**The Value of Apis.**

**Introduction.**

Welcome to module 3, the Value of APIs. What are they? I'll explain in just a moment. So far in this course, I've explored an organization's technical foundation in the cloud. Many businesses have a variety of systems and applications, and for traditional enterprises, many of those systems and applications were built on-premises

For traditional companies, legacy systems and applications are complex, expensive to maintain, and do not provide the speed and scale required to deliver seamless, digital experiences that consumers now expect. When it comes to digital transformation, companies typically have the following three primary goals-- modernize IT systems such as compute solutions,

modernize applications so they can remain relevant in today's comm era, and thirdly, leverage application programming interfaces or APIs to unlock and create value for customers. So in this module, I'll explore how APIs are a tool for both digital integration and digital transformation. I'll begin by defining legacy systems and identify why they struggle

to meet the demands of the digital age. Then I'll define APIs and how the can modernize legacy systems. Next, I'll explore examples of how APIs create new business value. And finally, I'll look at Apigee, a Google Cloud solution for developing and managing APIs. Let's jump in.

**Cloud change patterns.**

It's often assumed that modernizing an application with cloud technology can only be done in one way; move everything to the cloud all at once.

Yikes.

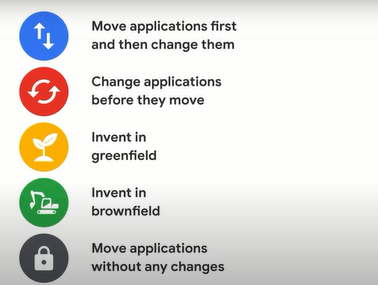
That can be risky, especially for large applications.

The good news is that's just one approach.

Moving an application to the cloud doesn't need to be done all at once.

Google Cloud has identified five common patterns that businesses can adopt when they want to modernize their applications.

A business can **move applications to the cloud first and then change them**, or they can **change their applications before they move**, **or they can invent in greenfield(“*nuevo terreno*”)**, **or invent in brownfield (“*terreno abandonado*”)**, **or they can just move their applications without any changes**.



Let's look at each of these in turn.

If an organization wants to take a relatively conservative approach to modernizing applications with the cloud, they might take a **move first then change approach**. This path typically starts with a lift and shift program for selected applications. The migration of these applications typically brings minimal changes to the ways of working within the organization, but once the applications are running in the cloud, they are then ready to be updated more easily than when they were running on premise. For example, a legacy application that is moved to the cloud could have its security improved by using the enhanced firewall and identity access management, IAM, capabilities of Google Cloud. Over time, further modernization can be explored potentially using APIs to change the way that the application interacts with data and other applications, or even making the application serverless so that it can become cloud native event-driven application, the most efficient form of application architecture.

After the first set of applications have been re-architected and optimized in the cloud, further applications can be moved. Think of this like renovating your house to maximize your space. You don't have to renovate every room all at once and you don't want to completely redo every room either. You can start to make changes as you're ready for them based on your needs and budget. Suppose, for instance, you want to start with the kitchen. You could replace the kitchen cabinets and countertops and still continue to use the electronic oven as is. Eventually, after you've put in a gas line, you can then replace your electric oven with a gas range.

If an organization wants to take a more aggressive approach to modernizing its applications, they can re-architect applications first to make them more cloud ready before migrating them. For our analogy, that might mean completely changing the design of the kitchen and the placement of the appliances for maximum efficiency and buying brand new appliances for the new design before doing the renovation work. For some organizations, their initial interest in the cloud is because they want the ability to build new, innovative applications quickly. They may not want to or be ready to move existing applications at this point. So when we talk a **greenfield** strategy, we're talking about building an entirely new infrastructure and applications in the cloud. It's like creating an office and buying furniture for it as part of your renovation project when you don't currently have an office in your existing home. This approach really only applies when an organization needs to develop new products or offerings, such as a B2C bank that wants to develop its digital banking channel. The organization doesn't need to touch older applications just yet. They could take either the move and change or change and move approach if they decide to modernize them at a later point. Inventing in greenfield allows you to build that innovative application that will help drive the business forward, but it does require agility, access to a diverse development skillset, and strong support from leadership.

