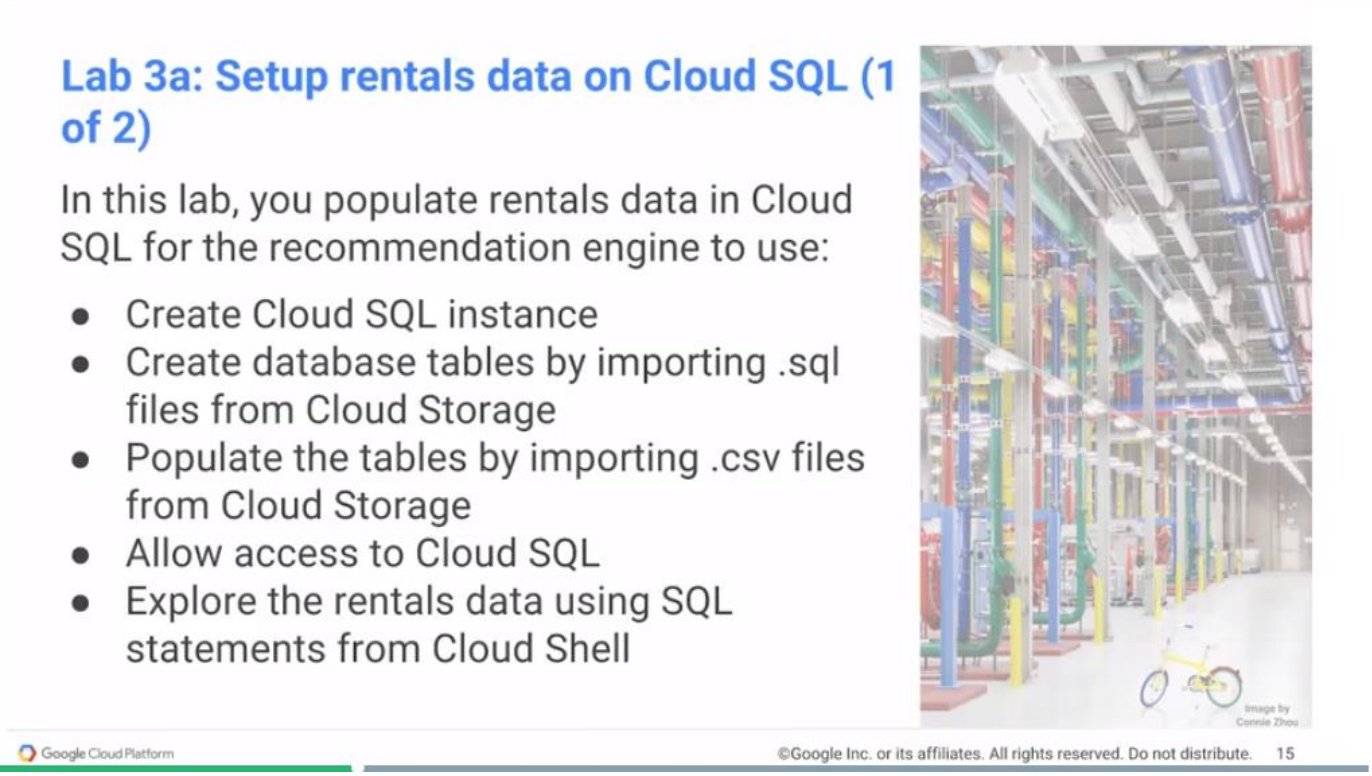
You also get very fast connections from GCE, Google Compute Engine, and GAE, Google App Engine. Why is that?

Well, because Cloud SQL is on a machine that lives in Google Cloud, and all these things are also in Google Cloud. So as long as you're running these apps in the same region, they get to take advantage of Google's petabit per second interconnect networking. Therefore, you get really fast connections between MySQL and other applications that are running on Google Cloud. Last, but not least, is this idea that you have Google Security taking care of your instances – you are not at the mercy of a randomly rotating set of security people. You basically have it being secured by people who do it as their job day in and day out.

Working with Cloud SQL



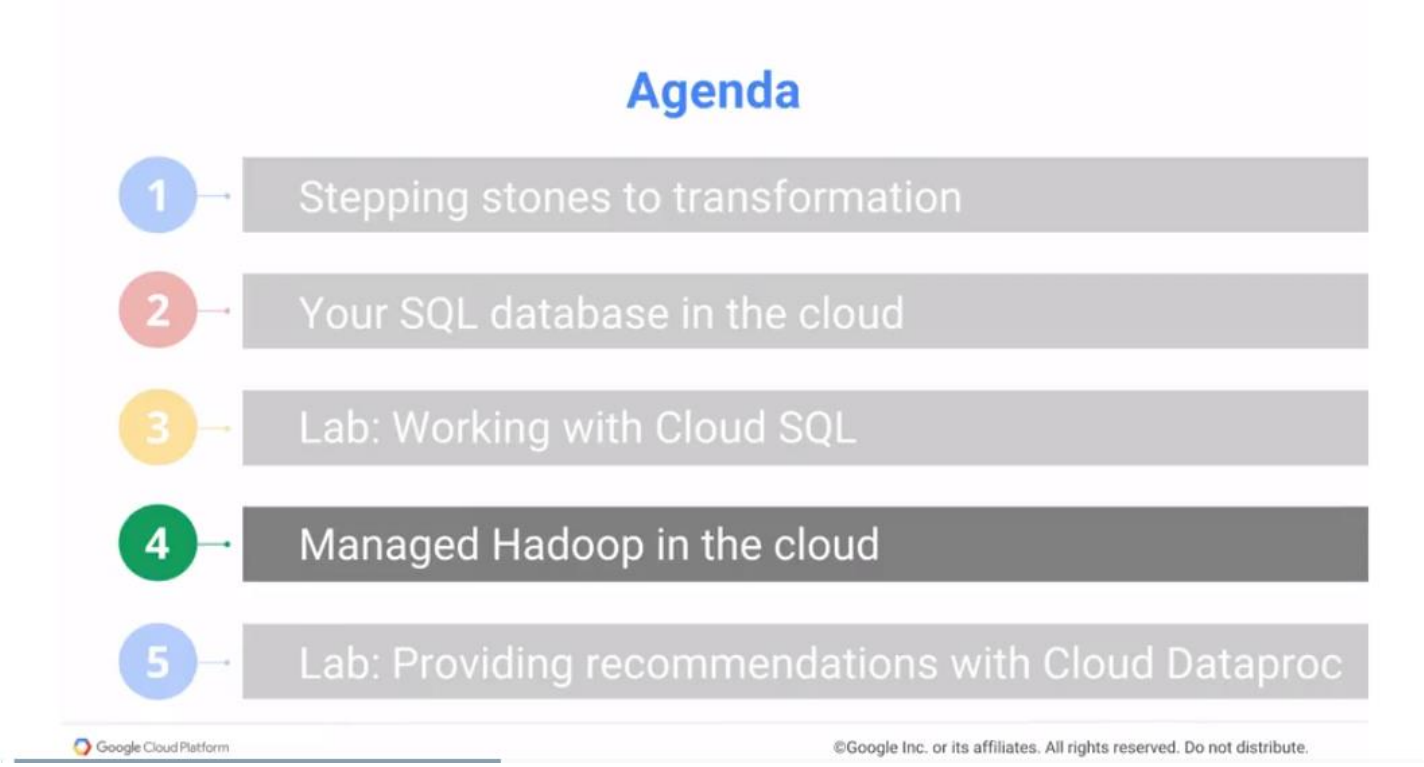
Create a Cloud SQL instance. And then populate that Cloud SQL instance with information about houses and ratings. So that in the next lab, we can use it to carry out product recommendations.



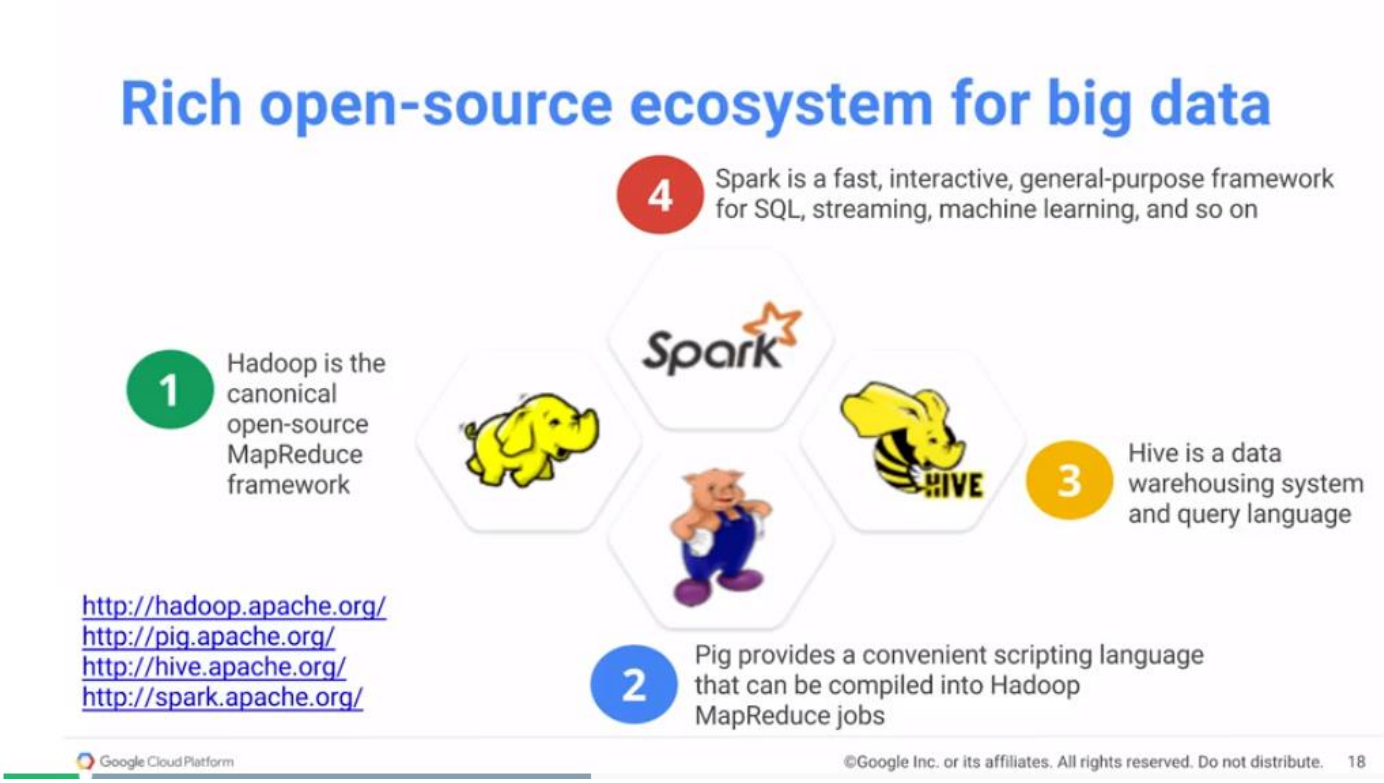
https://codelabs.developers.google.com/codelabs/cpb100-cloud-sql/

Lab 3A Review: https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/eV4ee/lab-3a-review

Managed Hadoop in the Cloud



So now that we have created a cloud SQL instance loaded the data, the next thing that we want to do is to use Hadoop - Spark in particular to create a recommendation engine. There's been a rich open source ecosystem that's come around the big data space. Hadoop, which is the Canonical open source MapReduce framework that was created by Doug Cutting after he read the papers out of Google that talked about MapReduce. The primary way in which you would use Hadoop was to write MapReduce programs in Java. This could get relatively verbose, so people looked for simpler ways to use Hadoop. And several simple solutions came about, simple, very powerful solutions came about. One was called Pig, and the idea behind Pig was that you use a scripting language. And you would write your MapReduce programs in that scripting language. And it could be much higher level, it is almost like an ETL language, extraction, transformation, loading of data language. You load data from here, do counting, operate on different datasets, do joins, and then you would basically store the data back.

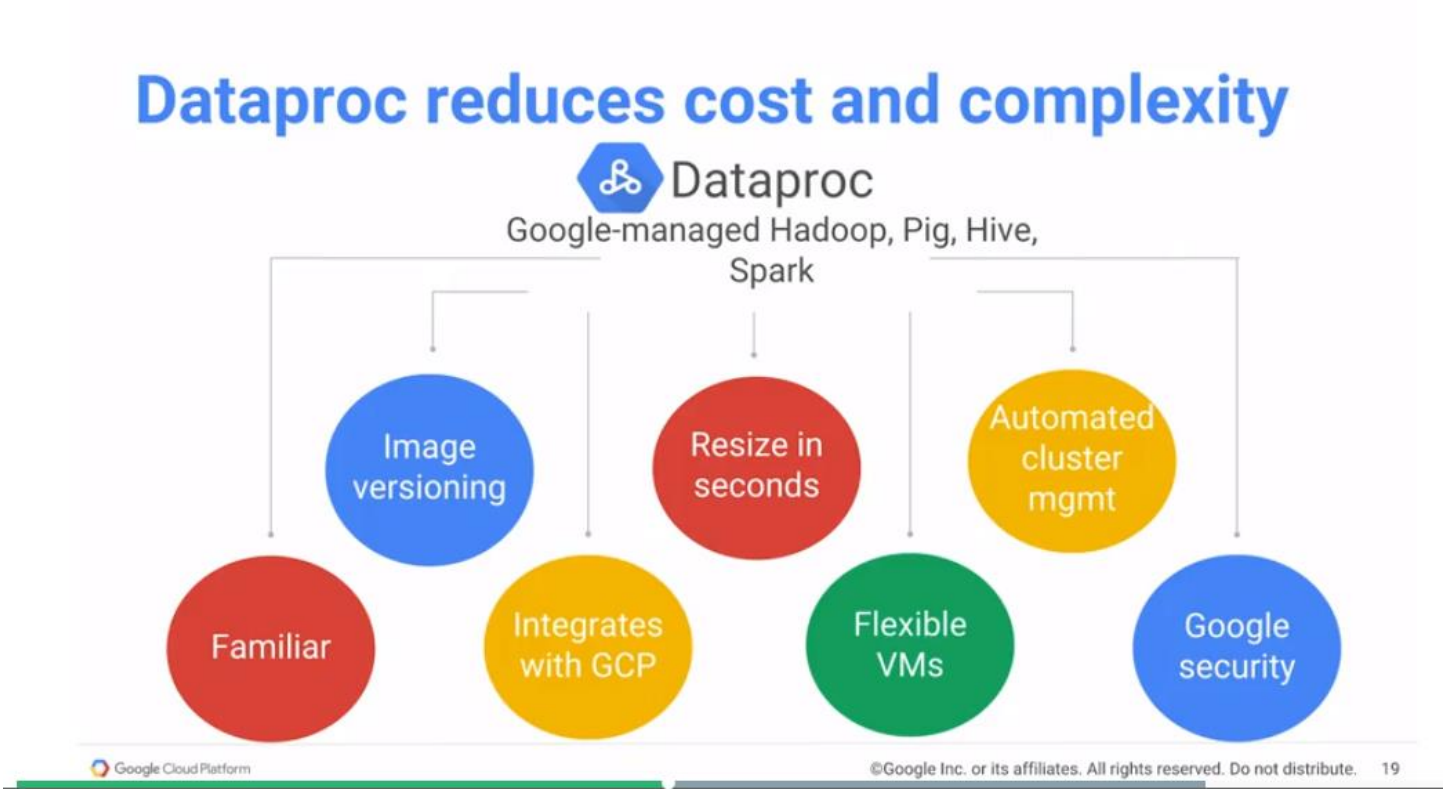


Pig provided a very convenient scripting language, and once you wrote those scripts, those script would get converted into Hadoop MapReduce programs, and they would run on the Hadoop cluster.

Another simplification that came about was this idea that most of the time you are dealing with structured data anyway. And if you have structured data, what if you have a schema, then you associate that schema on top of your structured data stored on a distributed file system? In the case of Hadoop, you would store it on HDFS. You have a distributed file system in which to store data and you perform queries using Hive.

Those were two simplifications that came about. Now, modern day programs on the Hadoop ecosystem tend to get written in Spark. Spark is very fast, it's interactive, and it has a bunch of libraries that allow you to deal with SQL, streaming data, machine learning, etc.

When people say that they're using big data on Hadoop, typically, they're talking about now they have Hadoop, MapReduce jobs, the low-level ones. They may have Pig Scripts, they may have Hive statements, they may have Python Spark programs, Spark car programs, etc. When Google says that we want to be able to take your Hadoop jobs to meet you where you are and to be able to run them on Google Cloud. We basically give you a way to run Hadoop, Pig, Hive, Spark, Presto, a variety of different Hadoop ecosystem things on the Google Cloud platform. The way you do this is with Dataproc - Dataproc is Google-managed Hadoop, Pig, Hive, Spark programs.



Dataproc is a cluster - it's not just a question of creating a single machine. Dataproc works on a cluster of machines that are automatically all set up. There are master nodes, worker nodes, you can set and all of those things. Their creation is extremely simple and straightforward. You can set up and initialize a Dataproc cluster in less than two minutes. You can also resize it. You can have a Dataproc cluster and add or remove a few more worker nodes and keep the cluster up. That kind of resizing is extremely simple and very straightforward. At the same time, it's very familiar, because it is your Hadoop, Pig, Hive, Spark jobs, they work unchanged. If you want, you can have HDFS and you can have your data in HDFS on the Dataproc cluster and read from HDFS.

However, a better way to do this is to take advantage of the very nice integration that Dataproc provides with GCP. What kind of integration?

Instead of storing your data on HDFS, you can store your data on Google Cloud storage. Now, what's the difference?

If you store your data on HDFS, you're basically taking your data, you're splitting it up into pieces, and storing it on the Dataproc cluster.

What that means, and this is probably what you're doing today if you're using HDFS, your cluster has to be up. Because if you delete your cluster, typically your data also goes away, and you can't do that. You are paying for compute even though all you need is a storage.

However, if you store your data on Google Cloud Storage, you get the huge advantage that your cluster life cycle and your storage life cycle are now separated. You can have your data on GCS. Bring up a cluster, have it complete a job, and then delete the cluster. So that's an important distinction. When you look at the cost of Dataproc, you should realize that you don't tend to keep a Dataproc cluster up and running 24/7 for months at a time.

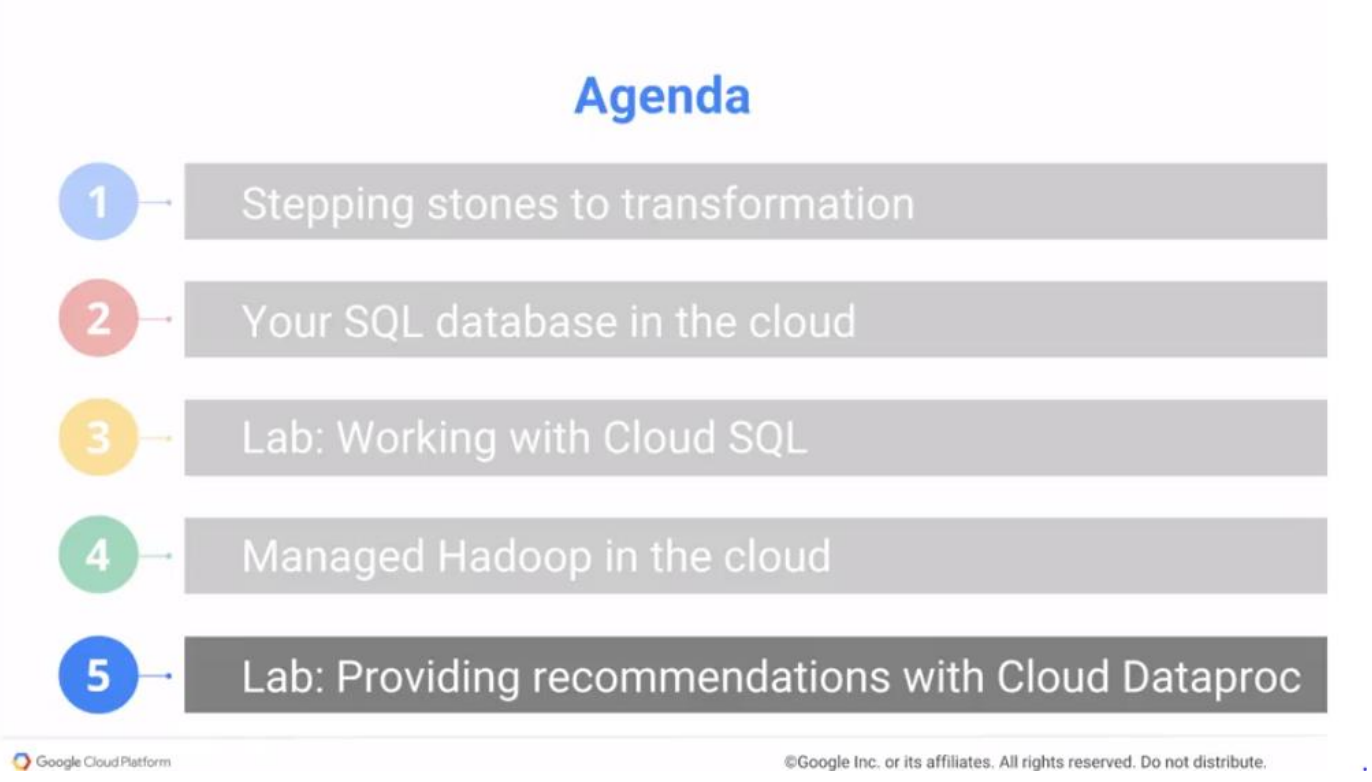
Instead, you think of a Dataproc cluster as a job specific resource. Every job that you want to run, create a new Dataproc cluster. And when the job is done, delete it. When deleting a cluster, you can't be moving data back and forth to it. And this is where storing your data on GCS is so powerful. You can store your data on Google Cloud Storage, and your data remains there. And the reason that we can do this, again, comes down to the quality of the networking within the Google data centers. Make sure that you store your data on GCS, ideally in a single region bucket. And create a Dataproc cluster in that same region.

And then you don't have to keep your Dataproc cluster up and around for way too long. In addition, Dataproc works very well with flexible virtual machines. This is basically the preemptable VMs that we talked about. So when you create your Dataproc cluster with a bunch nodes, create a few of them that are standard virtual machines, and a lot of them that are flexible virtual machines. If you get those virtual machines, you basically save yourself a lot of money. If you don't, well, to bad, but you still got your work done as you would have normally gotten them done using the standard machines.

And in addition to all of that, you basically get all the advantages of running on the Google Cloud, in terms of security, in terms of automated cluster management, in terms of image versioning. To make sure that all of your versions of Hadoop and Pig etc, they all work very well together.

So bottom line, Dataproc reduces the amount of cost that you're putting in. It reduces your complexity. It lets you just create a job and run it without worrying about managing that infrastructure yourself. Think about a Dataproc cluster as something that's specific to just that job that you're actually executing.

