Identity Architectures

Svet Voloshin

&

Ryan Clancy

Agenda

- On-Premise Architecture/Concepts
- Extending to the Web - Federated Identity
- Transitioning to the Cloud (full and hybrid models)
- Other things

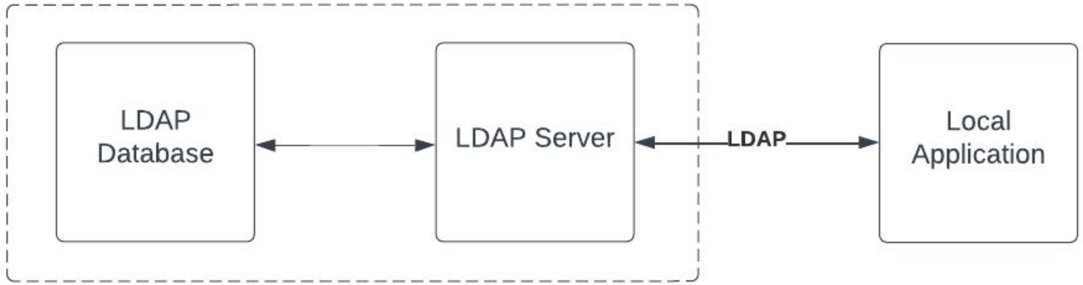
“Company X has an LDAP-compatible user store”

What this means:

LDAP Database: Where user identities are actually stored

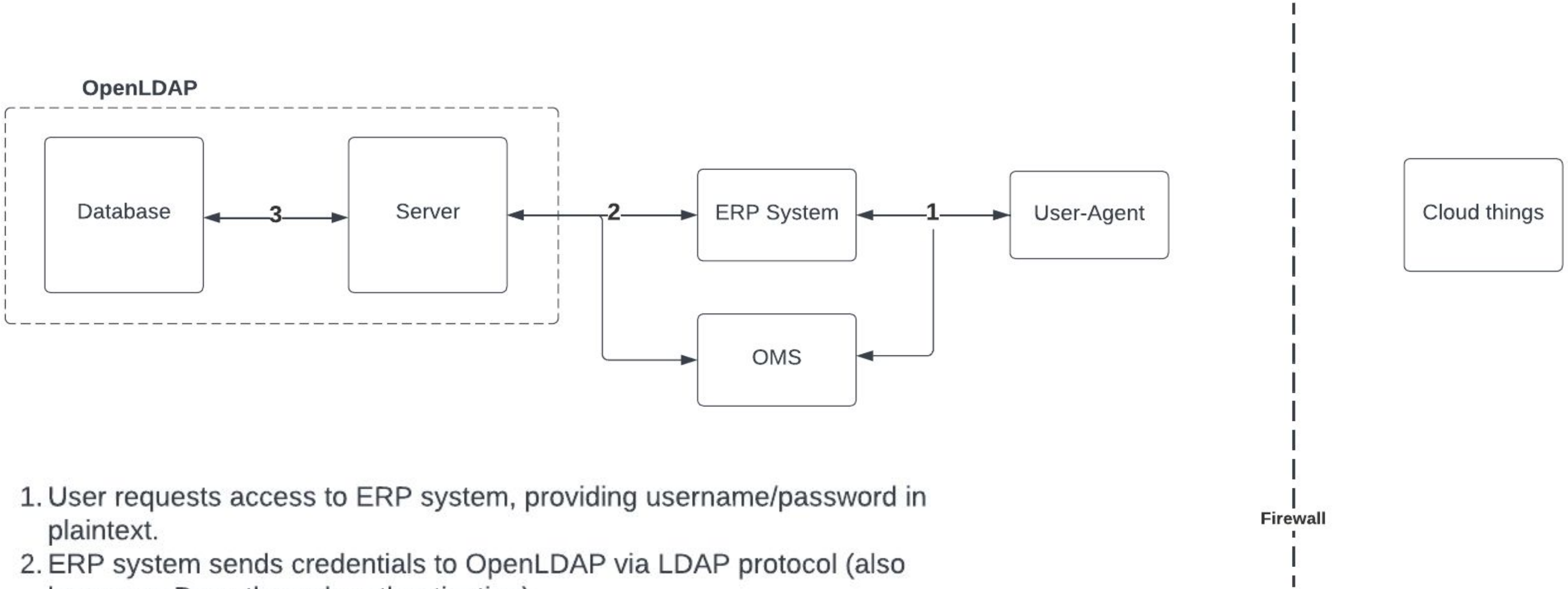
LDAP Server: Used to query/perform operations on the user data store

Client (Local Application): Any on-premise application that can make requests to an LDAP server (ex. ERP system)



