Hybrid ML Systems

Lak Lakshmanan

Hi, I'm Lak and I lead the team that is putting together this course and specialization.

In this module, we will look at building hybrid machine learning models.

Learn how to...

Build hybrid cloud machine learning models

Optimize TensorFlow graphs for mobile

You will learn how to:

- Build hybrid cloud machine learning models
- And how to optimize TensorFlow graphs for mobile

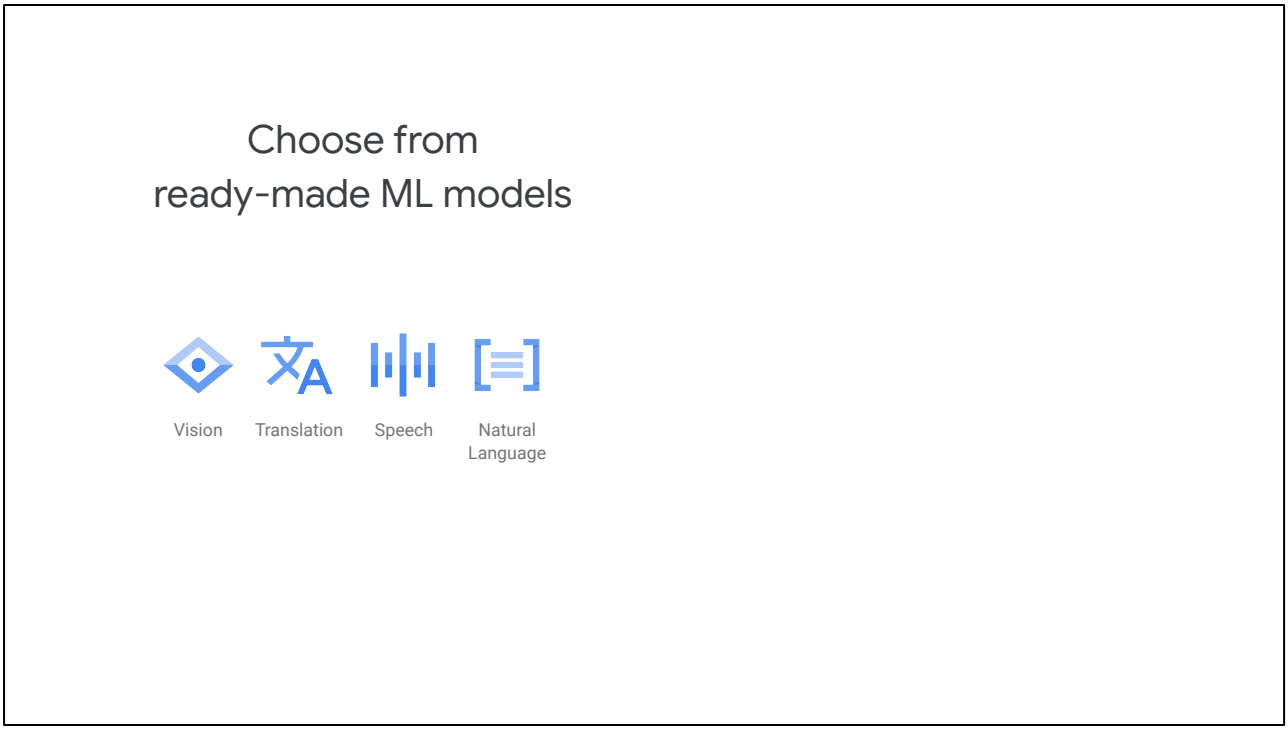
Agenda

Kubeflow for hybrid cloud

Optimizing TensorFlow for mobile

Let’s start by discussing a technology called Kubeflow which helps us build hybrid cloud machine learning models.

But why are we discussing hybrid in the first place? Why would you need anything other than Google Cloud?



Google Cloud is a great place to do machine learning.

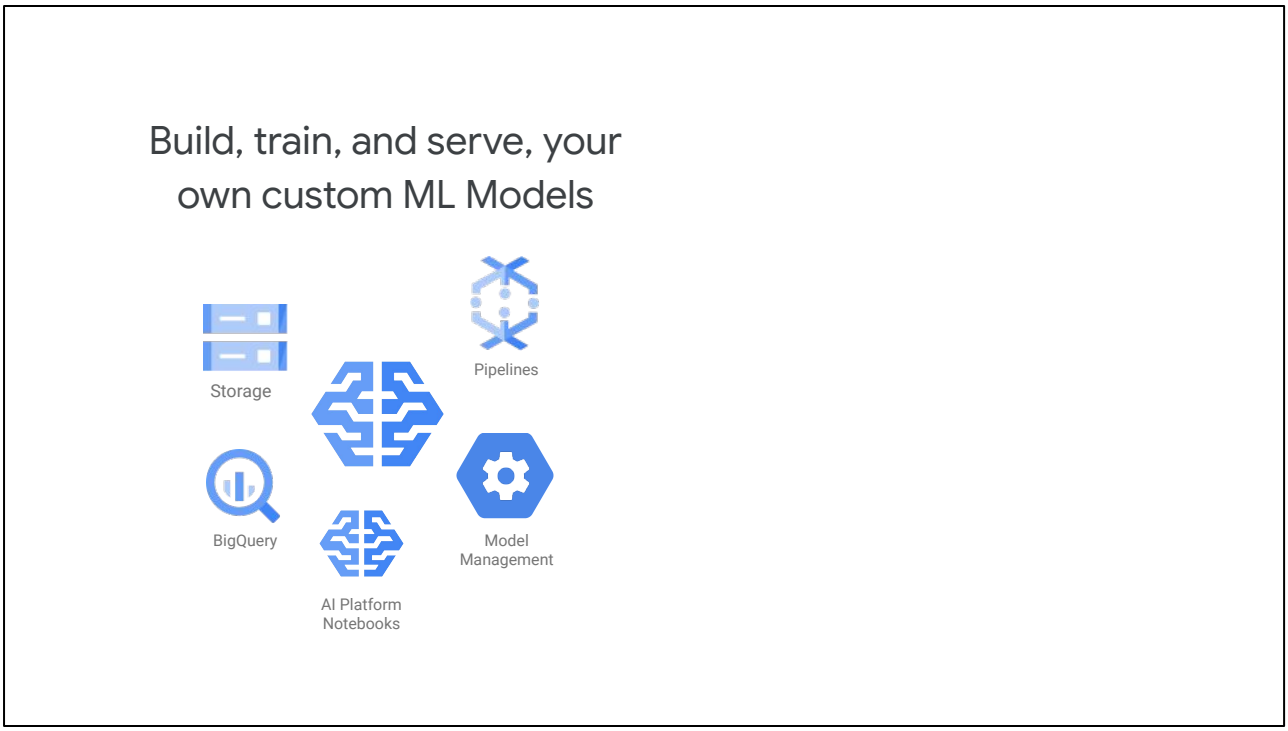
You have access to ready-made models like the Vision API, Natural Language API etc.

The key aspect of these models is that they are trained on Google's massive datasets.

Customize ready-made ML models

Auto-ML

Sometimes, ready-to-run models like the Vision API don't quite fit. In that case, you might want to train a family of models using your images and labels to customize and add to the Vision API. That's called Auto-ML and is possible only on the Cloud.



Even if you are building custom machine learning models, you have reason to do ML on GCP -- while TensorFlow is open-source, a serverless execution environment like Cloud ML Engine allows your data scientists to not have to worry about infrastructure. Plus, of course, the integration with distributed cloud storage and serverless and BigQuery make the overall development experience a lot better than if you had to provision and manage all that infrastructure yourself.

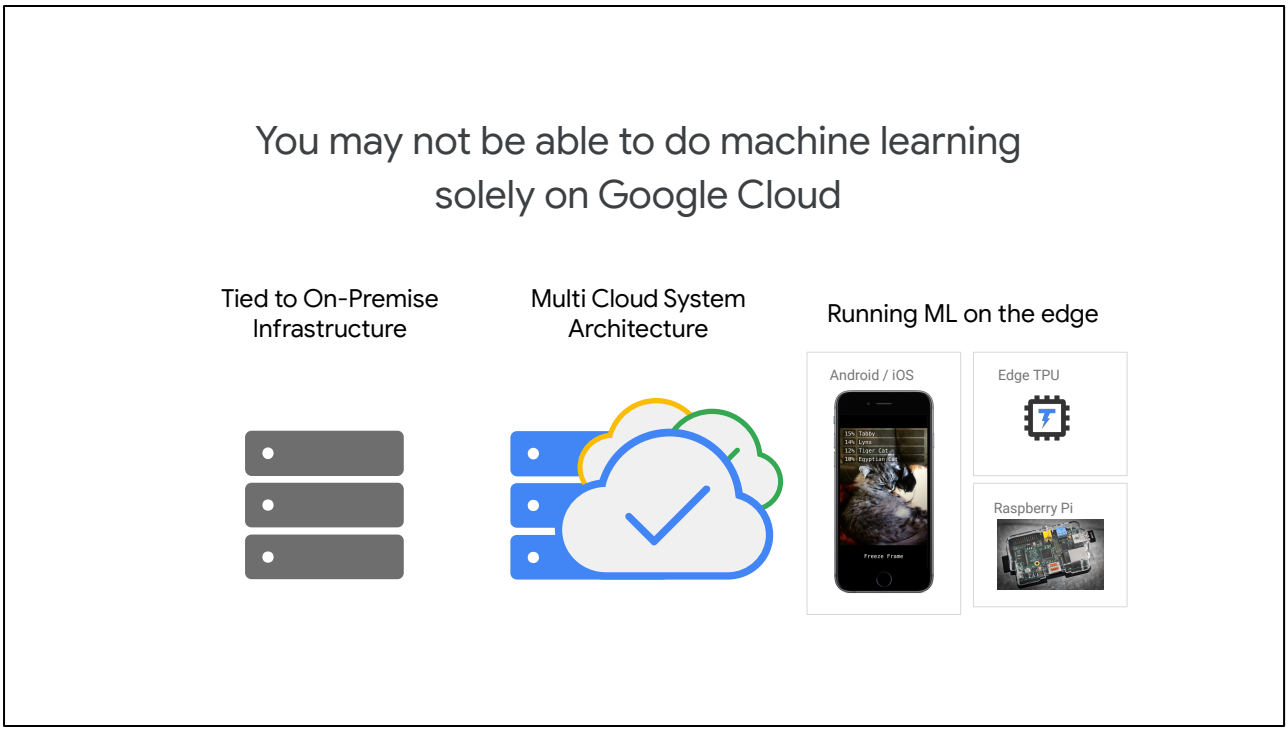
ML runtimes in a cloud-native environment

1. Prototype with AI Platform Notebooks or Deep Learning Image

2. Distribute and autoscale training and predictions with Cloud ML Engine

So far in this series of courses, we have assumed that you are in a cloud-native environment. So, we prototyped and developed our code using Cloud Datalab and once we had the code working on a small sample of data, we then submitted the training job to Cloud ML Engine to operate on the full dataset. We also served out model using Cloud ML Engine, so that we didn't have to worry about infrastructure.

There are times, however, when you can not be fully cloud-native.
What kinds of situations?



You may not be able to do machine learning solely on the cloud

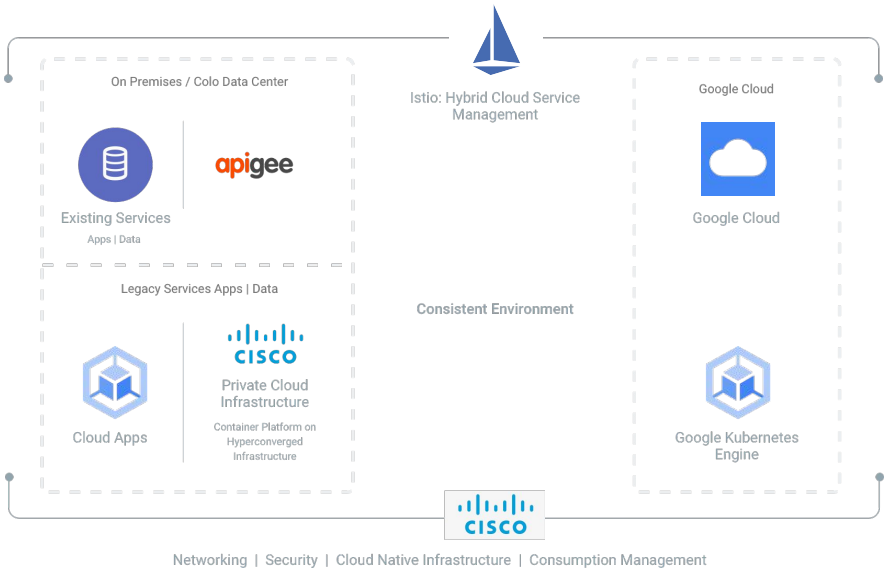
Perhaps, you are tied to on-premises infrastructure and your ultimate goal is to move to the public cloud perhaps in a few years.

Perhaps there are constraints about being able to move your training data off your on-prem cluster or data center. So you have to make do with the system you have.

Or maybe you are running machine learning on the edge, and connectivity constraints force you to have to do your predictions on the edge, on the device itself. So you have to do inference-on-the-edge. This is a common situation in internet-of-things.

Or maybe the data is being produced by a system that is running on a different cloud. Or the model predictions need to be consumed from another cloud. So, you need a multi-cloud solution, not a solution that is solely GCP.

Or maybe you are running machine learning on the edge, and connectivity constraints force you to have to do your predictions on the edge, on the device itself. So you have to do inference-on-the-edge. This is a common situation in internet-of-things.

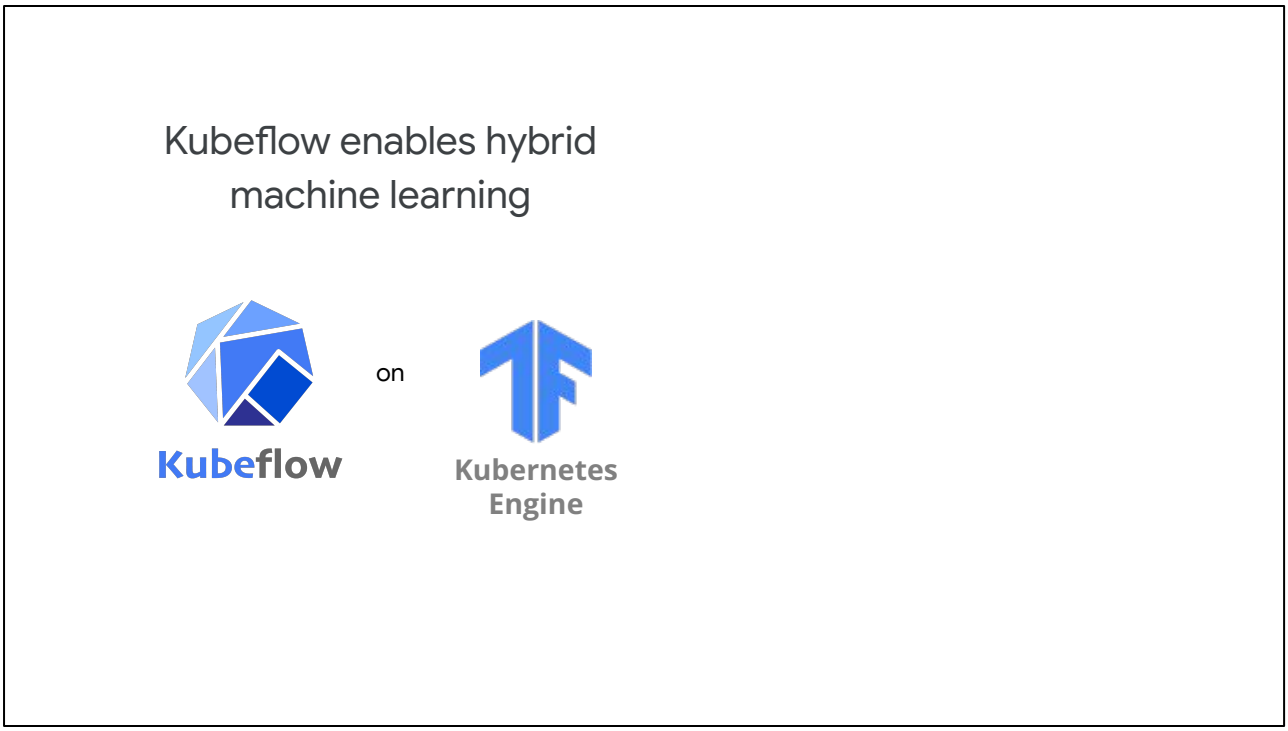


Here's an example of Cisco's hybrid cloud architecture -- Cisco partnered with Google Cloud to bridge their private cloud infrastructure and existing applications with Google Cloud Platform. Note the use of Google Kubernetes Engine to manage their container deployments.
Kubernetes is a container-orchestration system that was designed and then open-sourced by Google.

