Microsoft

Choosing a Cloud-Connected Backend for your Android, iOS, HTML5 or Windows App: BaaS, PaaS or IaaS?

Contents

Introduction

As a mobile app developer today, you make a number of decisions that significantly impact development time and app performance, not the least of which is how to implement your app's backend. Should you build it all yourself and host it on your own on-premises Linux or Windows machines or should you instead utilize virtual machines from a public cloud provider? If you go with a public cloud provider, should you stand up your own backend on Linux or Windows VMs (IaaS) or take advantage of things like OS patching from platform services (PaaS)? Perhaps you've even come across higher-level platform services such as Backend-as-a-Service (BaaS)—sometimes called Mobile Backend-as-a-Service (MBaaS) but still relevant to Windows 8 and Mac OSX client apps so more commonly referred to as BaaS—and are wondering if that's for you.

And if you do go the BaaS route, what criteria should you consider when evaluating BaaS providers?

This paper will help answer some of those questions by providing an overview of the pros and cons associated with each option.

This whitepaper will cover:

1. The range of options for hosting an app backend in the cloud—from mobile backend services to platform services to infrastructure services

    a. Definitions of Public Cloud, Hybrid Cloud, IaaS, PaaS, and BaaS
    b. General Considerations
    c. Mobile Apps with backends hosted in BaaS
    d. Mobile Apps with backends hosted in general PaaS
    e. Mobile Apps with backends hosted in IaaS

2. Considerations when evaluating a BaaS provider

    a. What every BaaS should provide
    b. General Considerations

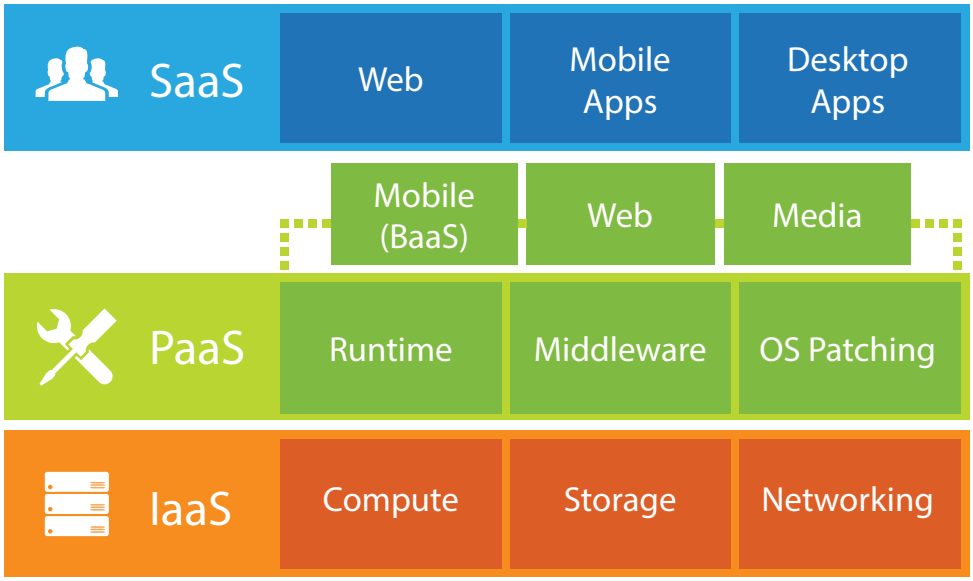
The Range of Options

From Backend-as-a-Service to Platform-as-a-Service to Infrastructure-as-a-Service

Definitions of IaaS, PaaS and BaaS

a. Public Cloud

The public cloud refers to compute, storage and networking resources provided by third parties such as Amazon Web Services, Microsoft Windows Azure and Google Cloud Platform on a subscription basis. These resources are typically consumable either as raw resources or packaged as higher level platform services. Developers and IT Pros typically adopt the public cloud for a combination of reasons commonly including speed, scale and economics.



Regulatory requirements and/or privacy concerns may prevent some workloads from ever fully moving to the public cloud. Those workloads can often still leverage certain cloud services—like a broadcast push notification service that assists with routing information through the necessary third-party servers or a cloud-based NoSQL option for logging. Some of the information herein will be relevant for this scenario, but the primary focus is for apps that can more heavily adopt the public cloud.

b. Hybrid Cloud

The term hybrid cloud encompasses scenarios where either an application utilizes both on-premises and public cloud resources or an application utilizes both infrastructure and platform services. This is typically accomplished using an integration service.