A **brownfield** strategy, on the other hand, is to invent a new application in the cloud environment that will replace an existing legacy application that remains on premises. The legacy application is only retired after the new application is built. In our analogy, it's like creating a new office in your house while continuing to use a cluttered desk space in the corner of a living room. You don't move any furniture or reorganize your documents until you know the new office space is set up. Although this redundancy can be comforting through minimizing risk, especially for mission critical applications, there are increased costs associated with running applications in both places.

Finally, it's worth noting that building cloud native applications isn't appropriate for all scenarios.

For some use cases, It's efficient to leverage the cloud just to modernize the infrastructure layer as we discussed in the previous module.

One possible use case is cloud storage for data to allow organizations to decommission on premises data centers.

Another use case is modernizing the infrastructure only to allow organizations to create a virtualized environment for disaster recovery.

Over the next few videos, we'll look at how the cloud can support application development and maintenance.

**Challenges in application development.**

Many businesses professionals share similar concerns around application development processes and timelines.

Creating a new application within an organization can be a challenge.

Have you had the experience of going to your tech team and suggesting a new application, only for them to tell you it will take 18 months or maybe even tell you it's not possible with the legacy systems already in place?

Traditionally, when business professionals want a new application, the tech or IT team has to do a lot of work to identify features, estimate capacity, define a technical architecture, consider integration with other systems, and allocate resources even before a line of code is written.

Once the requirements are agreed on, a new application will have to be designed, built, tested, integrated, and deployed.

But new needs often compete with existing projects for time and resources.

For some teams, this means spending just as much time creating and managing environments as is spent building business value.

Whether building an app on premises or in the cloud, developers still need to make decisions about overall network architecture, choice of database, and type of server.

All of these can slow down the application development process and even the launch of applications.

The challenges for building apps using an on premises infrastructure can outnumber those of cloud native apps and can often be frustrating for developers and business professionals.

Developers want to be creative and innovative by building new solutions, not spending hours maintaining the infrastructure.

When developers get too far removed from the task they enjoy, naturally, they start to seek out more interesting job opportunities that allow them to focus on building new apps and technologies.

In addition to losing a key team member that needs to be carefully replaced quickly, the organization loses the intangible knowledge that good developers take with them when they leave.

Developing cloud native applications avoids the hassle of trying to create something that is constrained by legacy systems and outdated processes.

Building a new application in the cloud means you can be more agile in your development.

It frees teams up from worrying about environment so they can focus on creating features which is where customers will get real value.

Updating already existing applications that have been typically built on premises presents difficulties, too.

Often, an application has been built with a monolithic architecture.

This means that as it's updated over time, its code base becomes bloated, making it difficult to change something without breaking something else.

And when an application is updated, the entire application needs to be deployed and tested, even if the change is only small.

This makes implementing updates a lengthy and potentially risky process.

When building new applications or modernizing existing ones, a microservice architecture can reduce these problems.

This type of architecture involves the separation of a large application into small, loosely coupled services.

The code base for each service is modular so it's easy to determine where the code needs to be changed.

And when a code change is required, the service can be updated and deployed independently.

In addition, each service can be scaled independently depending on its specific requirements.

Adopting an automated continuous integration and continuous deployment approach, also known as CI/CD, can help you increase your application release velocity and reliability.

With a robust CI/CD pipeline, you can test and roll out changes incrementally instead of making big releases with multiple changes.

This approach enables you to lower the risk of regressions, debug issues quickly, and roll back to the last stable build if necessary.

It also means you can update applications without interrupting services to your users.

Imagine being able to deliver new features to your customers every day instead of a few times a year.

Here's the important bit.

Some organizations have been able to adopt CI/CD to build applications faster but not always with the high quality that customers demand.

This is because they don't invest enough in building quality into the process.

When building an application, you need to consider how quickly your systems can recover from downtime.

If you're not able to recover from production infrastructure failures quickly, it doesn't matter how quickly you deliver software, you won't be able to deliver better customer experiences.

Google Cloud Developer Tools help you release software at a high velocity while balancing security and quality.

There are two tools we'll look at in this module: Google Kubernetes Engine and App Engine.

You might remember Google Kubernetes Engine from the last module.

Let's explore how it enables businesses to be more agile in app development.

