**15**

**Implementing Identity and Access Management**

The core principle of identity and access management in the cloud is that everyone and everything in it is an identity. In this chapter, we will learn how we can manage identities and control their behavior by granting them specific roles, allowing them to perform only those activities that are related to the primary job of an administrator. We will see that **Role-Based Access Control** (**RBAC**) is very important to keep our cloud environments secure. We will learn about authenticating and authorizing identities, how to deal with least privileged accounts, what eligible accounts are, and why a central depository is needed. We will learn how we can federate with Active Directory from the various public clouds.

After this chapter, you will have a good understanding of technologies such as federation, single sign-on, multi-factor authentication, privileged access management, and **Identity as a Service** (**IDaaS**).

In this chapter, we’re going to cover the following main topics:

* Understanding identity and access management
* Using a central identity store with Active Directory
* Designing access management across multi-cloud
* Exploring **Privileged Access Management (PAM)**
* Enabling account federation in multi-cloud
* Working with IDaaS

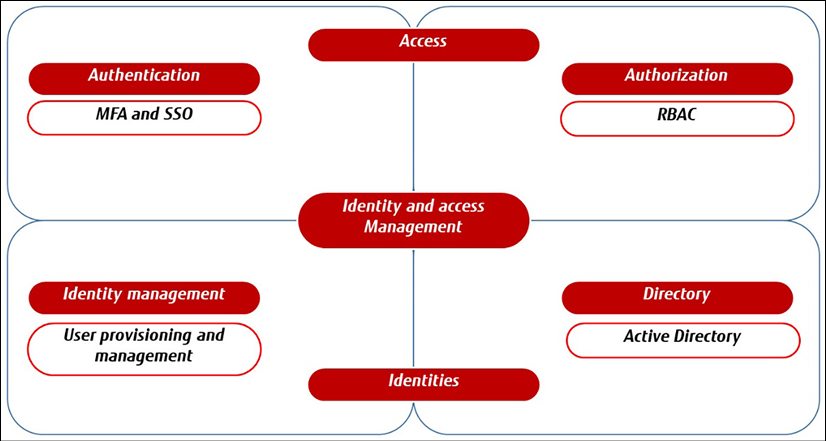
**Understanding identity and access management**

**Identity and access management** (**IAM**) is all about controlling access to IT systems that are critical to a business. A key element of IAM is **Role-Based Access Control**, or **RBAC** for short. In an RBAC model, we define who is allowed to have access to systems, what their role is, and what they are allowed to do according to that role. An important principle of RBAC is **least privilege**, meaning that a system administrator will only get the rights assigned that are required to perform the job. For example, a database administrator needs access to the database, but it’s not very likely that they will need access to network switches too.

In this chapter, we will discuss concepts such as **Single Sign-On** (**SSO**), **Multi-Factor Authentication** (**MFA**), and **Privileged Access Management** (**PAM**). Before we go into those, let’s have a look at the basics of IAM. There are three layers that we have to consider in our architecture:

* **Managed identities**: In this book, we’ve written a number of times that in cloud environments, everything should be perceived as an identity. Identities must be known: users, systems, APIs—everything that communicates with components in your cloud environment and with people or systems in the outside world.
* **Managed roles**: Roles must be defined in our cloud environments and assigned to identities. This includes the process of adding, removing, and updating roles. This is not only valid for people but also devices and systems. A system is an identity and has a specific role, for instance, a domain controller or application server. Thus, system authorizations must be defined and access rights must be given to resources.
* **Managed access**: This is the definition of who and what are given appropriate levels of access. To use the example of the database administrator once more, a database administrator needs access to the database and not to a network switch. If the database resides in a specific virtual network within the cloud environment, the administrator may need access to that network as well. However, that access should be limited; it is only needed to get to the database. That must be defined in the role.

On all three layers, the principle of least privilege is valid and must be followed through to achieve maximum protection of (sensitive) data in systems. The following diagram shows the main principles and related services of IAM:



*Figure 15.1: Main principles of IAM*

The next step is to define what an IAM system needs to do and what sort of tools it provides to control identities and access to our cloud systems.