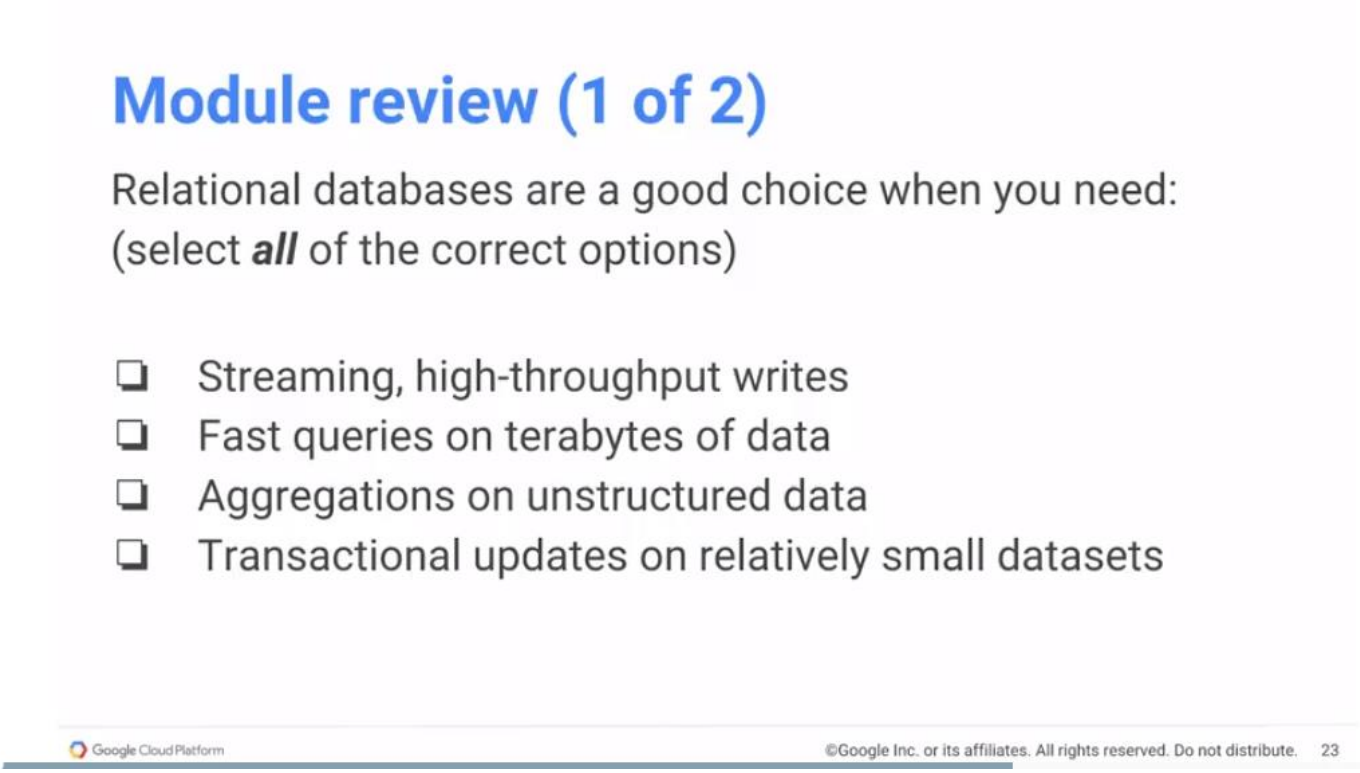
Providing Recommendations with Cloud Dataproc



https://codelabs.developers.google.com/codelabs/cpb100-dataproc/

Lab Review: https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/PV1MU/lab-3b-review

Module 3 Review



Let's do a quick module review. Relational databases are a good choice when you need which of the following? Streaming high-throughput writes, no. What about this is problematic for relational databases?

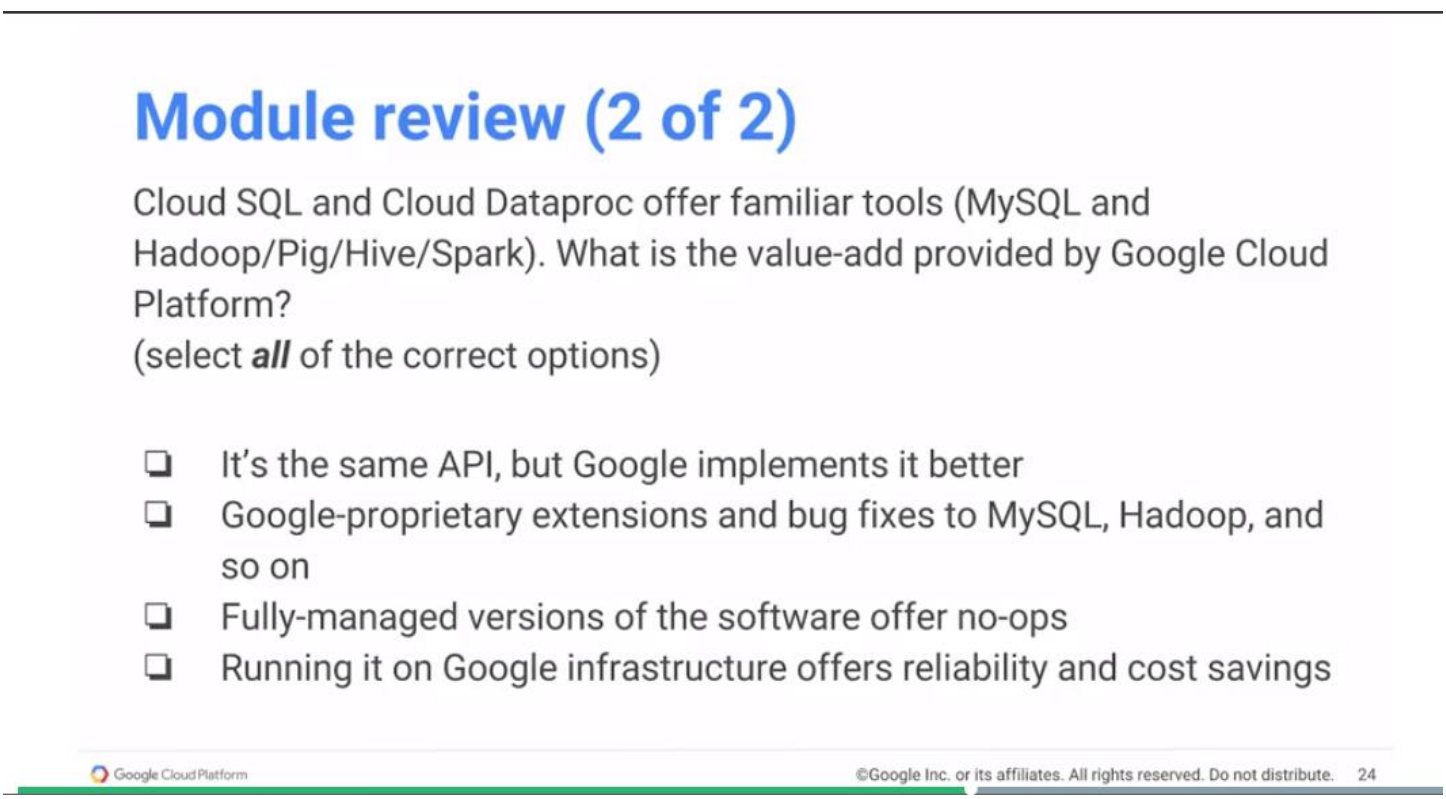
It's the high through-put. Relation databases may not be able to support very, very high through-put, I mean again, it depends on the through-put that you care about, but because they're transactional, rights tend to be relatively slow. Compare to storage mechanisms that don't try to manage transactions.

Second, fast queries on terabytes of data.

Now, the problem here is terabytes, it's this quantity. Relational databases scale pretty well to a few hundred gigabytes, but not more than that. And even at a few hundred gigabytes, you start running into problems of scale.

Third, aggregations on unstructured data. No, the problem here is unstructured. Aggregations are no problem. We can do sum, we can do average, we can do count. Those are all very standard SQL key words. The problem is unstructured. You cannot do a column of pixels in an image using a relational database. That's just it's not an appropriate use of relational database.

Transactional updates on relatively small datasets. Absolutely. The reason you want to use a relational database is if you need transactions and if you create datasets are a few hundred megs to a few gigs that's like the sweet spot for a relational database. It's perfect to use for that and that's a huge fraction of your use cases but you want to be aware of the exceptions, the cases were relational databases may not be the best choice. If you have high true put, if you have terabytes of data or if you have unstructured data. For those cases, we will talk about alternatives in the next chapter.



Next question, Cloud SQL and Cloud Dataproc offer familiar tools. MySQL, in the case of Cloud SQL.

Hadoop, Pig, Hive and Spark in the case of Cloud Dataproc. So what's a value-add Provided by GCP.

A, it's the same API, but Google implements it better.

No, it's the exact same code. In fact, if we find improvements, we basically go back and check them in back to the original open source code base. It is the exact same code. We're not making any changes. So that answers the second question too, there are no Google proprietary extensions. If there are bug fixes they go back into the main branch.

Third, fully managed versions of the software offer no-ops, absolutely. You basically get fully managed versions of MySQL of Spark that just run on the Dataproc cluster. You don't need to install anything, you don't have to worry about managing those kinds of infrastructure, that's all taken care of for you.

Google infrastructure offers reliability and cost savings. Absolutely. As we talked about. For example and this is very relevant in the case of something like Dataproc. The way you approach Hadoop and Pig and Hive jobs changes as soon as you think about putting your data on cloud storage. And reading it from your Dataproc instance.

Module 3 Resources

- Cloud SQL https://cloud.google.com/sql
- Cloud Dataproc https://cloud.google.com/dataproc
- Cloud Solutions https://cloud.google.com/solutions

Here are a few resources. So that's the documentation of CloudSQL, documentation of Dataproc. But let's go ahead and look at the third resource here of solutions. There are a variety of different solutions on Google Cloud platform. So here is cloud.google.com/solutions. And you have a variety of different fully worked out examples that you can take and adapt to your needs so there are things around media or on mobile application

solutions around big data, around financial services. But let's take a look at retell and commerce. There is retail and commerce. And now if you scroll down to solutions around retail and commerce, there is a solution around using machine learning for product recommendations. Go ahead and look at that documentation. And this should look relatively familiar to you. This is recommending houses given data on CloudSQL. And this is basically the same solution that we took and we adapted for our training course. And as you can see,there are a lot more solutions, and a lot more verticals. I strongly encourage you to go look at these solutions and use them as starting points for the things that you are developing.

MODULE 4: SCALING DATA ANALYSIS: COMPUTE WITH GCP

This module is about the more transformational technologies in Google Cloud platform that may not have immediate parallels to technologies that attendees are using (“what's next”).

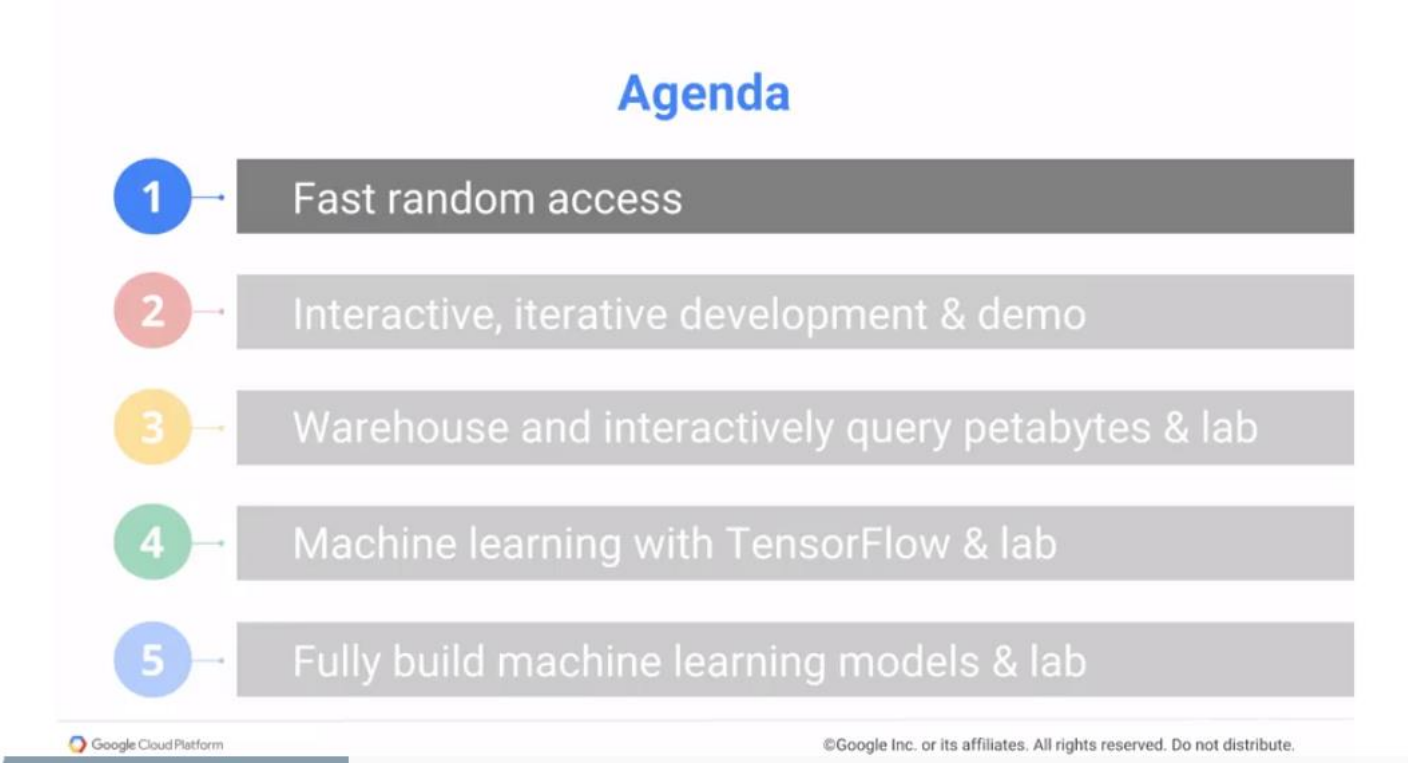
- Employ BigQuery and Cloud Datalab to carry out interactive data analysis
- Train and use a neural network using TensorFlow

Scaling Data Analysis. In this module, we look at things that are more transformational, things that you may not have done exactly the same way before. We will start out by looking at how to do fast random access. And if you're looking at fast random access, you need fast querying. You've probably used the relational database before, we look at other options, and how you can accomplish them on Google Cloud. Similarly, we look at how to do interactive, iterative development, but to do all of your processing on the Cloud using a notebook format.

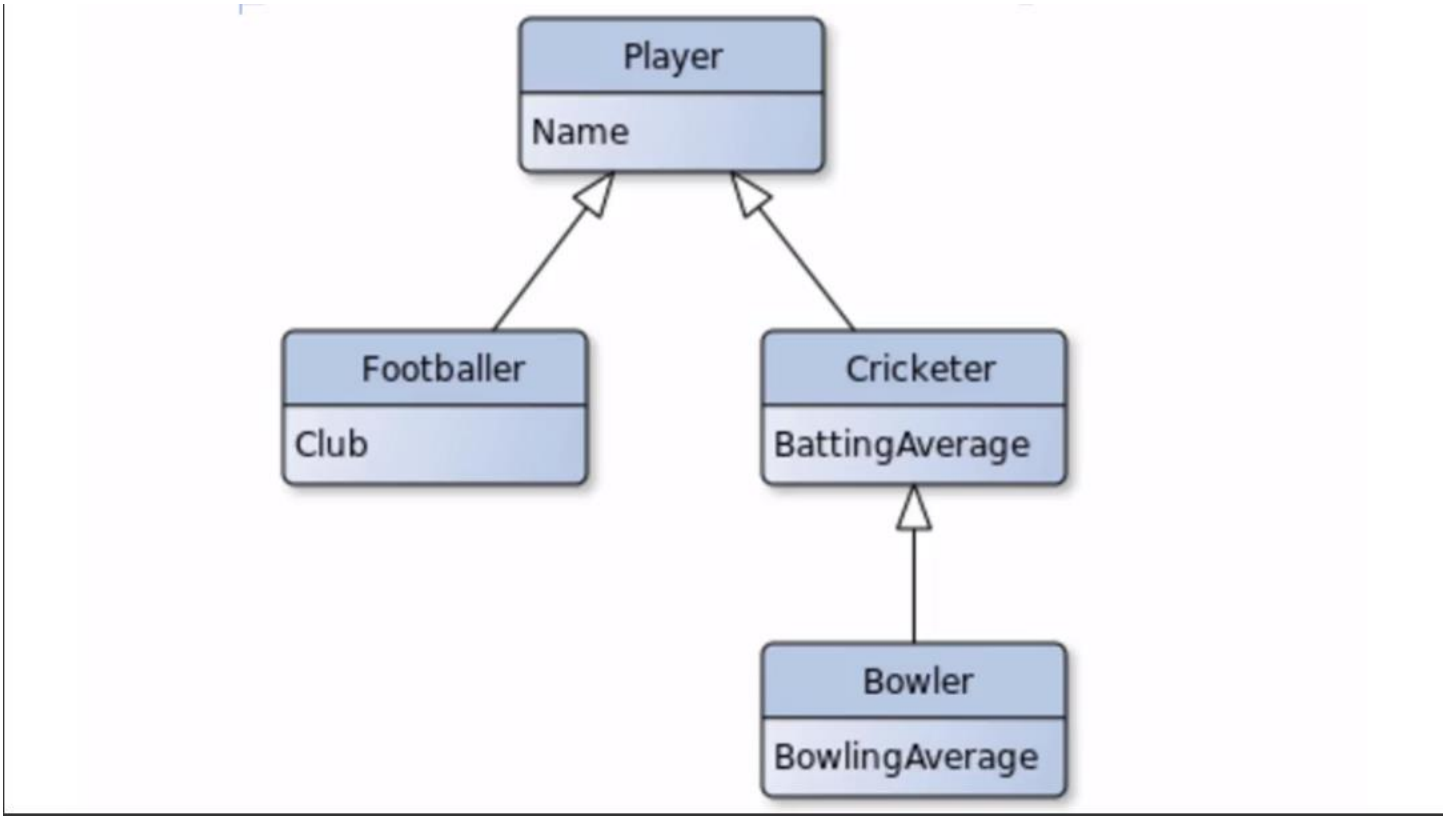
Then, we look at how to warehouse your data, and to carry out interactive querying of extremely large datasets, of petabytes scale datasets, but still do your querying interactively. And having done that, the next thing that we look at, and continuing on this whole idea of doing the things that are transformational, is that we look at transfer flow, and how you can do machine learning with transfer flow, again, on Google Cloud. And

then, finally, we will look at pre-built machine learning models that are available. That you could take those machine learning models and interpret them into your own applications.

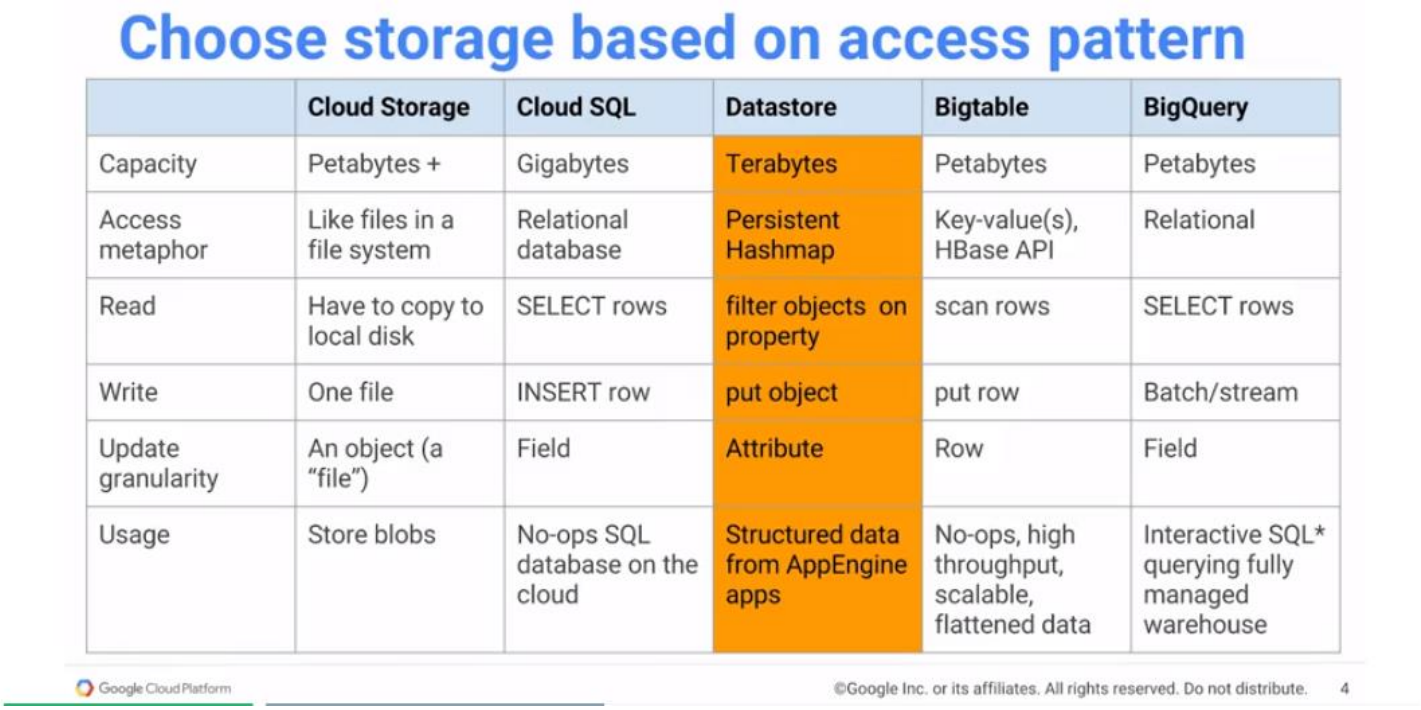
Fast Random Access



Let's look at some transformative changes, first Fast random access.



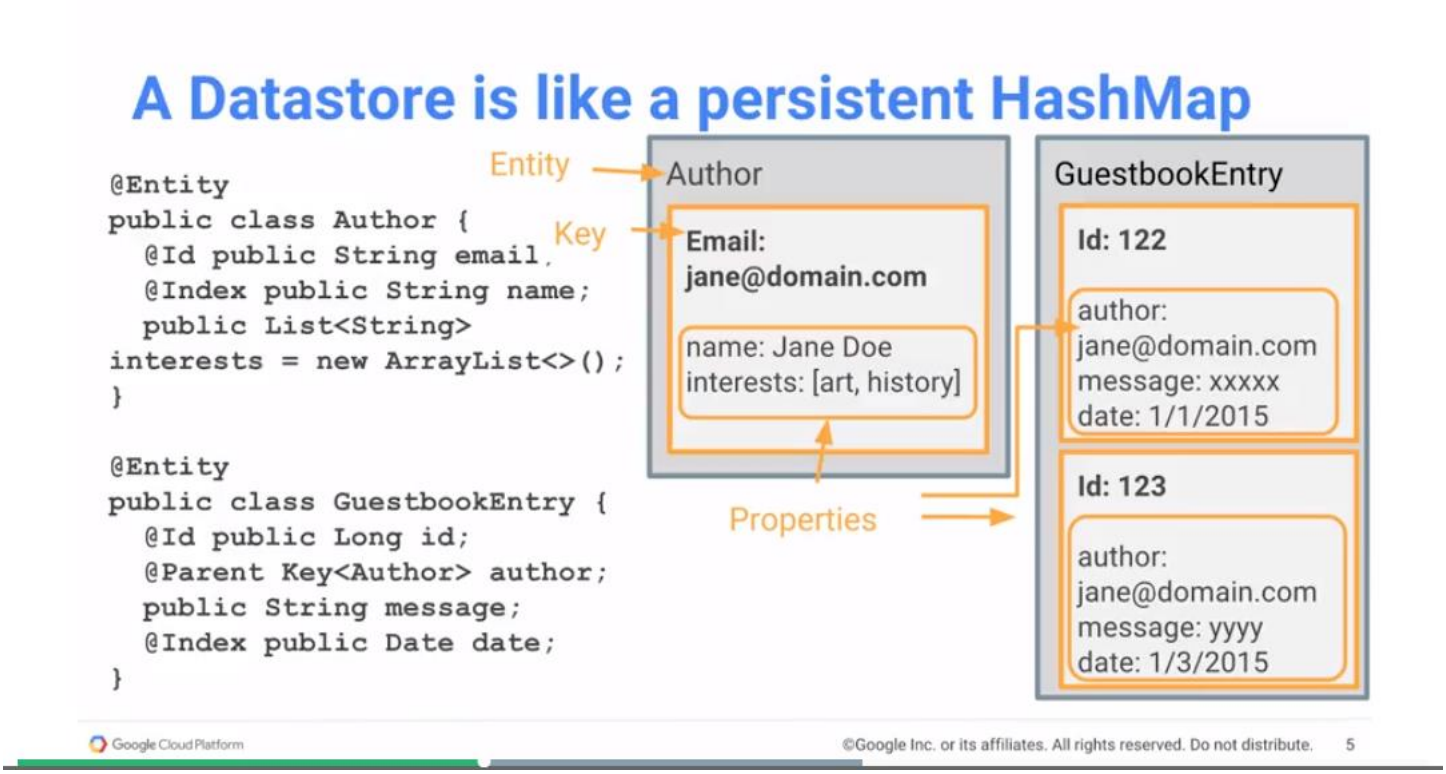
Many programming languages are object oriented, which means much of the software that you write deals with objects and the good thing about objects is that they're hierarchical. Let's say for example that in your program you have a hierarchical data structure like a Player. You have players who are either footballers or cricketers, and cricketers not all of whom bowl. You have some cricketers which have bowling averages; everyone bats, so all cricketers have batting averages. You have this hierarchy, and you want to represent this in a relational database, because you want to persist the data. If you use a relational database to persist an object hierarchy, you get into an object relational impedance, there's a mismatch here. For example, we said that all cricketers have batting averages, but not all of them have bowling averages. However in the table itself, if you go look at it there is a Name, a Club, a Batting Average, and a Bowling Average. What is a Batting Average of a football player? Makes absolutely no sense, but we have to have it. We basically go ahead and put in a null in there, and once you do that you have data integrity problems from this point onwards. So how do you prevent this kind of an issue with an object to relational mapping? Well one way is if you can store objects directly, and that's what Cloud Datastore on GCP lets you do. So Datastore scales up to terabytes of data, where in a relational database typically goes into a few gigabytes, Datastore can go up to terabytes and what you're storing in a Datastore conceptually is like a Hashmap (it is a NoSQL database – see https://cloud.google.com/datastore/docs/concepts/overview). There is a key or an id to an object - when you are writing to Datastore you're writing an entire object, but when you are reading from it, that's searching, you can search with the key but you can also search by a property. For example, you can look for all cricket players whose batting average is greater than 30 runs a game.