See: Windows Azure Service Bus, AWS Direct Connect, StorSimple, Windows Azure BizTalk Services

A note on saving and consuming data in the cloud

The major cloud providers typically offer tailored cloud-based services for storing relational data (SQL or MySQL), non-relational data (NoSQL), and binary objects (blobs). There can be significant differences between cloud-based data services and their on-premises versions, however. For that reason, and the fact that some sensitive data will never be able to move to the cloud for regulatory reasons, it's important to make sure your selected cloud provider not only offers its own relational, non-relational and binary storage options, but also options like SQL Server in a VM, access to third-party MongoDB, and the means by which to access on-premises data from mobile devices without storing it in the cloud.

No matter what cloud provider you choose for hosting your app's backend, always make sure that you have full control not only over the way it is saved, but also over where it is saved. Approach with significant caution if a cloud provider (BaaS or otherwise) only offers you the option to generally save an object to their service without designating it as relational, non-relational or binary. Dial that caution up a notch if that provider also fails to provide you with the option to designate to which data center(s) your data is saved and the ability to export the data without taking down your entire application.

c. IaaS (Infrasturcture-as-a-Service)

IaaS refers to on-demand infrastructure (typically, raw compute in the form of virtual machine instances) that scales and adapts to your variable business needs.

See: Windows Azure Virtual Machines, Amazon Web Services EC2, Google Compute Engine,

d. PaaS (Platform-as-a-Service)

PaaS is intended for net new development that can fully leverage the benefits of a public cloud environment and computing across pooled resources.

Applications running in PaaS typically enjoy additional benefits like OS patching and continuous availability (even during system upgrades). PaaS enables you to quickly build multi-tier applications, distribute processing across multiple resources, and scale each tier independently.

See: Windows Azure Cloud Services, Windows Azure Web Sites, Amazon Web Services Elastic Beanstalk, Google App Engine, Heroku

e. BaaS (Backend-as-a-Service or MBaaS, Mobile Backend-as-a-Service)

BaaS is a higher-level platform service specifically optimized for connected client development—whether that client is a mobile device, tablet or laptop. BaaS supplies a turnkey backend that provides some means of storing data in the cloud, authenticating users, and sending push notifications.

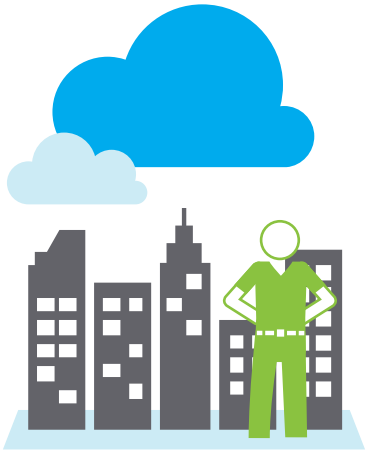
See: Windows Azure Mobile Services, Kinvey, Parse,

f. High-Volume, Low-Latency Push Notification Service

Although BaaS providers typically include push notifications as part of a turnkey backend, those push infrastructures are typically optimized for logic-based 1:few push notifications rather than a broadcast to millions of users at once or inbox scenarios with frequent notifications to each user. A high-volume, low-latency push notification service, which can and should be used in conjunction with BaaS push, is optimized to deliver a very large number of personalized push notifications in a very short amount of time.

In a turn-based game, for example, you would want to use BaaS push to alert Player 1 that Player 2 has just completed his or her turn, whereas you would use a high-volume, low-latency push notification service to send one of three offers to every user at once based on their favorite game or inbox turn-by-turn game and chat records.

See: Windows Azure Notification Hubs, Urban Airship, Amazon SNS Mobile Push



General Considerations

The central tension in deciding between infrastructure and platform services is degree of control and customizability versus speed of deployment and ease of maintenance.

If you have an existing backend hosted on-premises in Windows or Linux environments and want to move that to the cloud in order to take advantage of its elastic scalability, then you're typically looking for infrastructure services. A move to any platform service will typically require re-architecting your app.

Mobile Apps with backends hosted in mobile specific Backend-as-a-Service (BaaS)