Primarily, it needs to enable us to control identities. IAM should, therefore, contain a directory—basically, a database that holds all identities. Typically, this directory contains the entities that will access the systems. Almost all enterprises use **Active Directory (AD)** as their central directory. AD uses objects to represent an entity; an entity can be a user, a device, or a system. It also defines which domain an object belongs to.

Next, the IAM systems must be enabled to grant entities roles and the associated access. If a user is added and a role is assigned to that user, then the IAM system makes sure that access rights are provisioned to that user. Typically, roles and groups will be defined in the directory so that all that remains is to assign a user or object to that role or group. Access rights on the appropriate level are then automatically enabled. IAM should also facilitate a review process; only a few admins should have the right to add users to or remove users and objects from a directory. A user can request specific access, but they will always need a review and approval before rights are actually assigned. PAM and **Privileged Identity Management** (**PIM**) are tools to define that process. We will explore these concepts in the next sections.

**Using a central identity store with Active Directory**

One of the most used identity stores is still Active Directory. Before we get into AD itself, it’s important to understand that it should definitively not be confused with Azure Active Directory. The key difference is that Azure AD is a cloud-native IDaaS solution whereas AD is a traditional **Lightweight Directory Access Protocol (LDAP)**, a network protocol that determines how information is exchanged from directory services using, for instance, TCP/IP.

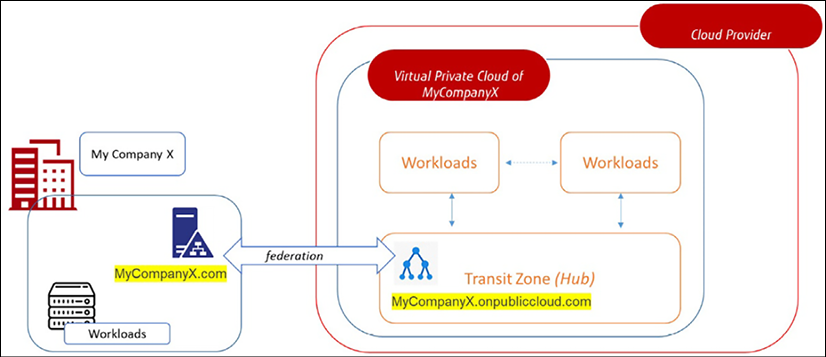
Understanding AD is not easy, but basic knowledge is necessary when talking about IAM. An enterprise should only have one central directory. Identities should only be kept in one place. That also comes with a risk—if a directory gets breached, an attacker will have access to all identities that exist within the enterprise. It’s crucial that the directory and the IAM system are very secure and that directory data is extremely well protected. This is an area where tools such as Saviynt and CyberArk come in; they add an extra security layer on top of IAM.

Both Saviynt and CyberArk offer solutions that are deployed on top of IAM, providing vaults and a way to secure access to systems, for instance, by hashing passwords in encrypted vaults so that users actually don’t see passwords, getting them instead from the tools. These tools can also record sessions or system logins to enable maximum visibility of activity in an environment, often referred to as an audit trail.

Let’s get back to the identity store and AD. The term is very much associated with Microsoft, as it was developed by that company for Windows domain networks. In the meantime, it has become a widely accepted term for the concept itself. AD comprises basically two major components that are both relevant in cloud environments. The first component is the directory itself; the second component is the domain services.

Domain services comprise a domain controller that authorizes and authenticates objects—users and computers—in a network. That network can be in a public cloud. It can also be a standalone network, but more often, the internal network of the enterprise is extended to a cloud. Extended may not be the right word, though. The enterprise on-premises network and cloud network(s) are merely connected or, to put it a better way, we connect the domains.

To be able to do that, domain controllers are needed in the public cloud. The domain controller makes sure that a specific part of the public cloud is now within our domain. For all of this, AD Federation Services can be used to federate the domain in the cloud with the directory that enterprises already have, commonly on-premises. The following diagram shows **My Company X** with the AD on-premises. There is an environment in a public cloud as well. That environment federates with the on-premises AD:



*Figure 15.2: Conceptual overview of AD federation*

Microsoft AD uses LDAP, Kerberos, and Domain Name Services for these services. LDAP enables the authentication and storing of data about objects and also applications. Kerberos is used to prove the identity of an object to another object that it communicates with. DNS enables the translation of IP addresses to domain names and vice versa.