This is done by using one of the indexed feeds, which we'll look at shortly.

You want to update, again you can update just the batting average of a player, and you can update this in a transactional way. So Datastore supports transactions, it allows you to read and write structured data. It is a pretty good replacement for use cases in your application, where you may be using a relational database to persist your data. However, this replacement is something that you would have to do explicitly. Unlike the things that we've talked about in the previous chapter you can't just, for example in the previous chapter we

said you have a spark program that you are running on a Hadoop cluster on-premise. You want to run it on GCP just run it on Dataproc, pretty much all of your code just migrates unchanged. If you have a MySQL code base, well whatever you're doing to your MySQL on-premise you can do MySQL on Google Cloud using Cloud SQL, those are easy migrations. Take what you have, take those use cases that you have, just move them to the cloud, but when we talk about something like Datastore, now it's not that easy a migration. You have to change the code that you're doing, where the way you interact with Datastore is different from the way you'd interact with a relational database. So how do you interact with Datastore? Well, the way you work with Datastore is that it's like a persistent Hashmap. For example, let's say we want to persist objects that are author objects. You'd say I have an author class, it's an Entity, that's the identity. It's an annotation that you add and I'm showing you Java here, but it works with a variety of object-oriented languages. And you say that the author is distinguished by their email address, the email address is an Id column, so you say add Id. We want to search for authors by name, so we'd like the name property to be indexed, and just to show you that you can have hazard relationships, an author has a bunch of different interests. Same thing about guestbook entries, you store guestbook entries, each entry has an id that makes it unique, it basically has a Parent Key and Author. These are the people who wrote the Guestbook Entry and that's a relationship. You have messages, we're never going to search apparently because it's not indexed. We're not going to search for guestbook entries based on the text of the message and we have dates, right? And that's something that we might want the search based on. Once you have an Entity, you have an Author, there's an Entity you have, an Author has an email which is the id, and a name which is the index. If you want to create an Author, you basically call the constructor, just as you would do for any plain old job or object. A new Author xjin@bu.edu, name is xjin, you have your Author object, but at this point the Author object is only in memory. You want to save it, you basically call save passing in the entity. ofy here is the objective file library, it's one of several Java libraries that help you deal with Datastore. In this case this code is showing you objective file, we save the entity and at this point the xjin object has been persisted. If you want to read it, if you want to search for it, what you can do is say load all authors and filter them by name Ha Jin, and because name is an indexed field we can do this. We can filter by name Ha Jin, and we will get back an iterable of authors.



Why iterable and not a list of authors?

Well because Datastore scales up to terabytes, so one of these columns that your search is based on, what comes back could be gigabytes of data, might be much more than can fit into memory, so we give you back an iterable. Well if you know you're going to get back only one item, such that's the second one here, you're loading authors and you're finding id xjin@bu.edu, at that point you're going to get one author back, so you best get back the author object. We call it JH within the code, and now we can update the name of jh, you can say jh.name = Jin Xuefei and then save that entity.

At this point we have a newly persisted object, the object that's persisted basically has a new name, and then if you want to delete the entity, we just say delete entity jh.

CRUD operations on a Datastore entity

```
@Entity
public class Author {
  @Id public String email;
  @Index public String name;
  public List<String>  interests = new ArrayList<>();
}

// CREATE
Author xjin = new Author("xjin@bu.edu", "Ha Jin"); xjin.interests.add("Misty Poetry");
ofy.save().entity(xjin);

// READ
Iterable<Author> authors = ofy().load().type(Author.class).filter("name","Ha Jin");
Author jh = ofy.load().type(Author.class).id("xjin@bu.edu").now();

// UPDATE
jh.name = "Jīn Xuěfēi (金雪飞)";
ofy().save().entity(jh).now();

// DELETE
ofy().delete().entity(jh).now();
```

Google Cloud Platform ©Google Inc. or its affiliates. All rights reserved. Do not distribute. 6

Create, read, update, delete - you can pretty much do everything that you do in a relational database from the transactional way using Datastore. Another access patron, so these are again options to using a relational database, you could use Datastore if you need transactional support for hierarchical data, something that relational databases don't handle very well.

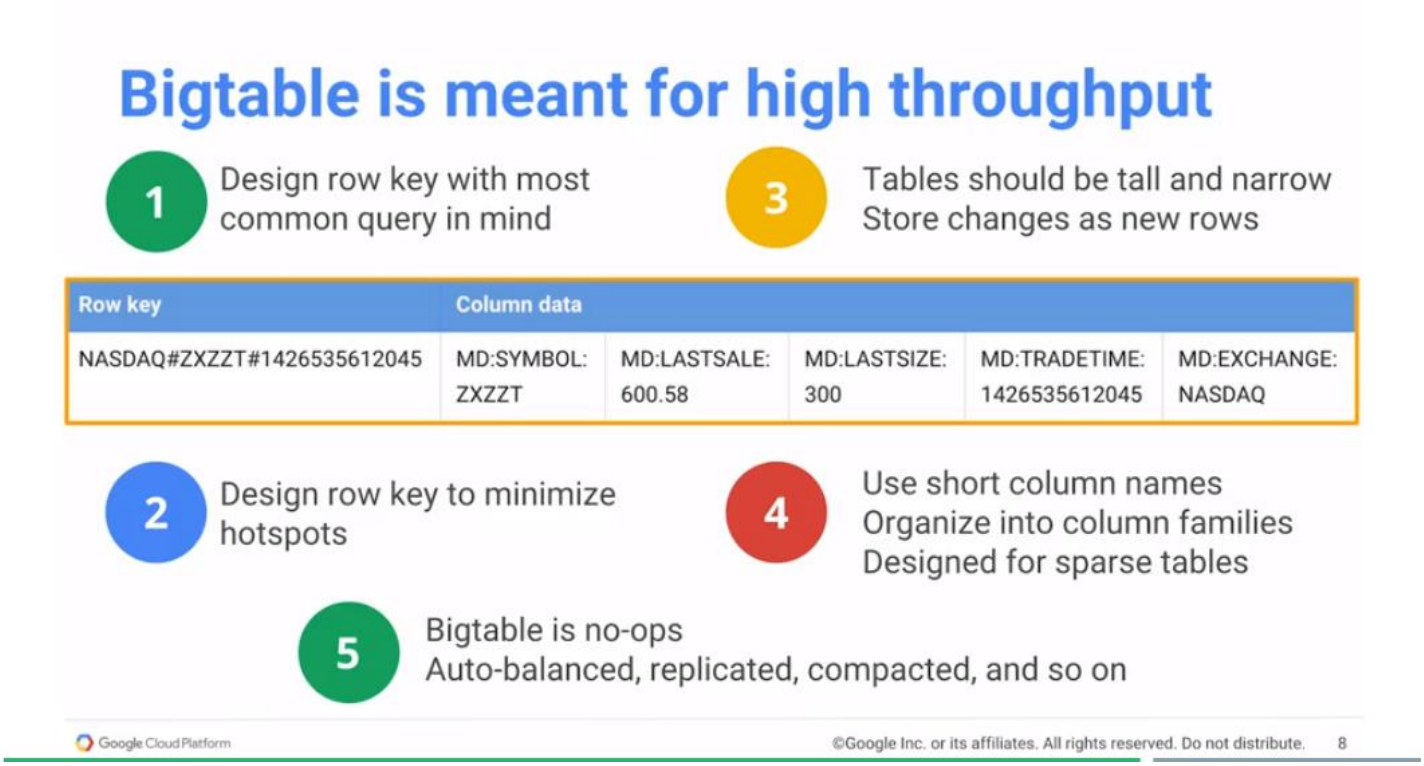
Choose storage based on access pattern

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Relational
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	Attribute	Row (write new row instead)	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse

Google Cloud Platform ©Google Inc. or its affiliates. All rights reserved. Do not distribute. 7

Another reason that relational databases may not work very well, and we've discussed this in the module review section of the previous chapter, is if you have high throughput needs. If you have sensors that are distributed all across the world, and you're basically getting back millions of messages a minute, that's not

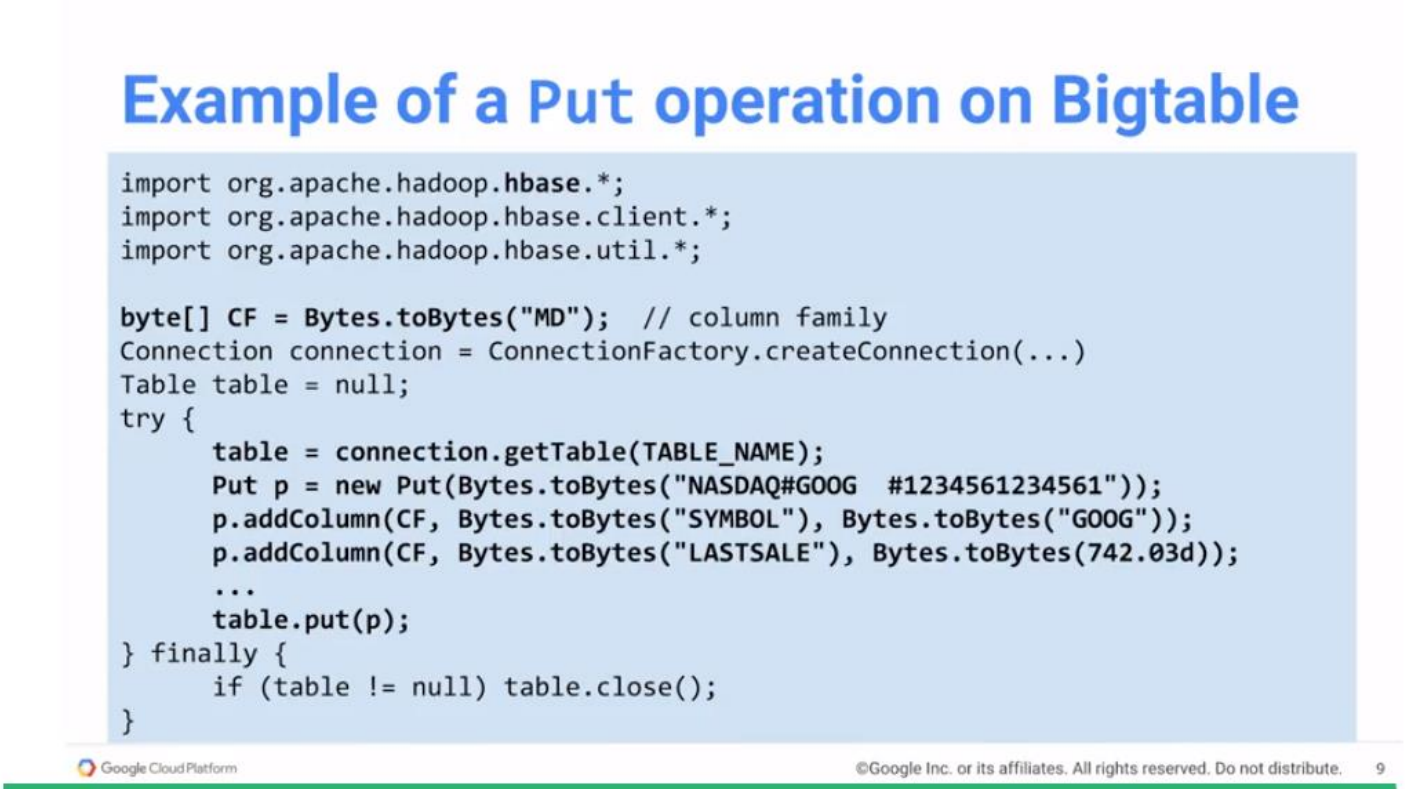
something the Cloud SQL can handle very well. That's not something a relational database can handle very well, that's essentially an open-only operation. We're just getting you data and we're saving, and we don't need transactional support. And because we are willing to give up transactional support, the capacity of Bigtable is no longer like the Terabytes that the Datastore can support, but Petabytes. On the other hand, what we've given up is the ability to update just a single field of the object, we have to write an entirely new row. The idea is that if we get a new object, we basically append it to the table and then we read from latest data and go backwards. So that the very first object that we find at the particular key is the latest version of that object.



So Bigtable is really good for high throughput scenarios, where you want to be not in the business of managing infrastructure, you want something to be as no-ops as possible. With Bigtable you basically deal with flattened data, it's not for hierarchical data, it's flattened and you search only based on the key. And because you can search only based on the key, the key itself and the way you design it becomes extremely important. Number one, you want to think about the key as being the query that you're going to make, because again you can only search based on the key. You cannot search based on any property, and because you can't search fast based on properties, you're going to be searching based on keys, you want your key itself to be designed such that you can find the stuff that you want quickly. And the key should be designed such that there are no hotspots, you don't want all of your objects, all of your rows, falling into the same bucket, you want things to be distributed there. Tables themselves should be tall and narrow. Tall because you keep appending to it. Narrow, why? The idea being that, if you have Boolean flags for example, rather than have a column for each flag, and have the value be 0 or 1. Maybe you just have a column that says, these are the only true flags on this object. This kind of thing becomes extremely useful if you're trying to store, for example, user's ratings. A user may rate only like five out of the thousands of items in your catalog, and browser has thousands of columns, one for every item. You simply store object comma rating for the things that they've actually rated, and that could be a new column.

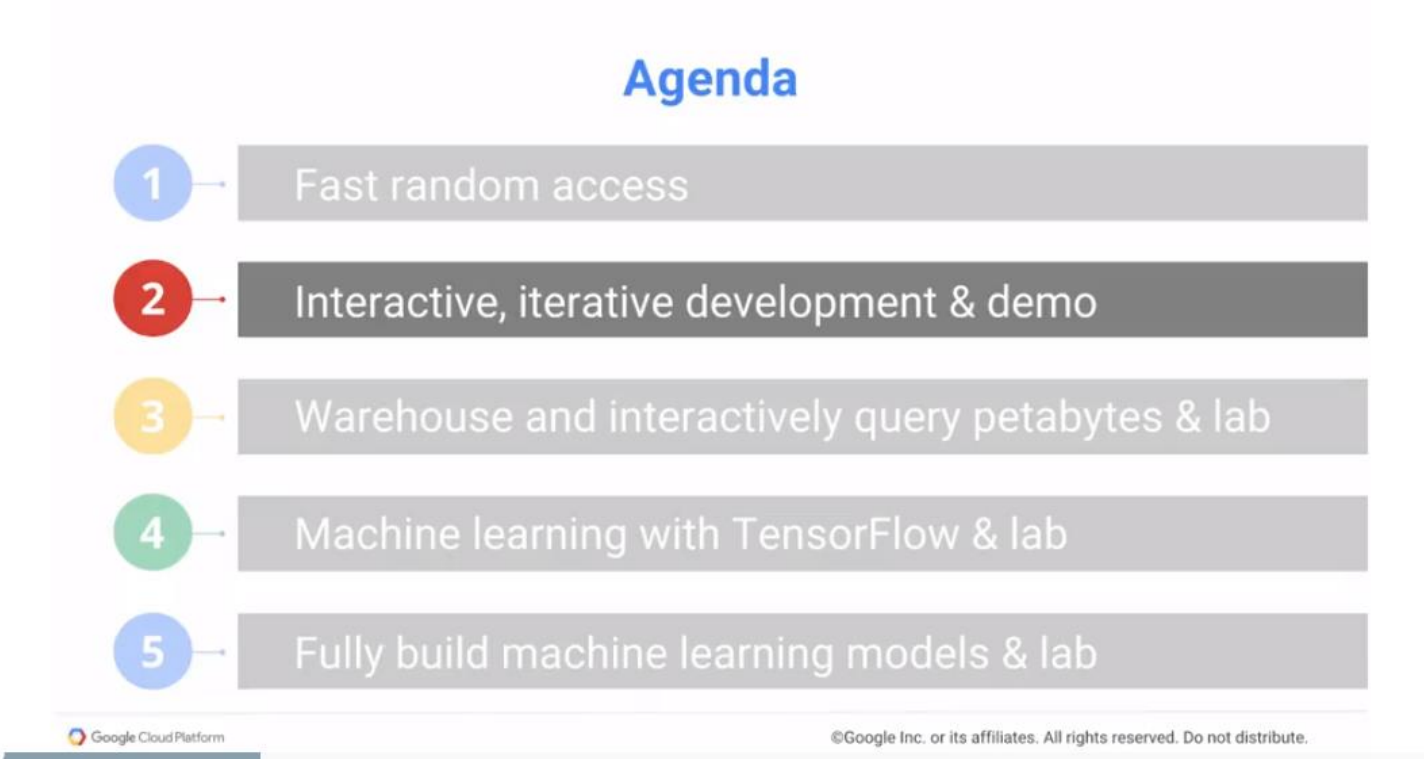
Even though we said that your columns have to be flattened, there is this concept of a column family. For example here MD is market data; MD colon symbol, MD colon last sale, this is a way to group together related columns.

The reason to use big tables and it's No-ops, it's automatically balanced, it's automatically replicated, its compacted, it's essential No-ops. You don't have to manage any of that infrastructure, you can deal with extremely high throughput data. This is how you work with Bigtable, you work with it using the hbase API. So that's why what if you're importing is org.apache.hadoop.hbase. You work with it the way you would normally work with hbase, you basically get a connection. Go to the connection and get your table, you create a put operation, you add all of your columns, and then you put that into the table and you've basically added a new row to the table.



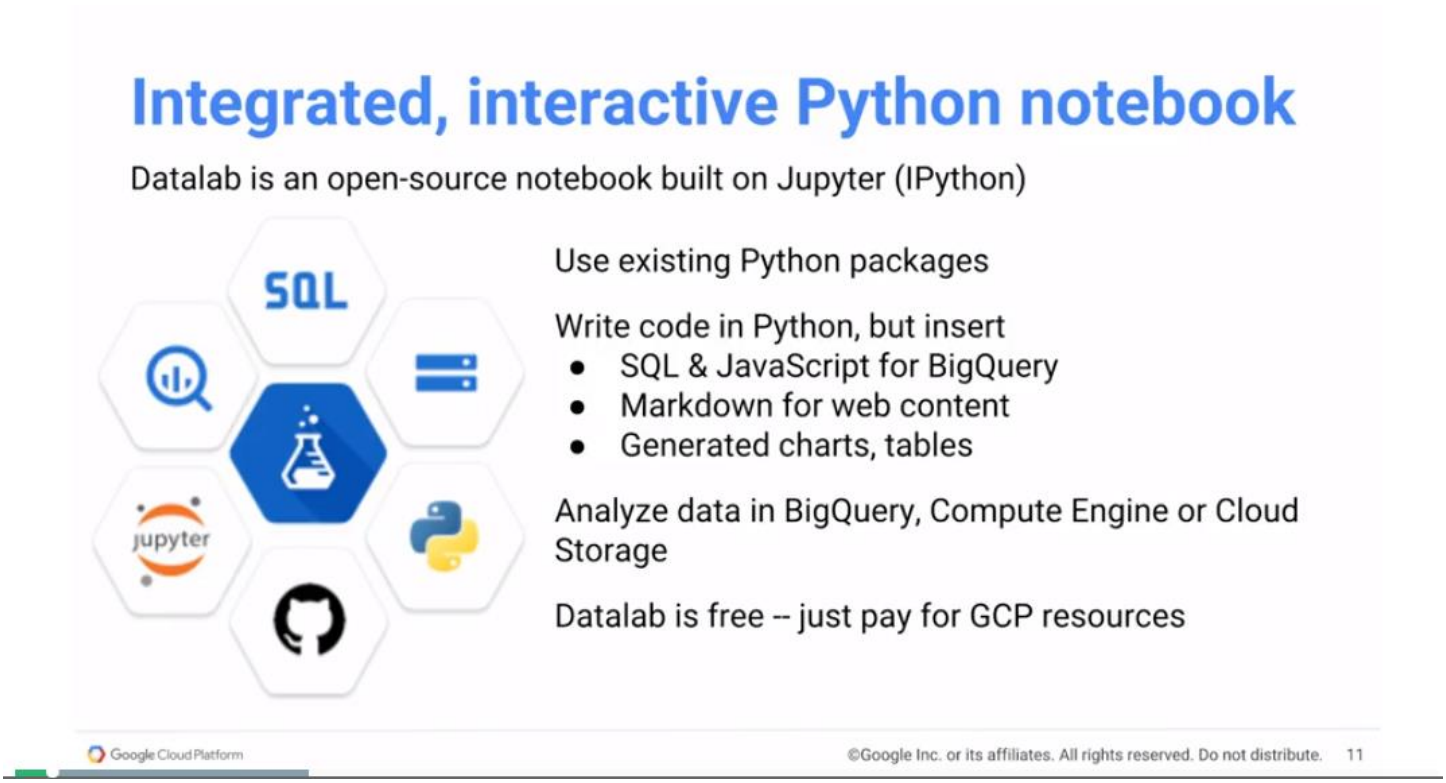
If you're familiar with hbase, it's exactly the same way.

Interactive, iterative development & demo



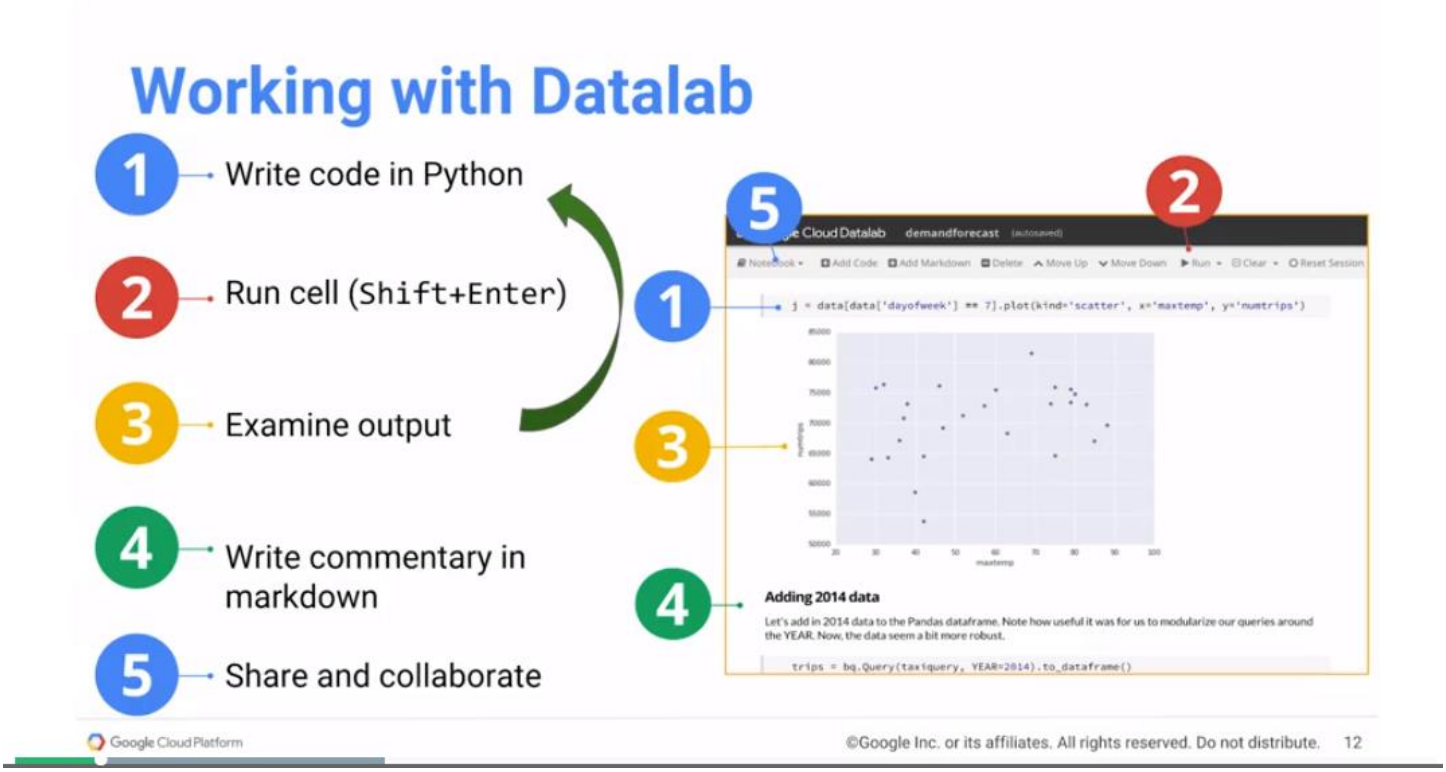
So, in this section, again we're talking about transformational cases. The next transformational case we want to talk about is about notebook development.