https://cloud.google.com/cisco/

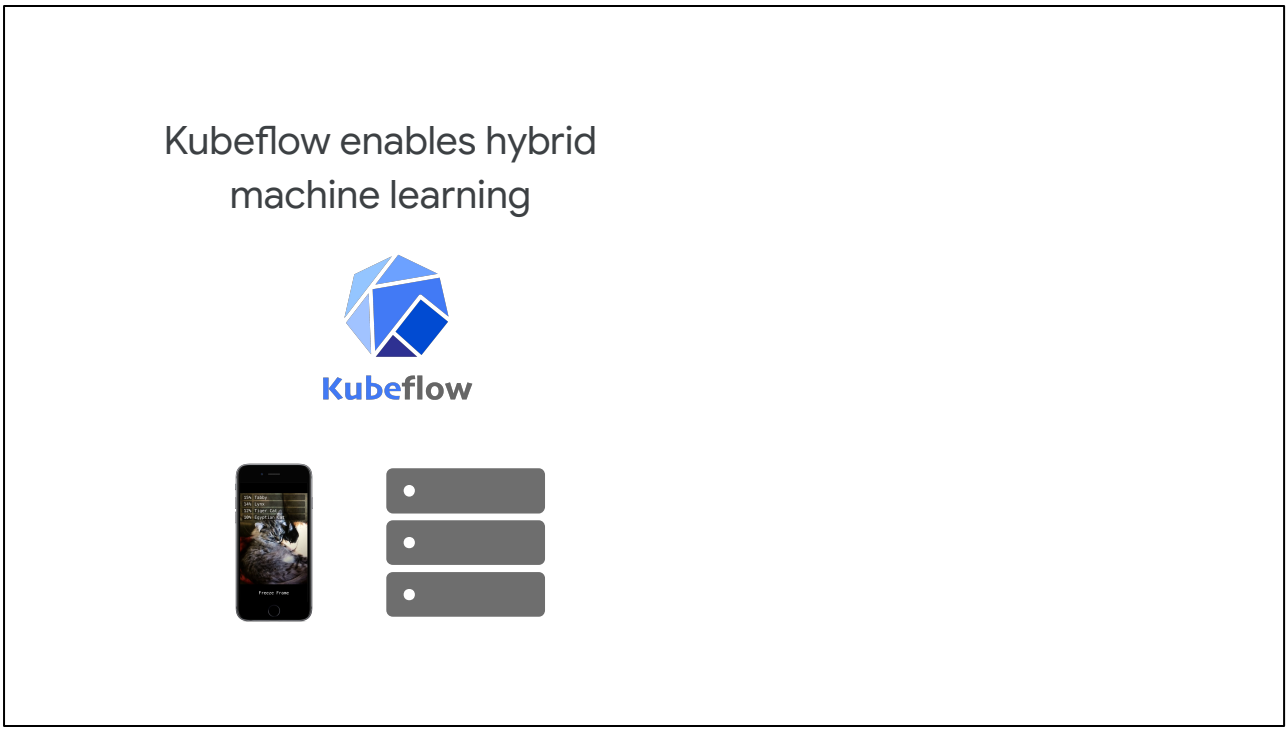
Kubernetes minimizes
infrastructure management

Using Kubernetes, it is possible to orchestrate containers whether they are running on-prem, or on the cloud. Any cloud. So, one possible solution to retain the ability to move fast, minimize your infrastructure management needs, and still retain the ability to move or burst to GCP is to use Kubernetes.

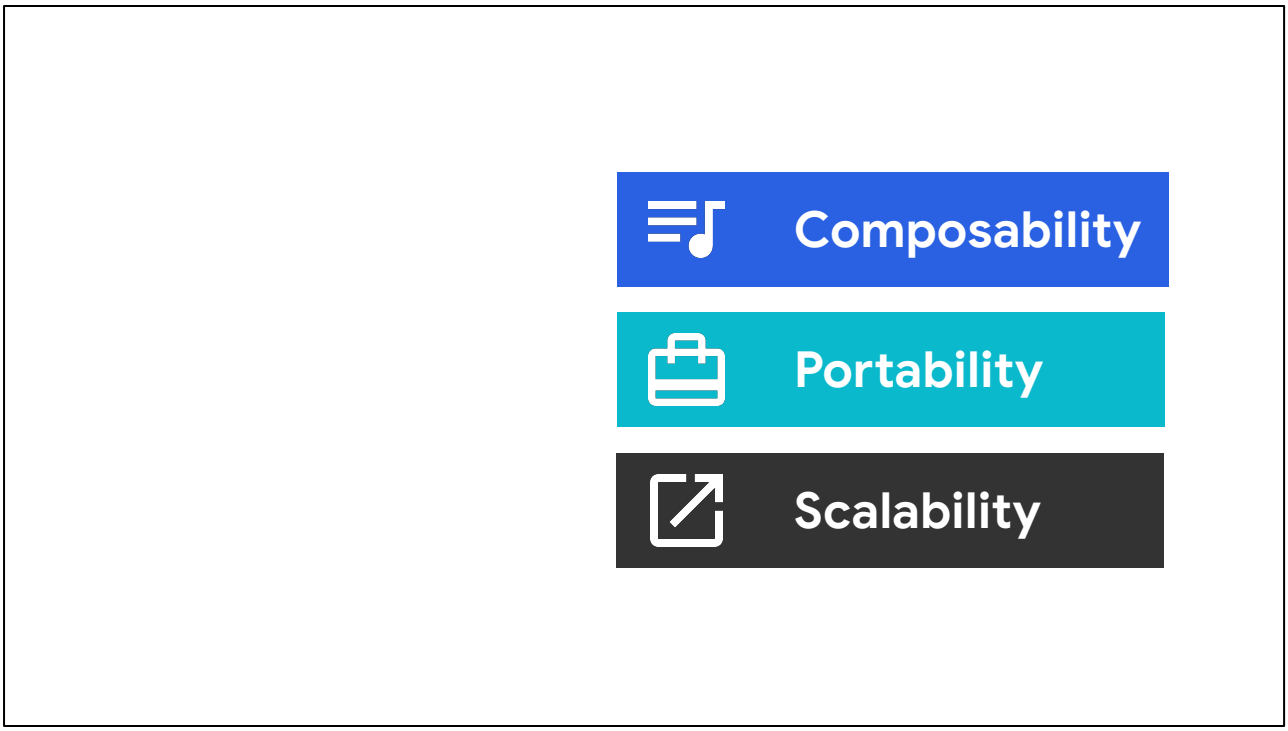


Specifically, a project called Kubeflow helps you migrate between cloud and on-prem environments. Kubeflow is an open source machine learning stack built on Kubernetes.

On Google Cloud, you can run Kubeflow on Google Kubernetes Engine, GKE.

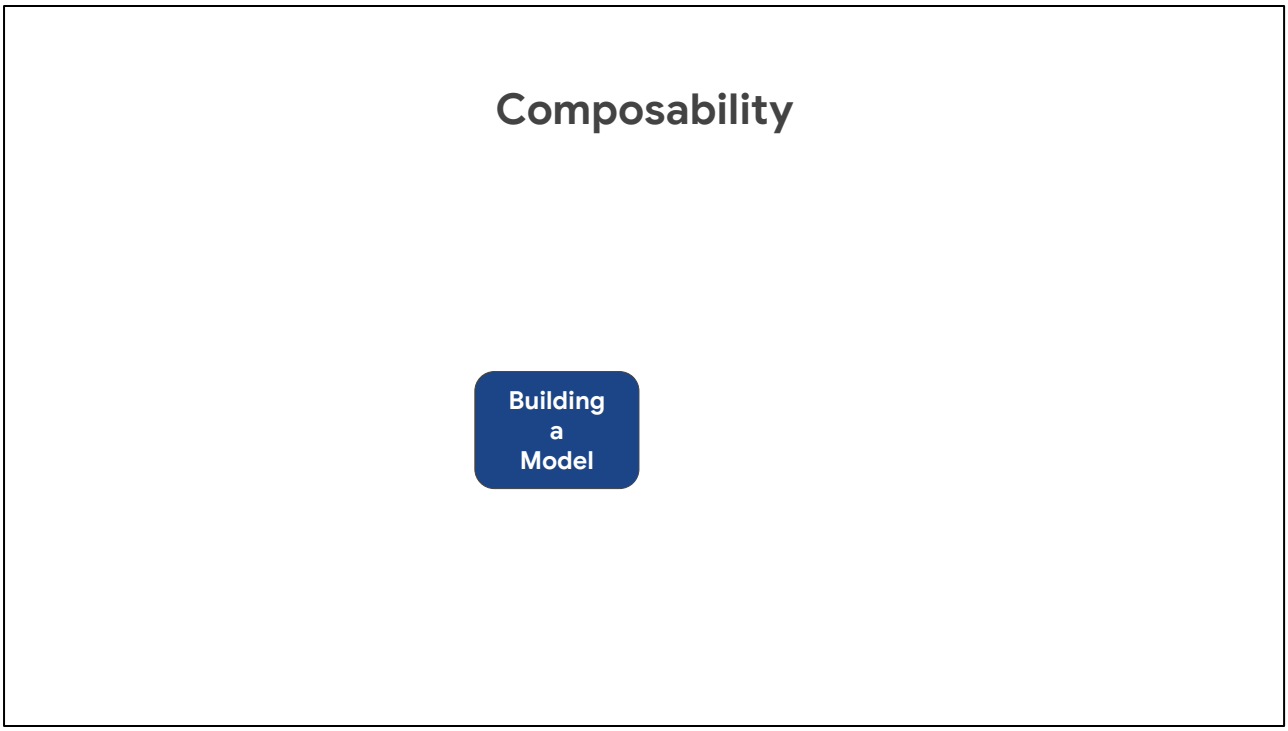


However, you can run Kubeflow on anything from a phone to a laptop to an on-prem cluster.
Your code remains the same. Some of the configuration settings change. That's it.



In order to build hybrid machine learning systems, that work well both on-prem and in the cloud, your ML framework has to support three things:

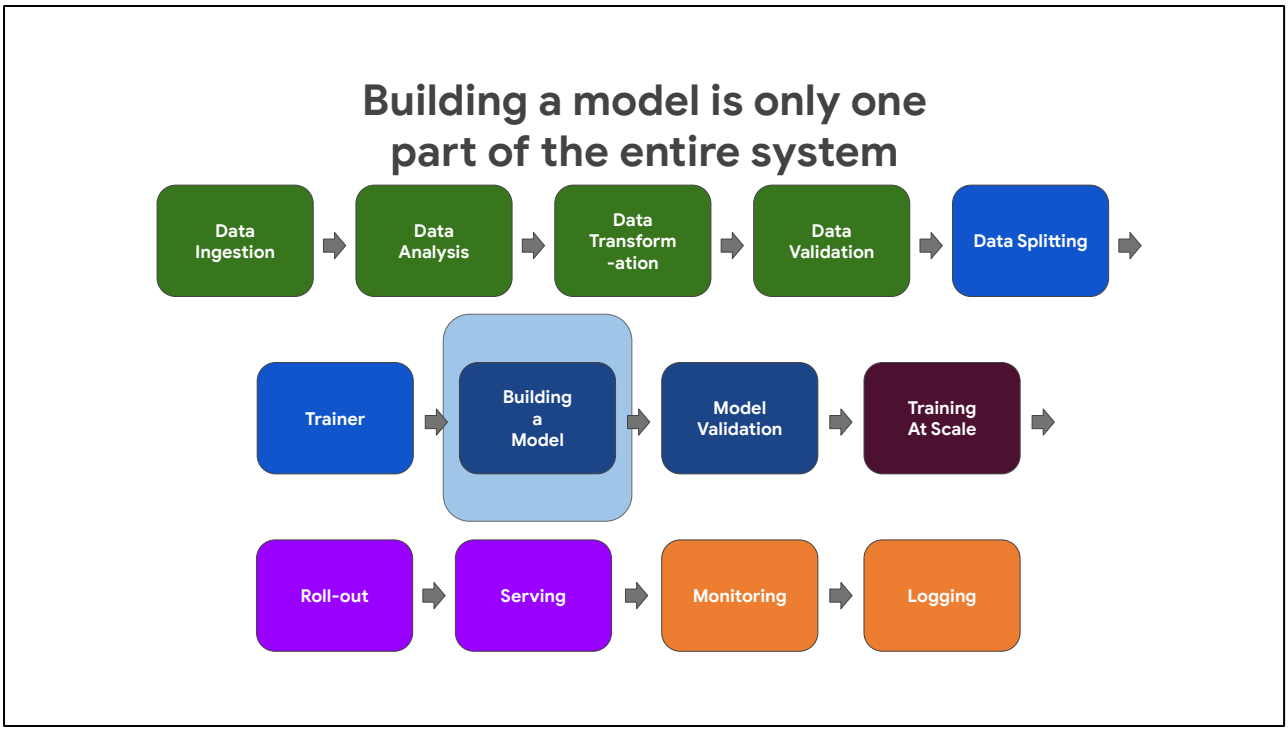
- Composability
- Portability
- Scalability



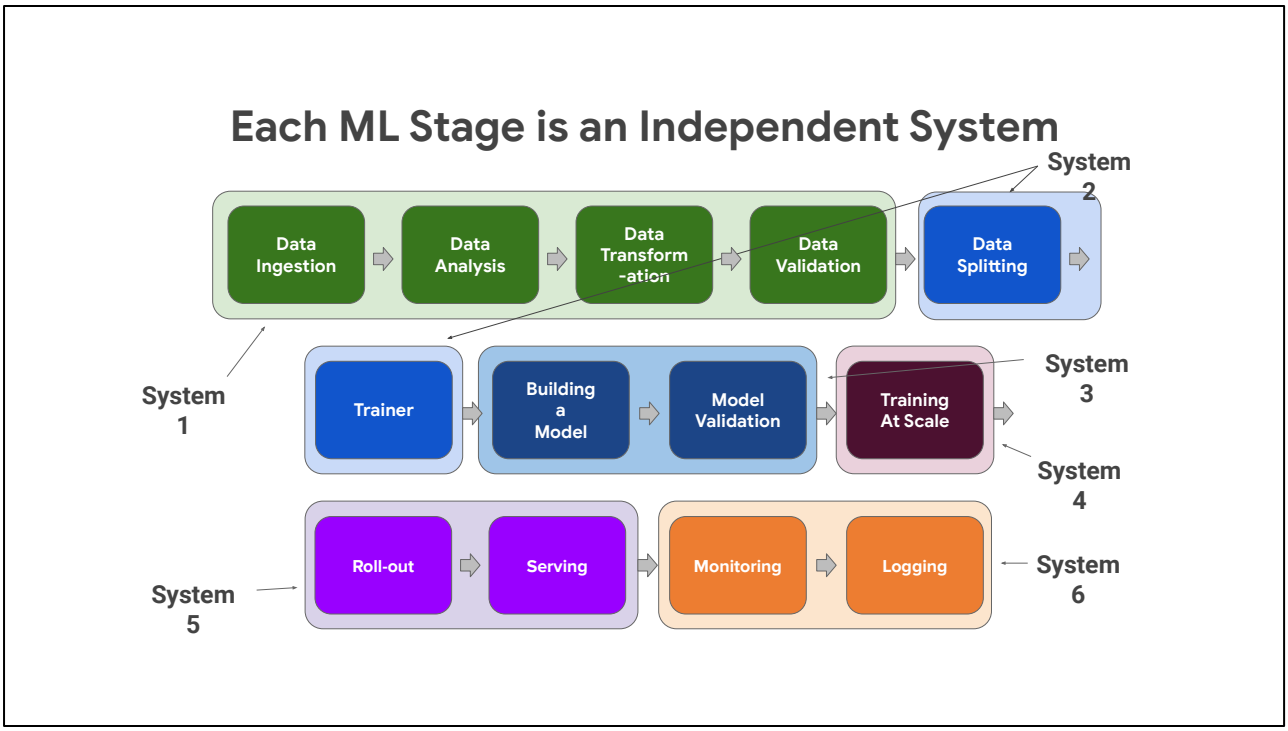
First composability.

When people think about ML, they think about building a model.

Tensorflow, pytorch, numpy, etc.