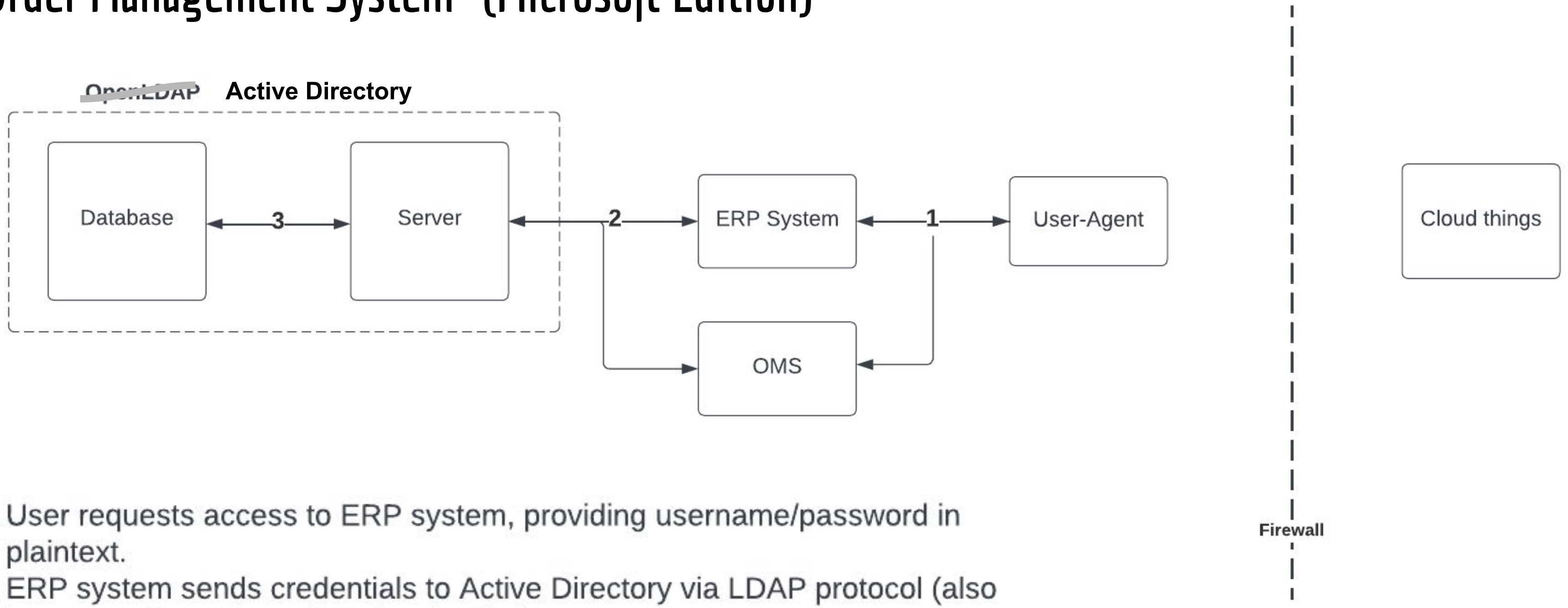
- If it helps to put a specific name in place of “LDAP-compatible user store”, use OpenLDAP (or Active Directory)
- LDAP is a **lightweight directory access protocol** that can be used to store and **organize data about users, devices, and other resources**. Its advantage is that it is **easy to use and implement**, and it can be used to **integrate with a variety of applications**.

“Company X has an LDAP-compatible user store **to authenticate with an on-premise ERP and Order Management System”**



1. User requests access to ERP system, providing username/password in plaintext.
2. ERP system sends credentials to OpenLDAP via LDAP protocol (also known as Pass-through authentication)
3. OpenLDAP server queries database to confirm valid credentials and access level. If correct, creates LDAP session/cookie and user can login

“Company X has an LDAP-compatible user store **to authenticate with an on-premise ERP and Order Management System” (Microsoft Edition)**



1. User requests access to ERP system, providing username/password in plaintext.
2. ERP system sends credentials to Active Directory via LDAP protocol (also known as Pass-through authentication)
3. Active Directory server queries database to confirm valid credentials and access level. If correct, creates LDAP session/cookie and user can login

Limitations of LDAP

- Username/password data is typically transmitted in **plain text**. This means that LDAP **cannot support authentication with external systems**, since there is **no way** to keep user credentials secure outside of firewall.

- **LDAP does *not* support SSO**. If you want to authenticate with an additional on-prem system, you will need to get an **entirely new session/cookie** for that app even if it is authenticating against the same credentials as your other on-prem systems.

- LDAP **does *not* support user provisioning/deprovisioning**. You need to do this manually or build an application/script to do this for you.

- Why was this a big deal anyways?
  - It allowed companies to have **one central repository** for user management

Question: How can we integrate external applications without changing our existing authentication architecture?

Federated Identity*

- Enables users to authenticate with systems across multiple domains (ex. On-prem, external) with a singular user profile.
- Prevents the need to expose user credentials in order to authenticate user identities with systems (aka. Claims-based Identity**)
- HTTP Redirects***
- Common Federated Identity protocols:
  - SAML
  - OAuth
  - OIDC
- Examples of Federated Identity systems:
  - ADFS
  - PingFederate

- *__Identity Federation__: The concept is based on **trust relationships (federation)** established between different domains or enterprises. In a federated environment, each participating domain maintains its own identity store and agrees to trust the identity assertions from the other participating domains.
- ****Claims-based Identity:** a security model that uses claims to represent user identities. Claims are statements about a user, such as their name, role, or location. Claims can be issued by identity providers (IdPs), such as Active Directory or Okta.
- *****HTTP redirects** are a versatile tool that can be used in a variety of ways to implement identity architectures. They are a simple and effective way to forward users and search engines from one URL to another, and they can be used to implement a variety of security features, such as SSO, federated identity, and MFA.