As a data scientist, one of the major changes that happened in the way I work with data is Datalab.

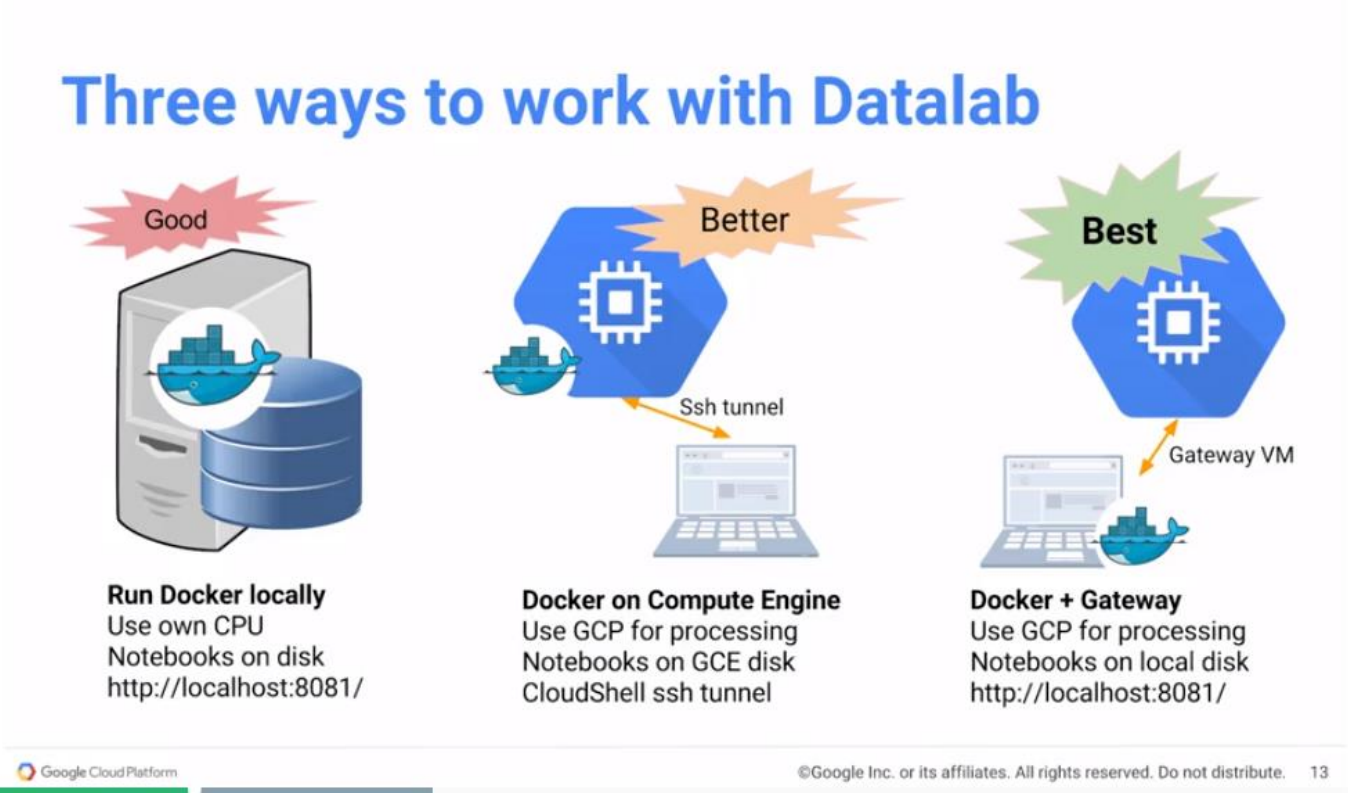


So, let's, the way you work with Datalab, is that you have a web page, that's your Datalab webpage. And in this webpage, you can basically write code in Python, and once you write the code in Python, you can run that

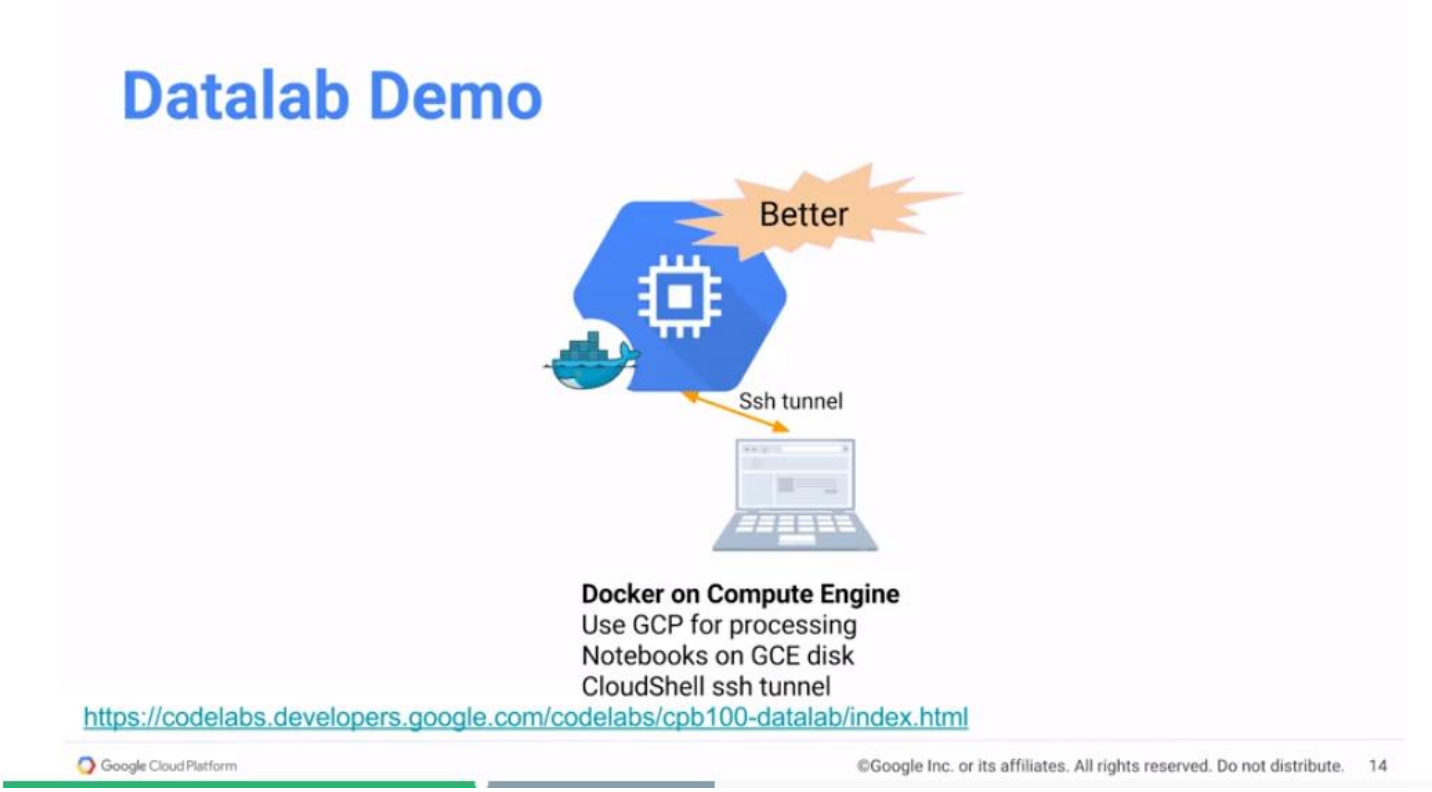
code by either hitting Shift Enter, or by clicking the Run button at the top of the menu. You can look at the output, you can go back, change the code, run the cell, change the code, run the cell and at that point some time you're basically satisfied with the way the code works you can go in and you can write some commentary. And the commentary can be mark down format, it can have titles, it can have bold emphasis and all of those kinds of things. It can have links, images, you write your commentary your documentation, and then because it's just a web page you share it and other people can come in. And they can execute your notebook and they can make changes to your code and rerun your notebook, it becomes an extremely good collaborative environment.



One issue still exists, and the issue is where does the web server run? Remember that this is the client client site, the web page is a client but you still need the web server. If you run the Datalab or iPython or Jupyter Web Server on your own laptop, then as soon as you close down your laptop, nobody can access your notebook. So, you really want to have this thing running on a cloud computer, and Datalab Is a way to run it on GCP. To work with this then, you have to think in terms of two things, where does a server run and where does a client run. So, one option is to run both locally. And this is if you're doing local development. Your own CPU, you store your Notebooks on disk And you access your notebook using localhost:8081/ for example, that's your port. And data log comes in a docker container, so you're basically running a docker image locally. So, that's good as long as you are the only one who needs to access this notebook. The second option, this is good if you have multiple people going to access a notebook, is to run the Docker container on Compute Engine.

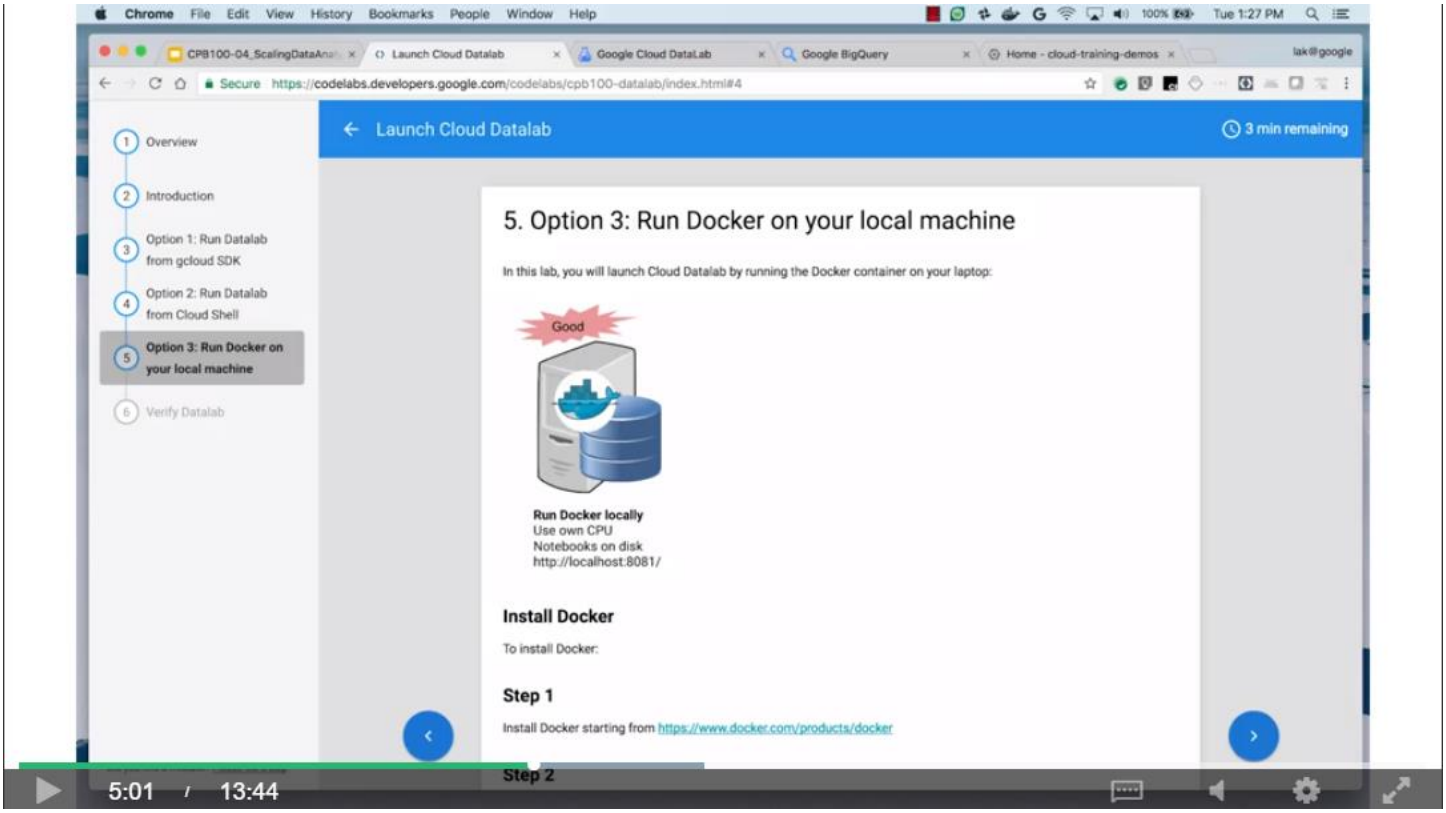


Set up a Google compute engine instance, then on that instance you run that Docker Container and this way then whenever you need to connect to the notebook. Use an ssh tunnel CloudShell will let you do this. So, you'll use an ssh tunnel where CloudShell to connect to this GCE instance and then inside your browser you're working with it, but remember that everything that you're running the code itself is getting executed on the computer engine instance.

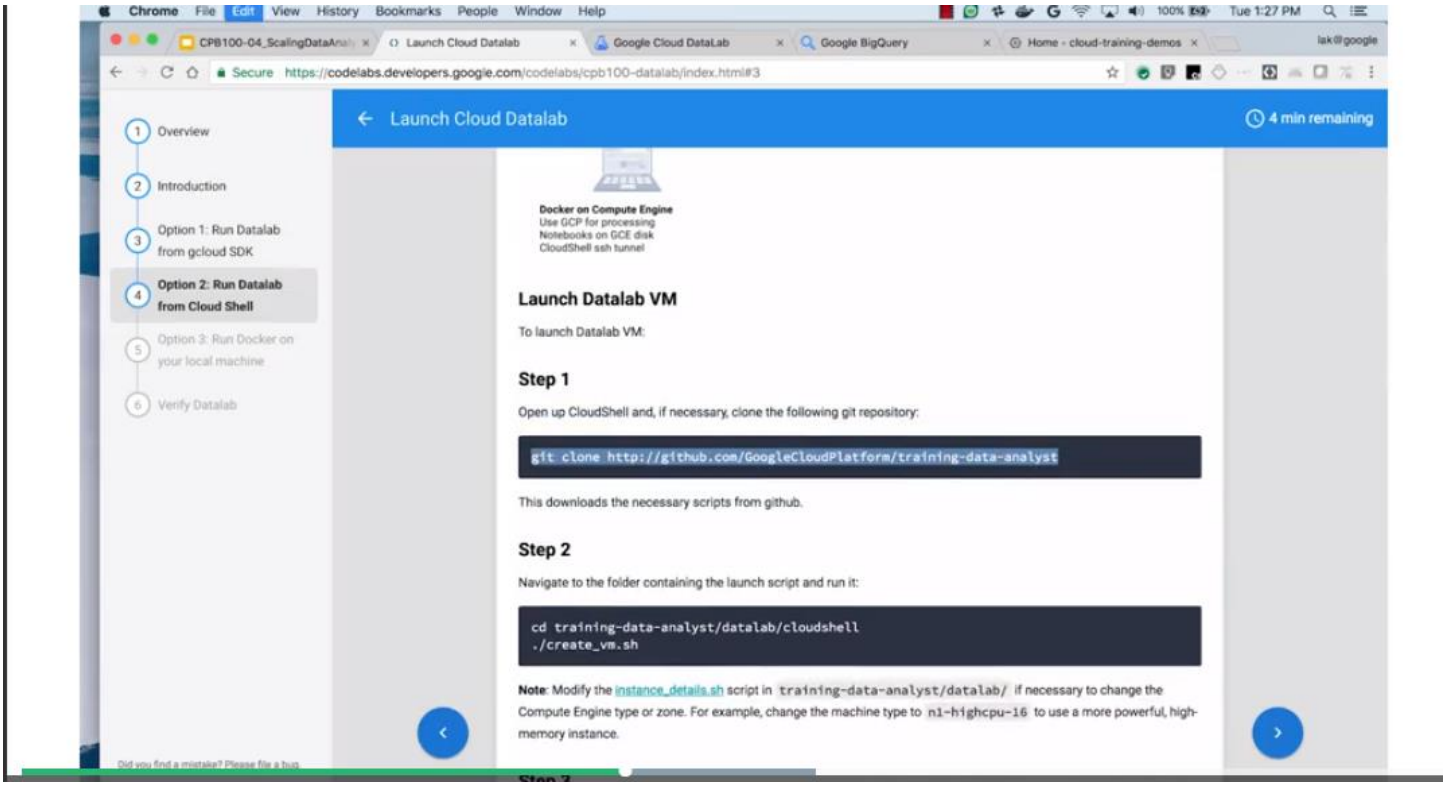


https://codelabs.developers.google.com/codelabs/cpb100-datalab/index.html.

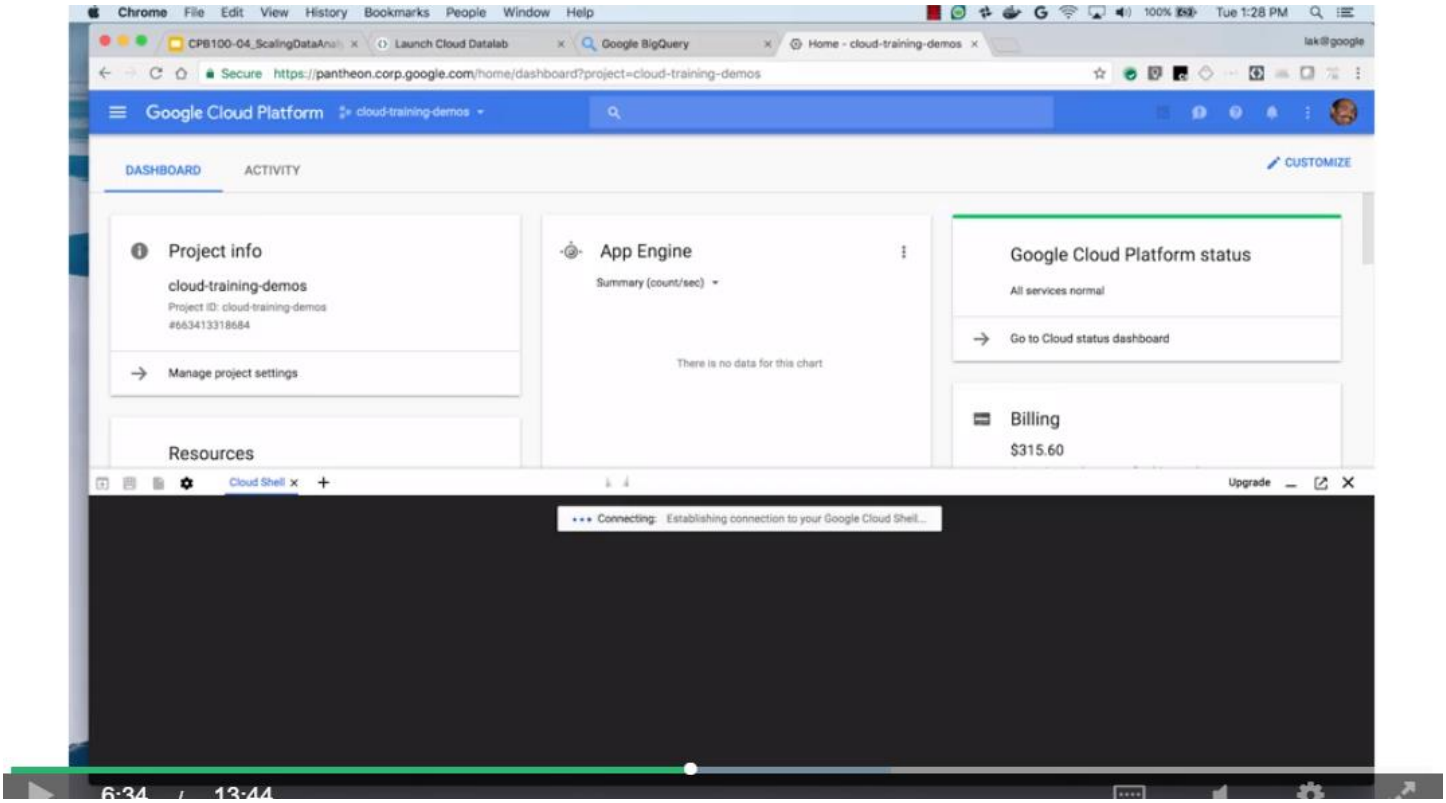
A third way to do this is to basically have it again on a computer engine instance, but access it through a gateway, so you basically have a proxy set up and this involves a little bit more in terms of setting up your browser. For the purposes of this class, the CloudShell approach is what we will use. But if you are going to be doing this a lot. If you are a data scientist, you do quite a bit of data science work. You might want to explore the third option, because remember that CloudShell is an ephemeral VM. And every 60 minutes or so it'll get recycled. And then you'd have to go create the SSH tunnel again. It's pretty easy to create an SSH tunnel, it's just a single script that you run, but it is still something that you might want to avoid doing, it can get pretty frustrating. So, if you're going to be doing this a lot, and if you're going to be doing this for more than 60 minutes, let's say if you're going to be doing this for more than 60 minutes, you might want to look at the third option. But for the purposes of this class, we will use the second option because it's very simple, very easy to get going. We already have CloudShell. It's easy to create a Compute Engine VM. So, let's go ahead and do that. The code lab gives you directions for doing all three of these things. Let's click on this link. And you will see that it takes you to option one. One option, two or option three. Option one from G-cloud SDK, option two from cloud shell and option three from local machine. These are actually reversed from the order in which we covered them but this is basically the order of recommendations if you will. In this class we'll use option number two. To run Datalab from Cloud Shell. If you want, you can follow along with me, pause the video if necessary so you can catch up, and we can go on. So, here what I'm going to be doing is, number one, I'm going to open up Cloud Shell. And, if necessary, I'm going to clone the following Git Repository. I already have the Git Repo, so that I can copy this. We need to open up cloud shelf. So, how do we open cloud shelf? The way you open cloud shelf okay is to base go to cancel this okay



The way you open up CloudShell is to go to console.cloud.google.com. And this takes you to the JSP web console.



And you can click on the Activate CloudShell. If you've been using Quick Labs to do your labs, then we will already have a data lab that's up an going for you. So, you can just look at this demo as how you would do it outside of the Quick Labs environment.



So, once we have Cloud Share up we can basically go ahead and do get clone In this case I don't need to get cloned because they already have it. So, there is the training data analyst already exists. So, the next step is to go to the datalab directory says cd training-data-analyst/datalab/cloudshell. Execute ./creatvm.sh. and at this

point it is creating US central 1a and in one standard NBM. If you want to change the zone or you wanted to change the type there is a file. Open a new cloud session Inside datalab - there is an instance details that I search and you can change then the zone or you can change. Change the machine type here. But us-central1 and n1-standard-1 are pretty reasonable. At this point notice that the datalabvm has been created, that's the name of my VM. It's in this zone, it's n1-standard-1. This is the IP address of the VM. This is going to be useful if you're setting up a different proxy that's the option, there are other options that exist, but for now, maybe we can basically we will connect to this through cloudshell. You can look at this VM by going to compute engine and if you look at the instances that are currently running there is the data lab VM that's currently running We can associate into this VM, we can do staff on that VM if you ever need to, right. So, you can associate into the VM and you can see what's running. And what should be running is docker, so if we go into this VM,

Docker, unfortunately, runs only as a root. So, after we go into this VM, I'm going to sudo su, so that I'm now root, and I can do docker ps. And this tells me that gcr.io/google_containers is running, but not yet datalab. So, if we can just wait for Datalab to get started.