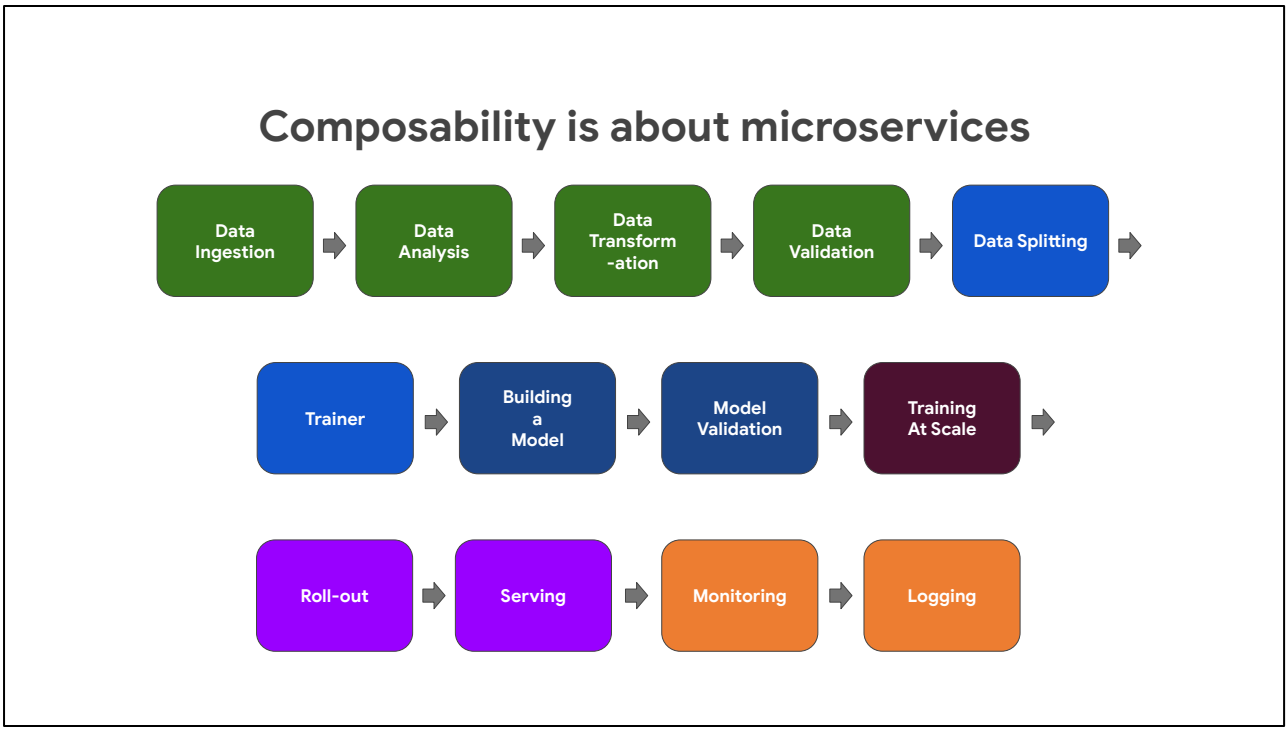


But the reality is 95% of the time is spent NOT building a model… it's all the other stuff.



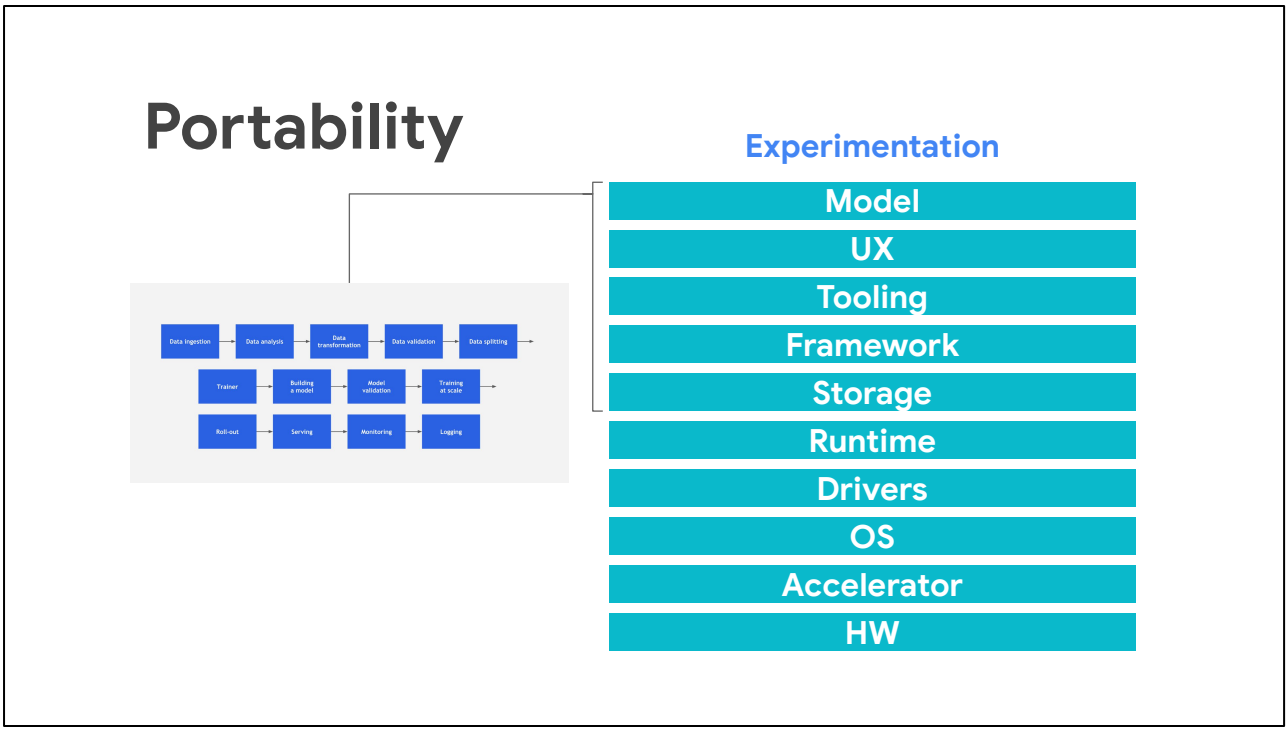
Each ML stage - data analysis, training, model validation, monitoring, they are all independent systems.

EVERYONE has a different way to handle all these boxes.



So when we say composability, it's about the ability to compose a BUNCH of microservices together.

And the option to use what makes sense for your problem.



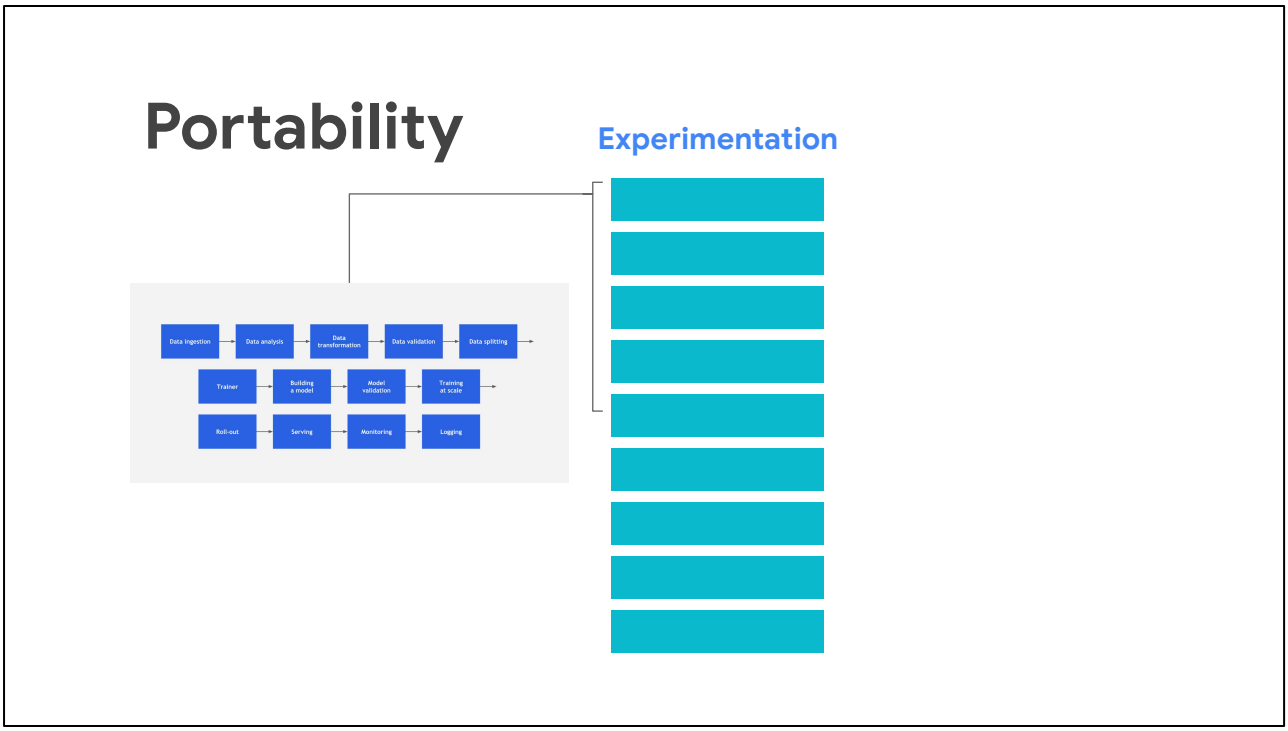
Now you've built your specific framework, you want to move it around. This is where we get into portability.

The stack you use is likely made up of all these components (and lots more)!

And all those microservices I detailed earlier only touch a small number of them.

But you do it, you configure every stage in the stack and it's FINALLY running.

What is this good for? What happens next? You need to think about the ML workflow.

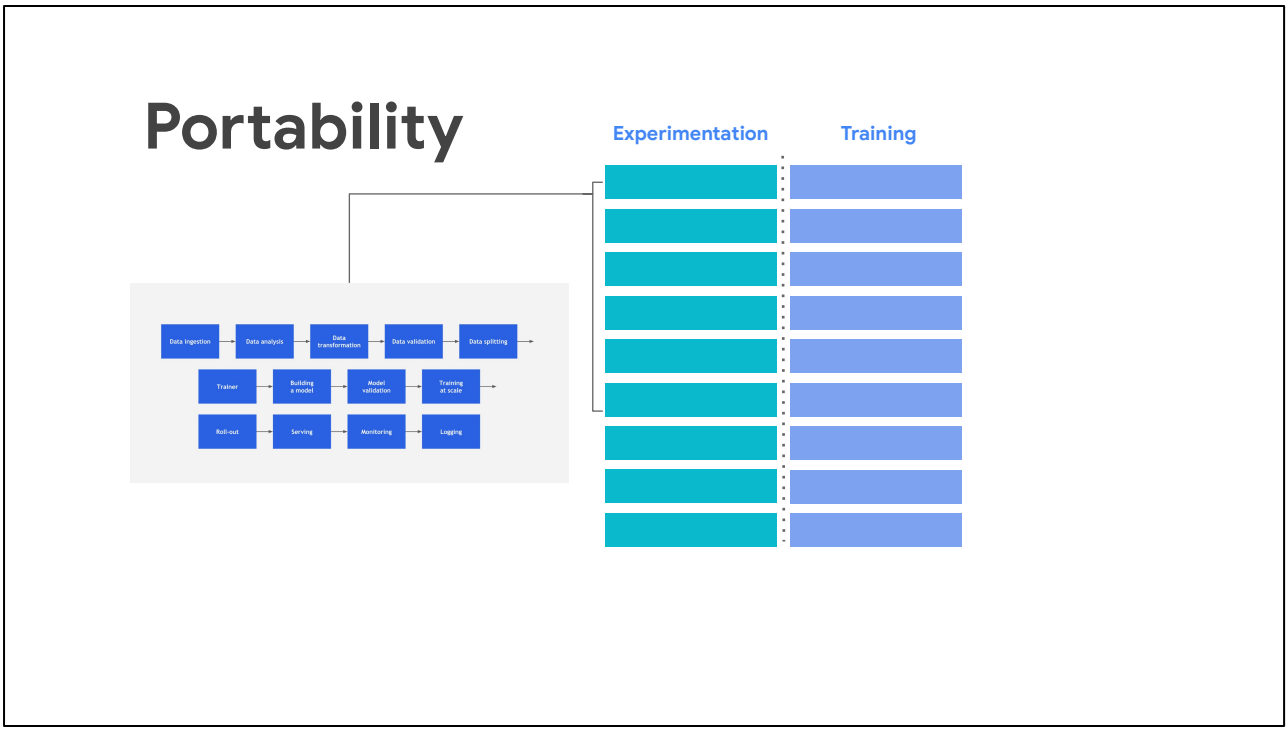


Remember that you did all this just so that you could develop the model.

We'll call that experimentation. But once you have the code running, what do you need to do?

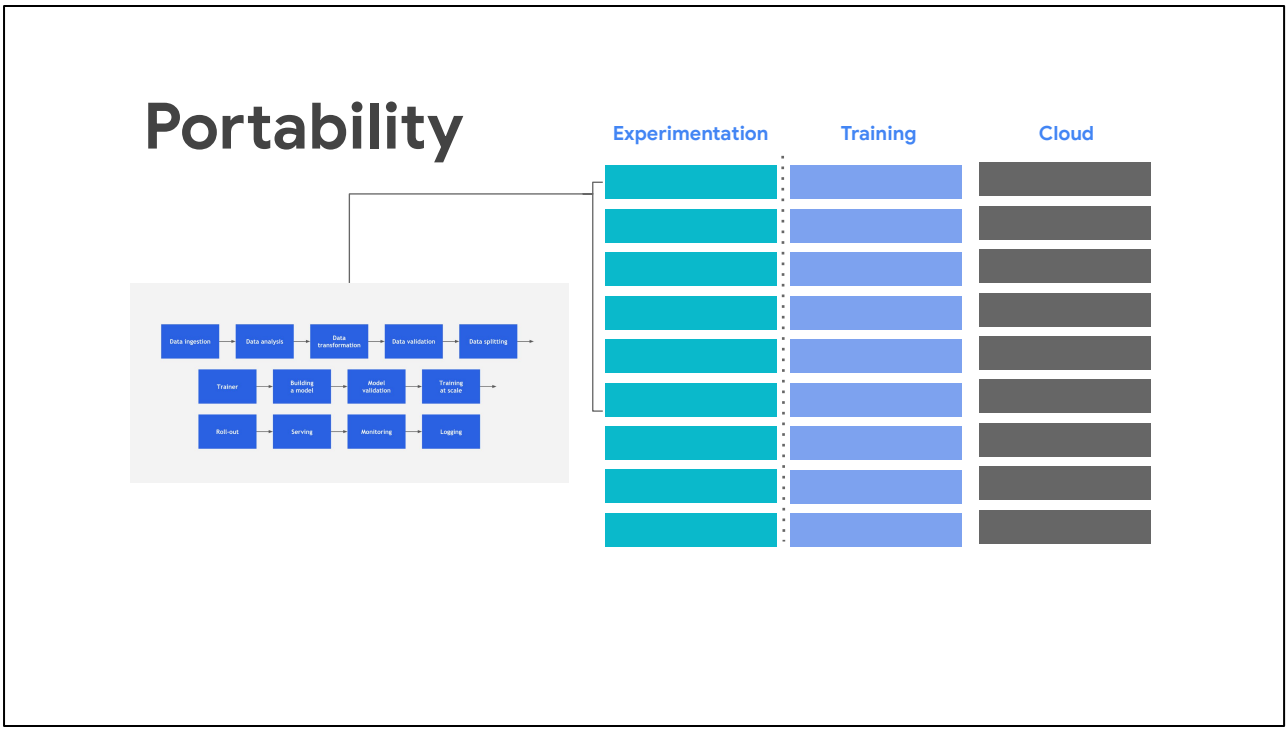
That's right, you need to train the model on the full dataset.

You probably can't do it on the small setup on which you did all your initial development.



So, you start up a training cluster…. And you've got to do it all over again.

All the configuration, all the libraries, all the testing. You've got to repeat it for the new environment.