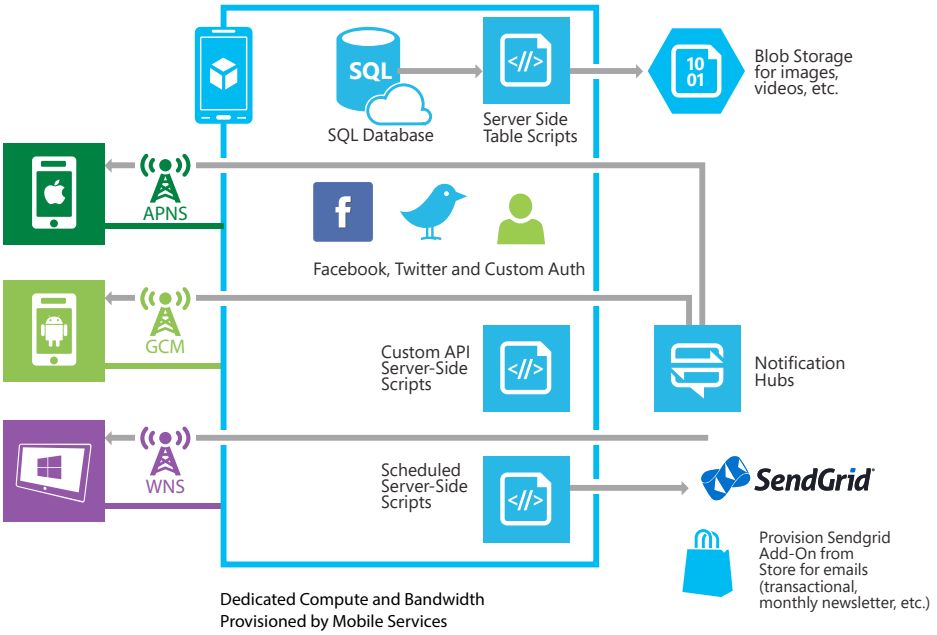
The primary advantages of building your iOS, Android or Windows apps on BaaS are that you:

- Accelerate your development time by leaving infrastructure provisioning and scaling to the provider
- Can share a single backend across apps available on multiple client platforms
- Receive a ready-made server-side coding environment for easily adding business logic to your app
- Benefit from multiple turnkey services specific to mobile app development—typically data, auth and push
- Are free to focus on delivering a phenomenal and differentiated user experience
- Can leave software patching and protocol updates to the BaaS provider

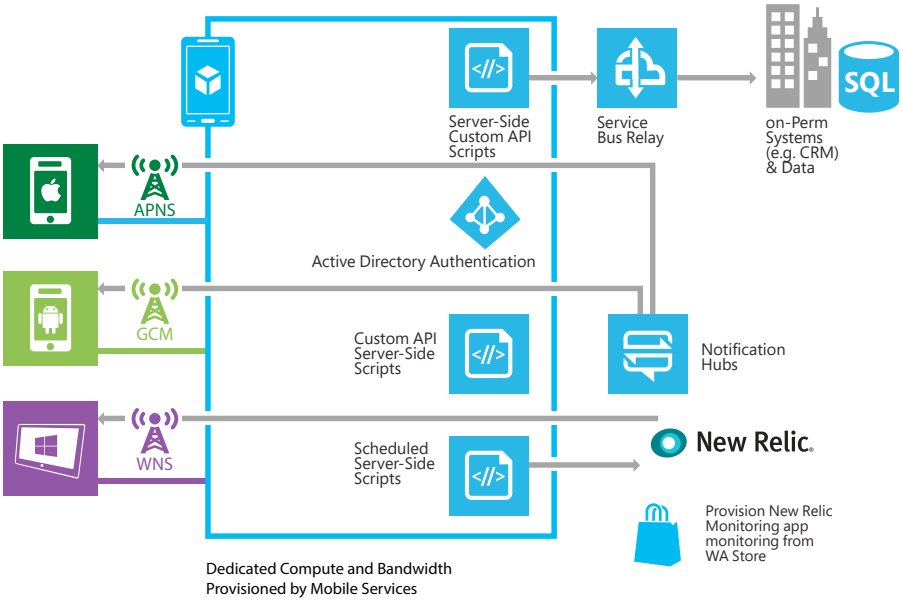
When you no longer need to manage your backend, you not only reduce development costs but also realize operational cost savings that are often far more significant. For many apps, reducing operational costs to nearly zero is the greatest benefit that BaaS affords.

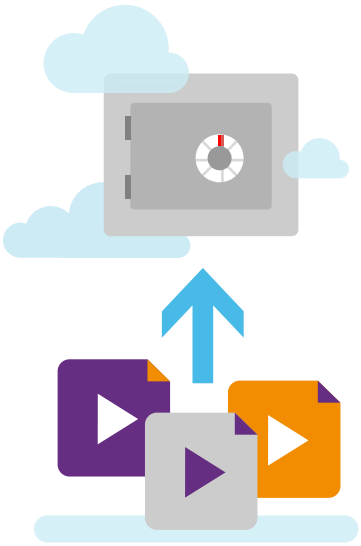
Windows Azure—alone among the three major cloud providers—offers a BaaS solution. Mobile Services, the Windows Azure backend-as-a-service, makes it quick to get started with data, authentication, push notifications and custom business logic for apps on iOS, Android, Windows and more. Note that, unlike many BaaS providers, Mobile Services provisions dedicated resources for your app backend so that you can enjoy consistent performance as you scale and avoid noisy neighbor problems. When you scale Mobile Services to consume multiple units, under the hood you're securing additional dedicated compute instances.