HTTP Redirects

HTTP redirects are a common way to forward users and search engines from one URL to another. They can be used in identity architectures to:

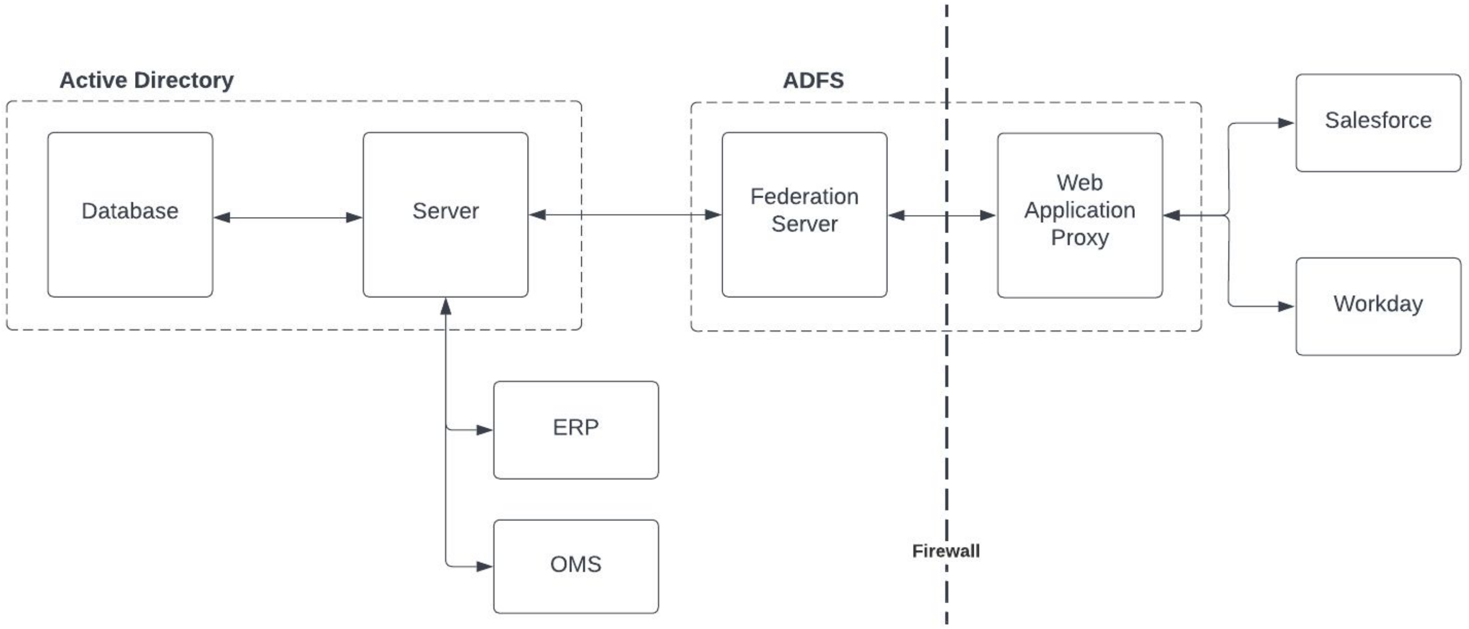
- **Implement single sign-on (SSO)**: SSO allows users to authenticate once and then access multiple applications without having to re-authenticate. HTTP redirects can be used to redirect users from the application they are trying to access to the identity provider (IdP) for authentication. Once the user has been authenticated, the IdP can redirect the user back to the application with an access token.
- **Implement Federated Identity**: Federated identity allows users to authenticate to one IdP and then access resources from multiple organizations that trust that IdP. HTTP redirects can be used to redirect users from the application they are trying to access to the IdP for authentication. Once the user has been authenticated, the IdP can redirect the user back to the application with an assertion about the user's identity. The application can then use the assertion to determine whether the user is authorized to access the resource they are trying to access.
- **Implement multi-factor authentication (MFA)**: MFA adds an additional layer of security to authentication by requiring users to provide two or more pieces of evidence to verify their identity. HTTP redirects can be used to redirect users from the application they are trying to access to a MFA server. Once the user has successfully authenticated with MFA, the MFA server can redirect the user back to the application.

HTTP redirects are a versatile tool that can be used in a variety of ways to implement identity architectures. They are a simple and effective way to forward users and search engines from one URL to another, and they can be used to implement a variety of security features, such as SSO, federated identity, and MFA.

Here are some additional things to consider when using HTTP redirects in identity architectures:

- **Security**: HTTP redirects can be used to implement phishing attacks. To protect against this, it is important to ensure that the redirects are coming from a trusted source.
- **Performance**: HTTP redirects can add latency to the authentication process. To minimize this impact, it is important to use redirects sparingly and to optimize the redirects for performance.
- **Compliance**: In some cases, HTTP redirects may be prohibited by regulations. It is important to check with your legal team to ensure that you are compliant with all applicable regulations before using HTTP redirects in your identity architecture.