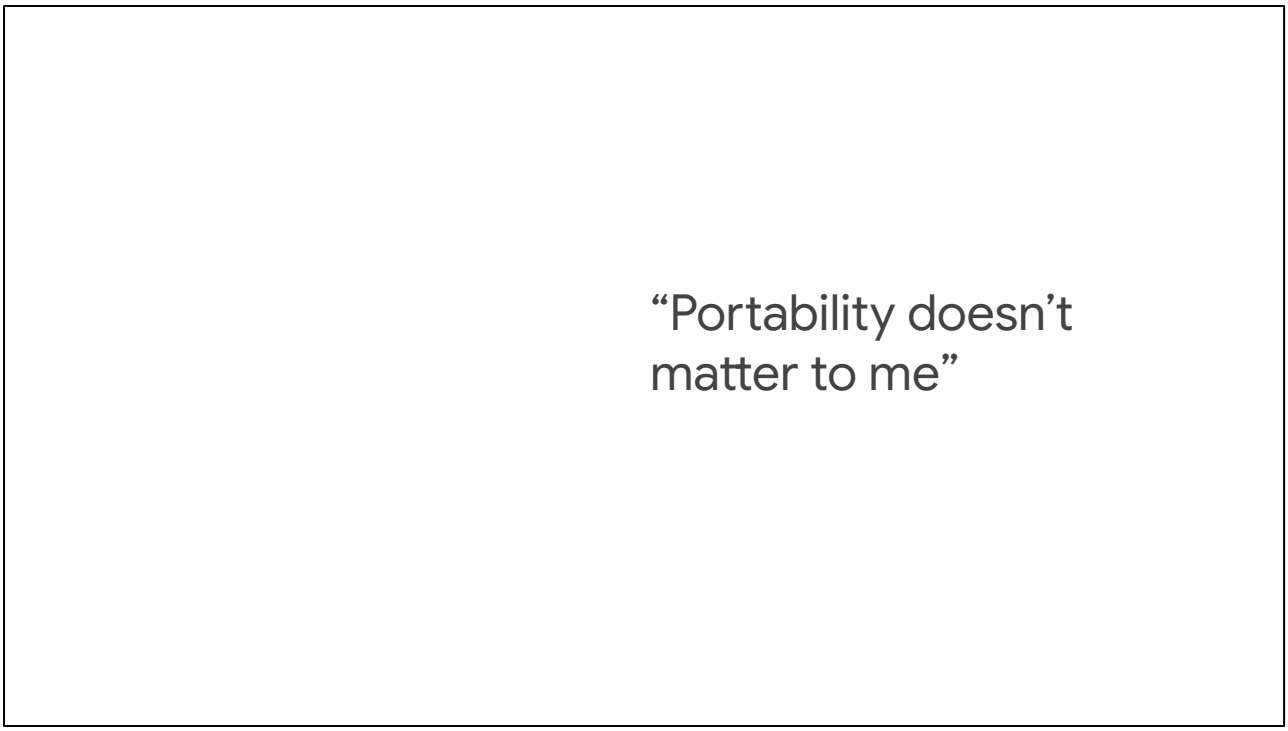


And then, chances are you've got to do it AGAIN to move to the cloud. Because, remember, we said, we want a hybrid environment.

An ML model that helps you train on the cloud, and predict on the edge.

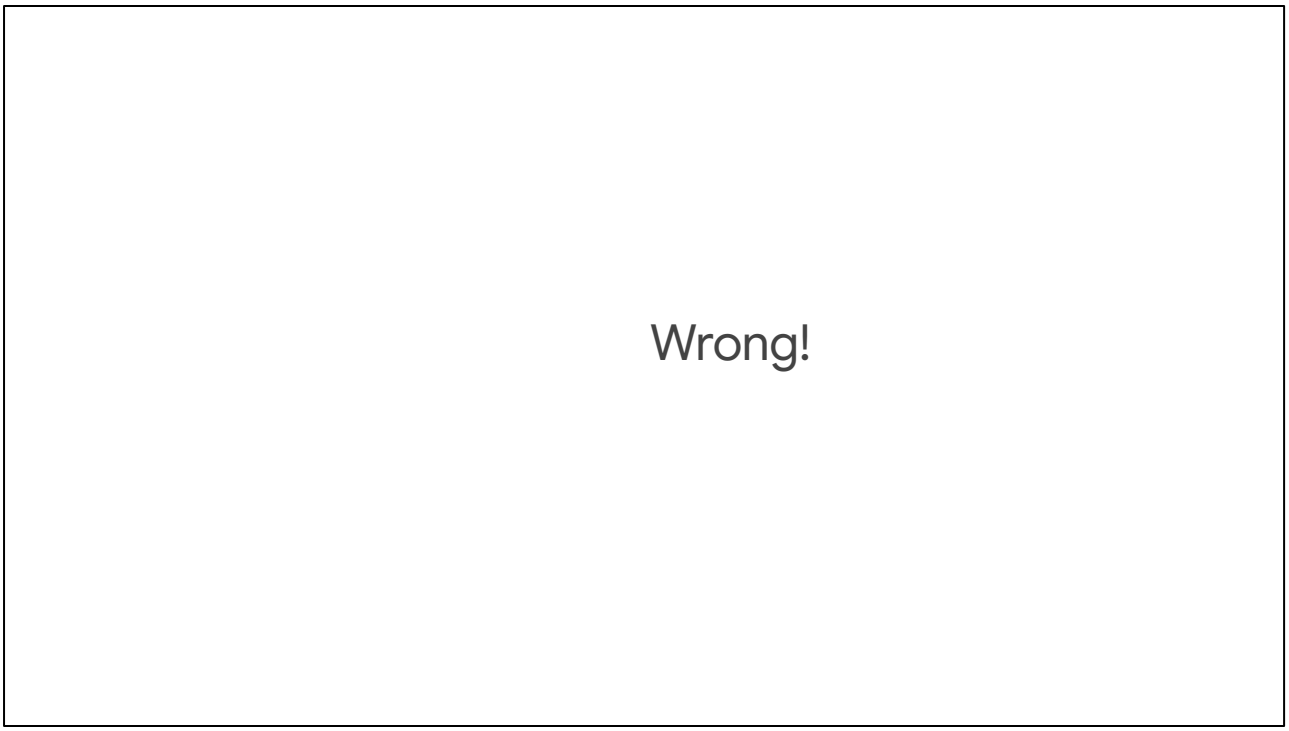
Or train on-the-cloud, but predict on-premises.

The point is that you have to configure the stack over and over again for each environment you need to support.

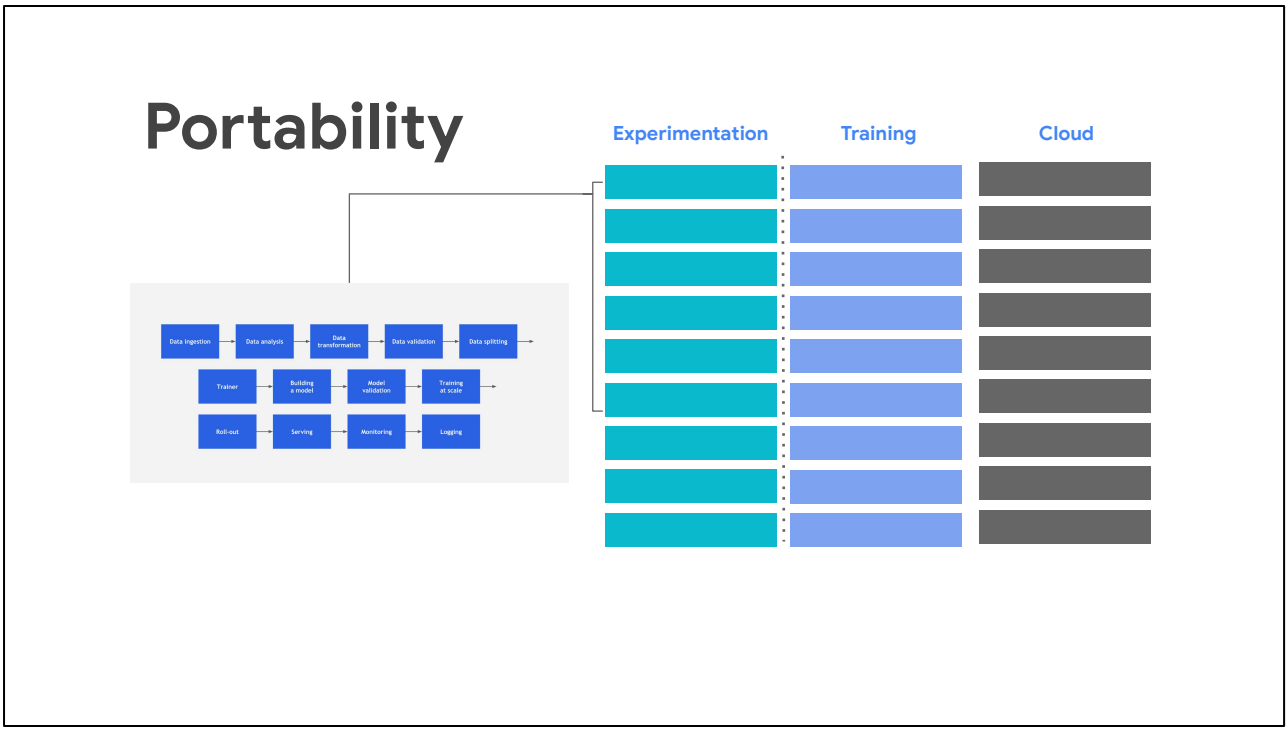


Maybe at this point, you are thinking “pfft … that doesn’t matter to me.”

I *never* have to change environments. I will use only one environment.



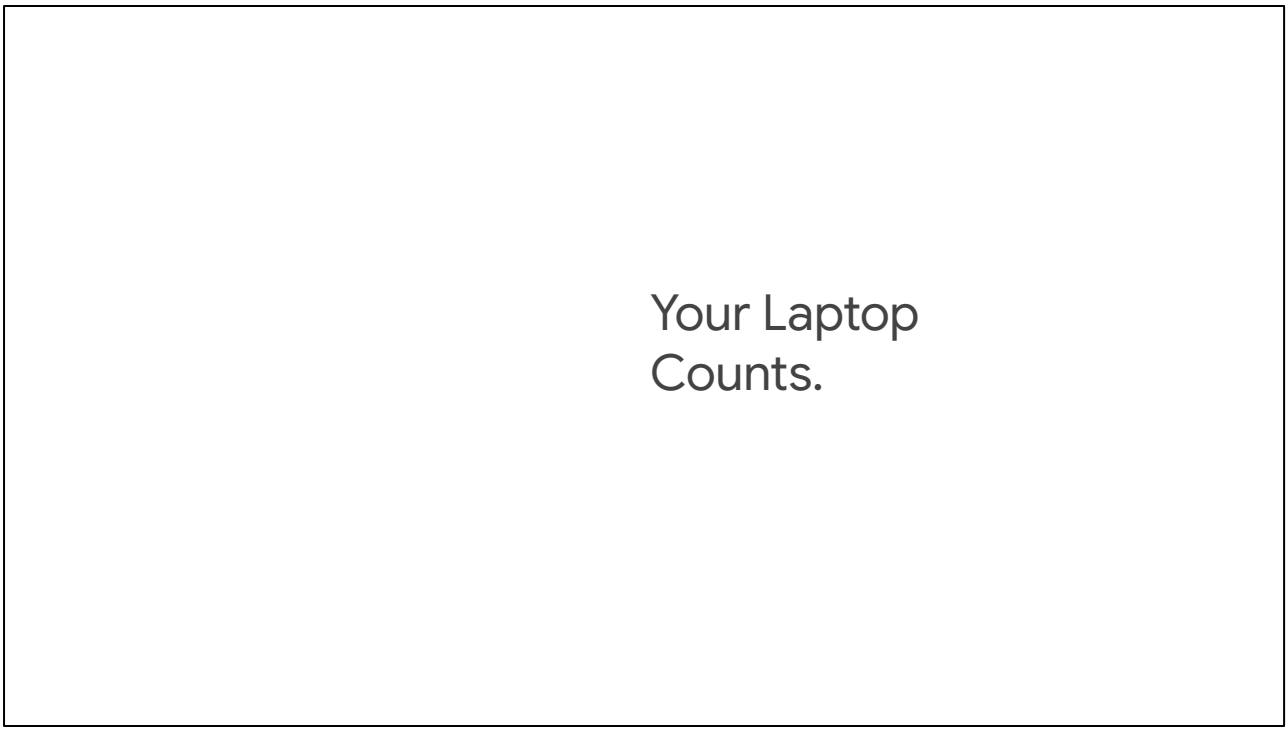
Wrong.



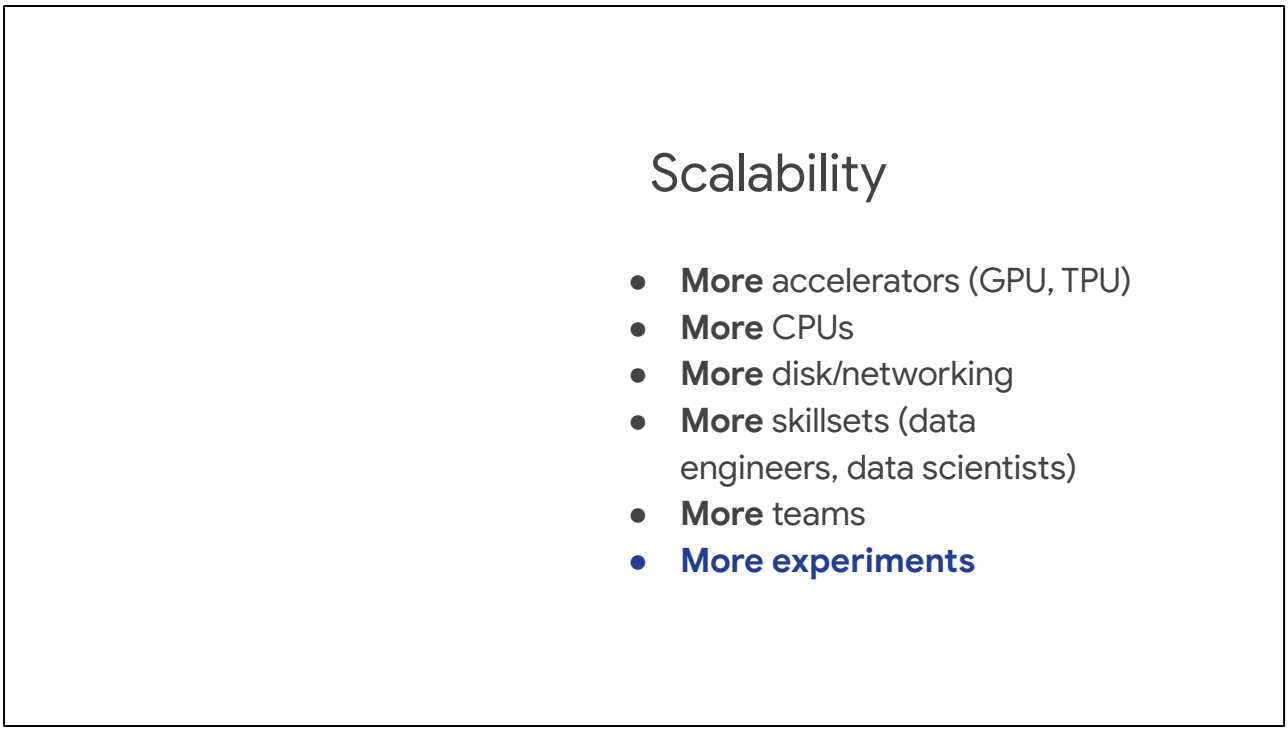
Portability? It is essential.

And then, of course, you've got to do it AGAIN when your inputs change, or your boss tells you to train faster by training on more machines.

You find that you have to change environments over and over again.