A standard greenfield app architecture might look like:



A standard app architecture that needs to also consume on-premises data might look like:





This architecture highlights a great characteristic of BaaS solutions—that apps available on multiple client platforms can still utilize the same backend.

If you have an app available on the iOS, Android and Windows platforms, for example, you can point each of them to the same cloud-based Mobile Services backend (named 'demoCrossPlat') in the following ways:

iOS (Objective-C)	self.client = [MSClient clientWithApplicationURLString:@"https://demoCrossPlat.azure-mobile.net/" applicationKey:@"yourMobileServicesIdentifier"];
Android (Java)	mClient = new MobileServiceClient("https://demoCrossPlat.azure-mobile.net/", "yourMobileServiceIdentifier", this);
Windows (C#)	public static MobileServiceClient MobileService = new MobileServiceClient("https://cloudvs.azure-mobile.net/", "hFleTmsVWpacTTmaTHZfJQrNWEvXAB51");

Once each client is connected to your backend hosted in Mobile Services and device tokens (iOS), registration ids (Android) and channel URIs (Windows) are stored in a 'DeviceInfo table, sending notification is easy. Say you want to send a notification anytime a new row is inserted in your app's SQL database. Without Mobile Services, that would require thousands of lines of code to establish a secure connection to your database, add a logic-based environment for server-scripts and open communication channels with three separate push notification services with very different protocols.

With Mobile Services, you can send a notification to all devices, regardless of platform, in a single script. Sending a notification to iOS and Android devices, for example, is as simple as selecting 'Insert' from the dropdown on your table and adding the following to that ready-made environment:

```
function sendNotifications() {
    var deviceInfoTable = tables.getTable('DeviceInfo');
    deviceInfoTable.where({ userId : user.userId }).read({
        success: function(deviceInfos){
            deviceInfos.forEach(function(deviceInfo){
                if (deviceInfo.uuid != request.parameters.uuid) {
                    if (deviceInfo.pushToken != null && deviceInfo.pushToken !=
'SimulatorToken') {
                        if (deviceInfo.platform == 'iOS') {
                            push.apns.send(deviceInfo.pushToken, {
                                alert: "New something created"
                            } , { //success / error block});
                        } else if (deviceInfo.platform == 'Android') {
                            push.gcm.send(deviceInfo.pushToken, "New something created", {
success / error block});
                        }
                    }
                }
            });
        }
    });
}
```

BaaS is generally a great option for net new mobile app development or adding new services to an app, like push notifications, even if the app needs to leverage on-premises data and systems (depending on the provider). With an integration technology, like Service Bus Relay, and a robust server-side scripting environment for writing and hosting custom APIs, it is possible to consume on-premises data and systems, such as CRM, from mobile devices.

Three canonical scenarios for using BaaS are the following:

- Greenfield (net new) mobile app development. Opting for a BaaS will significantly accelerate development time and reduce the ongoing cost of maintenance. By architecting your app for the cloud from day one using BaaS, you can reap the benefit of several value-add services.
- Improving local-only apps by adding a connected backend. By taking an app, particularly a game, from local-only to connected, you can dramatically increase user engagement. In games, for example, connectivity opens the door to global leaderboards, game history, multiplayer scenarios, push notifications and more. BaaS makes that all easier.

- From brand apps to customer-facing LOB and B2E (business-to-employee) apps, using a BaaS allows you to focus on features that deliver true value, instead of getting bogged down in glue code.

BaaS is generally not a great option in these two scenarios:

- You have an existing on-premises backend you're looking to lift and shift to the public cloud in order to better scale resources to match variable traffic. Virtual Machines are the best option in this case. (It's important to distinguish between moving an existing backend to the cloud and consuming existing on-prem resources in a backend obtained through a BaaS provider. In the former, the backend has already been written and changing it to fit any platform service, general or specialized, instead of a virtual machine, would require re-architecting for that platform. In the latter, the backend is supplied by the BaaS provider and hosted in the cloud and thus has a different architecture than a backend built for on-prem VMs. Even though the backend is hosted in the cloud, select BaaS providers provide a powerful enough server-side scripting environment for hosting APIs that, when combined with integration technologies, on-prem data can be accessed through a mobile application, even though it never actually moves to the cloud.)
- You have an existing backend hosted in the public cloud—either on VMs or general platform services. Moving completely to a BaaS will typically involve significant re-architecting, although it is still possible to leverage standalone BaaS services like user authentication or push notifications with only minor configuring.