This concludes the section about AD and how it’s used as an identity store. The next section will explain how access to clouds is controlled.

**Designing access management across multi-cloud**

In the previous section, we learned that we need to have federation with AD in our public cloud environment. The next question is, *how do we do that?* Azure uses **Azure Active Directory** (**AAD**). Just as a reminder—AAD is not the same as AD. AAD is an authentication service in Azure, using AD as the directory. The primary function of AAD is to synchronize identities to the cloud. For the synchronization, it uses Azure AD Connect.

With AAD, enterprises will have a system that provides employees of these enterprises with a mechanism to log in and access resources on different platforms. That can be resources in Azure itself or resources such as applications hosted on systems in the corporate network.

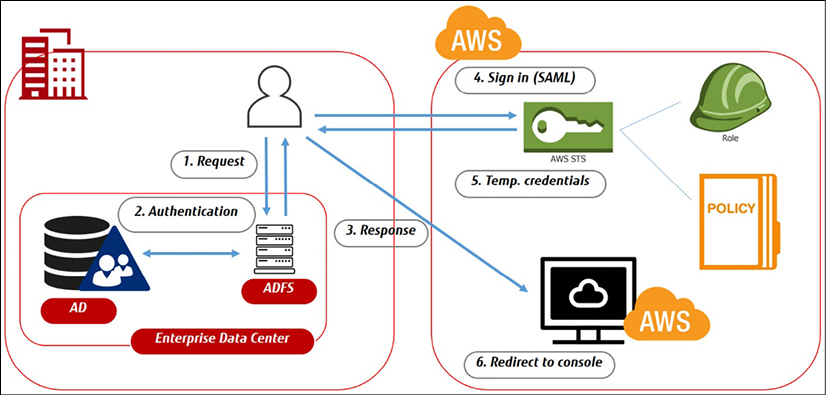
But AAD also provides access to SaaS solutions such as Microsoft 365 and applications that can integrate with Azure. AAD makes sure that users only have to log in once using SSO. It’s secured by MFA, meaning that when a user logs in by typing in a password, it is not enough. A second validation is needed to prove their identity. This can be a PIN code through a text message or an authenticator app on a mobile device, but also a fingerprint. If the user is authenticated, access is granted to federated services.

The federation between the domains in the corporate network and the corporate domain in Azure cloud is done with **Active Directory Federation Services (ADFS)**. Strictly speaking, there’s no hard requirement to use ADFS since AAD integrates with AD natively with hybrid entities, using password hash-sync or passthrough authentication. For third-party MFA you will still need ADFS though.

In the cloud, a corporate cloud domain is situated in the domain of the public cloud itself. In Azure, that is defined by onmicrosoft.com; this domain name address signifies that an environment resides in Azure.

Now, if we take company X, which has its domain specified as companyx.com and wants to have an environment in Azure, the domain in Azure would probably be companyx.onmicrosoft.com. Next, trust must be established between the corporate domain and the domain in Azure with ADFS. AD federation in Azure is shown in *Figure 15.2*.

In AWS, ADFS can be enabled as a component of the AWS Federated Authentication service. With ADFS, a user is authenticated against the central identity store, AD. After authentication, ADFS returns an assertion—a statement—that permits login to AWS using AWS’s **Security Token Service** (**STS**). STS returns temporary credentials based on AssumeRoleWithSAML, which then allows access to the AWS Management Console of the enterprise environments in AWS. The following diagram shows the concept:



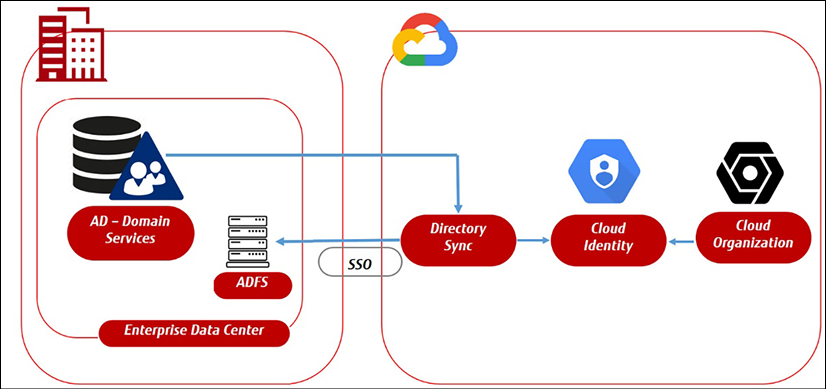
*Figure 15.3: Concept of AWS federated authentication*