**Google Kubernetes Engine.**

In the last module, I introduced containerization as a one-compute option for modernizing IT infrastructures.

In the context of application development, containerization allows developers to divide an application design into individual compartments.

The advantage is that parts of the code can be updated without affecting the whole application.

It also builds resilience because one error doesn't impact the whole application.

As businesses create and scale more and more applications, they need a way to manage or orchestrate their containers.

**Kubernetes is an open source container orchestration system for automating computer application deployment, scaling, and management.**

Google developed Kubernetes originally to support their own internal operations.

As with so many innovations, Google then made it available to anyone as open source technology.

Google Kubernetes Engine often shortened to GKE is the Google Cloud manage service for container orchestration.

GKE enables rapid application development and iteration by making it easy to deploy, update, and manage your applications and services.

Now, you might remember serverless computing from our previous module as another option for modernizing IT infrastructures.

The same option can be used for application development.

With serverless computing, your cloud provider manages even more of your architecture.

You write the code for the functions you want, and the cloud provider updates and adapts the container or VMs as needed to make that change.

Let's move on to the second Google Cloud solution for app modernization, App Engine.

It leverages serverless computing to enable businesses to develop applications.

Click on the next video to find out more.

**App Engine.**

I mentioned App Engine in the last module.

App Engine is a platform for building scalable Web applications and mobile back ends.

It allows you to concentrate on innovating your applications by managing the application infrastructure for you.

For example, when you're building an application, App Engine manages the hardware and networking infrastructure required to run your code so developers no longer need to spend valuable time doing this.

During deployment, App Engine will scale your application automatically in response to the amount of traffic it receives so you only pay for the resources you use.

Just upload your code, and Google will manage your app's availability.

You can easily run multiple versions of your app to test new features or designs with end users.

And because there are no service for you to provision or maintain, the monitoring and maintenance processes are easier too.

Let's look at an example.

EDP is one of the world's leading utility companies with a presence in countries across Europe, North and South America, and Asia.

It's an end-to-end operator involved in the generation, distribution, and trading of electricity and gas.

As a large company responsible for diverse operations, EDP has a complex IT infrastructure with over 400 applications.

Many of EDP's IT systems were legacy systems not designed to integrate with one another, leading to inefficient delivery of data.

In particular, EDP was experiencing problems with the performance of its customer account mobile app, which allows customers to check their usage, account, and payment details for their electricity and gas accounts.

EDP also needed additional capacity to meet peaks in demand.

To address these issues, EDP rebuilt the app in only two months using App Engine.

The auto-scaling functionality in App Engine means that their new app easily scales to meet peaks in demand, and customers can now access their data even when EDP's back end systems are under maintenance.

The new app has delivered significant gains for EDP in terms of both performance and customer satisfaction.

After EDP migrated its customer service app to App Engine, the average page loading time decreased by almost 90%, and its App Store reviews ratings jumped from 1.9 to 4.7 in just a couple of weeks, with downloads increasing as a result.

There are many more examples of customers who have leveraged to Google Kubernetes Engine and App Engine as part of their digital transformation.

Click on the links provided in the reading to learn about three customers in particular who increased developer velocity and provide amazing customer experiences.

**Quiz.**

1.What is a critical outcome of API management? Select the correct answer.

Digitizing and modernizing siloed business data.

Distributing and creating regular new business features.

Updating and repairing outdated business components.

Measuring and tracking business performance.

2.How can businesses use APIs to unlock value from their legacy systems? Select the correct answer.

By monitoring cloud resource usage of new applications.

By analyzing data from new applications.

By re-architecting their legacy systems.

By gaining access to data stored in legacy systems.

3.Michelle wants to manage her team's APIs and provide security policies for identity verification, authentication, and access control. What Google Cloud solution should she choose? Select the correct answer.

BigQuery

Google Kubernetes Engine

Cloud Identity

Apigee

4.What is the function of APIs? Select the correct answer.

They offer hybrid data storage.

They provide real-time analytics.

They enable rapid autoscaling of data.

They enable integration between systems.

5.Why do legacy systems struggle to meet modern consumer expectations? Select the correct answer.

They rapidly surpass physical capacity.

They only serve real-time data.

They ineffectively process batch data.

They scale slowly.