*“Company X has an **LDAP-compatible user store** to authenticate with an on-premise ERP system. They would also like to use those same credentials to login to their Salesforce org and Workday” (Microsoft Edition)*



How ADFS works

Active Directory Federation Services (AD FS) is a feature of the **Windows Server** operating system that extends end users' single sign-on (SSO) access to applications and systems outside the corporate firewall, or across different trusts or domains within a forest. Essentially, AD FS handles authentication by providing a secure mechanism for transmitting and validating claims between partner organizations. Here's an overview of how it works:

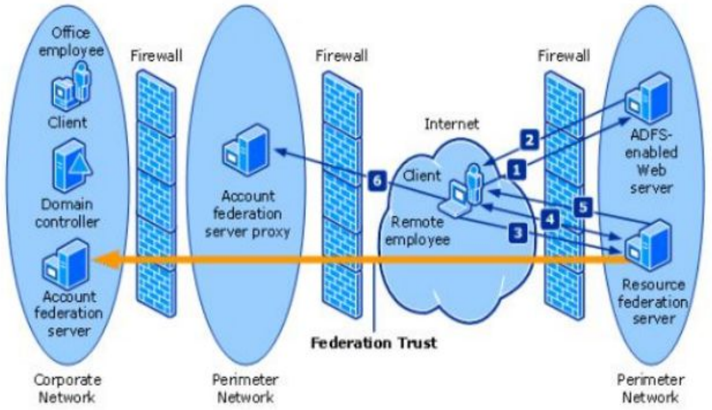
Basic Components

1. **Federation Server**: This server hosts the AD FS service and plays a crucial role in claims generation.
2. Federation **Proxy**: This optional component sits between the Federation Server and the internet to forward requests and responses.
3. **Claims**: These are **statements** (like **user ID, roles, or permissions**) that one subject, such as a user, asserts about itself to another subject, such as an application.
4. **Relaying Party**: This is the application or service that relies on the claims to provide authorization services. It could be a cloud application, a SharePoint site, or other services that trust claims from the AD FS server.
5. **Identity Provider**: In most AD FS configurations, this is the **AD FS Federation Server itself. It authenticates the user and issues security tokens, containing claims, that applications understand.**

Workflow

1. **Authentication Request**: A user tries to access a federated application (the "relying party"). If the user is not already authenticated, the application redirects the user's client to the AD FS service for authentication.
2. **Token Request**: AD FS prompts the user for **credentials** (which may be cached if they have already logged in). It authenticates the user against the **Active Directory**, which is acting as the **identity provider**.
3. **Claims Generation**: Upon successful authentication, AD FS generates a security token, typically in the form of a SAML (Security Assertion Markup Language) or WS-Federation token, containing the claims that pertain to the user.
4. **Token Signing**: The token is signed by AD FS to ensure its **integrity and validity**.
5. **Token Issuance**: The token is sent back to the user's client.
6. **Token Consumption**: The client presents the signed token to the federated application.
7. **Claims Extraction and Authorization**: The **federated** application **validates the signature** on the token and extracts the claims. These claims are then used to make **authorization decisions** within the application.
8. **Resource Access**: Once the claims have been **validated** and the user has been **authorized**, the application **grants** the appropriate **access** to the user.
9. **Single Sign-On (SSO)**: Now that the user has been **authenticated**, they can use **SSO** to access **other services** that **trust** the **same AD FS service**, without being prompted for credentials again.

This is a high-level overview. The actual details can get quite complex and may include many other features, like multi-factor authentication, claim transformations, and integration with other identity providers.



- All of this infrastructure needs to be managed by company
- ADFS does not do user provisioning/deprovisioning. Federation Servers only handle authentication of external systems (and some other stuff like SSO)

Scenario: *"Company X has an LDAP-compatible user store to authenticate with an on-premise ERP system. They would also like to use those same credentials to login to their Salesforce org and Workday. New users created in LDAP system should **automatically** be **provisioned** in Salesforce, and **inactive** users should automatically be **deprovisioned**."*

Question: *How can we do this with our **existing** authentication architecture?*

Answer: *We can't (yet), but if only we could make use of cloud-based solutions!*

Using Identity Management Solutions

1. **Identity Providers with Provisioning Support:** Solutions like Okta, OneLogin, and Azure AD provide built-in support for automated provisioning and deprovisioning of Salesforce accounts based on LDAP or Active Directory information.
2. **SCIM Protocol:** Salesforce supports the **System for Cross-domain Identity Management (SCIM) protocol**, which can be used for provisioning and deprovisioning users. Some identity providers can utilize SCIM to communicate between your LDAP system and Salesforce.

Custom Solutions

1. **API-based Provisioning**: You can build a **custom integration** using Salesforce's API and your LDAP system's API (or SDK). When a new user is added to LDAP, your custom solution can a**utomatically create a corresponding user in Salesforce**. When a user is marked as **inactive** in LDAP, the integration can **deactivate** the corresponding Salesforce user.

2. **Scheduled Scripts**: As a simpler approach, you could use scheduled scripts that **sync LDAP and Salesforce** by periodically querying LDAP for changes and applying those changes via the Salesforce API.

3. **Middleware**: You can use middleware solutions like MuleSoft, Dell Boomi, or WSO2 to handle the **user data transformation and communication between LDAP and Salesforce**.

Steps for Custom Solutions

1. **Detect Changes in LDAP**: Set up a mechanism to detect new or changed user entries in the LDAP system.
2. **Data Mapping**: Map LDAP attributes to Salesforce user attributes.
3. **Create or Update in Salesforce**: Use Salesforce APIs to create or update user accounts based on the information fetched from LDAP.
4. **Detect Inactive Users**: Implement a rule to identify deactivated or removed accounts in LDAP.
5. **Deactivate in Salesforce**: Automatically deactivate these users in Salesforce.
6. **Logging and Monitoring**: Implement logging and monitoring to keep track of what changes have been made, to troubleshoot issues, and to produce reports.
7. **Error Handling**: Implement robust error handling and notifications for exceptions.
8. **Testing**: Rigorously test the integration before deploying it in a production environment.

