viewds identity solutions

White Paper:

# Cloud Identity is Different

Three approaches to identity management for cloud services

## A Changing Landscape

Application development has changed a lot in the last few years. Developers used to target employees working in the office. Applications ran on the company's on premises servers and they relied on the company's identity infrastructure, usually Active Directory.

Today, organizations need to be much more flexible. They need to provide information and applications to a broader range of people, not just employees but contractors, partners, and customers. And they need to provide these applications to remote workers using mobile devices, anytime, anywhere. These requirements have pushed application development into the cloud in a big way.

While developing a cloud application is not very different from developing an on-premises application, cloud application architecture does differ in several important ways:

- Cloud applications usually involve several separate, cooperating services, rather than just one or two.
- Cloud application services don't run in known locations on known servers. A message from an application server to read a record from a cloud data storage service might be sent to one server, and the following update operation might go to an entirely different server. And, those servers may be geographically diverse.
- Cloud application services are usually completely out of the control of the application developer or administrator. They come and go according to the needs of the application service provider.
- Communications between cloud applications and cloud application services need to be stateless, or as stateless as possible, and not rely on long-running network sessions.

Identity services in the cloud are no different. The services the developer used to get 'for free' with Active Directory have a completely different architecture and set of programming interfaces. On-premises identity protocols such as Kerberos and LDAP simply do not work well in a cloud environment. And besides, the whole point of cloud-based identity is completely different to that of on-premises identity.

This whitepaper will outline these differences and discuss three approaches to dealing with identity in cloud-based applications.

# Identity in the Cloud is Different

## Accessibility

Probably the most important motivation for moving applications to the cloud is to make them more accessible, in other words, to expand the reach of the application beyond the corporate firewall and beyond the company's employees. Expanding the reach of an application to new user populations requires rethinking the conventional on-premises approach to identity.

On-premises identity services exist to provide strict control over access to applications and their reach ends at the corporate firewall. In the typical on-premises environment a new user must go through several steps to use an application. First, the IT department creates an account in Active Directory (and possibly other systems). Then the user gets approval to use the application. After that, IT adds the user to the appropriate groups and maybe even installs the software on the user's computer. All these steps are needed to ensure that IT maintains control over who can use the application.

The purpose of cloud-based identity systems is more-or-less the opposite of this. They traverse network and security boundaries, integrate with external identity systems, and make the application easily available to people inside and outside the organization, all with little or no input from IT. Users can sign up for the application using just their email address and/or use a social network identity (such as Facebook or Twitter) to authenticate. In the cloud, users define and control access to their own identity information. IT departments are becoming less involved.

## Scalability, performance and efficiency

Scalability and performance requirements are different in cloud applications. Some cloud identity systems (such as Facebook) handle billions of user accounts. Others (like Azure Active Directory) provide hundreds of millions of authentication transactions per day. This is sort of scale is not the norm for on-premises systems and handling it requires careful attention to product architecture.

Related to scalability and performance is efficiency. Almost all Active Directory systems use dedicated servers as domain controllers, for both security and administrative reasons. In

this scenario, once you have purchased a server, there are essentially no ongoing costs. Whether that domain controller uses 10% of the CPU and 5GB of RAM or 80% of the CPU and 10GB of RAM makes no difference to your bottom line. Purchasing virtual machines in the cloud is different. Because you are paying more per month the more computing resources you use, the efficiency of your identity system affects your monthly bill.

## API support

Cloud-based identity systems need to support a different set of application programming interfaces (APIs) than their on-premises counterparts. On-premises identity systems, like Active Directory, use LDAP for directory access and Kerberos for authentication and authorization. However, the cloud environment requires protocols that are stateless, can recover from connection failures seamlessly and can traverse firewalls and load balancers with no additional configuration. So, cloud APIs are based on HTTP/S and use a Representational State Transfer (REST) model. Consequently, there are a new set of identity protocols based on HTTP/S that provide equivalent identity functions to on-premises systems: OData and SCIM for directory access, OpenID Connect for authentication, and OAuth 2 and the XACML 3.0 REST/JSON profile for externalized authorization.

## Multi-tenancy

The concept of multi-tenancy is the ability to segregate sets of identity data from each other, usually along organizational lines (think of a cloud service with multiple customers, something like SalesForce). This concept is not required in on-premises identity systems because they are inherently single tenant solutions. But in the cloud, service providers maintain strict separation of the identities associated with their different customers, so their identity systems must be able to support that.

## Identity data model

The identity data model for cloud applications is also more complex than that of a typical on-premises identity system. The on-premises model is generally pretty straightforward: there are some organizational structures (for example OUs in the directory) that represent different business units or departments, there are users and computers whose lifecycle is controlled by IT, and there are groups (possibly nested) that enumerate users with certain rights. And that is pretty much it. Whereas, cloud identity systems must model a more

complicated ecosystem of identity information including users, organizations or tenants (and possibly tenants of tenants), a sophisticated relationship of applications, roles, and permissions, and trust relationships with external identity providers and consumers. Add on top of that the idea of social identity, for instance which products does the user like, which users like the same product, etc. and you can see that the cloud identity model is complex.

## Security and privacy

And finally, we come to the issues of security and privacy. If you are a cloud or managed service provider maintaining identity information for your customers or an organization storing identity data for your corporate employees and partner organizations, then you have the moral and legal responsibility to maintain that data securely. But at the same time, you want to give your users maximum flexibility in defining who can access that information and under what circumstances. It goes without saying that passwords should be hashed and encrypted before being stored on disk, but so should other personal information like social security numbers, credit card and bank details. In addition, the identity software you use must be bullet-proof to prevent hackers from compromising your system, your users' personal information, and potentially external systems as well.