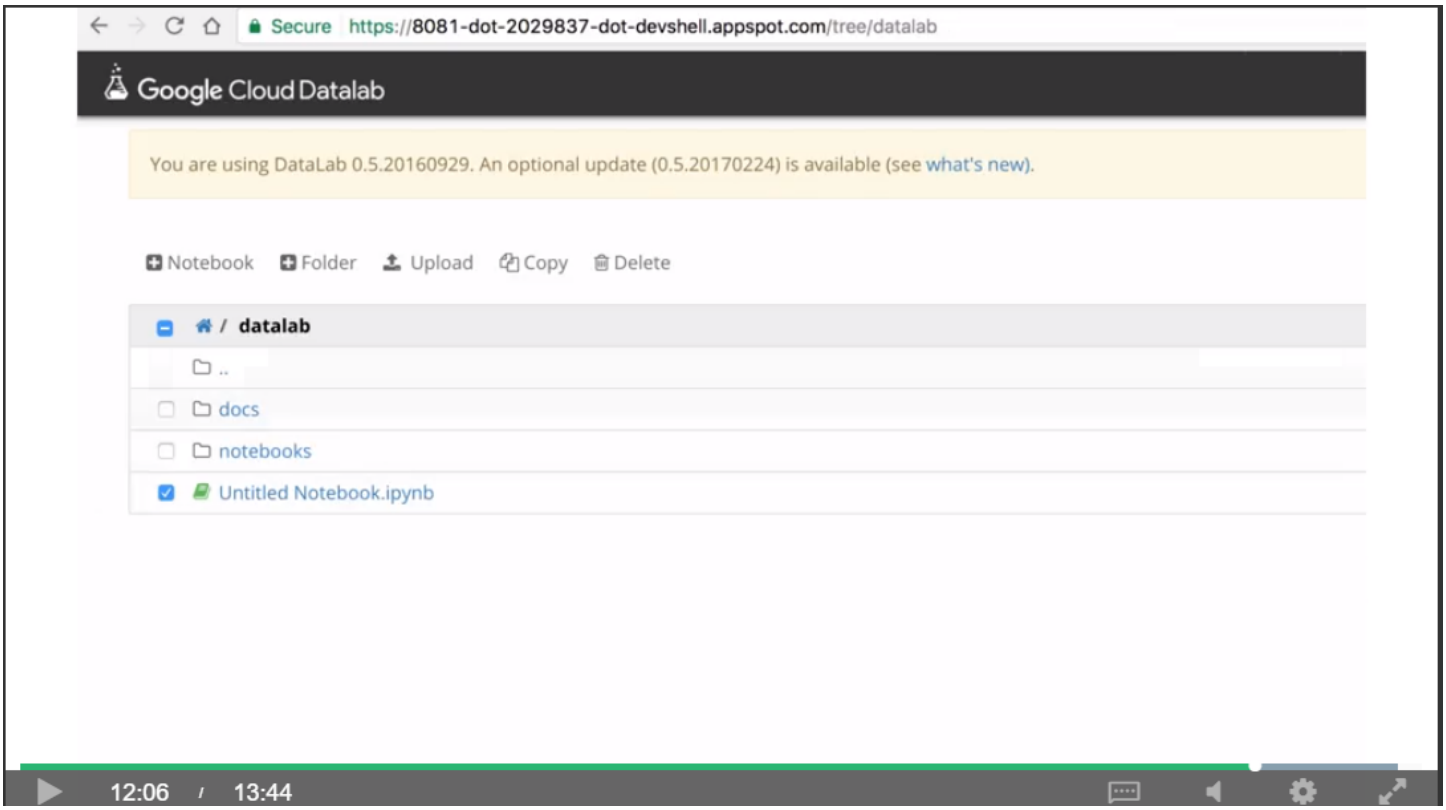
A couple more minutes.

When we do Docker PS, let me just clear this so it's obvious, that we do Docker PS now. Notice that Data Lab Cloud Training has also started. So, now, that Data Lab has started, we would be able to go into and connect to it. So, to connect to it, be in the same directory that you did create VM from, there is another script called start tunnel. So, go ahead and do ./start.tunnel, and this lets us tunnel into this VM from cloud shell and then click on this arrow to do a Web preview. Change the port to be 8081. So again, after you start the tunnel, move to this thing that says Web preview And change the port to be port 8081 and at this point you are now into Datalab. So, let's go ahead and look at how to use Datalab. So, I'll accept the conditions and now that we're into Datalab, let's go ahead and create a new notebook.

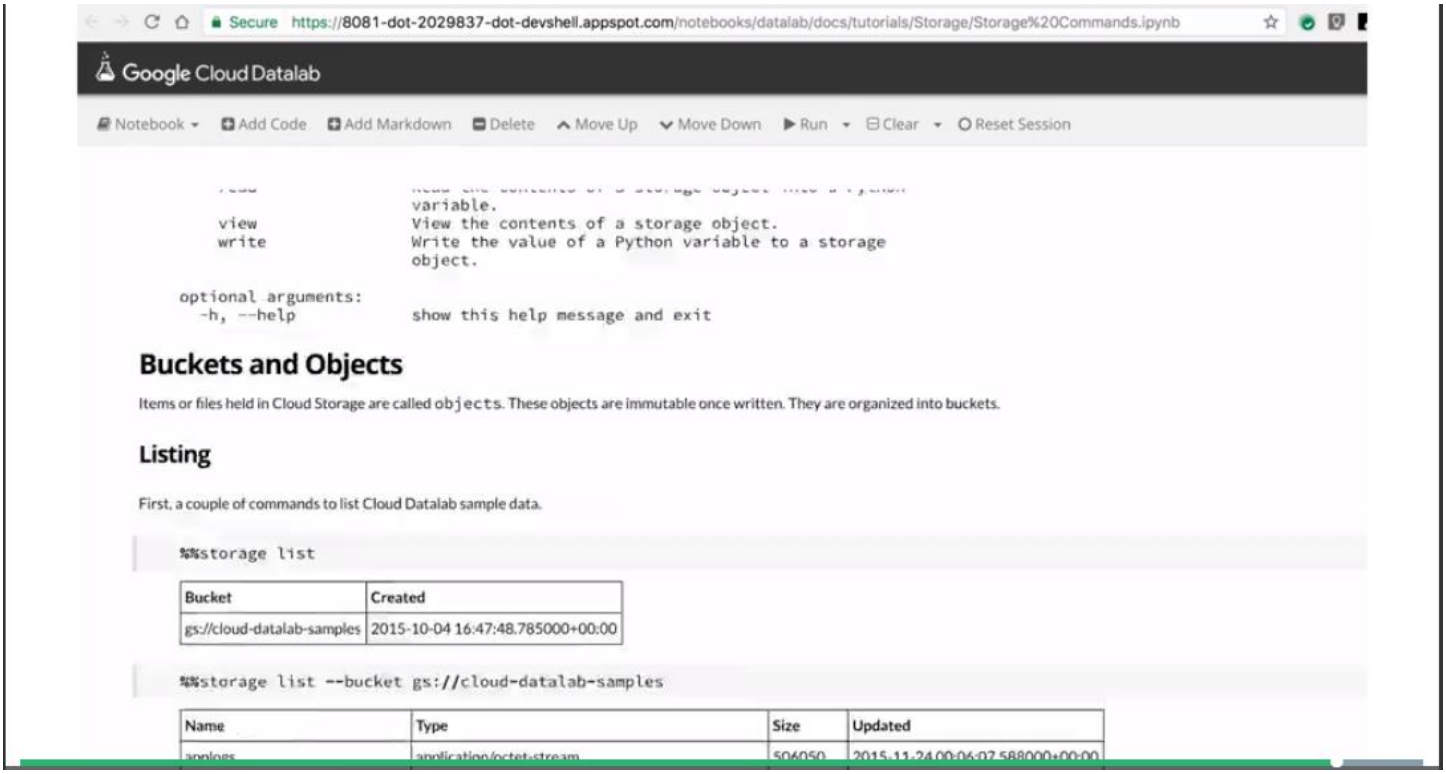
So, this is the new notebook, this is the kind of thing that I'd be working in. And it is Python, so I can do for example x = 2 and y = x + 2. print y, and then I can hit Run.

And well it's four. Surprise, two plus two is four. Great, but I can also go up here, I can add a mark down. I can move the mark down up, right? Let's see what <pre> 2 + 2 </pre> is. And then I can hit run. And you see that, let's see what 2 plus 2 is. That's basically text. I can go up here. I can add a title.

Fancy arithmetic and hit run and there we go. Right? There's Fancy arithmetic as a title With some code and this is basically a notebook that we can save. I'll save in CheckPoint. All right, this notebook is called Untitled Notebook, so I can go down here, I can click on this thing, right? And I can do a variety of things with it. There are other notebooks that already exist. So, actually not that. Other notebooks are already exist. So, if you go to Docs, there are the right sample notebooks. So let's go to the tutorials, let's go to the tutorials. And look at tutorial on how to work with BigQuery, or how to work with, let's look at how to work with the storage.

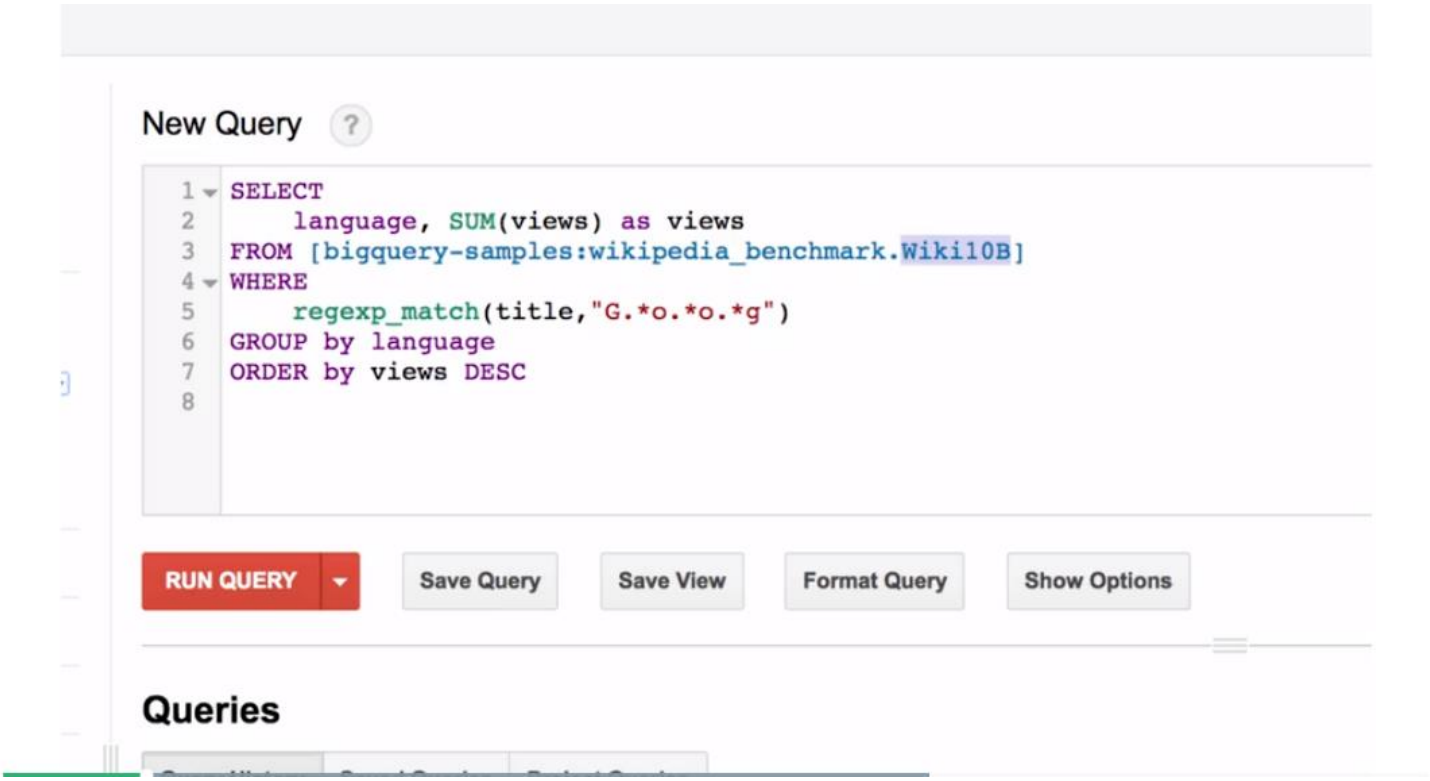


So storage there is storage campaigns iPad and notebook. And there it is, right here is an already fully pledged to notebook with text and with output and so on that has already been written and we can click on this thing and do Shift + Enter and we are running these things one at a time. All right? That's our storage list. So, that's all of the storage buckets that I have. I can say here, also that, so also notice that I'm just clicking on this and I'm hitting run. In fact, I was hitting shift/enter. But, that's shortcut to run.



For example, here is code that we could change. We can change buckets, etc. This is a way that you can share a notebook with someone else, they can change your code Rerun it so that is kind of the advantages that data lab gives you.

Warehouse and interactively query pedabytes



BigQuery is magical. But rather than me telling you what the query is, let's see it in action. Go to bigquery.cloud.google.com. In the BigQuery console, compose a query and that's basically going to be my

query. Search on 10 billion rows of the Wikipedia data. And for every row, I'm going to do a regular expression match on the title to see if it matches Google. And I'm going to see which language that mention of Google was in, and how many views there were of those mentions. We are going to look for pages in different languages that mention Google in the title and count the number of views of those. And we'll go to the Show Options, and say don't cache the results. BigQuery by default will cache the results for it for a few days, so that if you rerun the exact same query, you're not paying for it a second time. But we don't want to cheat, so we'll not cache any results. And let's go ahead and hide the options, run the query and 6.3 seconds later, 416 GB of data have been processed. And we find out that most mentions of Google that people viewed were that in English and second in Español. I mean, commons is obviously not a language. And the third was in German and Italian, followed by French, Japanese at number 8 and Dutch at number 9, etc. But the cool thing, realize, is that in 6 seconds, we were able to process nearly half a terabyte of data. This is awesome. This is magical. So, this is what we just did, we did a demo of BigQuery. This is a query. We looked at 10 billion rows of data. We did selection, then we grouped it, we ordered it and we got all that done in about 6 seconds.

What BigQuery gives you is an interactive way to analyze petabytes of data. The queries that you write can be standard SQL, SQL 2011, so it's a very familiar language to most people. You can also write User Defined Functions in JavaScript. And you can access data, CSV data, JSON data on Cloud Storage and run a query on them without actually ingesting them into BigQuery. Ingesting data into BigQuery though is very straightforward. If you have your data on disk, if it's small enough data, or if you have your data uploaded to Google Cloud or in Cloud Datastore, it's as easy as going to a web console and basically saying, here's my file. This is the destination. Here is a schema. These are the columns. These are the types. Load them in. It's very, very, very easy. And what you can do from the web console, you can also do from the command line using the BQ command. You can stream data in with Cloud Dataflow.

This is very, very helpful if, for example, you're receiving sensor data in real time, blog data time in real time, you can process them with Cloud Dataflow and stream them into BigQuery. And even as the data are streaming in, you can run queries on that data. Besides batch data on cloud storage or streaming data via Cloud Dataflow, a third option is to not load the data into BigQuery at all. Is to leave your data in raw form as CSV files or JSON files or Avro files and set up what's called a federated data source. You're basically creating a link to it and saying, here is the name, here is the schema, there is my file, so don't make a copy of that file and directly query that file.

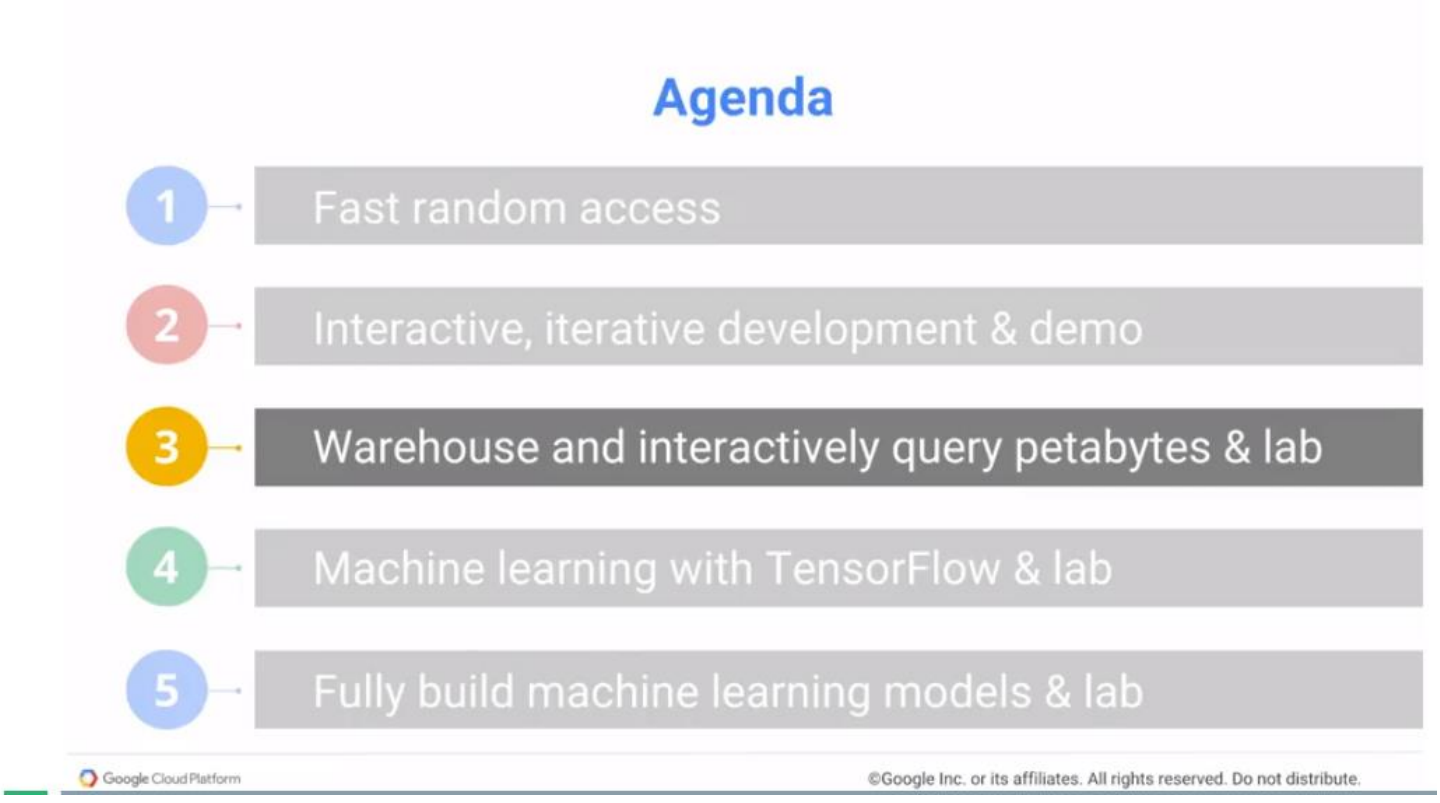
That last option there, Google Sheets, is extremely interesting. The idea is that you can have some data in a Google Sheet and query it with BigQuery. And now you are saying, wait a second, a Google Sheet, what? I probably have like 30 rows in my Google Sheet, why would I use a SQL BigQuery query to query it? Well, it's not about the data in the Google Sheet. It's about what you can join it with. You could, for example, have millions of rows, billions of rows of data in BigQuery and join it with the smaller data that you have in Google Sheets. You may have your customer data in Google Sheets, your sales data in BigQuery, and you'll be able to basically correlate that customer. Join them with this much more massive data that you have in BigQuery. Being able to join a table in Sheets with a table in BigQuery is extremely powerful.

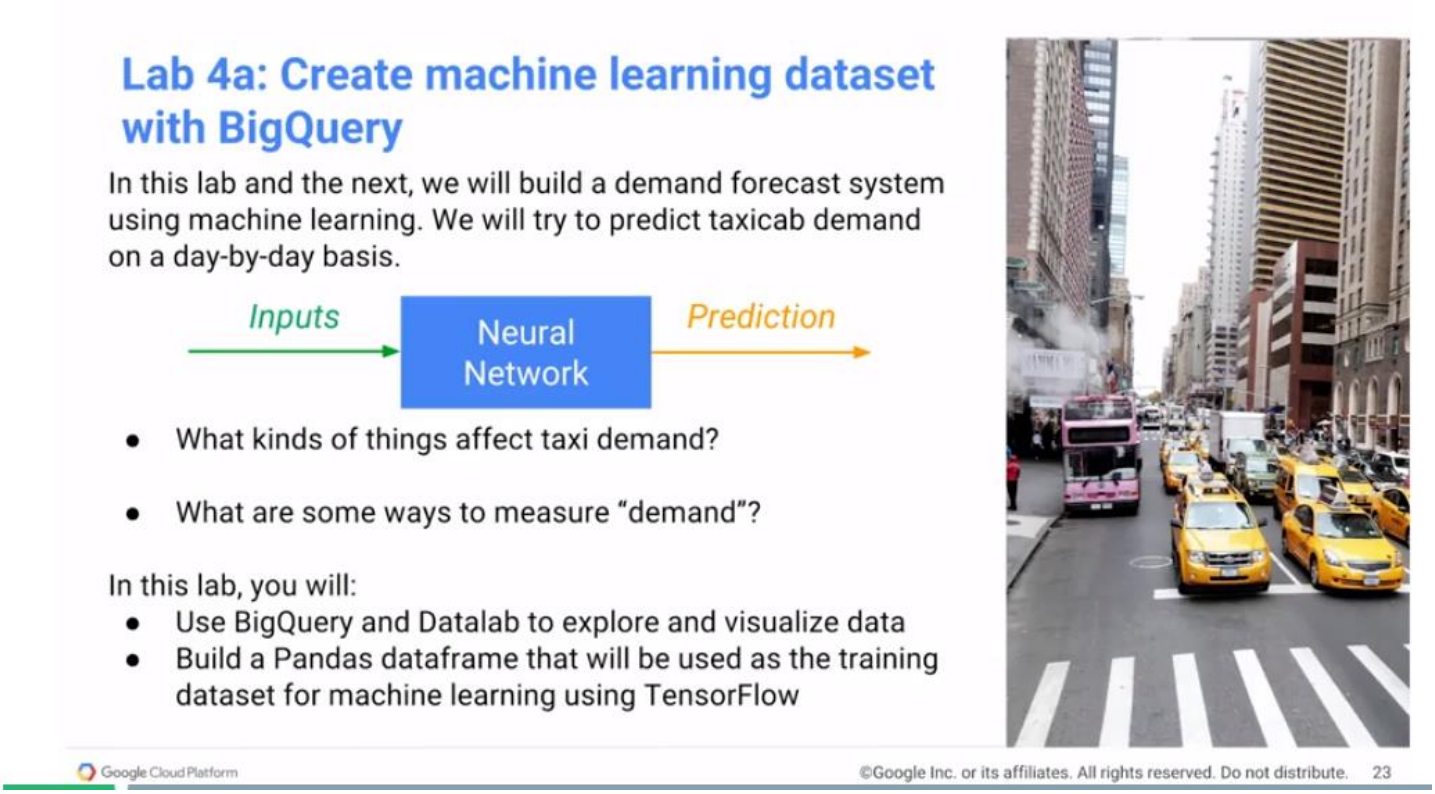
You can also write user defined functions in BigQuery, so you're not limited to SQL 2011. Here for example, I'm calling a method called urlDecode. That method is defined as a user defined function and it's written, it's implemented in JavaScript. And you can have natural JavaScript, so people have done even more very complex things like natural language processing. There's a JavaScript library that does natural language processing, and so they're able to basically write a UDF to process, for example, Stack Oveflow questions,

right? So that's free-form text. You can now process it with an NLP API and you can do queries on it. This is also a very powerful feature that makes BigQuery even more powerful than being able to run SQL queries. It's because you can now take JavaScript libraries, that can do a lot more than SQL, and combine them with the capabilities of your SQL programs. You can work with BigQuery from the console, as I just showed you. But running it from Datalab gives you another level of flexibility because Datalab is a way by which you can run Python programs. The good thing about Python is that it has really nice data analysis capabilities, data visualization capabilities. And now you can basically combine BigQuery with these very powerful graphical tools and data science tools. This is the way it works. You would basically go to a cell and you would mark that cell as a SQL cell. And that basically tells Datalab that this is not Python code that's going to follow, but a BigQuery SQL code. For example, create an SQL query named wxquery. Then in another cell, which is a Python cell, you would create a query, wxquery, which would basically link to that. Say, whenever I call wxquery, this is the SQL query that I want to run. I want to select the day of the year and all of those things and you supply a YEAR=2015. And what this does is that in this query, wherever $YEAR occurs, that's going to get replaced by 2015. If we run this query and we will get back a BigQuery result set, but we say take that BigQuery result set and convert it into a Pandas dataframe. All of the Pandas, Matplotlib and NumPy, all of those capabilities that data scientists in Python use, they are now completely available off the result of a BigQuery data set.