Also consider that, with a BaaS, you're often somewhat limited in your server-side language options. If your backend has be built in PHP or Python, then you probably require more granular control that you'll get with a BaaS and should consider a general PaaS option.

Mobile Apps with backends hosted in general PaaS

PaaS offers a higher-level abstraction on top of IaaS where you are no longer responsible for runtime, OS or certain middleware. Although PaaS options will generally accelerate net new development, they will not typically supply any features that are tailored to accelerating mobile development specifically. You will still need to stand up your own user authentication and push notification systems.

PaaS is a great option if you are looking to stand up a mobile backend soup to nuts in your preferred language, but don't want to think about patching the OS if security vulnerabilities surface. (Often higher-level services, though supporting native development for major mobile platforms, will restrict your language options for backend

development.) Also, PaaS provides predictable performance in a dedicated environment should you so designate, no noisy neighbor surprises if you so choose.

Keep in mind that general platform services still don't provide any turnkey services specific to mobile—you will still have to either stand up your own authentication and push notification systems or hook into third party APIs. General platform services like this will provide you more fine grained control over scale than a higher level BaaS, however, because you're still scaling based on compute and bandwidth on particular roles rather than API calls across your entire app (common among BaaS providers).

a. Windows Azure Cloud Services (Web/Worker Roles)

Windows Azure Cloud Services is Microsoft's general PaaS offering and lets you build multi-tier applications comprised of web and worker roles. Both web and worker roles contain application files and a configuration; the only difference between the two is that web roles come with a dedicated IIS web-server. That makes web roles ideal for hosting front-end web applications and worker roles better suited for apps that are asynchronous, long-running or perpetual (and independent of user interaction).

Using multiple roles allows you to distribute processing and scale each tier independently of the other. With Cloud Services, Microsoft handles routine maintenance, OS patching and triggering recovery from service and hardware failures.

In building a mobile app using Cloud Services, you will use a combination of web and worker roles in conjunction with a data store that can be either on-premises or in the cloud. The benefit of using web and worker roles versus virtual machines (IaaS) is twofold. First, you of course do not have to perform certain maintenance like OS patching. Second, you can more widely distribute the work amongst roles and scale each independently. Imagine, for example, that you have an application where users can take, edit and send photos. If a small percentage of users are each uploading many photos at once, you may not need to scale your front end web roles but can easily dial up the worker roles working on backend processing. If a larger percentage of users are each uploading a smaller number of photos, then you may need to scale your front end web roles but not the worker roles processing the images in the background.

b. Amazon Web Services (AWS) Elastic Beanstalk

AWS Elastic Beanstalk provides a layer on top of AWS EC2 images that makes it easier to deploy applications to the AWS cloud because once you upload your application, Beanstalk takes care of capacity provisioning, load balancing, auto-scaling and app health monitoring. With Beanstalk, it's easier to deploy new application versions to running environments (or rollback to previous versions), easy access to server log files and the ability to restart the application servers.

Beanstalk supports development in a variety of languages and frameworks

including .NET, node.js, PHP, Python, Java and Ruby. When building a mobile app on Beanstalk, you will follow essentially the same process as with building a mobile app on EC2 because Beanstalk is really just providing a management layer on top of the raw virtual machines.

It is important to note that, even though Beanstalk is marketed like a traditional PaaS, you still carry certain responsibilities, like patching the OS, that you do not with other PaaS providers.

c. Google App Engine (GAE)

Google App Engine (GAE) offers storage of properties and entities through App Engine Datastore as well as the option to use Google Cloud Storage for large and/or binary objects like images or other media files.