SCIM in a Nutshell

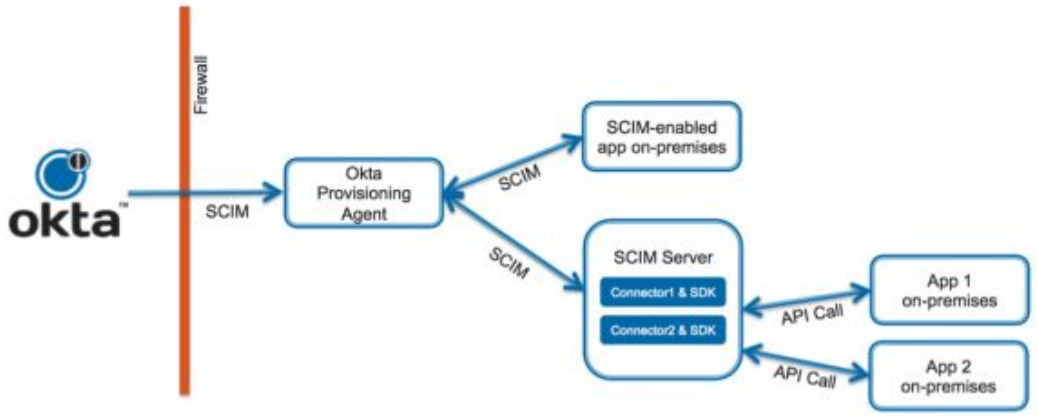
- SCIM is a set of **REST endpoints** provided by cloud applications to allow for management of user identities

- SCIM is a concept that is specific to cloud applications, this is why User Provisioning/Deprovisioning was not possible “out-of-the-box (OOTB)” until this point

- Allows for bi-directional provisioning/deprovisioning between cloud systems
    - *On creation of a user in Azure AD/Okta, create user in Salesforce*
    - *On creation of a user in Salesforce, create user in Azure AD/Okta*

- **Salesforce Documentation**
    - Create SF User via SCIM
    - Deactivate/Reactivate SF User via SCIM

SCIM Protocol



The System for Cross-domain Identity Management (SCIM) is an open standard designed to simplify the complexities of identity management across multiple systems. SCIM aims to improve the automation and interoperability of user identity information among cloud-based applications and services, as well as enterprise domain services. It provides a standard way of representing users, groups, and other identity-related objects.

Core Components of SCIM

1. **SCIM Schema**: Defines the attributes and data model for representing users and groups. The schema is extensible, allowing for custom attributes and objects to be added.
2. **SCIM Endpoints**: These are the URLs exposed by the service provider for creating, reading, updating, and deleting user and group resources. Standard endpoints often include /Users, /Groups, and /ServiceProviderConfig.
3. **SCIM Protocol**: Built on standard web technologies like HTTP, JSON, and XML. It specifies CRUD operations (Create, Read, Update, Delete) that can be performed on user and group resources.
4. **SCIM Filters**: Allow for the querying of resources based on attribute values.

Workflow

Here's a basic workflow for SCIM-based provisioning and deprovisioning:

1. **SCIM Client and Server**: The SCIM client (often an Identity Provider) communicates with a SCIM server (the application, like Salesforce, where identities need to be managed).
2. **User Creation**: When a new user is created in the identity source, the SCIM client sends an HTTP POST request to the /Users endpoint of the SCIM server to create a new user resource.
3. **User Update**: If a user's attributes change, the SCIM client sends an HTTP PUT or PATCH request to the specific user's endpoint (e.g., /Users/{id}) to update the user resource.
4. **User Deletion**: When a user is deactivated or deleted in the identity source, the SCIM client sends an HTTP DELETE request to the specific user's endpoint (e.g., /Users/{id}) to remove the user resource.
5. **Group Management**: Similar operations can be performed for group management through the /Groups endpoint.
6. **Bulk Operations**: SCIM also supports bulk operations for creating, updating, or deleting multiple resources in a single request.

Transitioning to the Cloud - Azure AD and Okta

Azure AD and Okta are both IDaaS* products that support a wide host of features, including:

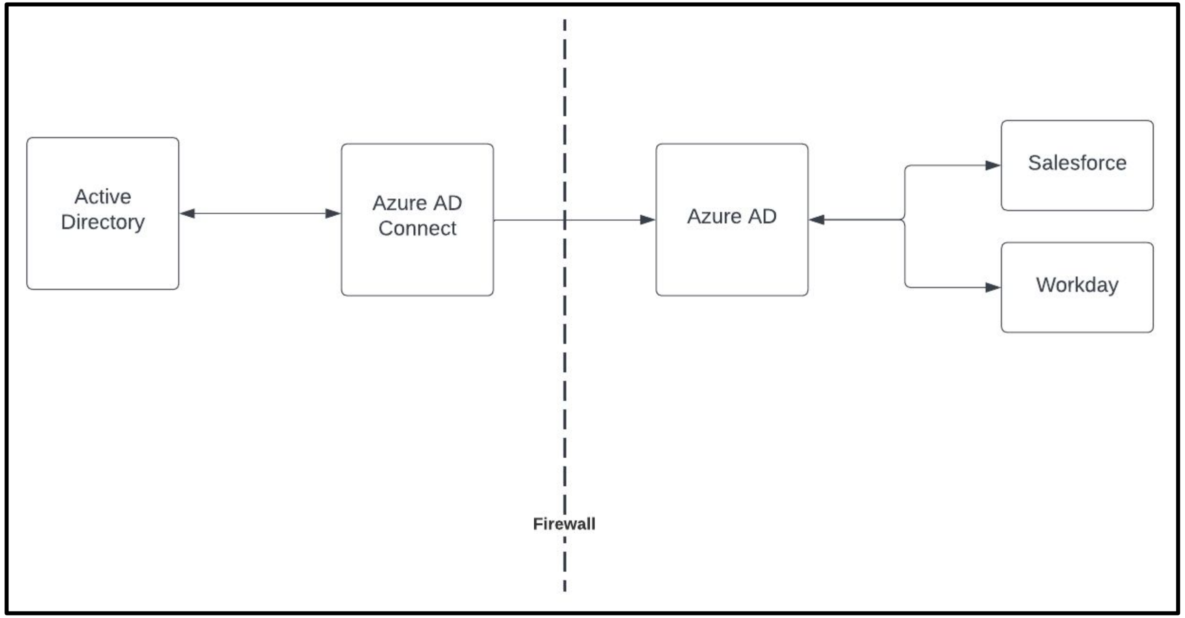
- Directory Management/Synchronization
- Single Sign-On
- MFA (multi-factor authentication)
- Integration with Cloud/On-Prem applications
- User Provisioning/Deprovisioning in external systems (SCIM)
- Conditional Access policies
- A lot more

One of the biggest benefits of Azure AD/Okta over the previously covered architectures is that companies do *not* need to directly manage the identity infrastructure!!

*IDaaS: Identity as a service.

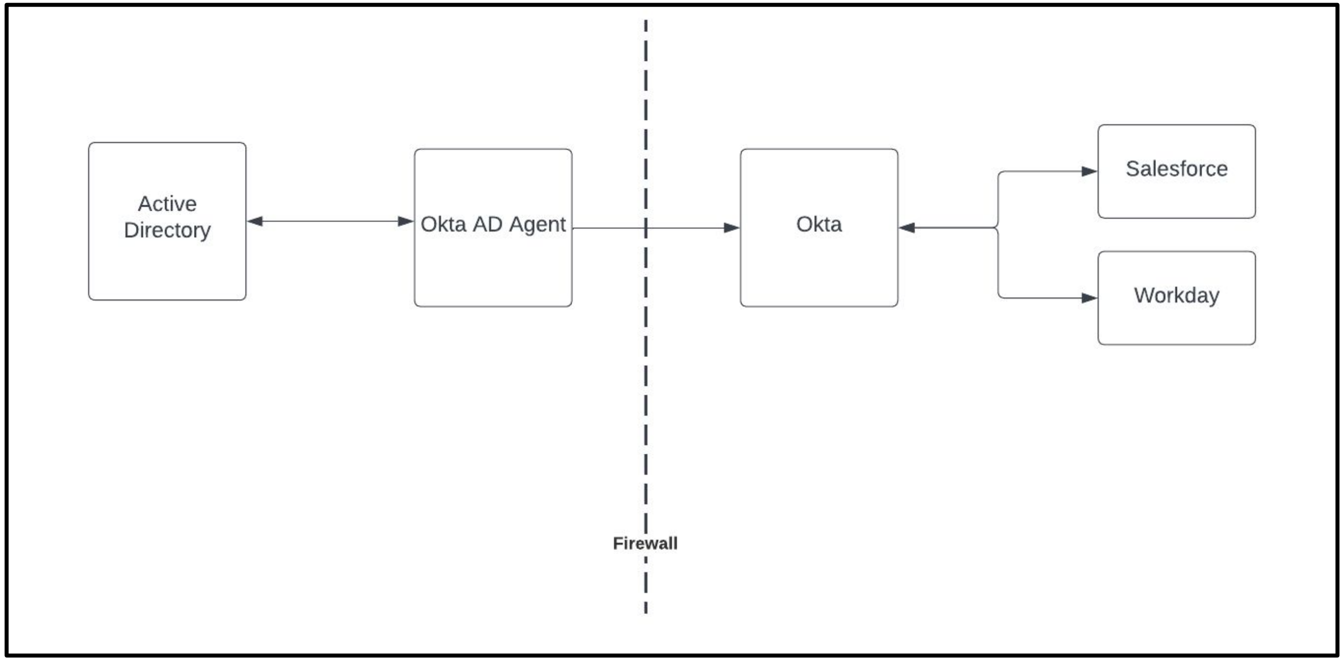
User Provisioning/Deprovisioning with Azure AD (and on-prem user store)

- User created in on-premise Active Directory
- Azure AD Connect Provisioning Agent synchronizes on-prem Active Directory with Azure AD user store
- Azure AD makes REST callouts to Salesforce to provision/deprovision users via SCIM protocol



User Provisioning/Deprovisioning with Okta (and on-prem user store)