AssumeRoleWithSAML is something specific to AWS. This function in STS provides a way to authenticate against the identity store with role-based AWS access. It uses **Security Assertion Markup Language** (**SAML**), an open standard for exchanging authentication and authorization data between parties, such as the identity store at a corporate level and the cloud provider. Yes, it’s comparable to LDAP, but SAML is more commonly used in the cloud.

A lot of companies still use ADFS, but it’s no longer a hard requirement to use AD in AWS or GCP. It is possible to integrate AD directly into other clouds. Refer to <https://learn.microsoft.com/en-us/azure/active-directory/saas-apps/aws-multi-accounts-tutorial> and <https://cloud.google.com/architecture/identity/federating-gcp-with-azure-active-directory>.

Also, GCP embraces SAML to do AD federation. At GCP, it starts with Google Cloud Identity, the service that GCP uses for IAM. But Google also understands that enterprises typically already have an identity store with AD. We can set up federation between GCP’s Cloud Identity or G Suite and enable user provisioning from AD, including SSO. SSO is done through SAML.

For the actual federation, we use Google Cloud Directory Sync, a free service from Google. The concept is shown in the following diagram:



*Figure 15.4: Concept of Google Directory Sync*

Of course, AD can be used in OCI to provide centralized authentication and authorization for OCI resources as well. We need to take a number of steps to set this up, as we did with the other clouds. First, we need to create a **virtual cloud network** (**VCN**) in OCI and configure a VCN and subnet for AD. In the VCN, we can now create an AD controller by using an AD server image. The best practice is to use an image from the marketplace. We must configure the AD domain controller by setting up the AD forest and domain and adding users and groups. Now we can use the AD server as the identity provider and start adding groups and users to the AD server. Be aware that we also must create a policy that allows access to OCI resources based on the AD groups and users. In the OCI console, we configure the AD identity provider as the default identity provider for our tenancy and configure the **single sign-on** (**SSO**) settings.

It’s important to note that this is a general overview of the process, and more detailed steps and specific configurations may vary depending on the specific use case. In all of these concepts, it’s also important to understand how the corporate AD is set up. This setup needs to be mapped to the IAM policies in the cloud platform. AD has a logical division with forests, trees, domains, and organizational units. Forests are the top-level segment of an AD and contain the root-level domain. Objects such as computers and users are grouped into domains. A group of domains forms a **tree**. Domains and trees form a forest – simply put, a collection of groups with users and systems in one domain that trust each other.

In a public cloud, this division of forests, trees, and domains might not map by default to the structure that a public cloud has. Using GCP as an example, organizations are the container boundaries that hold the resources within GCP. Organizations contain all the projects that can be hierarchically subdivided into folders. These structures have to be mapped to the AD structure; otherwise, federation will fail, leading to objects and users that can’t authenticate or have access to specific resources in cloud environments.

It’s beyond the scope of this book to do a deep dive into AD, but in the *Further reading* section, we’ve listed literature that provides more in-depth insights.

**Working with least-privilege access**

Least privilege is an important principle in IAM that ensures that users are only given the minimum level of access needed to perform their job functions. This principle is based on the idea that by limiting the access of users, organizations can reduce the risk of security breaches and data loss.

In an ideal scenario, a user should only have access to the specific resources and data that are necessary for them to perform their job. **Just in Time (JIT)** grants them access to do their job only, no longer than is strictly required.

This reduces the risk of unauthorized access to sensitive information and prevents users from accidentally or intentionally causing damage to a system. By implementing least privilege, organizations can ensure that users are only able to access the resources they need to perform their job, while preventing them from accessing or modifying sensitive information.