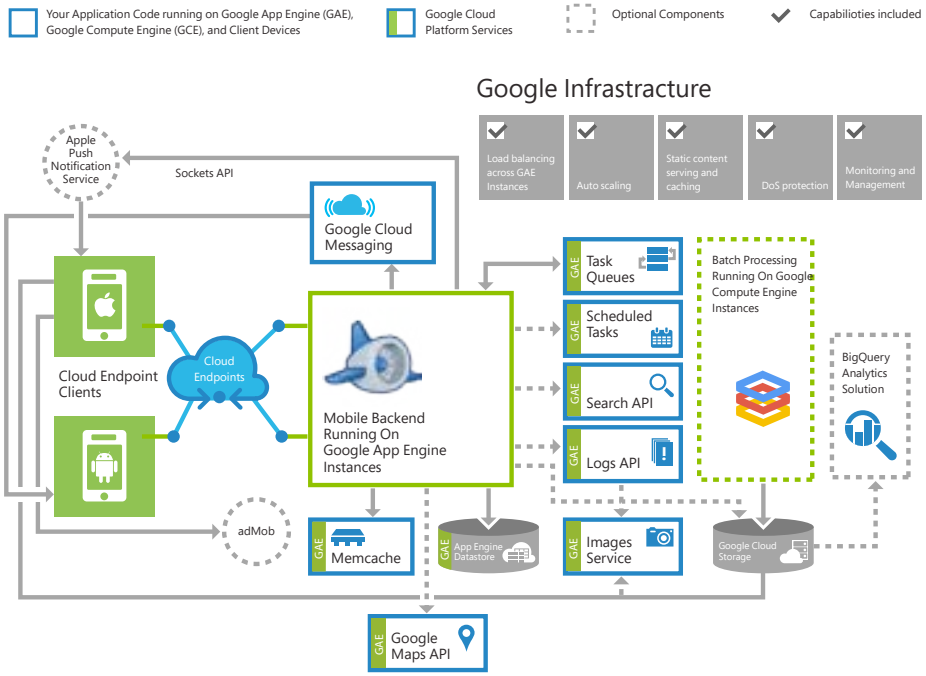
Applications with backends hosted in GAE can also utilize higher-level platform services like Memcache, Task Queues and Endpoints.

- Memcache is a distributed in-memory data cache called Memcache to store and retrieve the most frequently accessed data more quickly.
- Task Queues for asynchronous request processing.
- Google Cloud Endpoints offers an easier way to expose a REST API from your own mobile backend (like the Custom API feature in Windows Azure Mobile Services) and a means of consuming third-party APIs. Endpoints itself is not an app backend but rather provides an easier communication gateway between mobile clients and the backend hosted in GAE o GCE. Endpoints does not offer compute or an environment for hosting an app backend, just a few authentication options and easier access to core compute.
- Easy push notifications through Google Cloud Messaging (GCM) for Android apps, but only a Sockets API-based approach to using Apple Push Notification Service (APNS) for iOS apps.

It is important to note that with app's built on GAE's shared computing environment, you may run into some performance issues if you have a noisy neighbor. If you encounter performance issues in any shared environment, you can always offload a portion of the work to a dedicated server, but keep in mind that will likely require some re-architecting on your part.

The Google Cloud Platform team provides a step-by-step walkthrough of building iOS and Android apps on GAE here. A sample mobile app architecture utilizing GAE is also provided below for reference:

Mobile Solutions on the Google Cloud Platform



https://cloud.google.com/i/articles/mobile-app-10.png

Mobile Apps with backends hosted in virtual machines (IaaS)

Virtual machines provide you with maximum customizability and bring you as close to the actual metal of the machine as possible in the public cloud. You choose the instance size, OS and language. In addition to being the most customizable mobile backend option, they are also typically the cheapest.

Virtual Machines will not provide any middleware, runtime or environment specific to mobile app development. You will have to write the glue code for every connection. You will have to create an environment for server-side logic. You will have to stand up a push notification system that can easily span thousands of lines of code. You will take responsibility for all updates—from patching the machine's OS to responding to changed 3rd party protocols.

Virtual Machines are a good option for extending mobile app backends already hosted on-premises. When extending an existing application to the cloud, first confirm that the cloud provider supports images that match your on-premises workloads, often both Windows and Linux. After that, evaluate options for integration, messaging, storage and higher-level platform services like a standalone broadcast push notification service that can hook into any backend (on-premises, hybrid or fully cloud-based).