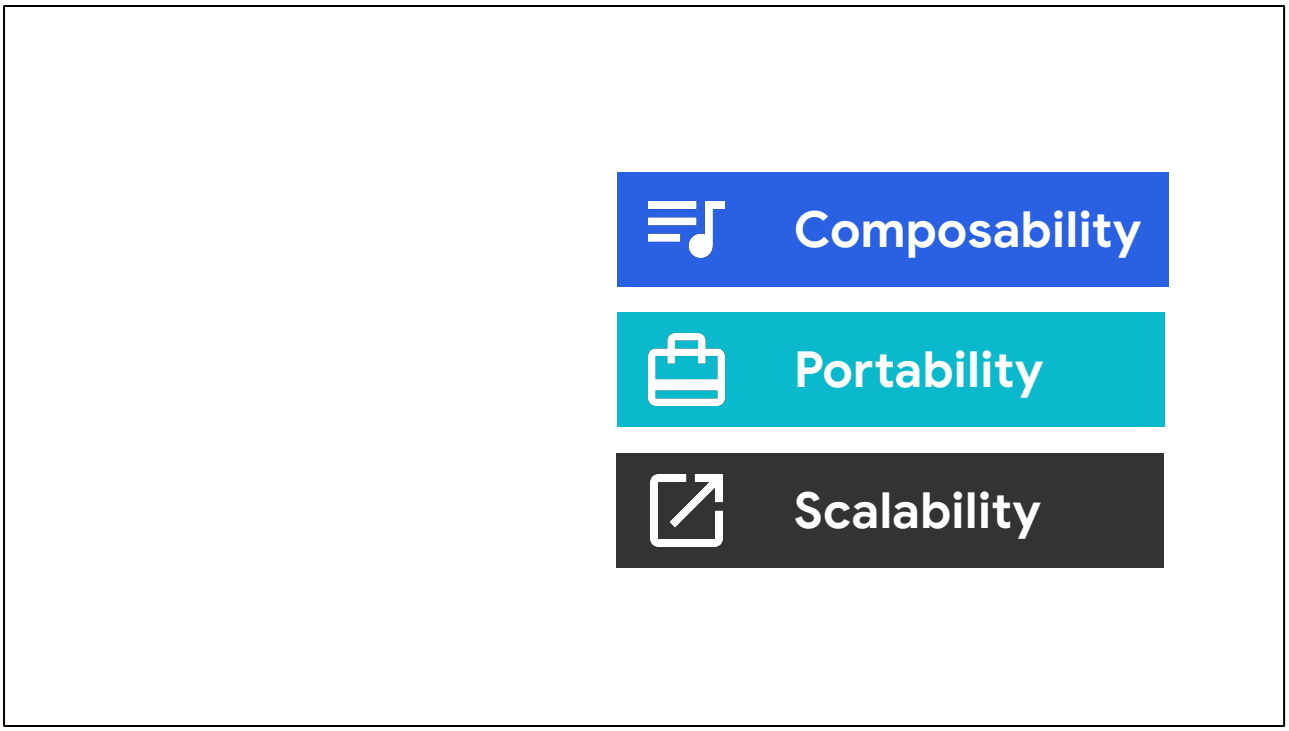
Your laptop counts as environment #1. And you don’t do production services on your laptop. So …



Finally scalability.

You always hear about Kubernetes being able to scale, and that's true! But scalability in ML means so many more things:

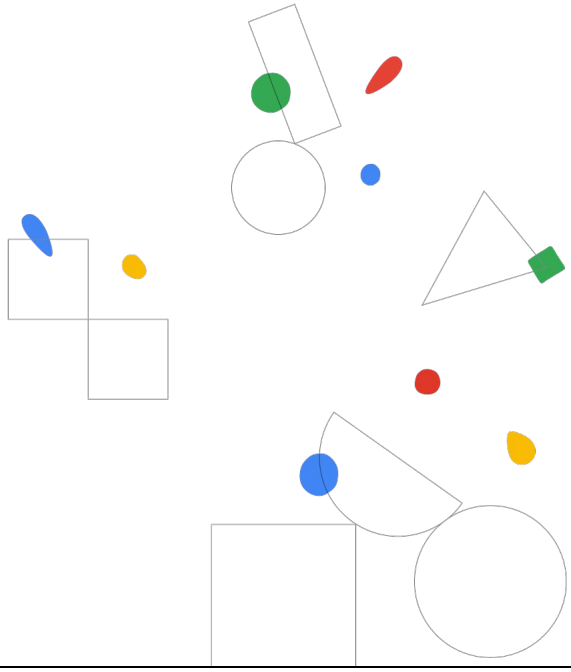
- Accelerators
- Disk
- Skillsets (software engineers vs. researchers)
- Teams
- Experiments



So that's what we think of when we think of machine learning in a hybrid cloud environment. Composability. Portability. Scalability.

Google Cloud

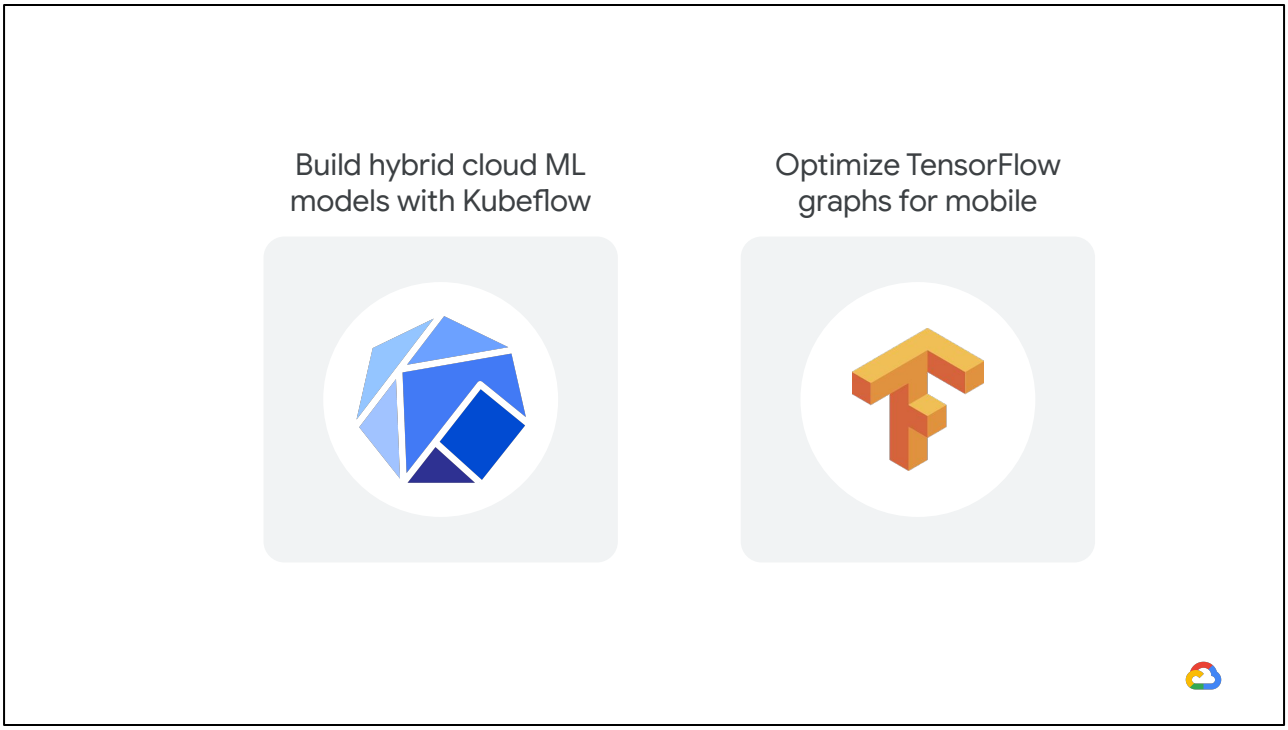
Kubeflow



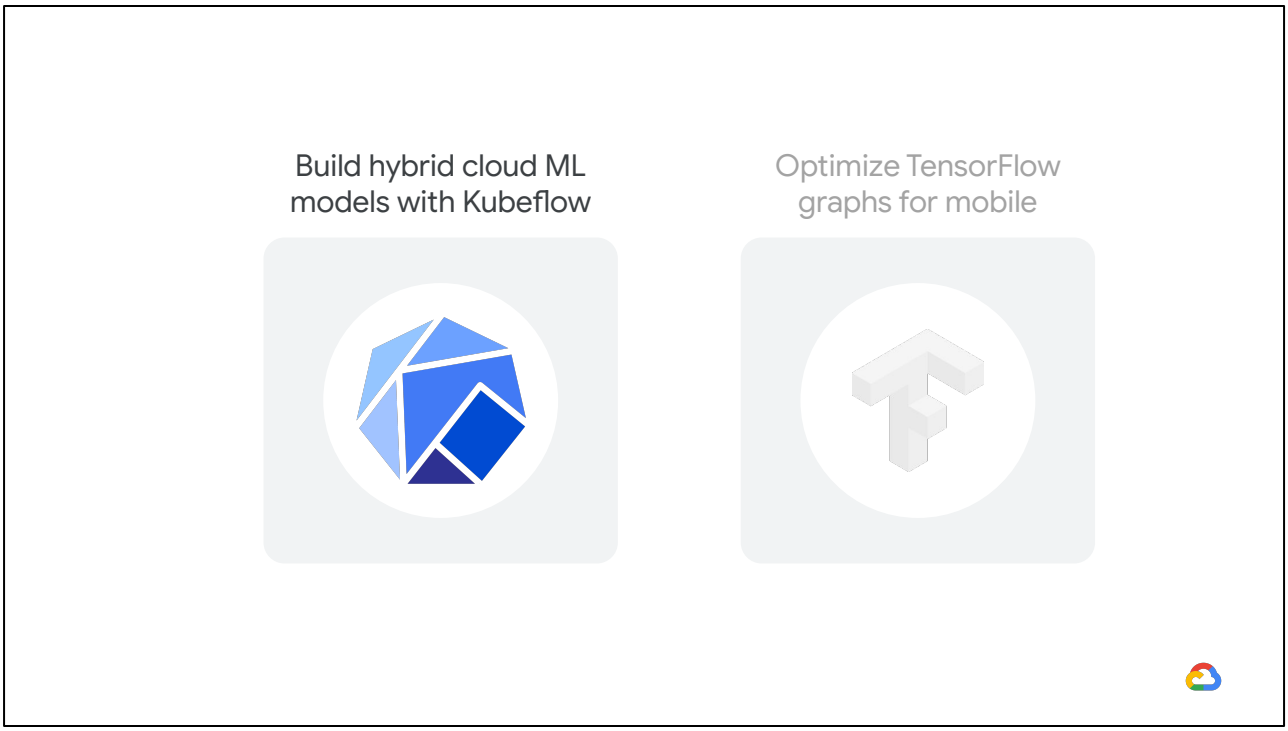
Module 04
Building Hybrid ML systems

Welcome back.

In this module you will learn how to

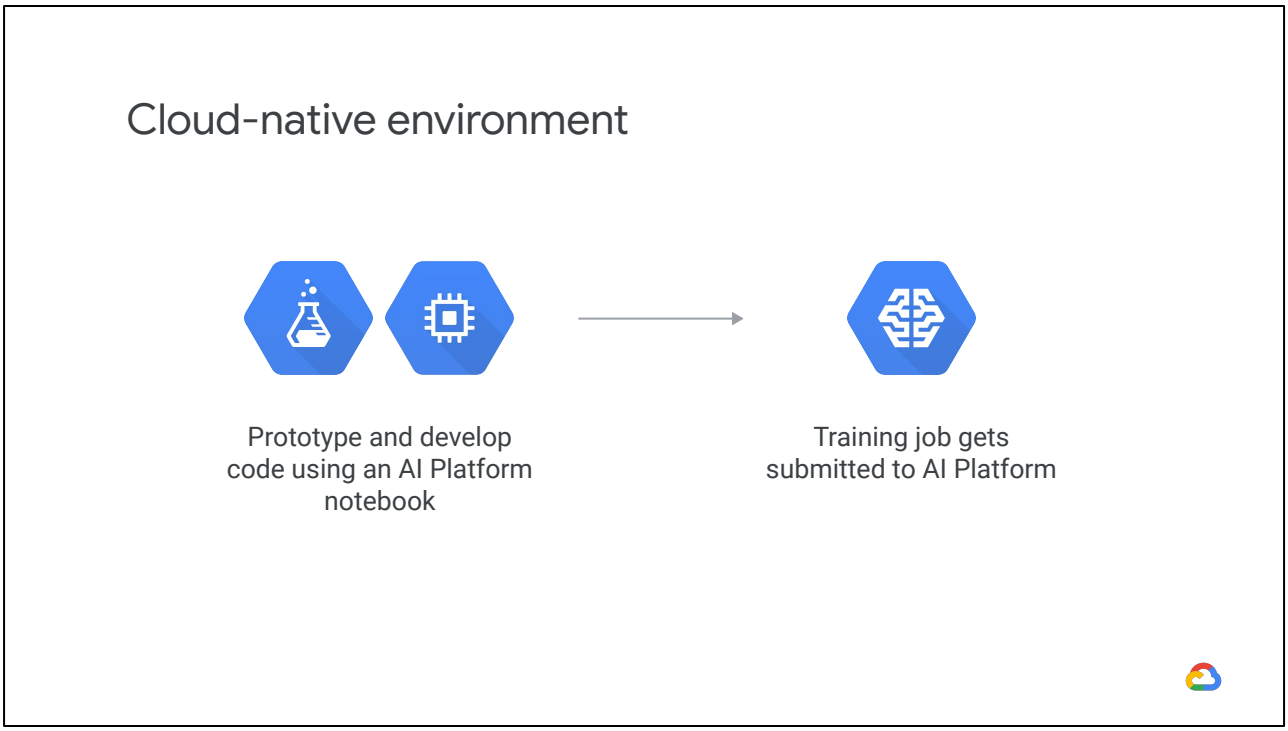


build hybrid cloud machine learning models with Kubeflow and how to optimize TensorFlow graphs for mobile.



To begin, let's explore Kubeflow, an open-source machine learning platform designed to enable the use of machine learning pipelines to orchestrate complicated workflows running on Kubernetes.

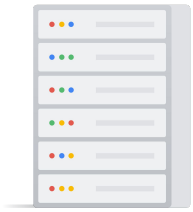
Kubeflow helps build hybrid cloud machine learning models. But why are we discussing hybrid in the first place? Why would you need anything other than Google Cloud?



So far in this course we've focused on cloud-native environments, which involve prototyping and developing code using an AI Platform notebook.

Once that code is working on a small sample of data, the training job gets submitted to AI Platform to operate on the full dataset.

There are, however, scenarios when cloud-native, or conducting machine learning solely on Google Cloud, is not an option. Let's look at some of these scenarios now.