(https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/IfBRN/interactive-iterative-development-demo)

Warehouse Lab





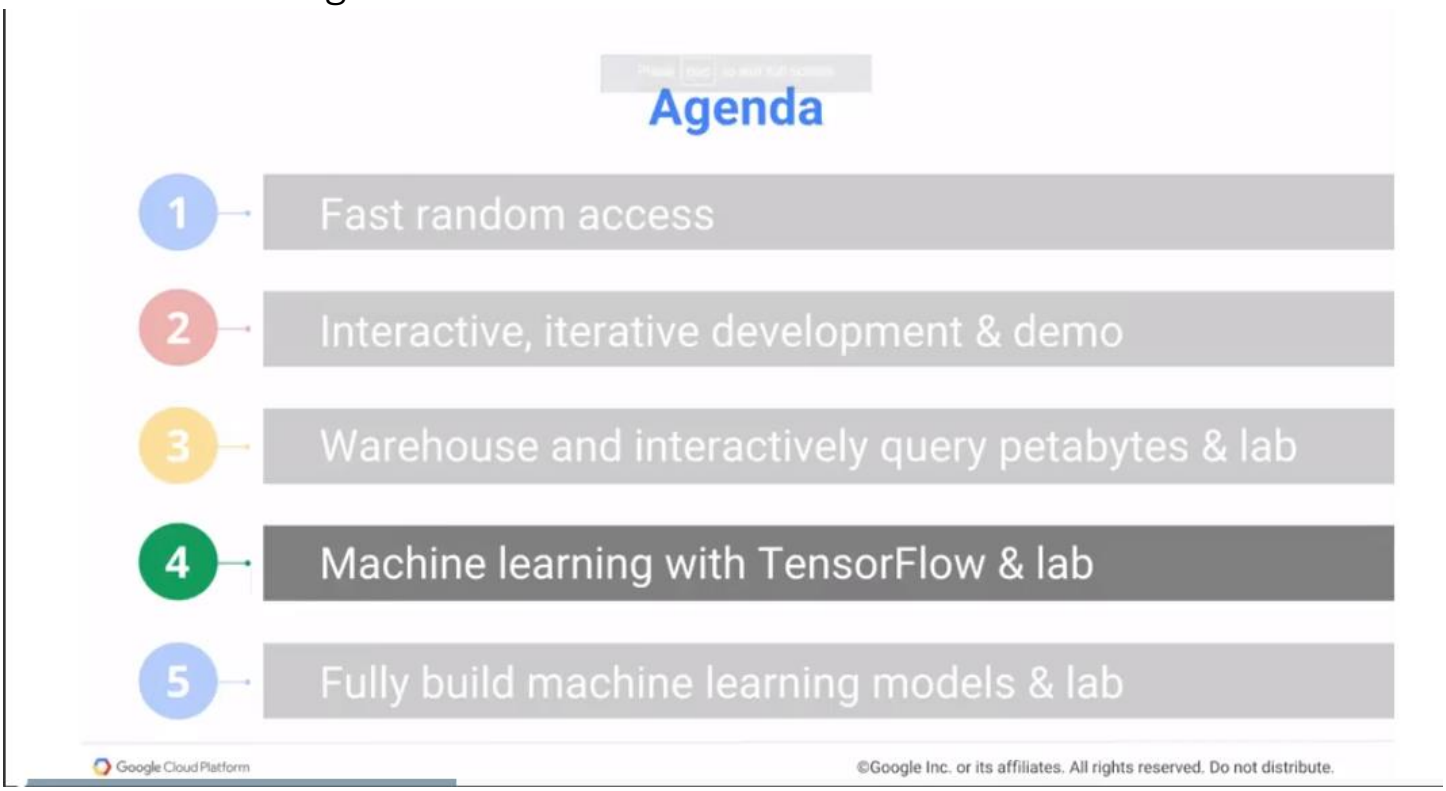
We are going to create machine learning datasets in Datalab using BigQuery. And the key thing to realize, is that each of those queries that we're doing, is actually processing a billion taxicab rides. This is the kind of thing that would take hours to do. Instead, you're going to get so use to doing them, and getting the results back in a matter of seconds.

https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/supplement/nlZ0j/create-ml-dataset-with-bigquery-lab-4a

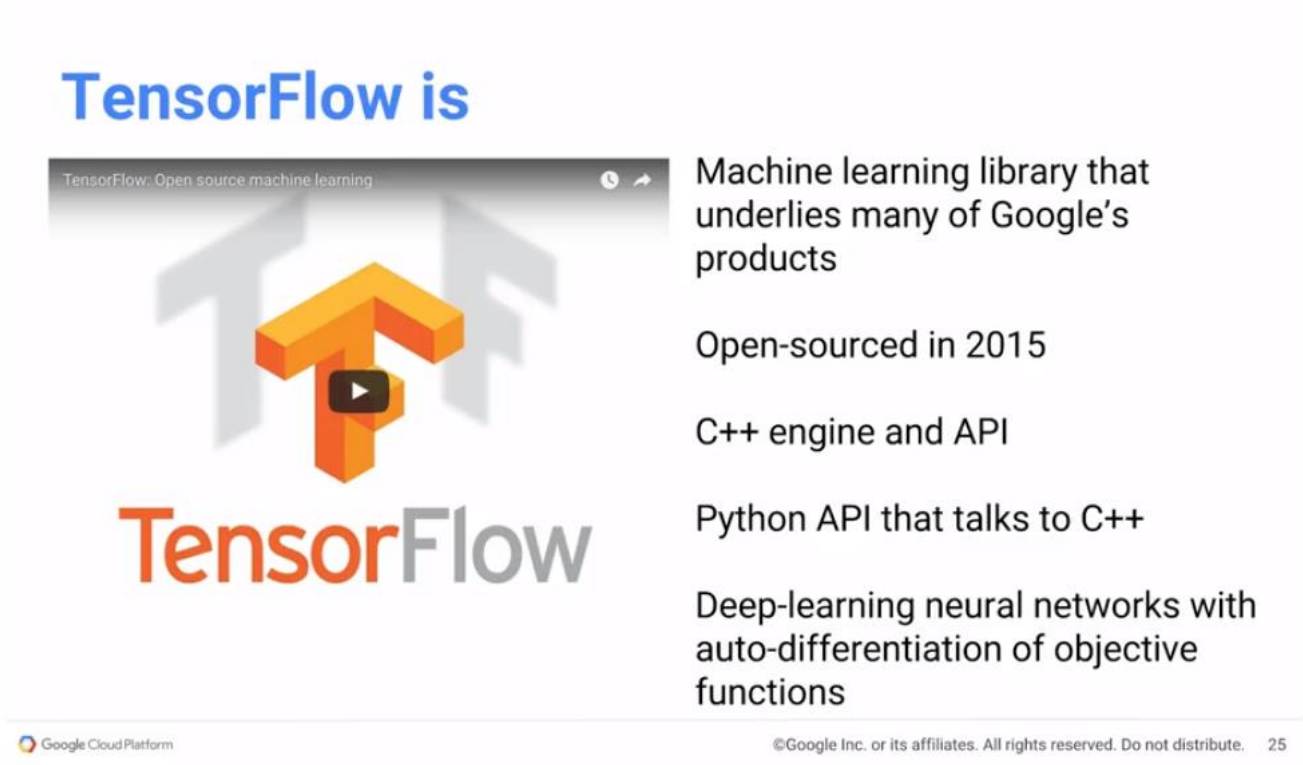
Review:

https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/8QI6L/lab-4a-review

Machine Learning with TensorFlow



To do machine learning, we're going to use TensorFlow. TensorFlow is a machine learning library that underlies many of Google's products. We open-sourced this in 2015. And TensorFlow is actually a C++ engine. The reason it's C++ is so that we can use GPUs, we can use CPUs, we can run on Android phones, etc.



(The video https://www.coursera.org/learn/gcp-big-data-ml-fundamentals/lecture/2g2lq/machine-learning-with-tensorflow walks through the graphical creation of a neural net.)

But people don't want to write code in C++, so you have an API, and that API is in Python. The Python API talks to C++, gets the job done.

TensorFlow is essentially a numerical processing library, but it has a variety of features that make it particularly good for deep neural networks and training of deep neural networks. So first of all, so because we are going to be talking about neural networks, what exactly is a neural network? Let's go ahead and look at this pretty cool site called Playground. I'm going to go in to playground.tensorflow.org. Go ahead and remove all these so that we have an idea what it is that we want to do. What it is that we want to do is that we have some data, and the data is that we have blue dots and we have orange dots. And the idea is that given a dot, we want to be able to predict whether it's orange or it's a blue. And in order to do that we have two pieces of information. We have the x and we have the y. Okay, the x here is from -66, and the y here is from -66. And given the x and y, we want to be able to predict if a dot at this point for example. x is 5 and y is 4. Is that dot going to be blue or orange? What do you think?

Well, I think you would say orange, because everything far away seems to be orange. But at this point, the background image is a prediction. And the prediction is that it's going to be blue. Everything to the right of this is going to be blue. Everything to the left of this line is going to be orange. And the way this prediction comes about is by taking the two x's, this x and this x, adding them together, and that's basically what my result is going to be. All right, so it's basically going to be a sum of the x and y with a certain weight. So -0.18 times x plus -0.28 times y. Add the two things up. If it's less than 0, it's orange, and if it's greater than 0, it's blue. And because these two weights are negative, you can kind of see that all the negative data is here and all the positive data is here. So that's basically a prediction that's pretty bad, right? The prediction is that everything here is going to be blue, everything there is going to be orange, that's not true. But let's see if we can change these weights. There's a weight here on x, there's a weight here on y. Let's say, go ahead and tune these weights to come up with a better prediction. And as you can see, it is not possible to come up with a better prediction that can linearly combine x and y to basically separate blue dots and orange dots. So let's stop this, this is going nowhere. And let's think back about this problem. Remember when I said when x is 5 and y is 4, what was the color? You can intuitively say it will be orange. The reason that you thought it would be orange was because of the distance. Everything that was close to the center was blue. Everything far away from the center was orange.

And going back to elementary school, what was the formula for a distance?

It's square root of x squared plus y squared. There is an x squared term and there is a y squared term. Let's, instead of just using x and y, let's add x squared and let's add y squared. So now we have four inputs, not just x and y, but x squared and y squared. Given these four inputs, let's come up with weights for all four of these in such a way that it separates blue dots and yellow dots. Let us start, and lo and behold, that's my prediction now. The prediction is that everything inside of this is going to be blue, and everything outside of that is going to be orange, and that seems to capture the data pretty well. It captures our intuition of what this data say very well. This idea is called feature engineering. One of the ways that we can improve our machine learning models' predictions is to kind of get human insight into the problem. And the insight that we had was that this was based on distance. We knew that distance involved x squared and y squared. We threw that into the network and we said, train yourself with weights. But let's say we don't have that insight.

All I have is x and y and I want to basically do this prediction. So rather than do feature engineering, another thing that we can do is that we can create a neural network. I'll create a layer of these and what it's doing is, this guy is x and y added together, and to that, I'm applying some function. I could do rectified linear unit, tan hyperbolic, sigmoid, whatever. It doesn't really matter which one we choose, let's just pick Tanh. I'll do Tanh there, Tanh here, Tanh here, Tanh here, Tanh here, so five different Tanhs. Add them all up. Why did I pick five? Who knows? I just picked five. Why did I pick Tanh? Who knows? I just picked Tanh, right. I just picked something. I have a neural network. I'll basically go ahead and train it. It is now no longer just two sets of weights. It's two weights here, two weights here, two weights here. So that's ten weights here plus five weights here. Therefore, 15 weights that we get to basically tweak around, so go find me a set of weights that capture this data.

Training can take some time, and it basically comes back with everything inside this triangle like shape is going to be blue, and everything outside it is going to be orange.

Is that reasonable?

It is not going to be perfect, but it's a pretty reasonable approximation to this data, and it'll help you predict with good accuracy how well this is going to do. And that's the point of a neural network. The idea is to basically capture what the data are in such a way that you can do the prediction later on.

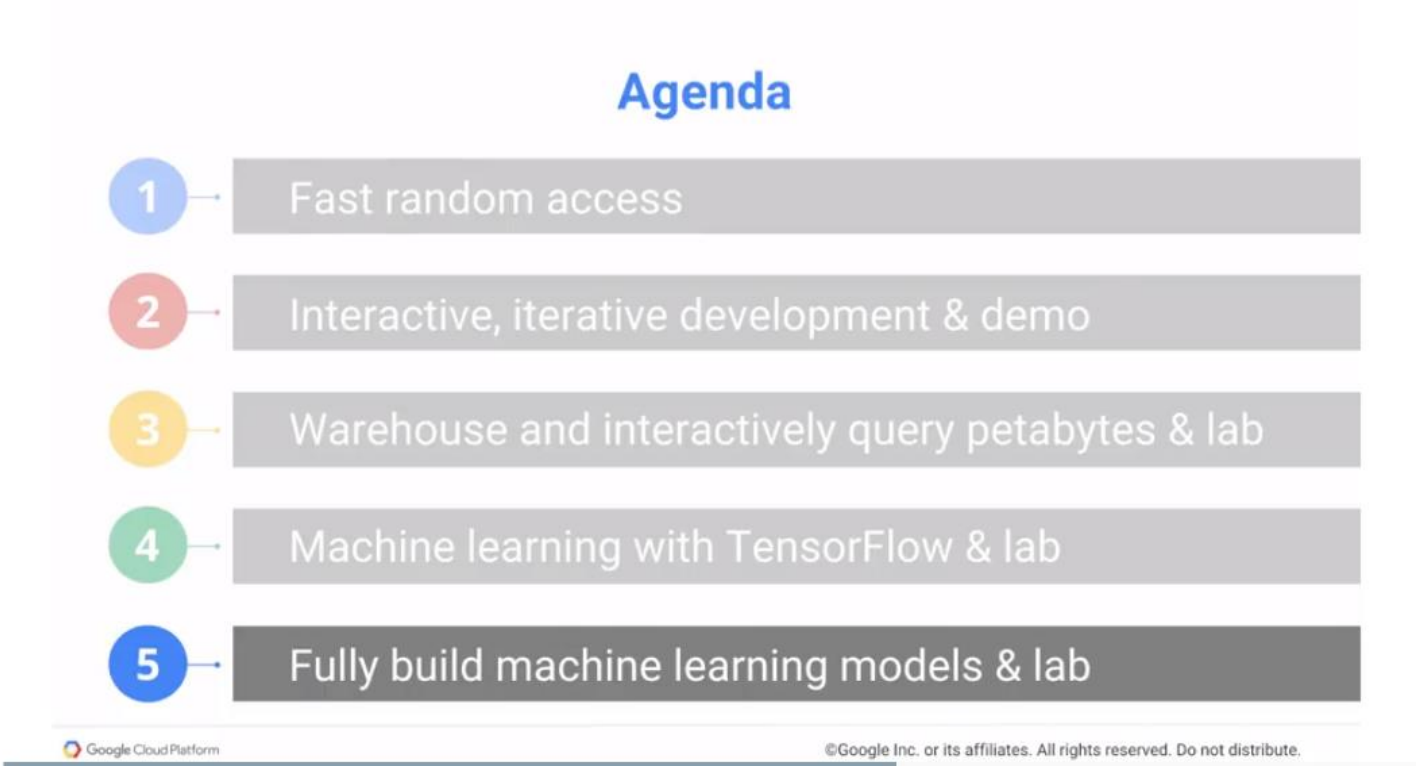
Notice that what we did here was rather than take our human insight, we were able to use a neural network to essentially get at a good enough end result. This is a neural network with one hidden layer, that's one layer of these neurons, so there are five of those nodes. We could also create extra hidden layers.

So now we have ten sets of weights here, ten sets of weights here, and then five sets of weights, so that's now a whole bunch more weights, right? Now this is ten, but each of these has five, so that's 5 times 5, that's 25, so 10 plus 25 is 35, plus 5 is 40 weights. We now have a model that's a lot more complex. And you basically get, again, reasonably good results.

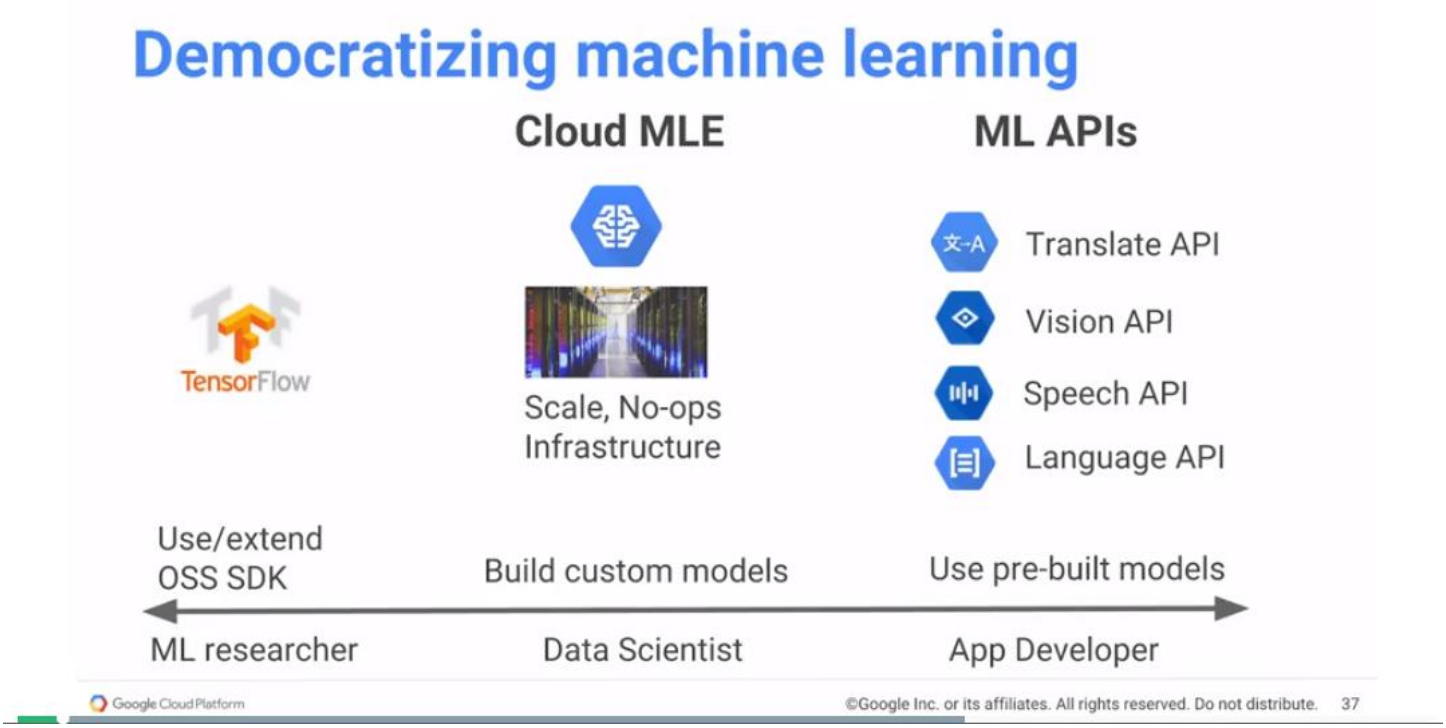
And the basic rule of thumb is to go with the simplest possible network that gives you good enough performance. In this case we would go with just one hidden layer, but here we have to choose a number of nodes, and let's say we start with two nodes, and let's see, does this do well? And it turns out that no, two nodes are not enough for this problem. It doesn't do very well. There are all these errors here in terms of capturing them. Let's stop that. Let's say we add a third node, right? And then we say, start this.

And with three nodes, it seems to be fine. Except maybe some of these guys. The ones at the edges are probably not completely right, but it's close enough. In this situation, I would probably go with just three nodes, right? That's the simplest neural network that gives me good enough performance.

Fully build machine learning models



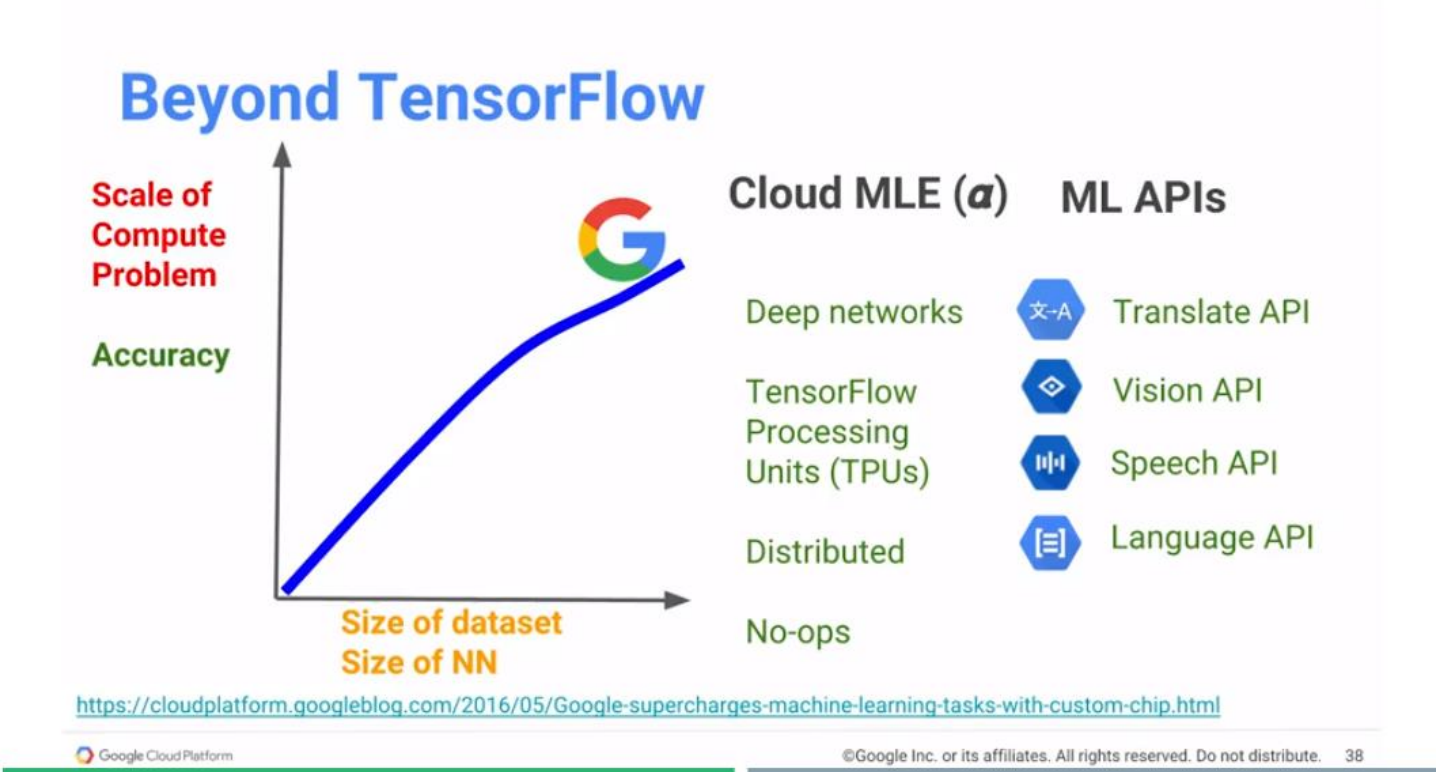
In the previous section we looked at doing machine learning from scratch using TensorFlow. But, there is a full spectrum of machine learning models available. You would use TensorFlow if your machine learning researcher interested in extending the open source SDK If you're interested in creating new machine learning models for research, etc.