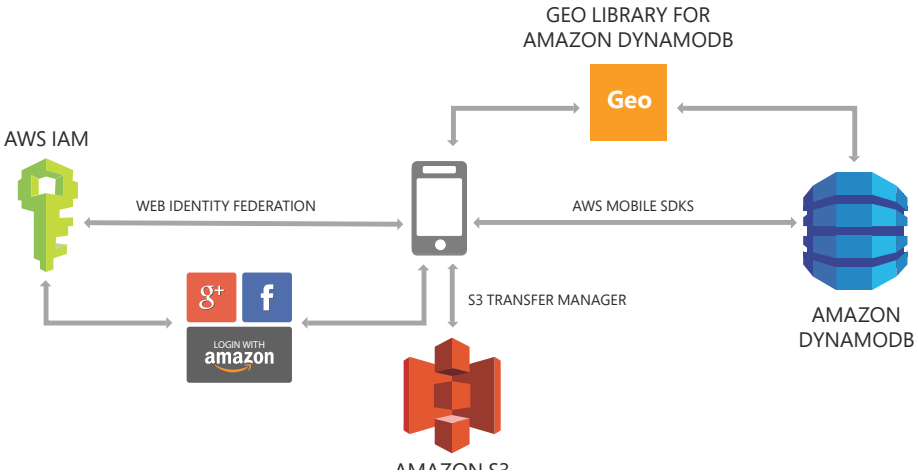
a. Windows Azure Virtual Machines

Windows Azure also offers Virtual Machines—Windows, SQL, Oracle, and a variety of Linux distributions (CentOS, SUSE, Ubuntu, etc.). Although the virtual machine gallery isn't quite as extensive as the AWS gallery, for example, the license portability (on-premises Windows Server to Windows Server in a virtual machine) and the easy access provided to additional platform services optimized for mobile make it a solid option for many customers and mobile scenarios.

b. Amazon Web Services (AWS) Elastic Compute Cloud (EC2)

AWS Elastic Compute Cloud (EC2) is a web service that provides access to a range of virtual machine instances in various sizes, operating systems and optimized for different workloads. EC2 facilitates running low-level processes that are able to write directly to disk. The wide variety of AWS instances—from high-memory instances to those optimized for compute-intensive workloads—allow you to allocate different backend tasks to different instances and scale each independently. This also pairs nicely with the AWS Elastic Load Balancer (ELB), which automatically balances traffic across all designated machines, detects failing machines and transfers loads to running machines.

Mobile Photo Share - Architecture



(Source: Glen Dierkes AWS Re:Invent 2013 Session: Building Cloud Backed Mobile Apps http://www.slideshare.net/AmazonWebServices/building-cloudbacked-mobile-apps-mbl402-aws-reinvent-2013)

AWS provides SDKs for the browser, Android and iOS that include a library, code samples and documentation for using AWS S3, SQS, SNS and DynamoDB with your app backend, whether hosted in EC2 or a PaaS.

c. Google Compute Engine (GCE)

Google Compute Engine (GCE) offers a variety of Linux images and access to a wider suite of cloud services including Google App Engine (GAE), Google Cloud Storage, and Google BigQuery. In order to create a web-facing interface, most applications will need to use GAE in conjunction with any workload based on GCE.

It is important to note that GCE only supports virtual machine images with Linux distributions; it does not offer Windows or SQL Server images. If you have some on-premises Windows-based workloads, you may limit your options by selecting a provider that only offers select Linux distributions.

Considerations when choosing a backend-as-a-service provider

Note:

Standing up your own push notification service can easily run 1000s of lines of code because every push notification service requires different channels and protocols. A huge BaaS value-add is not only provisioning this infrastructure for you, but also maintaining it. Push notification services like APNS for iOS, GCM for Google and WNS for Windows often change their registration and connection requirements. If you roll your own push, you're responsible for making these updates to your system; if you use a BaaS, the provider takes care of it all under the covers.

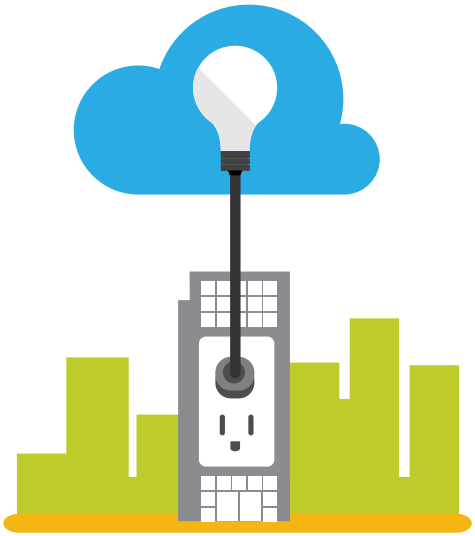
What every BaaS should provide

At a minimum, a BaaS should provide a means of:

- Storing any type of app data not just in the in the cloud, generally, but the datacenter of your choice, specifically
- Authenticating users via 3rd party credentials (Facebook, Twitter, Active Directory, etc.)
- Configuring and sending push notifications
- Environment for server-side code to add business logic

A robust solution will also provide:

- Easy consumption of 3rd party APIs and integrated billing
- Access to additional cloud services like broadcast push, networking, integration, raw VMs, cache etc.
- Offline sync capabilities
- Means of tunneling back to and consuming on-premises data and systems



General Considerations

There are well over 40 BaaS providers with solutions in market today the most prominent of which include Windows Azure Mobile Services, Parse, and Kinvey. Before selecting a provider, it's important to consider the following:

- Breadth of features (Does the provider cover a wide range of features like data storage, user authentication, geo, push notifications?)
- Feature richness (Do your data options include SQL, NoSQL, blob, Oracle, SQL Server and consumption of on-premises data? Can you choose the data center in which your data resides? Does the provider offer authentication via Facebook, Twitter, Microsoft and Google credentials? What about custom identity and Active Directory?)
- Power of the provided push infrastructure (Is there the option for both logic-based push and broadcast push? Can you easily personalize broadcast messages based on interest and device? What push notification providers are supported? Only GCM for Android and APNS for iOS or also WNS for Windows, ADM for Kindle and browser notifications for Mavericks OSX?)
- Extensibility (Can you utilize other cloud-based assets, third party APIs and on-premises data and systems? Can you upload and share any node.js module on the server side?)
- Auto-scale (Are you responsible for manually scaling your service up and down to match demand or does the provider take care of that for you?)
- Billing (Will all services from that provider and third party APIs utilized appear on the same monthly bill or are you expected to manage and scale multiple accounts independently? For consumer facing apps, can you be billed based on number of API calls? For employee facing apps, can you be billed based on number of authenticated users?)
- Comprehensiveness of SLA and support (Is the SLA based on overall uptime, API calls delivered or something else? What support options are available if you require assistance?)
- Global footprint (How many data centers does the provider operate worldwide? How does that map to your current and future expected customer base?)
- Vendor risk (What is the chance the provider is acquired or goes out of business? are supported? Only GCM for Android and APNS for iOS or also WNS for Windows, ADM for Kindle and browser notifications for Mavericks OSX?)

There is another general scale consideration inherent with any higher-level platform service. While every BaaS provider will tout essentially infinite scale and is probably built to deliver on that promise, that doesn't always tell the whole story when it comes to performance and cost. Because you're utilizing a layer above actual virtual machine or role instances, what you pay for in API calls and what you receive in terms of performance won't be the same as it would if you were closer to the metal.

With push notifications, for instance, just because you pay for 5 million API calls or 5 million pushes/month (depending on provider), that doesn't mean you'd be able to use all of those right away. If the underpinning service is a shared (multi-tenant) environment, then a noisy neighbor could prevent you from being able to send 5 million notifications in a minute even though there's generally enough compute to process those requests. If the underpinning service is a dedicated (single tenant) environment, then the capacity of the underlying compute could prevent you from being able to send 5 million notifications in a minute because the underlying instance might only be able to process 15,000 notifications/minute. In that case, you'd have to scale up to consume multiple underlying instances but would likely run into the problem of distributing requests evenly and instantaneously across each of those instances.

This is an example of a problem you may run into when you choose to abstract from the metal, relinquish some granularity of control and use a BaaS for the benefits outlined in the previous section. This example highlights why it's so important to choose a vendor with high extensibility. Being able to utilize a broadcast push service in conjunction with your selected BaaS would head off this sort of problem because you're able to offload one of the variable compute needs (low-latency push notifications) to additional infrastructure.

Conclusion

Ultimately, if you are building a mobile app for a business—either consumer facing or employee facing—that needs to extend on-premises systems and data, you should narrow the field of consideration to only those providers that offer a suite of services encompassing virtual machines, messaging and a variety of data storage options. Although it's possible to pair a standalone BaaS with core services offered by AWS, Google Cloud Platform or Windows Azure, that requires managing multiple bills, incurring additional cost and latency to enable communication between providers, and running the risk of requiring significant additional work to move and re-architect your app if the standalone BaaS is acquired or goes out of business. When deciding between with which of the major cloud providers to host your mobile app's backend, think about whether you are limited to virtual machines or can take advantage of higher level platform services to accelerate your development time and decrease your ongoing maintenance costs.

If you are delivering a net new mobile app that does not need to leverage on-premises data or systems—such as a conference app, standalone marketing contest app, game or new business built entirely in the cloud—you can consider a wider field of BaaS providers. Just make sure to carefully conside overall platform stability and extensibility and control over your data just as much as the individual feature roster.

Written by

Miranda Luna [@MLunes90]

Miranda is a Product Manager for Windows Azure services relevant to mobile app development, including Mobile Services and Notification Hubs. When she's not holding down the fort in Redmond, you can usually find her riding horses and drinking Stumptown cold brew.

Windows Azure Mobile Services

Windows Azure Mobile Services makes it fast and easy to build mobile apps that scale. Within minutes, you can store data in the cloud, authenticate users, and push notifications to millions of devices.