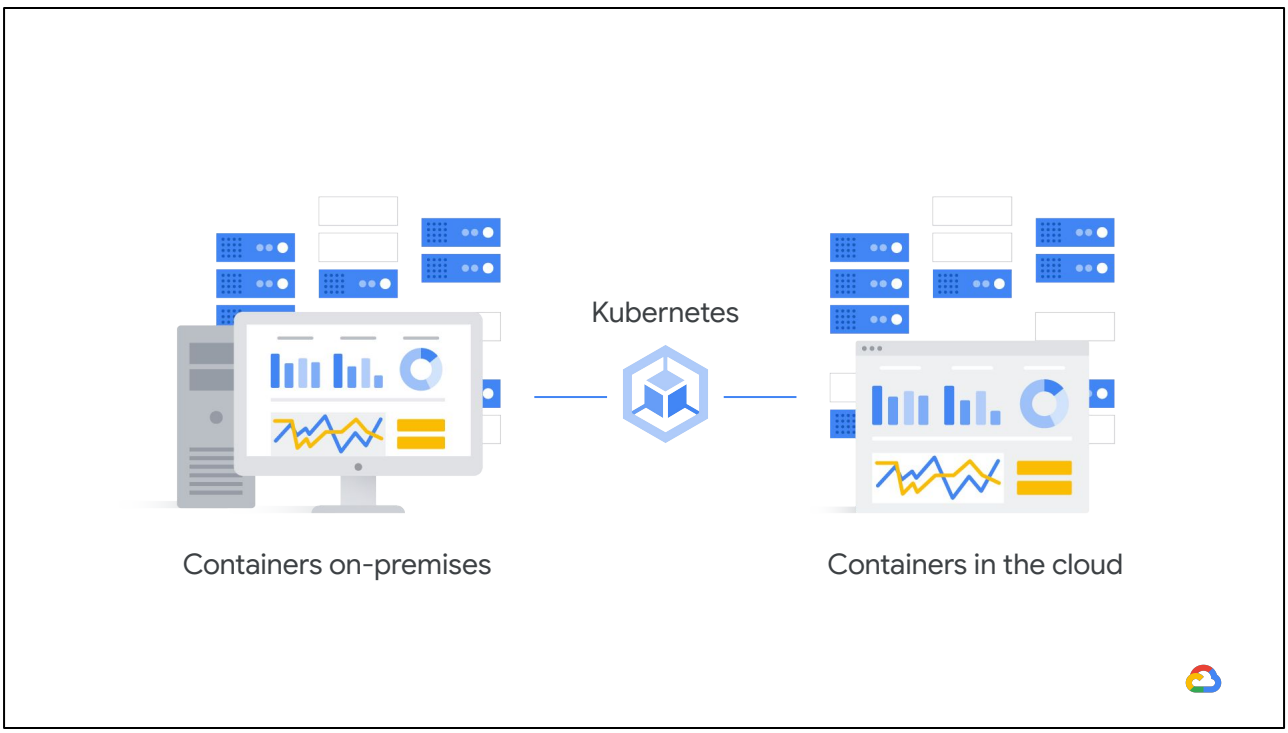


Tied to on-premises architecture

Multi-cloud solution architecture

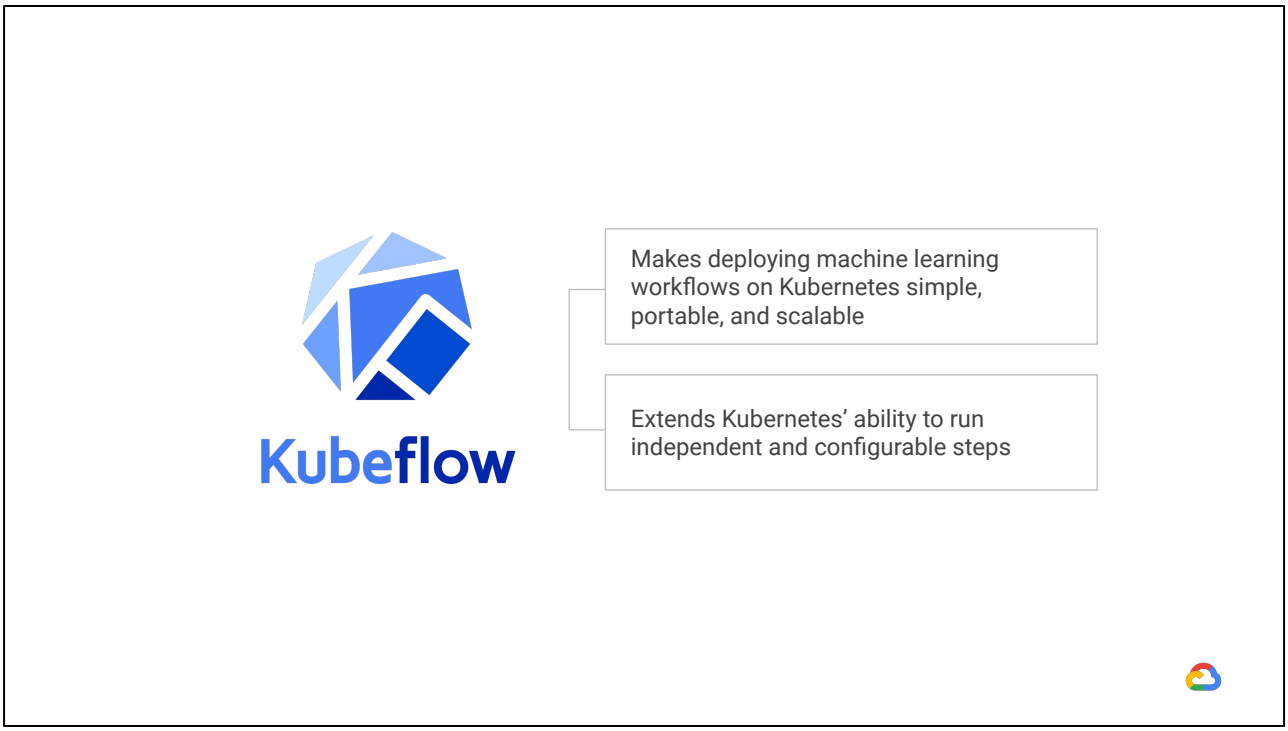
Running ML on the edge

- Maybe you're tied to on-premises infrastructure, or there are other constraints preventing the option to move your training data off an on-prem cluster or data center.
- Alternatively, perhaps you require a multi-cloud solution architecture, one that does not rely solely on Google Cloud. This could be because you're working with data that is being produced by a system that is running on a different cloud provider, or because model predictions need to be consumed from another cloud.
- Or maybe you are in an initial phase of an ML project running machine learning on the edge, where you're working on a local developer workstation. Training at scale typically happens in a cloud environment; however, inference and distributed training can happen at the edge. The edge means that predictions happen on a smart device, which is common in internet-of-things.



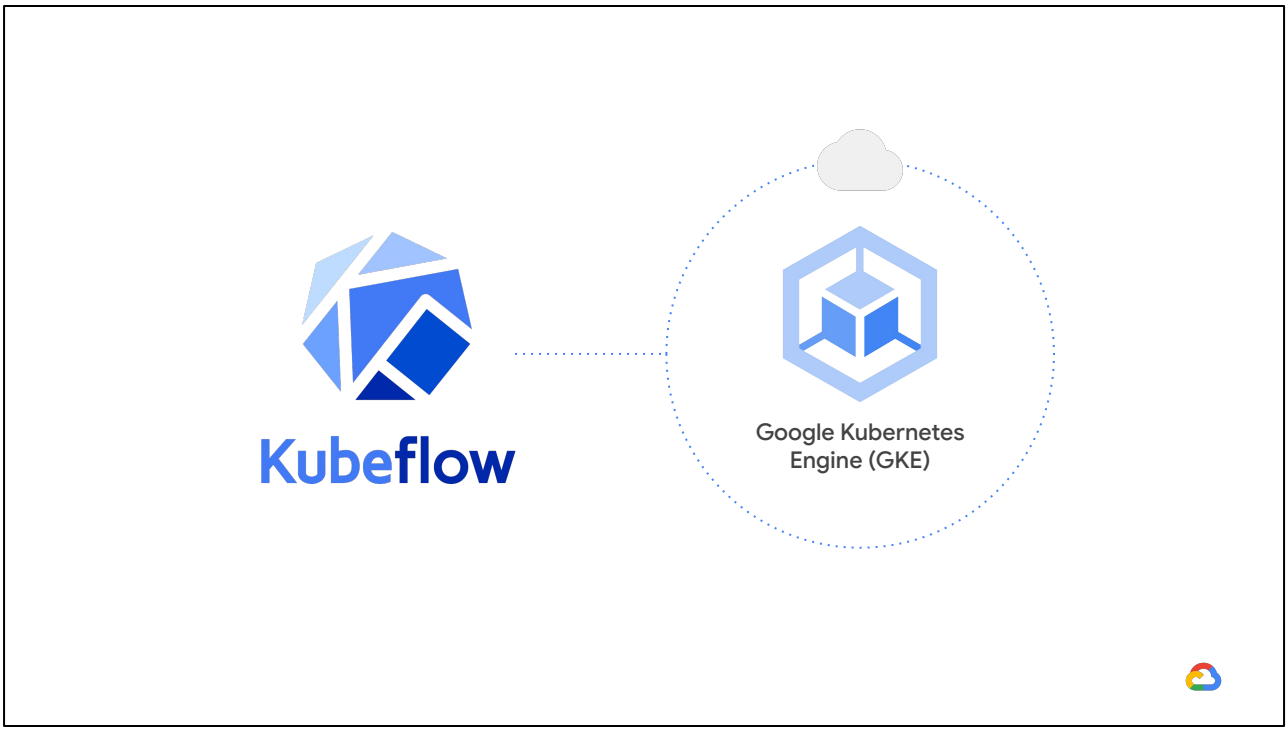
Using Kubernetes, it's possible to orchestrate containers that run either on-premises or in the cloud. And that can be any cloud.

Using Kubernetes allows for speed, the ability to minimize infrastructure management needs, all while being able to move or burst to Google Cloud.

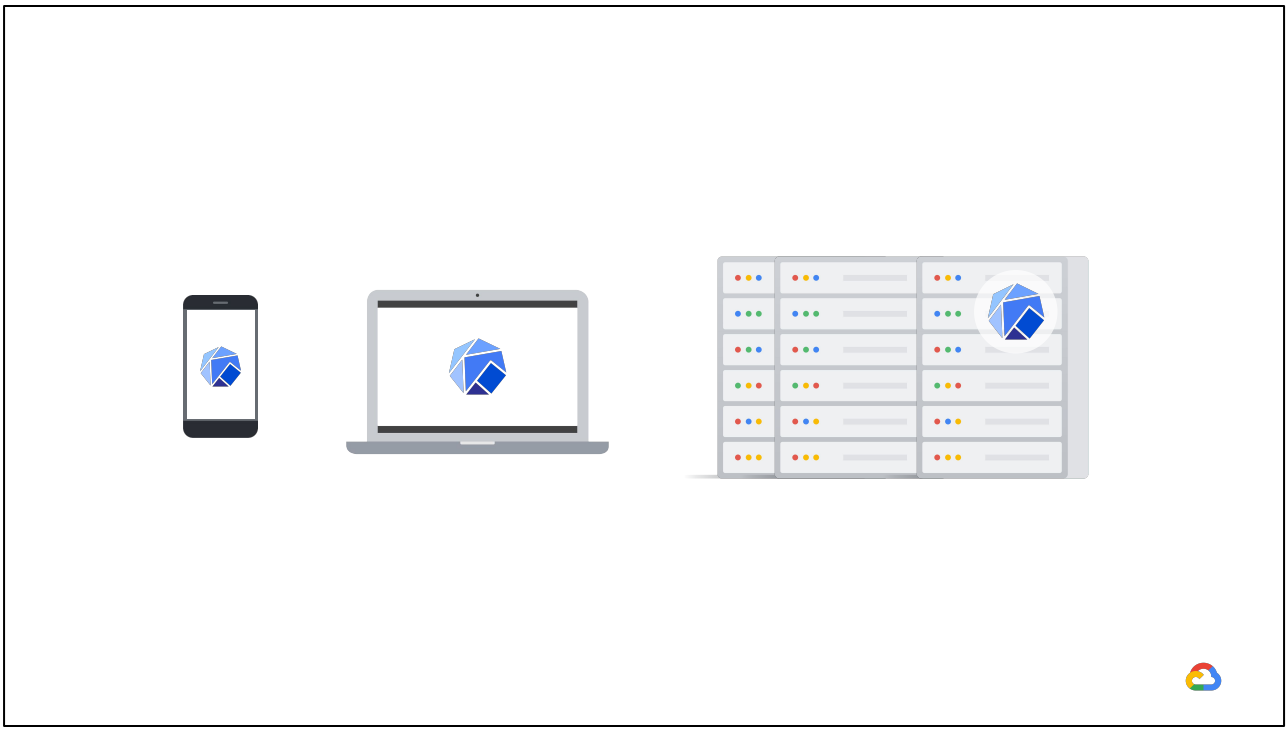


Kubeflow is the machine learning toolkit for Kubernetes, and it brings a number of benefits.

- It makes deploying machine learning workflows on Kubernetes simple, portable, and scalable.
- It also extends Kubernetes' ability to run independent and configurable steps, with machine learning specific frameworks and libraries.



And since Kubeflow is open source, it can run Google Kubernetes Engine, which is part of Google Cloud.



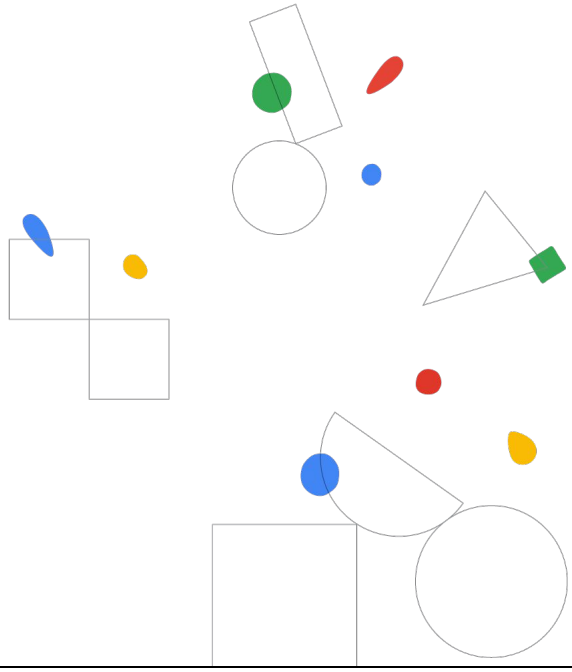
However, Kubeflow can actually run on anything–whether it's a phone, a laptop, or an on-premise cluster.

Regardless of where it's run, the code remains the same. Some of the configuration settings just change.

Google Cloud

Lab

Kubeflow Pipelines with Google Cloud's AI Platform

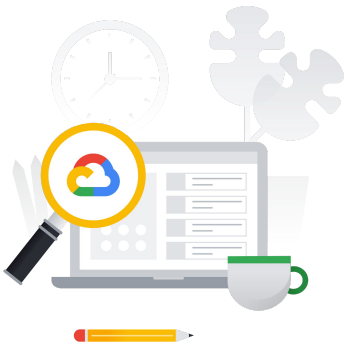


Module 04

Building Hybrid ML systems

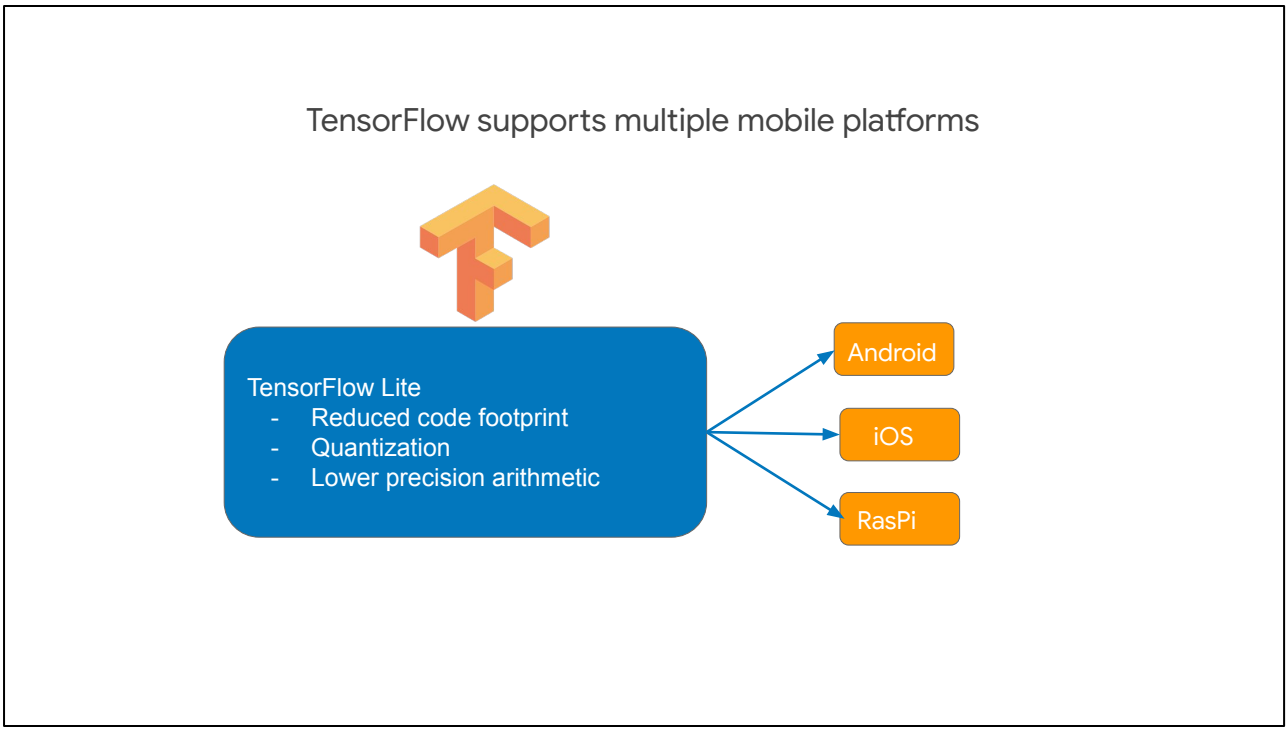
This lab provides hands-on practice installing and using Kubeflow Pipelines to orchestrate Google Cloud services in an end-to-end ML pipeline.

Lab objectives



- Create a Kubernetes cluster and install Kubeflow Pipelines.
- Launch an AI Platform notebook.
- Create and run an AI Platform Pipelines instance.
- Run a Python function-based pipeline.

- To begin, you'll create a Kubernetes cluster and install Kubeflow Pipelines.
- Next, you'll launch an AI Platform notebook.
- From there, you'll create and run an AI Platform Pipelines instance.
- And finally, you'll run a Python function-based pipeline.



TF supports multiple mobile platforms, including Android, iOS and RasPi. In this talk, we focus on mobile devices.

Mobile TensorFlow makes sense when there is a poor or missing network connection, or where sending continuous data to a server would be too expensive.

The purpose is to help developers make lean mobile apps using TensorFlow, both by continuing to reduce the code footprint, and by supporting quantization and lower precision arithmetic that reduce model size.