On the other hand, if you are in industry as a data scientist and you want to build a machine learning model on your data set. And the machine learning model that you're building is something that's pretty well understood, you would want to use Cloud ML Engine. This is basically going to give you no-ops, so that you are not in the business of managing infrastructure, to be able to do machine learning at scale over real world data sets. In the previous lab, we were applying TensorFlow on a very small data set just about 700 rows long because that's all that would fit. The real world, you would do it on a large data set, and you would need much more powerful infrastructure, and rather than manage all that infrastructure yourself, you would typically just submit the job to Cloud ML and have that taken care of.

The third option in the spectrum is this idea that there are variety of machine learning models that already exist and it's kind of painful to reinvent the wheel. What if you are investigating how to transcribe audio – you don't build your own speech model. Just use a speech model that has already been developed because how are you going to get the amount of data that's needed to train that model? The machine learning APIs are all about taking pre-built models and incorporating them into your applications. You are using machine learning, but you're not training a machine learning model, when you use ML APIs.

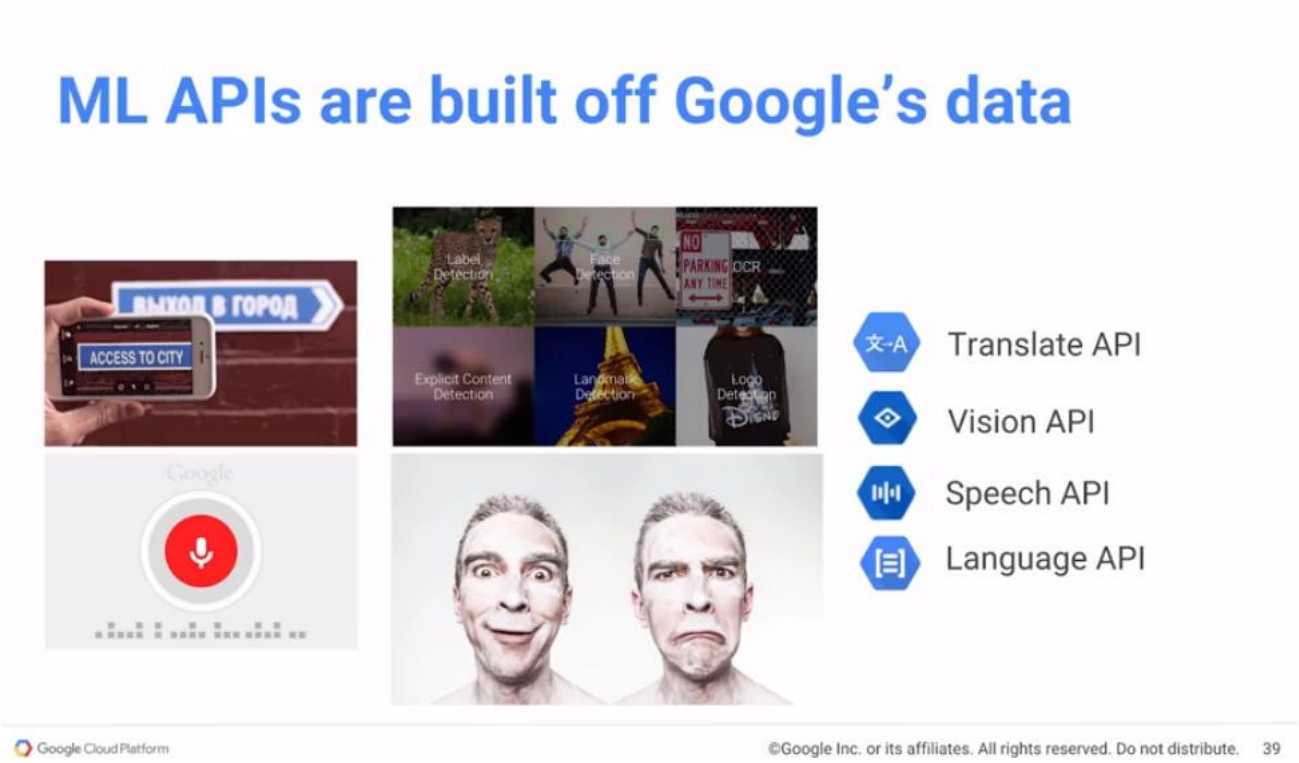
And the reason that you want to use the ML APIs is that the quality of a machine learning model increases with the amount of data that you have in it. So as the amount of data increases, your accuracy goes up. And the reason your accuracy goes is up, as you increase the amount of data, is because you can use larger and larger and larger newer networks you also have a huge computer problem. And that the scale at which Google operates, very deep networks, to use TensorFlow processing units, we do distributer training and we can do no-ops. Ccloud ML gives you all these things if you have a data set that's large enough to solve the problem.



2:45

You tend to have that kind of a data set for things that are relevant to your business. But maybe not for something like classic image recognition or classic speech recognition and natural language processing. This

is where you basically want to take advantage of the amount of data that Google has collected in order to build our vision models and our speech models and our language models. And the good thing is that the speech and vision and language models that Google has created are available to you as machine learning API's.



The machine learning APIs and other words are built off Google's data. If you ever use the Android app, where you can point the application, add a foreign language sign and get it translated. That app uses translation, it uses optical character recognition and OCR, Optical Character Recognition is part of the Vision API and translation is part of the Translate API. Similarly, if you're being on Android and you've done a voice on Google or you use Google maps and you've done a voice search, you've basically using the capability that goes into the speech API. Being able to take audio and turn it into words so you can do something with those words. Or the sentiment analysis of things like reviews. Is it a positive mention? Is it a negative mention? Those are the kinds of things that these models already exist. Google has built them off of Google's collected data. We can take advantage of it with the ML APIs.

Lab: Machine Learning APIs

https://codelabs.developers.google.com/codelabs/cpb100-translate-api/

Module Resources

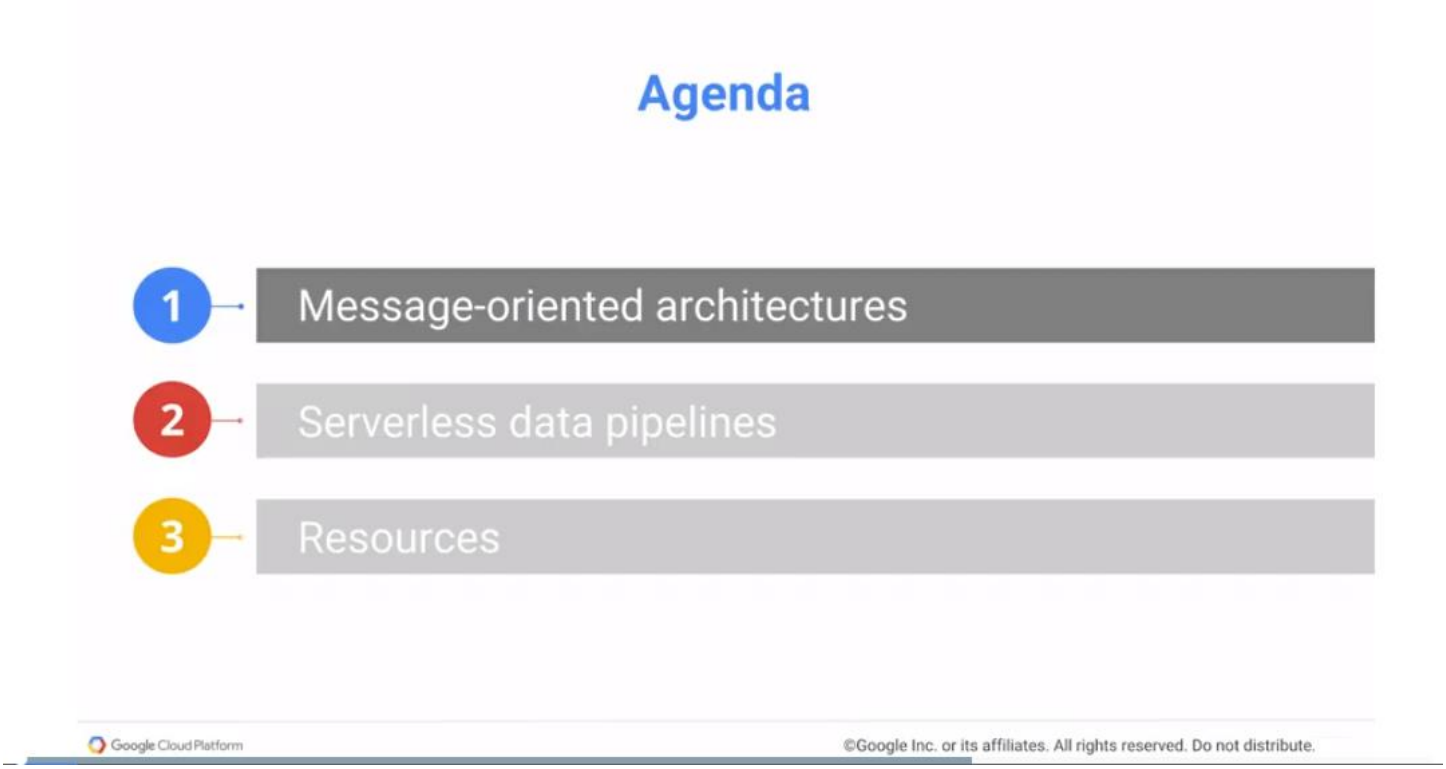
- Cloud Datastore: https://cloud.google.com/datastore/
- Cloud Bigtable: https://cloud.google.com/bigtable/
- Google BigQuery: https://cloud.google.com/bigquery/
- Cloud Datalab: https://cloud.google.com/datalab/
- TensorFlow: https://www.tensorflow.org/
- Cloud Machine Learning: https://cloud.google.com/ml/
- Vision API: https://cloud.google.com/vision/
- Translate API: https://cloud.google.com/translate/

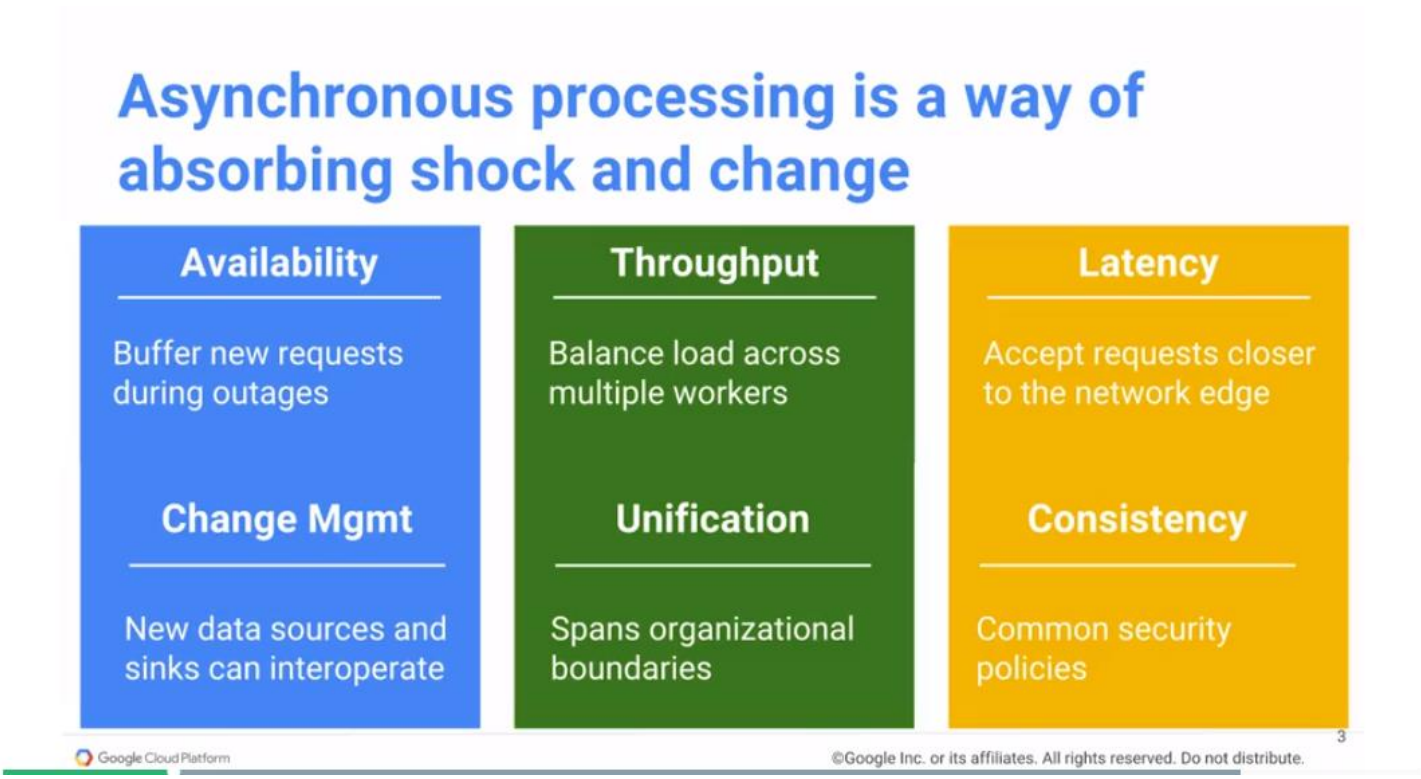
- Speech API: https://cloud.google.com/speech/

MODULE 5: DATA PROCESSING ARCHITECTURES: SCALABLE INGEST, TRANSFORM AND LOAD

In this last module of the class, we do a very quick overview of message-oriented architectures and serverless data pipelines. We look at serverless data pipelines, in particular cloud data flow, with much more detail in the course on serverless data analysis, which is part of the data engineer track. In this module though, we will look at it very, very briefly, very quickly so you know what it is and you know where to find it if you need it.

Message-Oriented Architectures

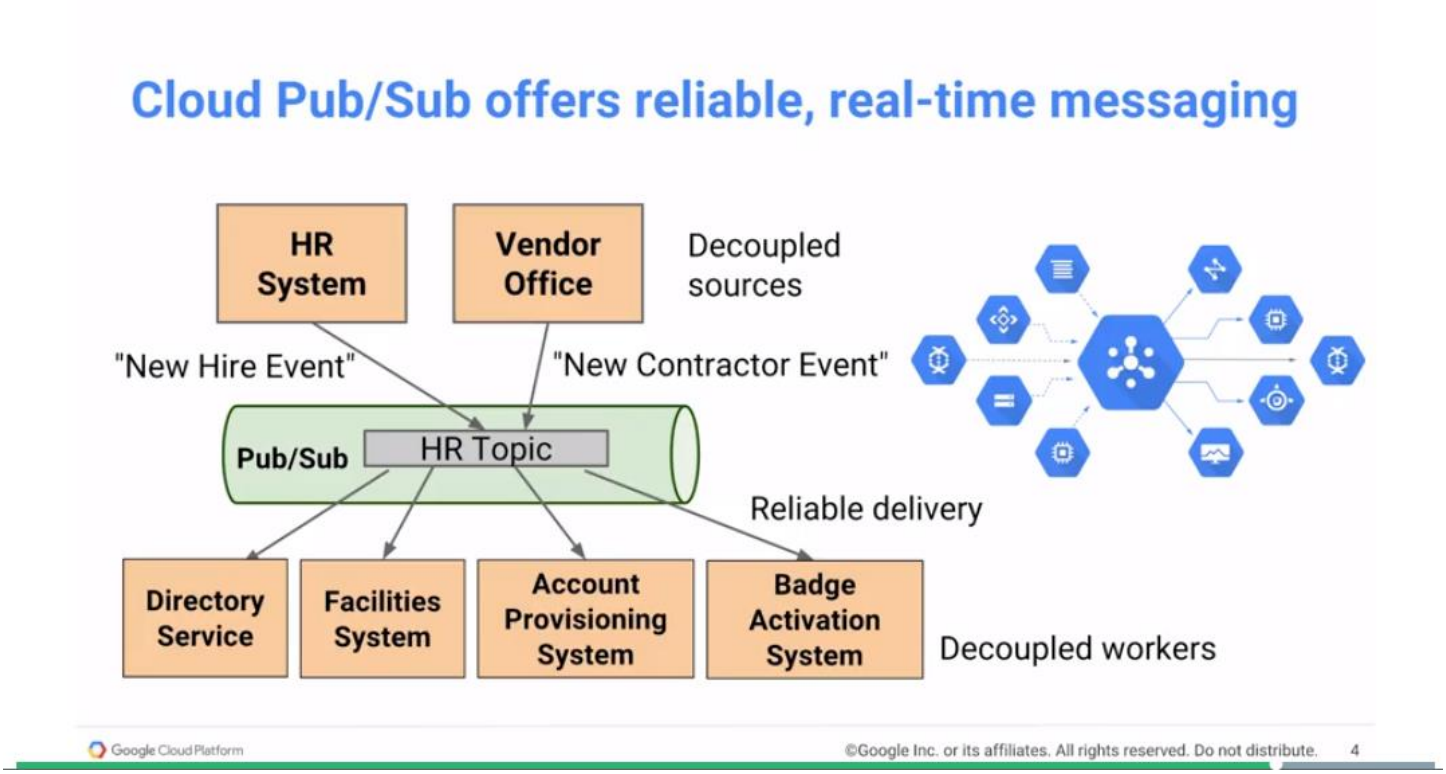




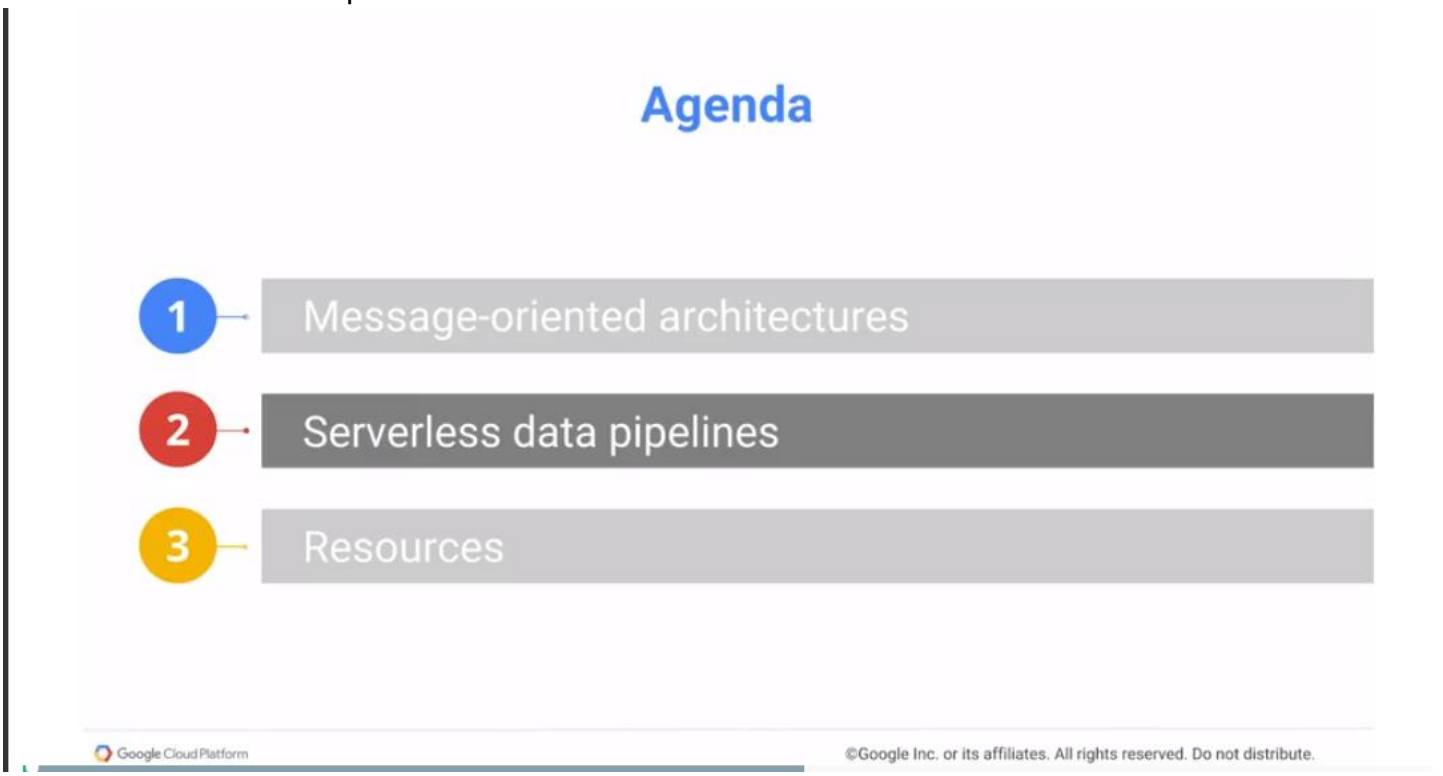
Asynchronous processing is a great way to absorb shock and change. For example, if you have an application and you've built this application for 100 users. And in order for it to handle 100 users, it's doing all of these things that it needs to do, and then you basically get a spike in usage. Instead of getting 100 users, you now suddenly have 4,000 users. At this point you have two approaches, one is for your application to just crash. The other is for you to kind of queue things up, such that people get responses but they're a little bit delayed.

That's what asynchronous processing helps you do. The idea is that when somebody submits a job, it goes in. And rather than giving them a response immediately, you basically have the receiving code and the processing code separated. And they're separated by a messaging system. In other words, new requests come in, they go into a message queue and then you have consumers of this message queue actually processing these requests. Asynchronous processing is a very common design paradigm if you need to build highly available systems. Any request that's sent to the system will get processed. Well, what happens if you have an outage? Well, if you need high availability, then asynchronous processing is one solution. Another thing is to balance load across multiple workers - balance high throughput. And the third reason to do that is to reduce coupling. You may have the people producing images separate from the people consuming those messages. Rather than having the system producing messages held up by the fact that the people receiving the message aren't yet able to accommodate a new library or new way of doing things. Separate them and reduce the coupling by using an asynchronous system, by using a message queue. This allows more agility within your organization. It's a great way to reduce latency so that you can accept requests really close to the network edge. The idea being that the person making the request doesn't have to make the request all the way up to the service. Instead, they can make a request to the closest point on the network. Then the request can travel on an internal Google fiber all the way through. It is a good way for you to manage consistency. Such that you can apply the exact same security policies to message processing, regardless of where the message comes from. Whereas, if you're relying on the client to process these messages, then you may have some issues.

On GCP message-oriented architectures are implemented with Cloud Pub/Sub. Cloud Pu/Sub offers reliable, real-time messaging that's accessible through HTTP, You can have your HR system basically sending a new hire event or vendor office sending a new contractor event and these are decoupled sources. They know nothing about each other, but they publish their events to a common Pub/Sub topic, the HR topic. And then you could have multiple consumers, each of whom has a subscription to this HR topic. Some of these subscriptions could be pull subscriptions. In other words, whenever the system, the client is ready to process a new message, it goes ahead and asks, are there any new messages? Or it could be a push in which, basically, the client system says, call this endpoint whenever there's a new message for me. And that new endpoint would get called by Pub/Sub whenever there's a new message. In this way, Pub/Sub can give you reliable delivery, you can get completely decoupled workers. Cloud Pub/Sub is a good way to handle asynchronous processing.