Least privilege also helps organizations to comply with various regulatory requirements, such as the EU regulations under the **General Data Protection Regulation** (**GDPR**) and the **Health Insurance Portability and Accountability Act**(**HIPAA**). These regulations require organizations to implement strict controls on access to sensitive data and to ensure that only authorized users have access to it.

Least privilege can even become a significant way of lowering costs. By limiting the access of users, organizations can reduce the need for expensive security measures, such as firewalls and intrusion detection systems, and also reduce the risk of system downtime caused by unauthorized access. Having said that, least privilege access doesn’t mean that organizations do not need to implement other security measures.

What do we need to do if we implement least privilege access? The following steps are best practices:

1. **Identify and classify sensitive data**. This includes identifying data that is sensitive from a regulatory, legal, or business perspective.
2. **Create roles and responsibilities**. Once the sensitive data has been identified, we should create roles and responsibilities for each employee. This includes determining what access each employee needs to perform their job and what access they should be denied.
3. **Implement access controls**. These controls are required to ensure that users are only able to access the resources they need to perform their job. This includes implementing authentication, authorization, and access control mechanisms such as **Role-Based Access Control (RBAC)**, but also **attribute-based access control** (**ABAC**).
4. **Regularly review and update access rights**. This should include regularly reviewing user accounts, role assignments, and access rights to ensure that they are still valid and necessary. A good addition to this is Microsoft Entra and, specifically, Microsoft Entra Permissions Management. This is a **Cloud Infrastructure Entitlement Management (CIEM)** product that provides visibility and control over permissions for any identity and resources in Azure, AWS, and GCP.
5. **Monitor and audit access**. This includes implementing logging, monitoring, and reporting mechanisms to track user activity and detect any unusual activity.

This is not a one-time activity. We must evaluate these steps every time we implement new systems or applications to ensure that access is restricted only to the necessary parties.

**Exploring Privileged Access Management (PAM)**

In previous sections, the principle of least privilege was introduced—users only get the minimum set of rights to the systems that they are authorized for/require.

PAM uses the principle of least privilege to grant users only the access they need to perform their job functions. This is achieved by using roles and policies to limit access to specific resources and actions, and by using MFA to ensure the identity of users. PAM also allows you to monitor and audit access to your resources, so you can detect and respond to any unauthorized access. Additionally, PAM provides a feature called Session-Based Authentication, enabling customers to create short-lived access credentials for users that need privileged access to resources, which last for a predefined duration.

Least privilege works with non-privileged accounts or **least-privileged user accounts** (**LUAs**). Typically, there are two types of LUA:

* Standard user accounts
* Guest user accounts

Both types of accounts are very limited in terms of user rights.

There are situations where these accounts simply aren’t sufficient and inhibit people from trying to do their job. The user would then need elevated rights—rights that are temporarily assigned so that the user can continue with their work. An account with such elevated rights is called a **privileged account**. Examples of privileged accounts are the following:

* **Domain administrative accounts**: Accessing all resources in the domain
* **AD accounts**: Accessing AD with rights to, for example, add or remove identities
* **Application accounts**: Accessing applications and databases to run, for example, batch jobs or execute scripts

A special category would be break glass accounts, sometimes referred to as emergency accounts. These are accounts that function as a last resort when users are completely locked out of an environment. The break glass account is an account that has access to all resources and has all the rights to literally unlock the environments again.

The issue with these accounts is that they form a much bigger risk than standard, non-privileged accounts. If privileged accounts get breached, a hacker can have control over critical systems and functions. PAM is a solution that mitigates these risks. In short, PAM makes sure that elevated rights can only be used by specified accounts for specific systems at a specified time and for specified reasons.

The principle behind this is called JIT and just enough administration. In this principle, an administrator can decide that specific users need certain privileges to perform tasks on systems. But these users will not get these privileges permanently; that would be a violation of the least privilege principle. These users will get eligible accounts, meaning that when a user needs to perform a certain task, the rights to do so will be elevated. To enable the eligible account, the user will need permission that expires after a pre-set time window.

So, the user requests permission to enable the elevated rights from the privileged account. Permission is granted for one, two, or the number of hours that are needed to perform the tasks. After the time has expired, the rights are automatically withdrawn.