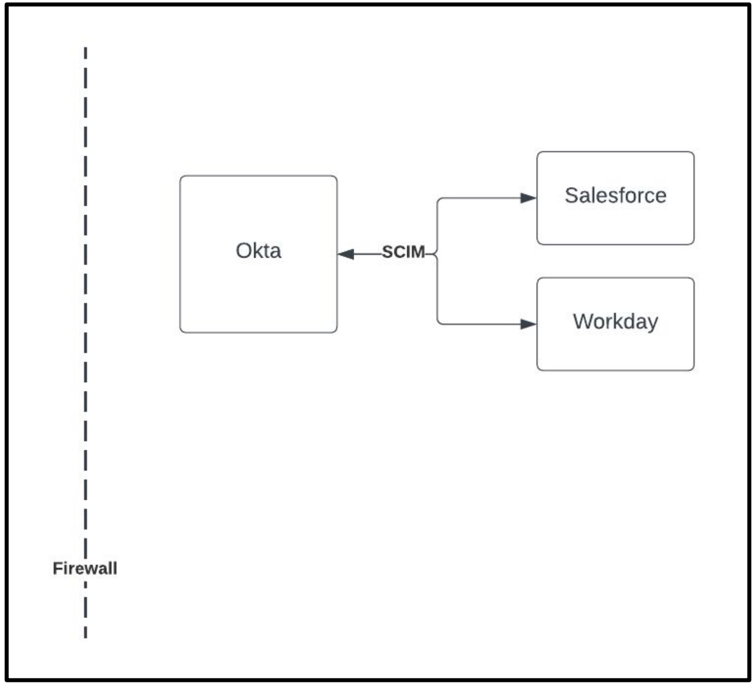
- User created in on-premise Active Directory
- Okta AD Agent synchronizes on-prem Active Directory with Okta user store
- Okta makes REST callouts to Salesforce to provision/deprovision users via SCIM protocol



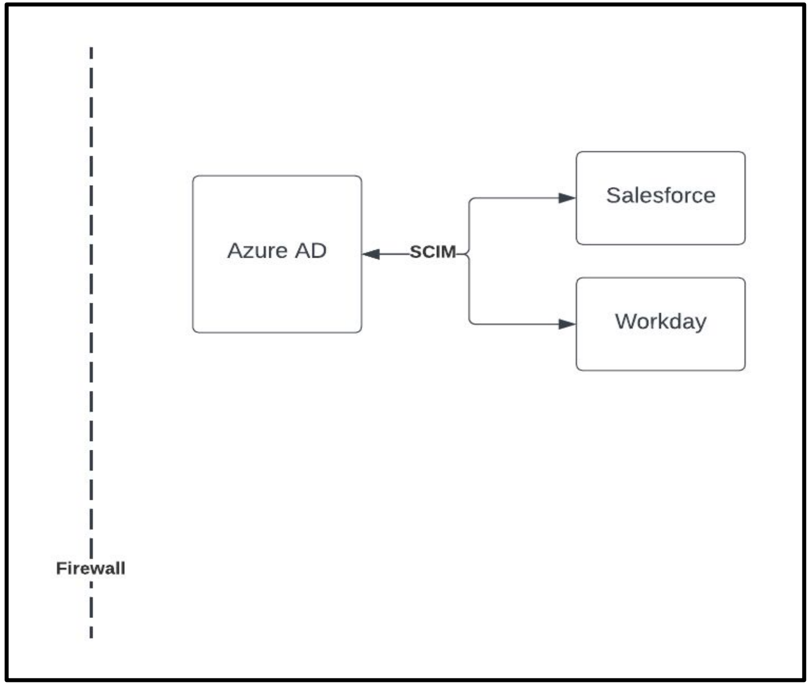
What if your user store isn't on-prem?