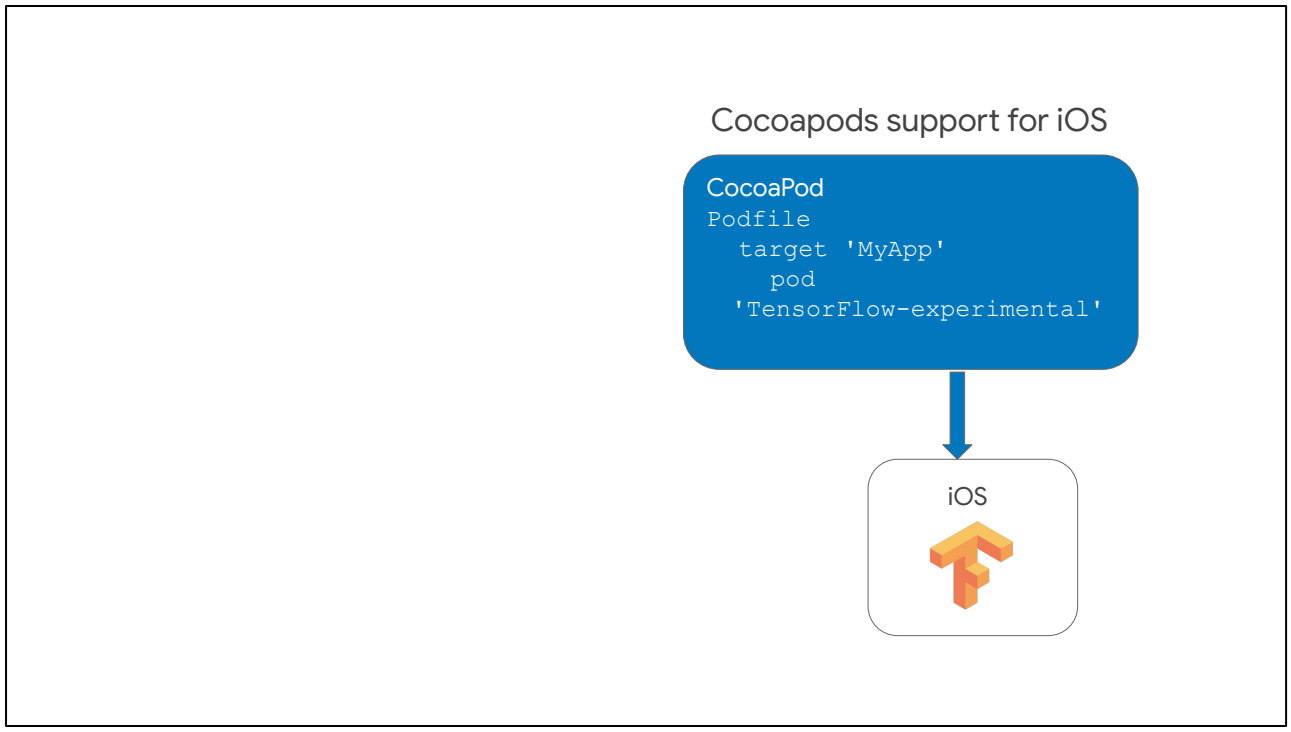
Build with Bazel by starting with a git clone

Install:
- TensorFlow
- Bazel
- Android Studio (optional)
- Android SDK
- Android NDK

Config:
- Edit tensorflow/WORKSPACE

```
android_sdk_repository(
    name = "androidsdk",
    api_level = 23,
    build_tools_version = "25.0.2",
    path =
    "<path-to-android-sdk>",
)
android_ndk_repository(
    Name = "androidndk",
    Path = "<path-to-android-ndk>",
    api_level=14
)
```

You can build a TensorFlow-shared object on Android from Android Studio using a continuous integration tool called Bazel.



And for iOS, we added CocoaPod integration as well. It’s quite simple.

Understand how to Code with the API

```
c.inferenceInterface =
    new TensorFlowInferenceInterface(assetManager, modelFilename);

// Copy the input data into TensorFlow.
inferenceInterface.feed(inputName, floatValues, 1, inputSize, inputSize, 3);

// Run the inference call.
inferenceInterface.run(outputNames, logStats);

// Copy the output Tensor back into the output array.
inferenceInterface.fetch(outputName, outputs);
```

Let’s take a look how you can use the Tensorflow API.

The Android inference library integrates with Tensorflow for Java applications. The library is a thin wrapper from Java to the Native implementation and the performance impact is minimal.

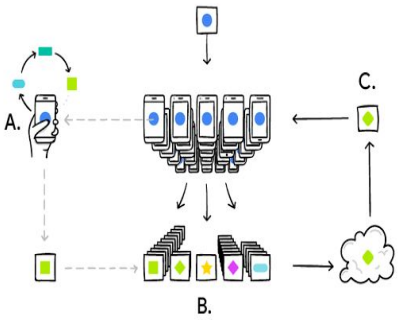
At first, create TensorFlowInferenceInterface, opening the model file from the asset in the APK. Then, set up an input feed using feed API. On mobile, the input data tends to be retrieved from various sensors, cameras etc.

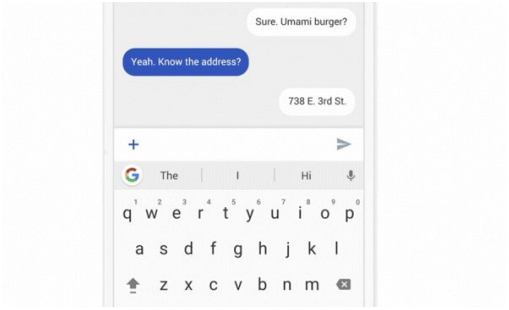
Then run the inference, and finally you can fetch the result using fetch method.

You would notice that those are all blocking calls.

So you would want to run them in a worker thread than the main thread since an API call would take long time.

Even though we have talked primarily about prediction on mobile,
a new frontier is federated learning





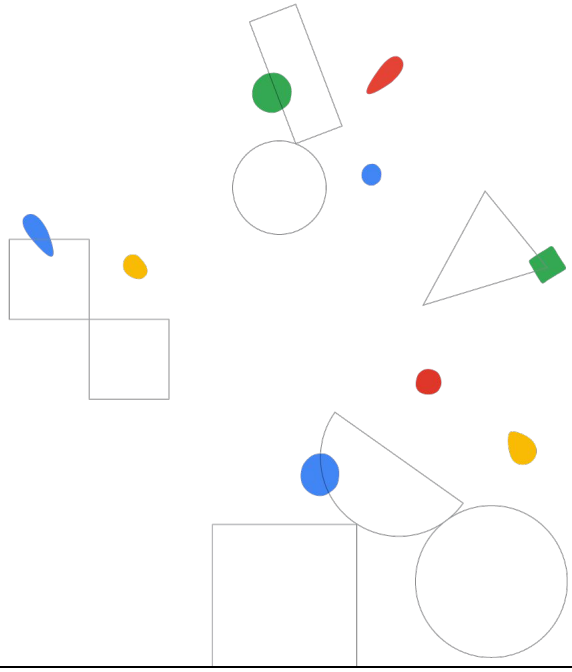
Federated learning in Google Keyboard
https://research.googleblog.com/2017/04/federated-learning-collaborative.html

Even though we have talked primarily about prediction on mobile, a new frontier is federated learning.

The idea is to continuously train the model on device, and then combine model updates from a federation of users' devices to update the overall all. The goal is for each user to get a customized experience, and still retain their privacy.

Google Cloud

Optimizing TensorFlow for mobile

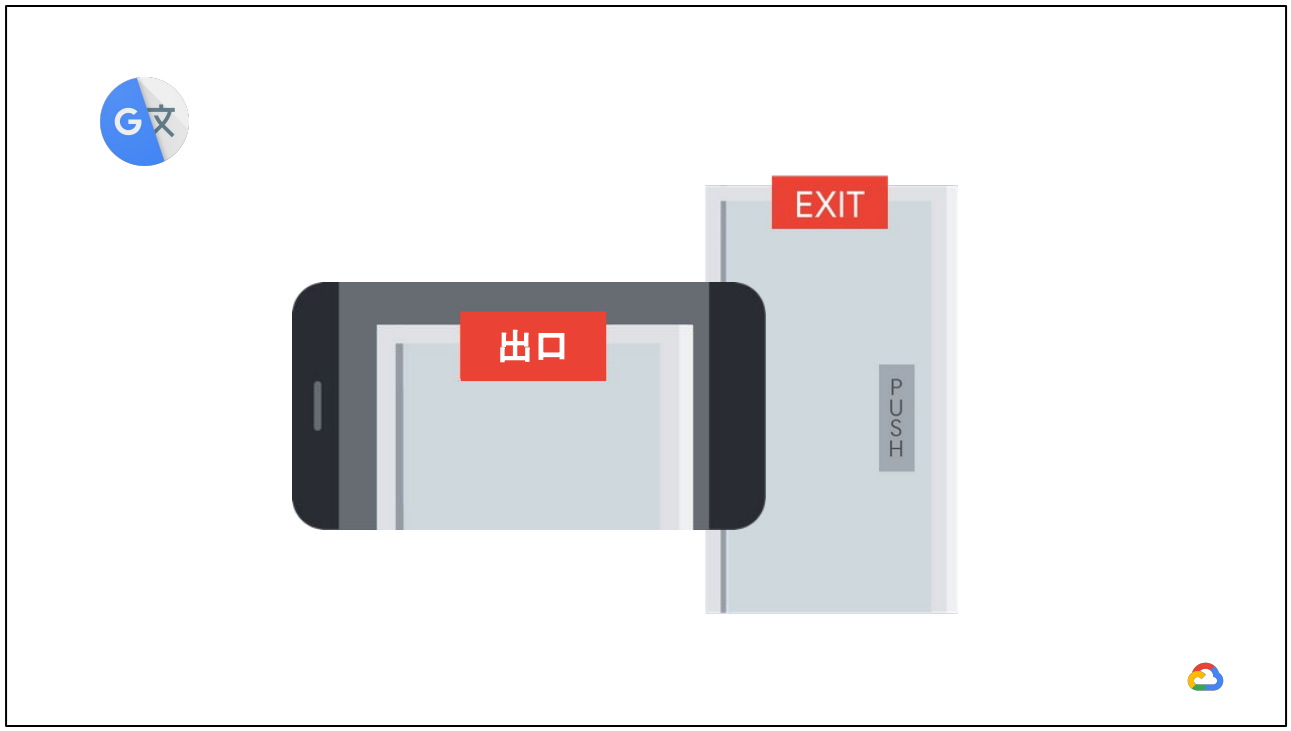


Module 04

Building Hybrid ML systems

Let’s look at a second scenario where hybrid models are necessary.

Earlier we explored how more and more applications are combining machine learning with mobile applications.



Take Google Translate, for example, which is composed of several models.

It uses one model to find a sign, another model to read the sign using optical character recognition, a third model to translate the sign, a fourth model to superimpose the translated text, and possibly even a fifth model to select the best font to use.

Add some intelligence

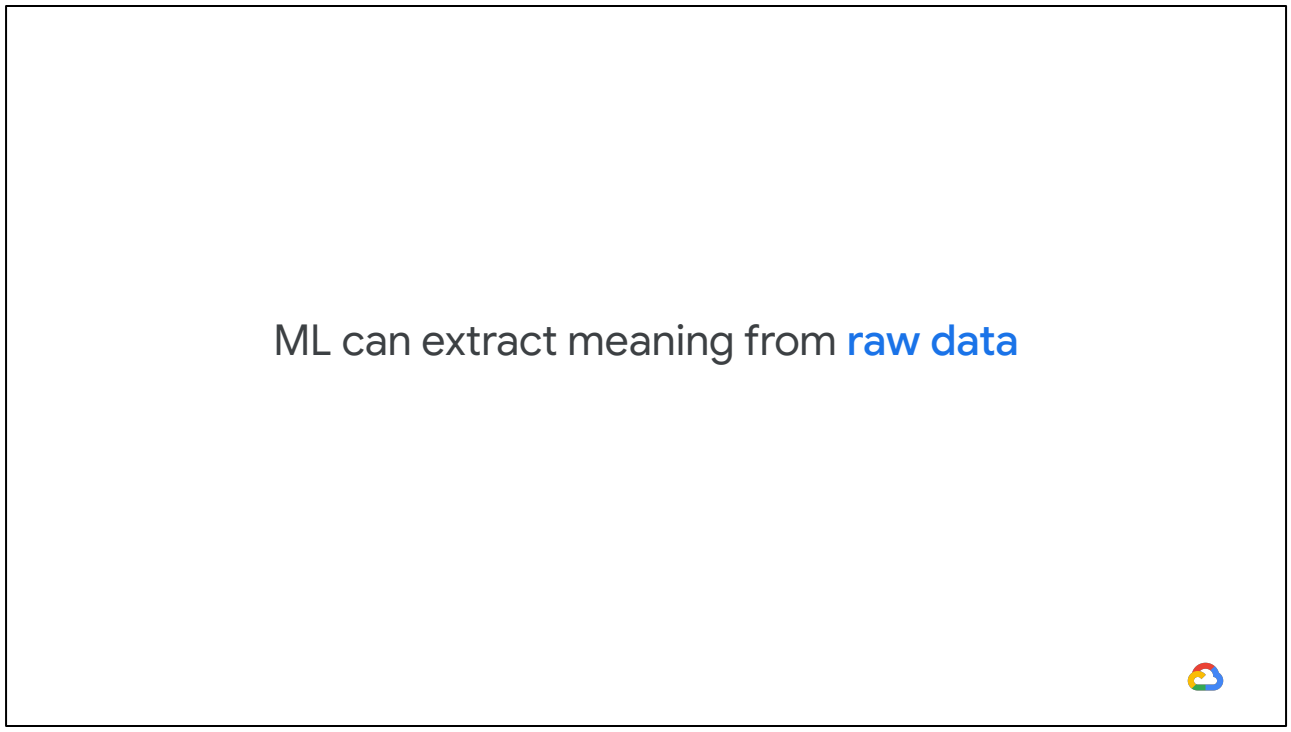
- Image and voice recognition
- Translation
- Natural language processing

ML allows you to add some "intelligence" to your mobile apps, such as image and voice recognition, translation and natural language processing.

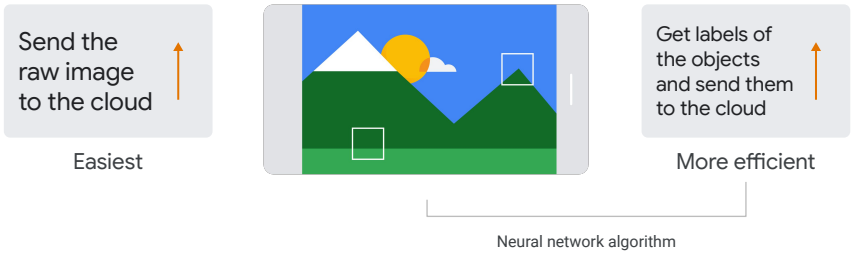
Add some intelligence

- Image and voice recognition
- Translation
- Natural language processing
- Smarter analytics

You can also apply machine learning to gain smarter analytics on mobile-specific data. For example, to detect certain patterns from motion sensor data, or GPS tracking data.

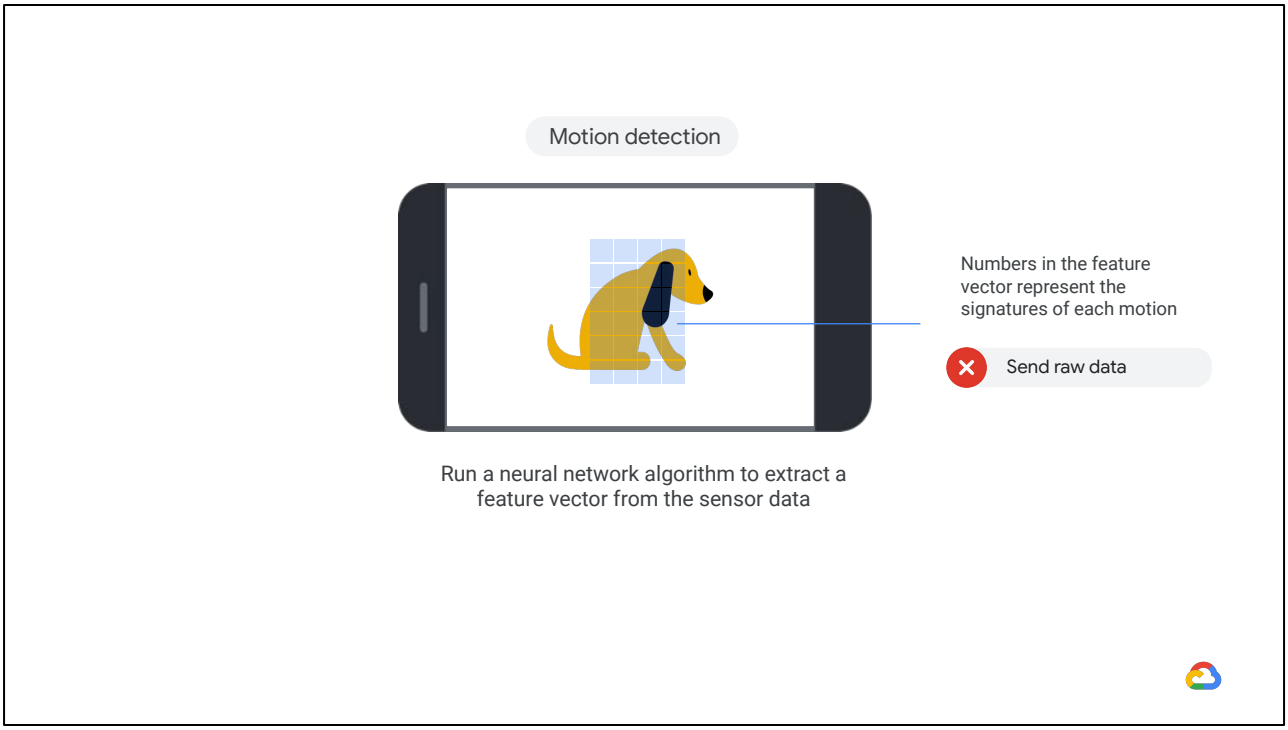


This is all thanks to the fact that ML can extract meaning from raw data.