**PAM on cloud platforms**

How does this work on cloud platforms? PAM only works if the principle of least privilege is applied—what privileged accounts are needed, and what roles they will have. The cloud platforms all have an extensive role-based model that can be applied, enabling execution at a granular access level with a separation of duties for resources.

Cloud providers work with only a few built-in general role types: roles that can do everything in the cloud tenant, roles that can do specific things in certain areas of the tenant, and roles that can only view things. In addition to those, there’s often a role for the purpose of adding users and roles in the tenant. That’s not sufficient for a role-based access model that requires more granularity. Cloud providers provide that and have roles specifically for network administrators, database administrators, or even very particular roles just for managing the backups of specific websites.

With a clear overview of our accounts and the roles that these accounts have, PAM can be configured in cloud environments. Azure offers PIM as their solution to identify and set eligible accounts. Be aware that this requires a premium license for AD in Azure. PIM sets eligible accounts, activates JIT, and configures MFA.

In AWS, PAM features are included in the IAM solution. Like Azure, AWS offers a role-based access model and the possibility of having privileged accounts, using elevated rights, SSO, and MFA. The logic starts with requests being denied by default, except for the root account, which has full access. There must be an explicit allow in an identity or resource policy that overrides the default policy. If these policies exist and are validated, then access is allowed. AWS IAM checks every policy that is connected to the request.

As with many other services on their platform, AWS allows third parties to provide solutions on top of the native technology—in this case, AWS IAM. Both Saviynt and CyberArk are two vendors among the third parties that have developed PAM solutions for AWS.

One solution that is worthwhile mentioning here is IAM Access Analyzer, a solution that helps to identify resources in an organization and accounts that are shared with an external entity. This tool is able to provide recommendations for modifications to IAM policies to ensure that access is following least privilege. It will present findings, showing the resources that have access permissions, how the resources are shared through, for instance, S3 bucket policies or access control lists, and the level of access.

GCP offers cloud identities and, like other clouds, a model for role-based access. One particular feature that needs mentioning here is Recommender. This feature provides usage recommendations to optimize GCP environments, but it also comprises tools to manage IAM in GCP. IAM Recommender automatically detects identities that may lead to security risks and can even remove unwanted access to resources. It uses smart access control.

From the Google Cloud console, the IAM page lists all accounts and the permissions that these accounts have. IAM Recommender displays the number of unused permissions over the past 90 days and makes recommendations, such as replacing a role with an account with a predefined role in Cloud Identity, or creating a custom role with the appropriate rights. By doing so, we enforce least privilege in GCP.

We have studied Azure, AWS, and GCP so far. It’s fair to say that the basic principles in these clouds to set up IAM are quite similar. Let’s go over the steps to implement IAM, this time using OCI. We can use the following services:

* **Identity**: creates and manages users, groups, and policies.
* **Policy**: defines specific permissions for users, groups, and resources.
* **Compartment**: organizes resources into compartments and then uses policies to control access to those compartments. This is comparable to setting up Resource Groups in Azure and AWS and projects in GCP.
* **Users**: creates and manages users and their associated credentials.
* **Groups**: creates and manages groups of users, making it easier to grant or revoke access for multiple users at once.
* **Roles**: assigns permissions to users and groups for specific tasks or job functions.
* **Multi-Factor Authentication (MFA)**:provides an additional layer of security by requiring users to provide a second form of authentication, such as a fingerprint or a one-time passcode, in addition to their password.
* **Audit and Governance**: monitors and audits access to resources, so we can detect and respond to any unauthorized access.
* **Session-based Authentication**: creates short-lived access credentials for users that need privileged access to OCI resources that last for a predefined duration.

This concludes the comparison of IAM services between various clouds, but what if we really are multi-cloud and want to federate IAM across the clouds? That’s the topic of the next section.