So, even though cloud identity systems provide the same basic functions (identity, authentication, authorization, and audit) as their on-premises counterparts, cloud identity really is quite different.

# Three Approaches to Identity Management in the Cloud

## Write Your Own Identity System

The first approach to developing your application's identity infrastructure is to simply do it yourself. Application developers can and often do build basic functioning identity systems reasonably quickly using a combination of custom code and freely available open-source components. However, such solutions rarely work well in the longer term for a number of reasons.

First, you must build your identity system so it will scale as your user base grows. Identity systems are by nature very read-heavy, generating several hundred times the number of read operations as write operations. (Think of the number of times you login compared to the number of times you update your details.) And identity updates are usually very localized, requiring at most the update of one or two objects. SQL databases and the object-relational mappers (ORMs) used by many web application platforms are optimized for an entirely different kind of workload, where many rows in many updates are updated together in a single transaction. Configuring and managing the SQL partitioning and replication so that your application can reliably and efficiently serve a global audience is also highly specialized knowledge.

Second, identity technology itself is complex. Implementing even basic username and password authentication properly requires specialist knowledge. Get it wrong and your applications are left vulnerable to bugs, the inadvertent disclosure of potentially sensitive information, or even the theft of your data. It is far safer and more efficient to let your application developers focus on the features of your application and to leverage the expertise of identity specialists for your identity infrastructure.

And finally, identity requirements change as applications and policies mature. The first version of your application might only need a username, password and email address. However, version two might require links to social media, profile and preference information, contact details, roles and relationships with other identities and for version three you might want role-based access control to resources and user-based consent to personal details. Redesigning your identity system every time you have a new requirement

is expensive and prone to error. On the other hand, trying to design an identity system that can accommodate all possible future requirements will most likely result in elaborate and extraneous code that you must test and maintain. And every time you add a new feature to your application, you must modify your identity system to support the appropriate authorization policies for that feature.

So, although it is fairly easy to use your application platform to build a rudimentary identity system, building an identity system that will evolve and grow to support your changing application requirements is a substantial investment, and a substantial risk.

## Use a Cloud-based Identity Service

The second approach to providing identity infrastructure for your applications is to use a cloud-based identity service. There are a number of different service providers out there including the big cloud application development platforms such as Microsoft's Azure Active Directory, Google ID, and Salesforce and smaller specialist identity as a service providers like Okta and Onelogin.

Using a hosted service has several advantages over building your own identity system. Such services are generally well designed and implemented, they support common identity standards, provide a robust, reliable and scalable service and they are 'just there'. You don't have to manage any additional cloud infrastructure to support your application identity needs.

However, there are some significant downsides to using such service providers. First, you give up control of your identity information. Instead of storing sensitive identity data on servers that you control and administer, you store it on servers controlled by a third party. This might not be a big deal. The public identity services probably have stronger security controls and administration processes than you do. But your legal or security requirements may prohibit letting sensitive identity information out of your control. Or, allowing it to be handled by a foreign organization subject to a foreign government's privacy and disclosure rules.

Another potential problem with public identity services is their relative inflexibility. These providers have created infrastructure capable of supporting identity on a large scale, but

they have traded off features and configurability that might be important for your applications. For instance, some identity services provide only a fixed identity schema. If you need to store additional attributes with your identity data, you must implement your own parallel identity store.

However, the real problem with committing to a public identity service is vendor lock-in. Once you have stored all your identity data with a particular service and written or configured your applications to work with the particular features and quirks of that vendor's system, it will be very difficult to move your application to a different identity infrastructure. You have effectively locked your applications to the vendor's identity infrastructure, and you will be subject to whatever service, feature, and cost changes that vendor decides to make in the future. So, while you enjoyed agility at the start of your project by leveraging an existing service, you have basically given up that agility for the life of your application.

## Use Packaged Identity Software

The third approach to providing the identity infrastructure for your applications is to use packaged cloud identity software. This is essentially an extension of the software model used with Active Directory for on-premises identity. You must provide the hardware (virtualized or otherwise) and operations manpower but the software itself is complete and ready to deploy after some basic configuration.

The packaged software approach has a couple of disadvantages. It requires setup and configuration on your part, so it is not 'just there' like a public identity service. Depending on the software package you choose, this might be a significant effort or not. Although once you have worked it out, your cloud automation software can handle subsequent installations. Then there are the ongoing operational costs. Not only do you have to manage your cloud applications, you also must manage the identity software as well. Although once again, this can be largely automated, so the incremental costs should be small.

On the other hand, there are several substantial advantages to using packaged software for your cloud identity needs. You get a richer feature set and more configuration flexibility. So, instead of having to shoehorn your application identity into a one-size-fits-

all identity data model, you can tailor your data model to suit your needs. You remain in control of your identity data. You control where it lives, how it is managed, and how it is accessed. This means that you can be sure that your identity data is secured according to your requirements, not according to those of a third-party or foreign government. And most importantly, you aren't locked into a service providers' feature set or pricing model.

## Conclusion

Cloud identity is a totally different beast than on-premises identity and the software you need to create your application's identity infrastructure has to be designed and built with cloud requirements in mind. Building your own identity infrastructure is almost never a good idea. Taking advantage of a cloud identity service can be an effective approach, particularly when you are just starting out, but has significant downsides in terms of control, flexibility and vendor lock-in. Using packaged identity software for your cloud application's identity needs does require that you deploy and manage the identity software along with your cloud application, but has several significant benefits including a richer feature set and increased control over your identity data, as well as the fact that you are never locked-in to a third party's pricing model.