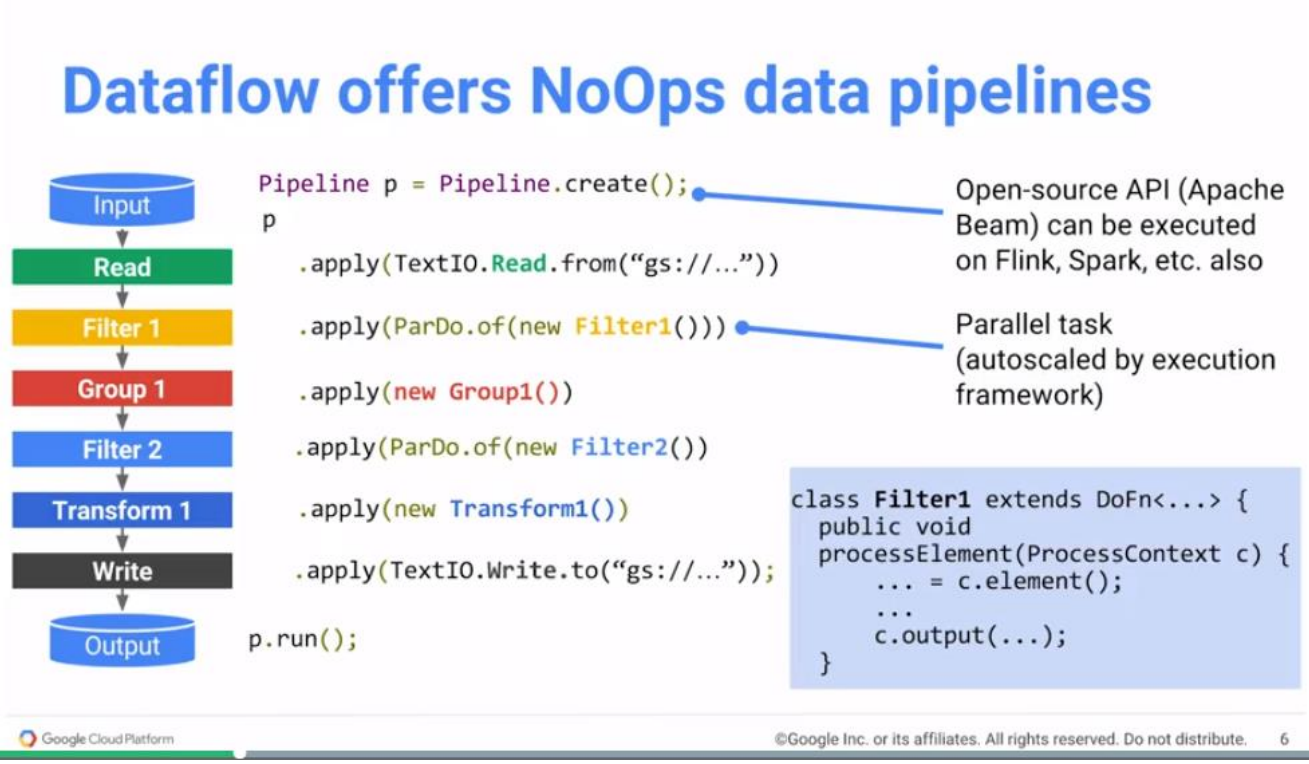


Serverless Data Pipeline

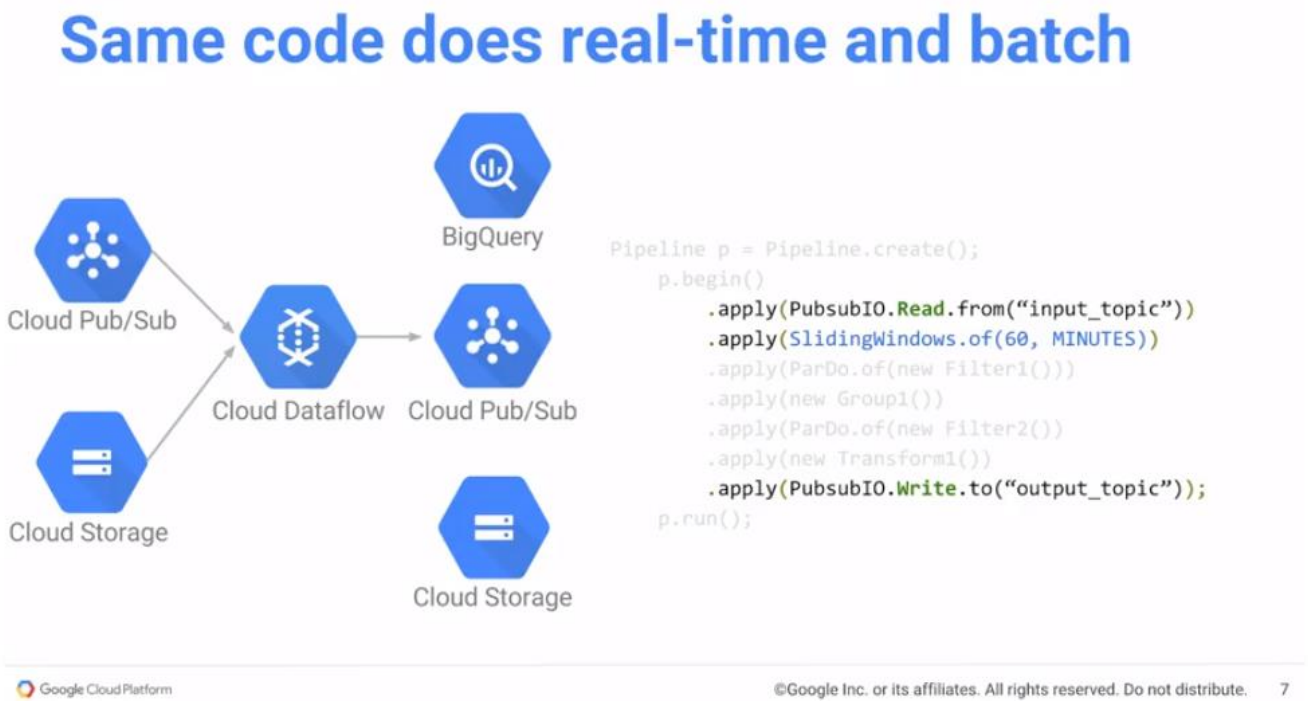


The other architecture that we want to talk about is this whole idea of building data pipelines.

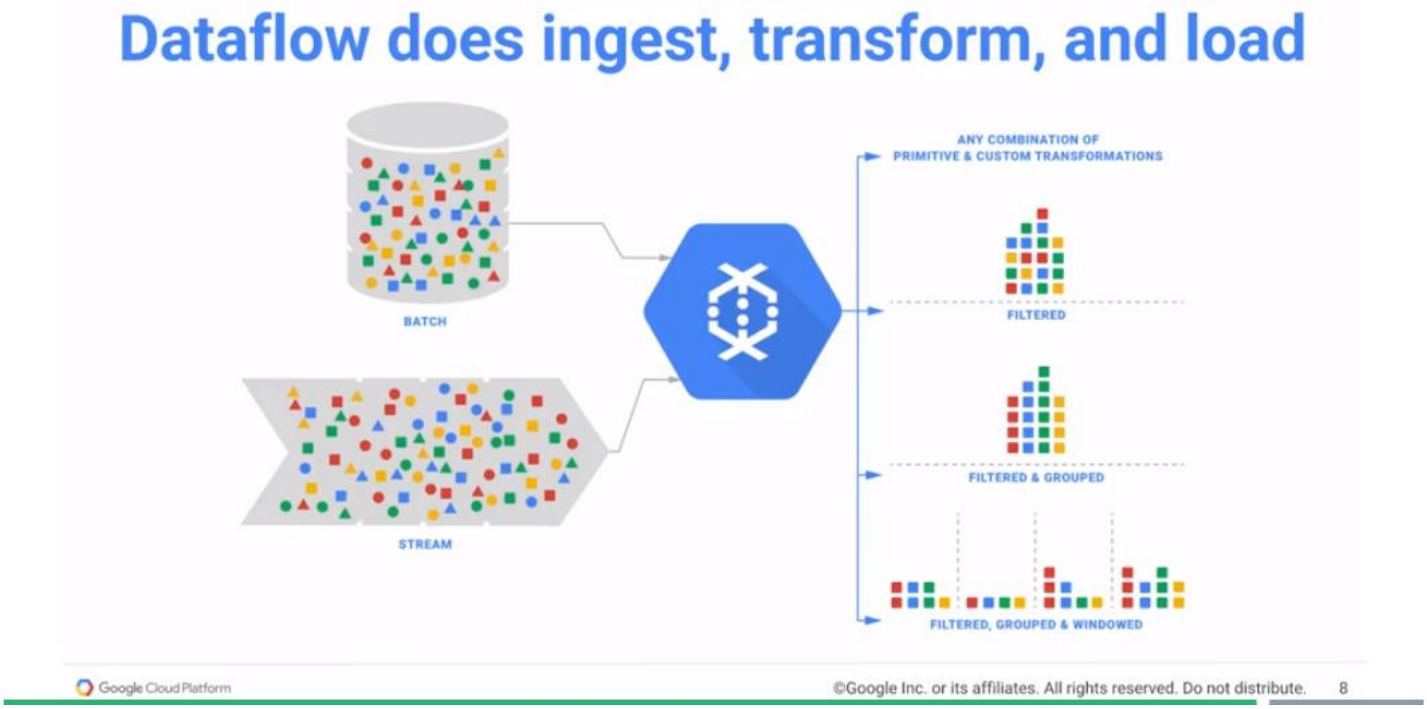
Dataflow, Cloud Dataflow, is the execution framework for Apache Beam pipelines. Apache Beam is an open-source API that lets you define a data pipeline. This API here that we are showing you is Apache Beam. And you're basically creating a pipeline, reading some text, and the text that we're reading is from Cloud Storage.



We are reading it line by line and for every line, we're applying Filter1. The result of that goes to Group1, the result of that goes to Filter2, the result of that goes to Transform1. And then the result gets written out to another Cloud Storage. The neat thing is, Filter1 is wrapped into a ParDo, parallel do. And what this means is that every line that's read from the input source basically is processed by Filter1. But this is done in parallel across tens of machines, hundreds of machines, as many machines as you need. And you don't need to create these machines beforehand. Dataflow manages the provisioning of these machines, the auto-scaling if necessary, of your pipelines. Such that Filter1 just happens at scale, completely distributed and then everything comes streaming back into Group1. And then the results of Group1 again get done in parallel using Filter2. And then Transform1, and then it basically goes into this sync. Now, each of these things, Filter1, Group1, Filter2, Transform1, these are all classes that you write in Java. You can do Beam in Java or you can do it in Python. I'm showing you Java here, but you can do it in Python. And the code that you write, the Beam pipeline is an open source API. And you have other executors besides Google Cloud Dataflow. So you can execute it on fling, on Spark, and a variety of other things. With Dataflow, what you get is a completely NoOps data pipeline. A job can be submitted to the cloud and it is executed. In this case, all of this processing happens on some text input, and some text output gets written out.



A very neat thing about Dataflow is that it can have that intermediate processing be identical. Filter1, Group1, Filter2, Transform1. But you can swap out the input. Rather than reading from text, in this case, I'm now reading from Pub/Sub. I'm now reading messages that come in into a particular topic. And remember that I'm doing grouping. Grouping is an aggregation operation, so doing grouping where? If you're accounting or a sum, you need to basically do these things within a particular window. In this case I'm saying the window that I'm going to do all of these things on is over a period of 60 minutes. I'm applying a 60 minute window, the default of every is 1 minute. I'm applying a 60-minute window every minute. And every minute, I'm basically writing out a new message to the output topic. That consists of filtered, grouped, Filter2 transformed data. That's a new message that now goes to Pub/SubIO. I can change the second PubSubIO, the output sync, to be BigQueryIO. And then I'll be writing to BigQuery, I'll be writing table rows to BigQuery. And then now other dashboards etc., can basically run BigQuery queries on the streaming data as it comes in. And that's the way that you generally get real-time insight. The cool thing is the processing that you do. Whether the data comes from Pub/Sub or the data comes from Cloud Storage, the processing that you do in Dataflow remains the same.



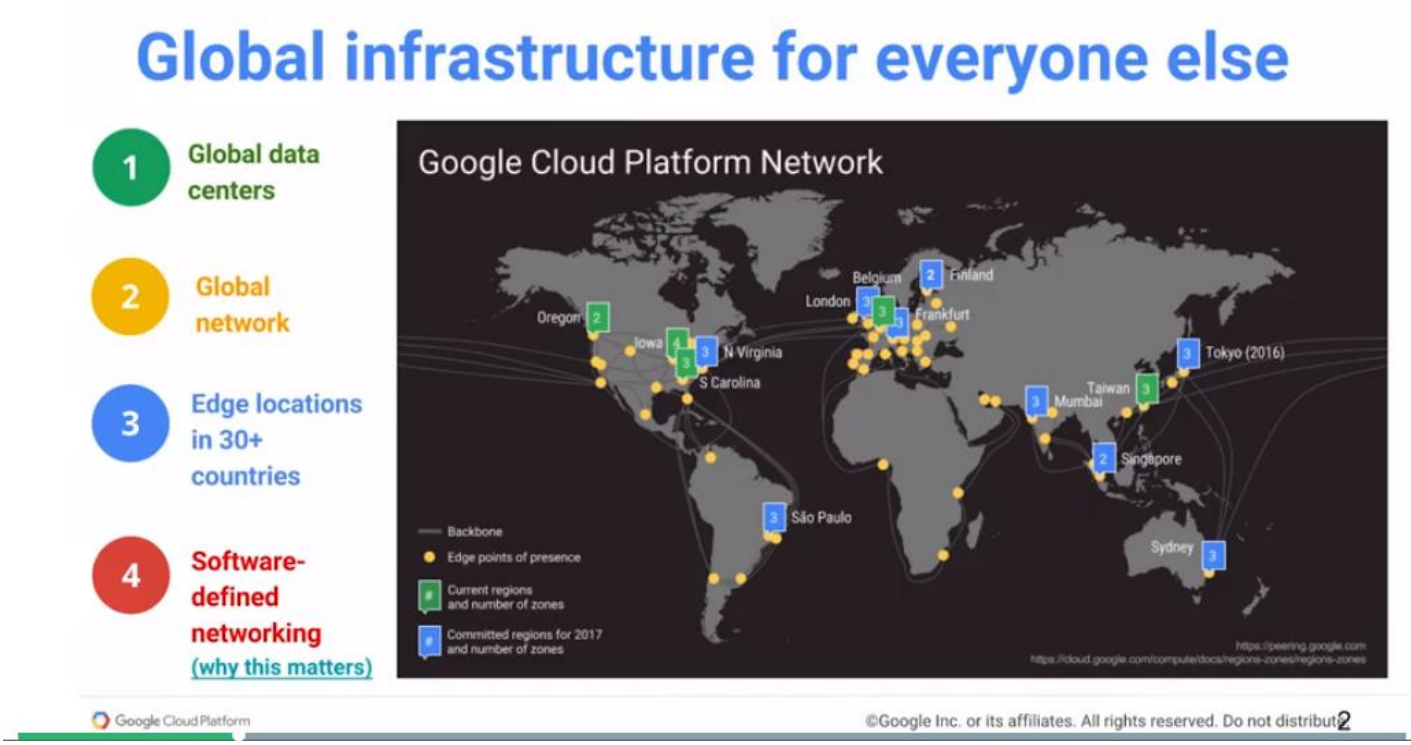
Dataflow helps you do ingest, helps you do transformations. Help you do load, so it can do filtering, you can do grouping, and you can do windowing. Dataflow is where we see a lot of data pipelines migrating. Because you really want to be able to process historical data and real-time data in and identical way. That is the only way that you'll be able to build a machine learning pipeline. For example, that is trained on historical data that operates on real time arriving data. And Dataflow is essential glue in order to be able to do this kind of data processing.

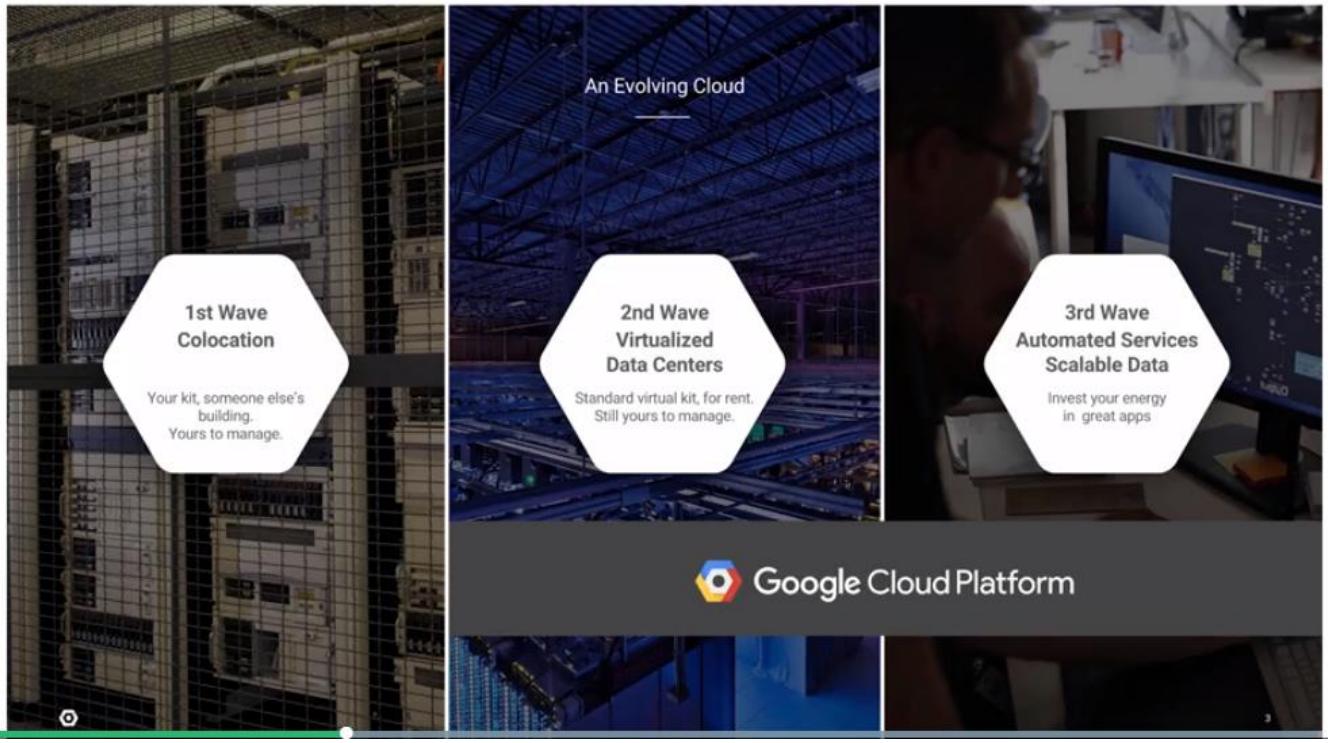
Module 5 Resources

- Cloud Pub/Sub: https://cloud.google.com/pubsub/
- Cloud Dataflow: https://cloud.google.com/dataflow/
- Reliable task scheduling on Google Compute Engine: https://cloud.google.com/solutions/reliable-task-scheduling-compute-engine
- Real-time data analysis with Kubernetes, Cloud Pub/Sub, and BigQuery: https://cloud.google.com/solutions/real-time/kubernetes-pubsub-bigquery
- Processing logs at scale using Cloud Dataflow: https://cloud.google.com/solutions/processing-logs-at-scale-using-dataflow

MODULE 6: SUMMARY OF GOOGLE CLOUD PLATFORM, BIG DATA, AND ML

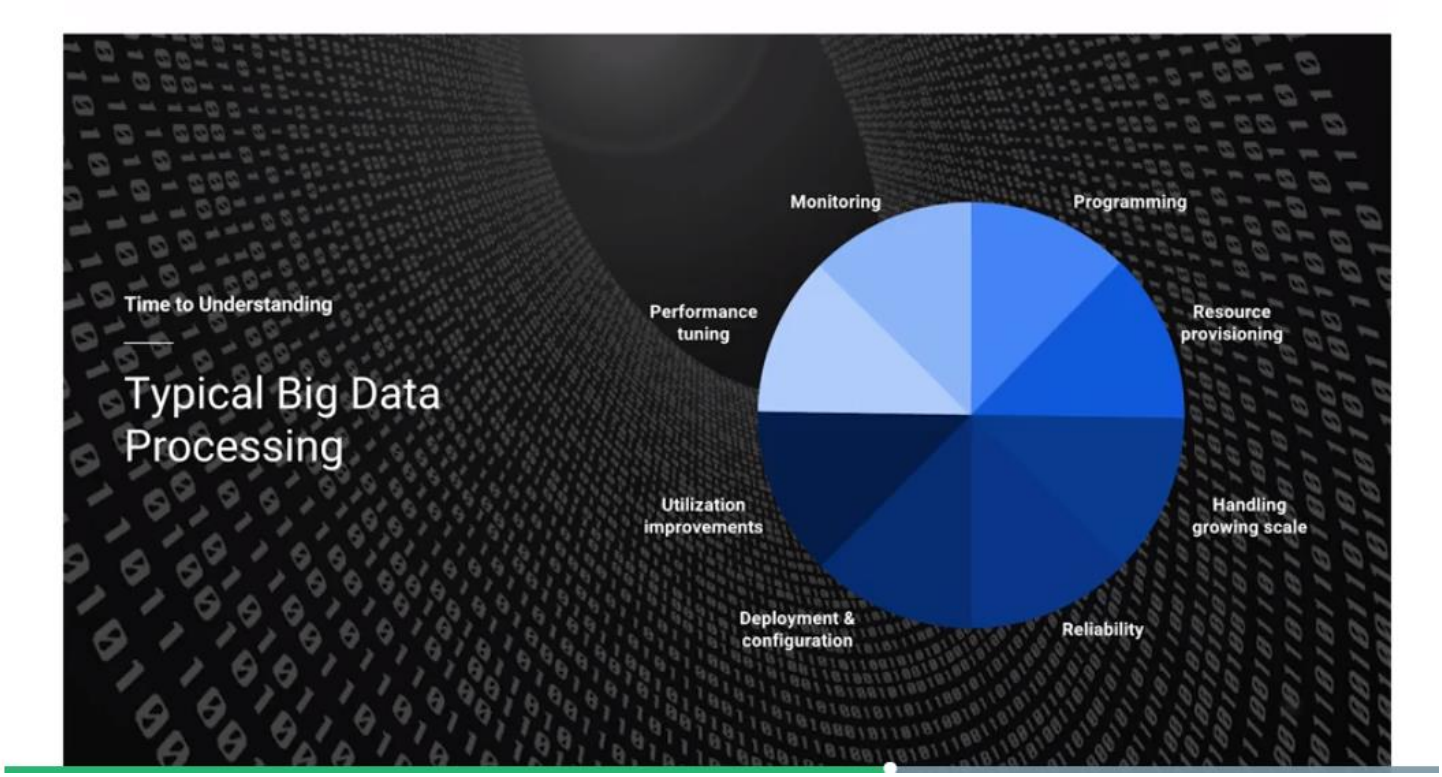
In summary then, we've looked at google infrastructure. We basically looked at how google provides global infrastructure in terms of global data centers at global network, Edge locations in a variety of countries, software-defined networking and so on, such that you can build applications similar to the way we do, used on the GCP platform.





We reviewed how the Cloud is evolving, where you may have been doing everything on location, on-premise, and migrated to a virtualized data center. And if you're using Cloud at the level of provisioning virtual machines and spinning up a VM every time you need to run a job, you're essentially doing the second wave of cloud, but you're still managing these virtual machines, this infrastructure. In this course we encourage you to look beyond managing your own machines to the third wave of completely elastic, automated services and scalable data. The idea being, whether you are writing SQL queries or BigQuery, or data pipelines or data flow, or you're doing machine learning, models with Cloud Machine Learning, you are using no ops auto-scaling

services on GCP, so that you are focused on the business applications, and Google's focused on the infrastructure.



So the point of this is that typical big data processing involves programming, provisioning resources, handling growing scale, working on how to manage reliability you're deploying, configuring, looking at how much of your machines are getting used, figuring out how to optimize the use of those machines, looking at tuning off the performance, monitoring these machines, and in the time that you have left, creating new features and programming new things. On GCP though, working with Big Data is programming. This helps you focus on insight, not on managing and provisioning infrastructure.

In summary, GCP offers you ways to spend less on ops and administration, incorporate real-time data into apps and architectures, apply machine learning broadly easily, and as an end goal, create citizen data scientists, so that everybody in your organization can work with data. That should be your goal and that's something that GCP can enable for you.

Module 6 Resources

- Big data and machine learning blog: https://cloud.google.com/blog/big-data/
- Google Cloud Platform blog: https://cloudplatform.googleblog.com/
- Google Cloud Platform curated articles: https://medium.com/google-cloud