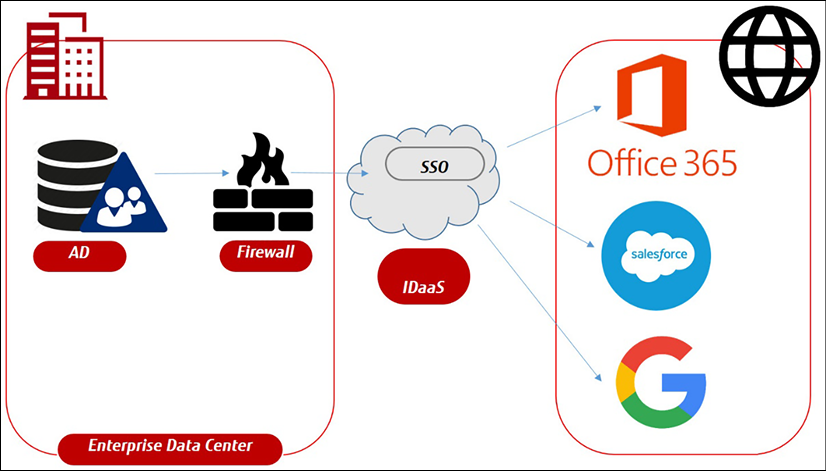
**Enabling account federation in multi-cloud**

Businesses are shifting more and more from software to services by adopting SaaS solutions, for instance. Typically, a user would have to log in to separate SaaS solutions, since these are provisioned from a service provider. The risk is that users create new passwords to log in to SaaS solutions. It’s easy to lose control of who has access to what. This can be solved through SSO, but the directories of SaaS solutions or web applications need to be federated in that case.

In the field of account federation, Okta has become an increasingly popular IAM solution in recent years. To avoid confusion, it’s not an alternative to AD. AD is typically the primary, central directory; Okta is a solution that utilizes AD and takes care of the federation to web applications using SSO. That’s what Okta does—it enables IAM with SSO on top of AD, delivering IDaaS.

IDaaS is a cloud-based service that provides organizations with a way to manage and authenticate users’ identities across multiple cloud environments. As more and more businesses adopt a multi-cloud strategy, IDaaS has become an essential tool to ensure secure access to cloud-based resources.

One of the main benefits of IDaaS is its ability to centralize identity management. Instead of managing identities across multiple cloud environments individually, IDaaS allows organizations to manage all identities in one central location. This makes it easier to control access to resources, enforce security policies, and track user activity. Additionally, IDaaS can be integrated with existing on-premises identity management systems, allowing for the seamless management of both on-premises and cloud-based resources. The basic principle of IDaaS is shown in *Figure 15.5*:



*Figure 15.5: High-level concept of IDaaS*

The other benefits of IDaaS are the capabilities to provide MFA and SSO, but the most important feature concerns the centralization of the management and monitoring of user access to resources. IDaaS provides granular access controls, allowing organizations to specify exactly who can access which resources and under what conditions. Additionally, IDaaS can provide real-time monitoring of user activity, allowing organizations to quickly identify and respond to security breaches.

To be clear, doing so will move all identities fully to the cloud. It’s really a decision for the **Chief Information Security Officer** (**CISO**) or **Chief Information Officer** (**CIO**) to adopt this shift.

Enterprises will need federation using cloud solutions, though. Companies tend to have complex IT environments, comprising on-premises systems, IaaS, PaaS, and SaaS. To be able to connect everything, as it were, all in one domain, federation is required. An IDaaS solution can be a valuable option to connect an existing AD to all these different solutions and enable secure access management to them.

**Summary**

Security starts with IAM—making sure that we have control over who’s accessing our environments and what they are allowed to do in systems. In this chapter, we have learned what identities are and that we need a central identity store. From this identity store, we have to federate between the different cloud solutions that an enterprise has. We have learned how we can set up federation and how IDaaS can be a good solution for this.

We’ve studied concepts of authorization and authentication in the major cloud platforms. An important concept is least privilege. After this chapter, you should be able to make a distinction between standard accounts and privileged accounts. Lastly, we learned what benefits PAM has in securing access to our clouds.

The reason to have our cloud environments maximally secured is to protect our data. We have studied identities, access management, and security policies to protect our infrastructure. In the next chapter, we will learn how we should define security policies to keep our data safe.

**Questions**

1. In IAM, we have three layers that we must consider to identify identities. Can you name them?
2. Both AWS and GCP use a specific protocol for authentication. What’s that protocol?
3. If a standard account isn’t sufficient, we can “promote” users temporarily with another account that holds more rights. How do we name these accounts?

**Further reading**

* *Mastering Active Directory*, by Dishan Francis, Packt Publishing