So, if you want to perform image recognition with your mobile app, the easiest way is to send the raw image to the cloud, and let the cloud service recognize the objects in the image.

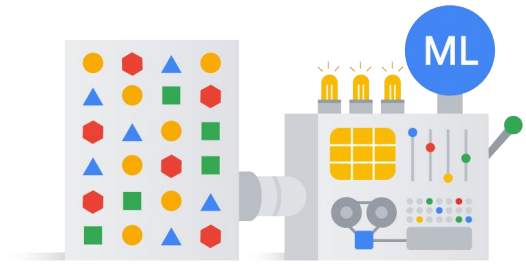
However, if you have a neural network algorithm running on your mobile app, you can get labels of the objects and send them to the cloud. It's a more efficient way to collect the object labels on the cloud service.



Now let’s say you perform motion detection with your mobile app. In this case, you can run a neural network algorithm to extract a feature vector from the sensor data.

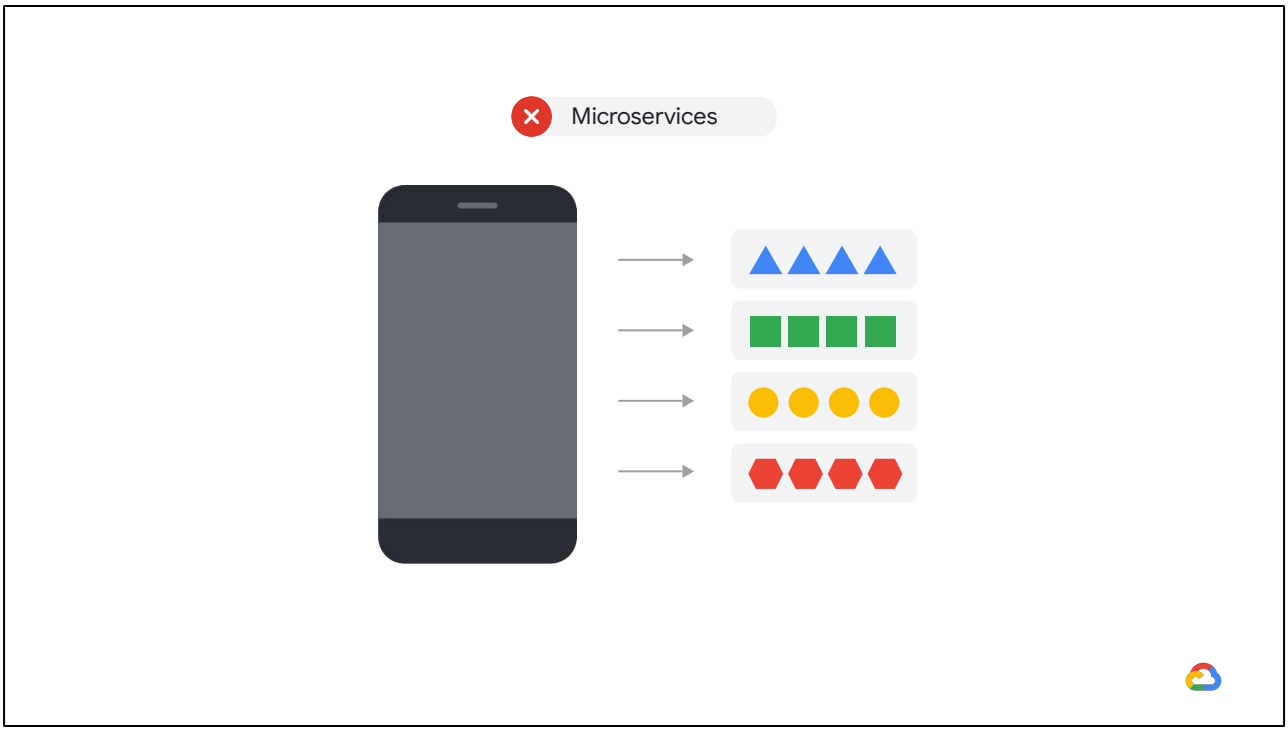
The numbers in the feature vector represent the "signatures" of each motion. This means you don't have to send the raw motion data to a cloud service.

By applying ML to mobile apps



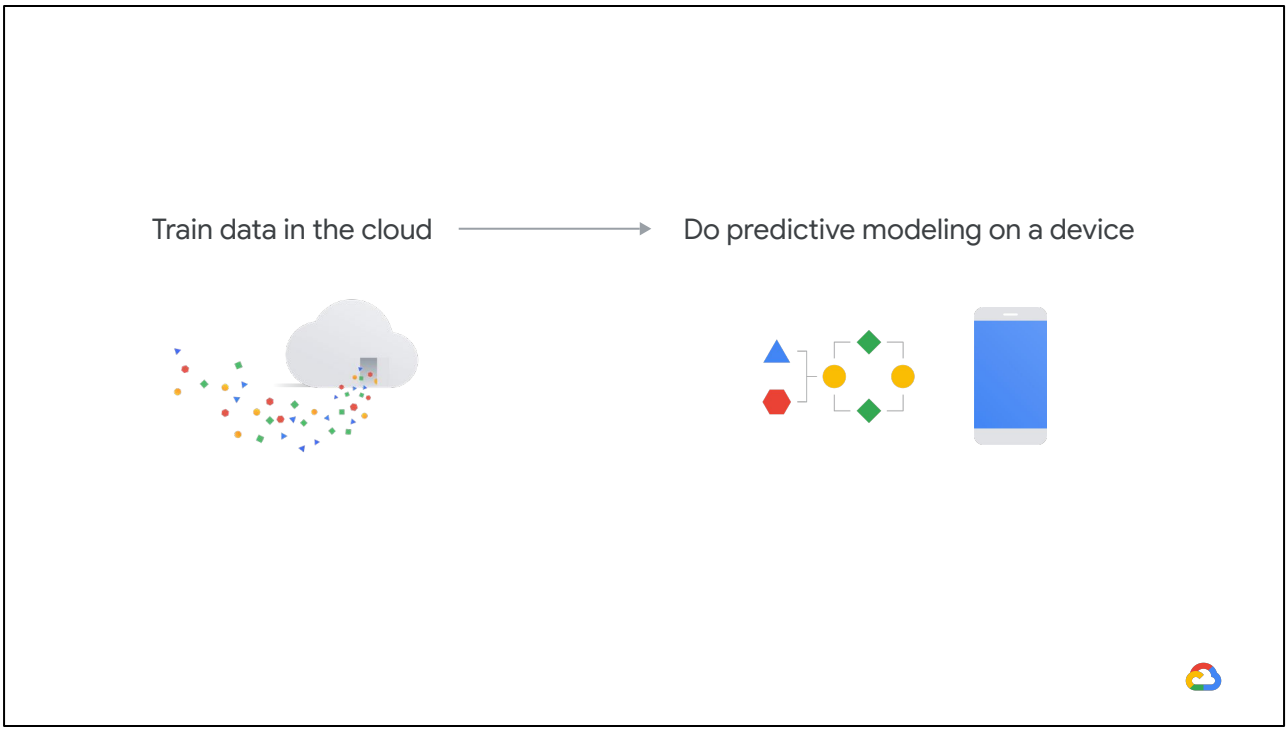
- Reduce network bandwidth
- Get faster response times

Also, by applying machine learning to mobile apps, you can reduce network bandwidth and get faster response times when communicating with cloud services.



It’s important to note that you often can’t use the microservices approach for mobile devices, as they can add unwanted latency.

Since you can’t delegate to a microservice, like you can when running in the cloud, you’ll now want a library, not a process.



In these types of situations, it's best to train models in the cloud and do predictive modeling on a device.

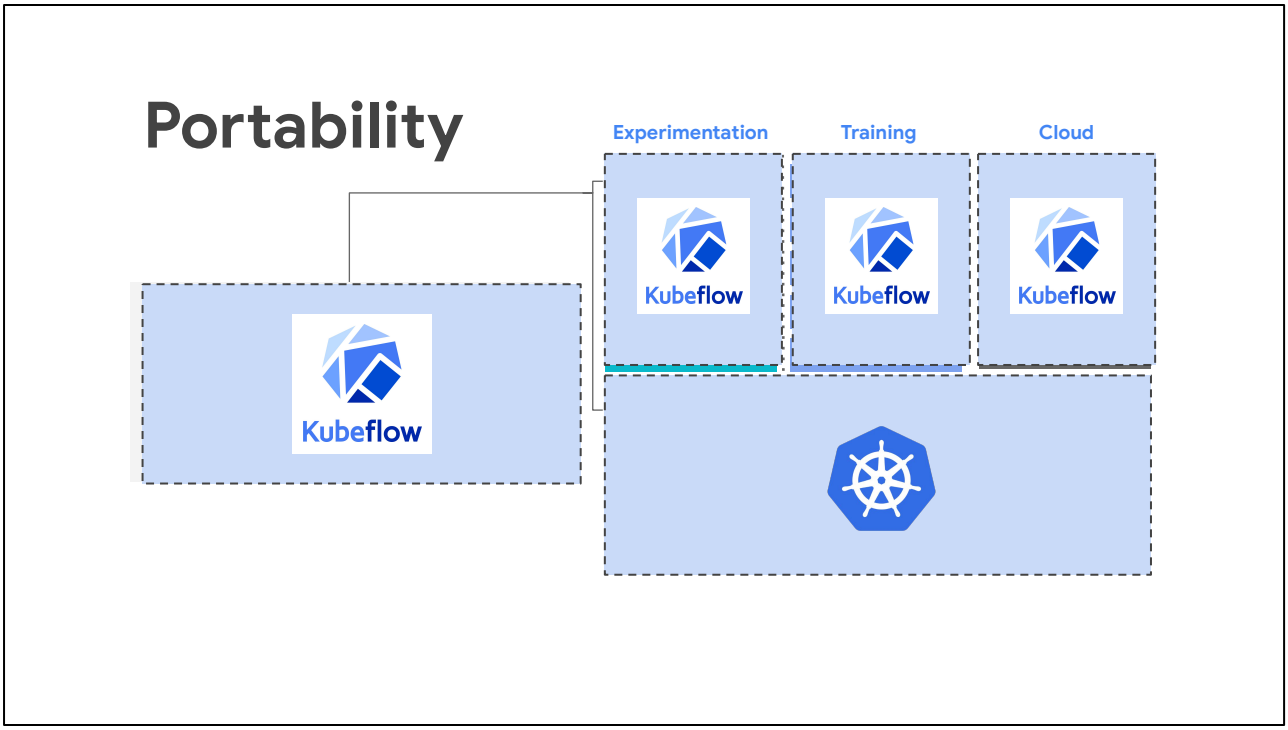
This means embedding the model within the device itself.

Summary

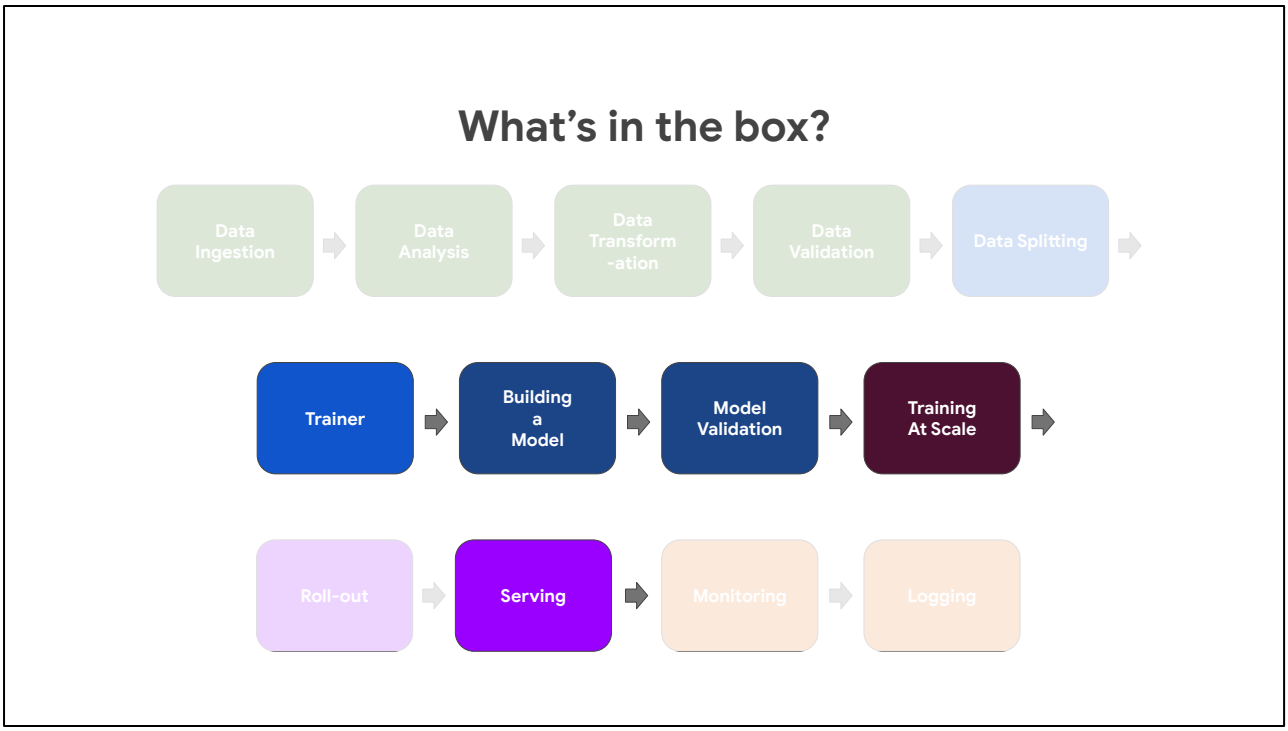
Build hybrid cloud machine learning models

Optimize TensorFlow graphs for mobile

In this module, we showed you two technologies -- Kubeflow and TensorFlow Lite -- that are important in hybrid machine learning systems.



Kubeflow gives you composability, portability and scalability while preserving the ability to run everywhere.



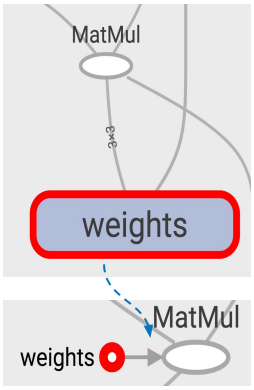
Specifically, Kubeflow offers portability and composability between your on-prem environment and Cloud ML Engine.

The tradeoff is that Kubeflow is not serverless. You will have to do cluster management.

Still, retaining the ability to move to cloud and serverless at some point in the future, or for some fraction of your workloads, provides flexibility.

The presence of Kubeflow also limits lockin. You can always take your models off Google Cloud, and you have a way to continue training and serving those models.

Freezing a graph can do load time optimization



Converts variables to constants and removes checkpoints

TensorFlow Lite makes specific compromises to enable ML inference on low-power devices.

For example, you can convert variable nodes into constant nodes, which streamlines your model because constant nodes are embedded in the graph itself.

However, you sacrifice maintainability and portability since you can not resume training from that model graph.

Transform your graph to remove nodes you don't use in prediction

quantize_weights
quantize_nodes

Add quantization

Another compromise you might make is to use a less accurate model on device. Perhaps you quantize the nodes or use a smaller model.

Of course, we hope that you choose to train and serve ML models on Google Cloud so that you don't have to make these compromises.

But if business and real-world considerations require you to be able to train or serve machine learning models outside a cloud environment, it is good to know that you do have options.

Kubeflow and TensorFlow Lite are good to know about. To have in your backpocket when such situations arise.