Well then this is much easier

Okta

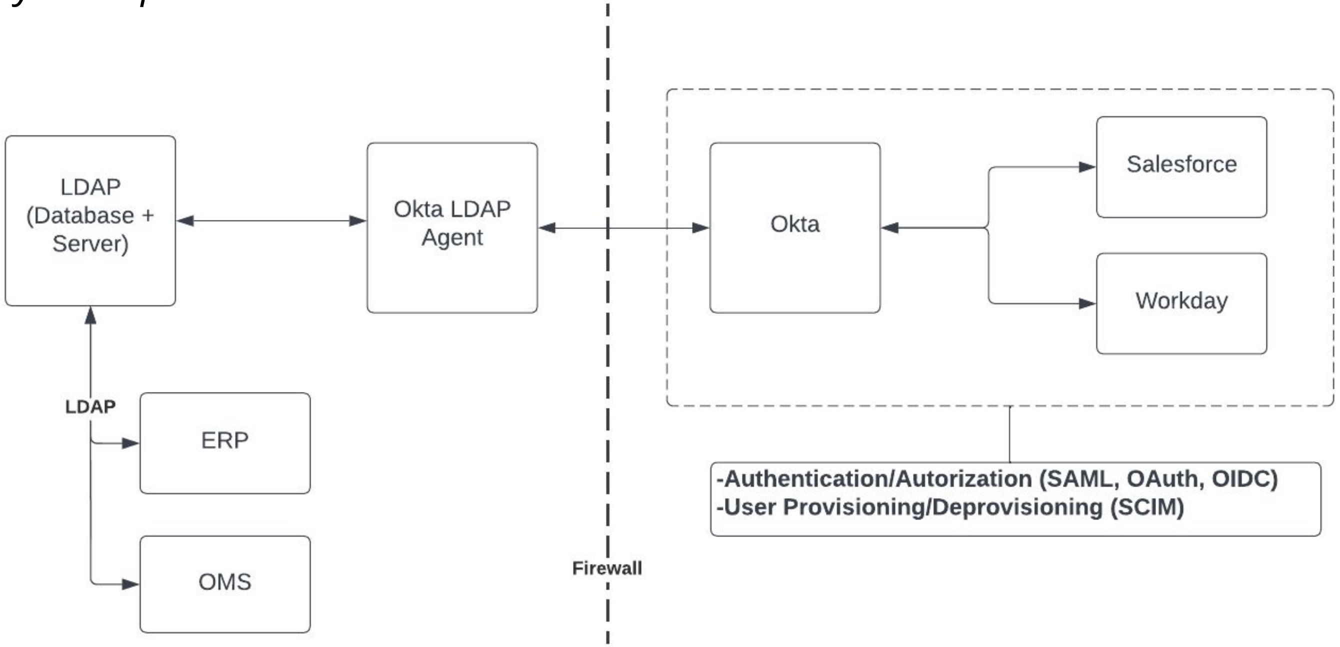


Azure AD



Putting it all together

"Company X has an LDAP-compatible user store to authenticate with an on-premise ERP system. They would also like to use those same credentials to login to their Salesforce org and Workday. New users created in LDAP system should automatically be provisioned in Salesforce, and inactive users should automatically be deprovisioned."


LDAP (Database + Server)
Okta LDAP Agent
Okta
Salesforce
Workday
LDAP
ERP
OMS
Firewall
-Authentication/Autorization (SAML, OAuth, OIDC)
-User Provisioning/Deprovisioning (SCIM)

SCIM vs. JIT Provisioning (within Salesforce context)

- JIT Provisioning (via SAML JitHandler or OIDC RegistrationHandler apex class) is used to create or update users **at the time they login to Salesforce**.
  - Can **create** users in Salesforce when users login from external IDP
  - Can **update** users in Salesforce when users login from external IDP
  - **Cannot deactivate/deprovision** users at all
- SCIM Provisioning (via REST API) is used to standardize identities between Identity Providers and Service Providers and is typically done **via batch synchronization jobs**
  - Can perform any CRUD operation on users in Salesforce, most notably user deprovisioning
- **Recommendation:** Use JIT Provisioning with external users (where one cannot determine all users accessing system until login), and use SCIM with internal users (directly managing Identity domain and user onboarding/offboarding)

Just-In-Time (JIT) User Provisioning

Just-In-Time (JIT) provisioning is an identity management concept where a user's account is created and updated dynamically at the moment it is required, rather than in advance. In traditional identity management, accounts are often provisioned ahead of time, which can be cumbersome to manage and can lead to orphaned or unused accounts. JIT provisioning seeks to minimize these issues by automating the account creation and update process, usually as part of a Single Sign-On (SSO) workflow.

How JIT Provisioning Works

1. **Initial Login Attempt**: A user attempts to access an application (the service provider) for the first time via an SSO link or direct access.
2. **Authentication**: The identity provider (IdP) authenticates the user, usually through username and password, multi-factor authentication, or another method.
3. **Token or Assertion Creation**: After successful authentication, the IdP generates a security assertion, often a SAML (Security Assertion Markup Language) or OIDC (OpenID Connect) token, which contains information (claims) about the user. This can include a unique identifier, role information, email address, etc.
4. **Token Transmission**: The security assertion is then sent to the application (service provider).
5. **Dynamic Account Creation**: The application receives the assertion and, instead of just using it for authentication, it also checks if a user account with the given details already exists.
    a. **If it exists**: The application updates the user's account details based on the information in the assertion.
    b. **If it doesn't exist**: The application automatically creates a new account using the information provided in the assertion.
6. **Resource Access**: Once the user account is either found or created, the user is granted access to the application's resources.`

Advantages of JIT Provisioning

1. **Reduced Administrative Overhead**: Since accounts are created on-demand, there's no need to pre-provision accounts, reducing manual effort.
2. **Improved Security**: JIT provisioning minimizes the existence of orphaned or unused accounts, reducing the attack surface.
3. **Up-to-date Information**: User information is updated dynamically, ensuring that the user's account information is always current.
4. **Scalability**: JIT provisioning can scale to a large number of users without requiring additional administrative effort.

Disadvantages

1. **Limited Attribute Syncing**: JIT is often limited to the attributes included in the SAML or OIDC assertion, which might not include all the information required for account setup.
2. **Reduced Control**: Automatic provisioning may not always adhere to specific organizational rules for account creation, leading to possible governance issues.
3. **Error Handling**: Errors in JIT provisioning could directly impact user access, requiring robust error handling mechanisms.

JIT provisioning is widely used in cloud-based applications and services that support SSO protocols like SAML or OIDC. It's particularly useful in scenarios where organizations are leveraging third-party applications and want to streamline the user access process.

Should you migrate your user store to the cloud?

- It depends…
- From a CTA scenario standpoint: if you are **decommissioning** all your on-prem applications in favor of cloud apps, then it’s worth consideration
  - Need to think about timing, data migration, user provisioning, etc
  - What is the benefit of doing this?
- If you are **not decommissioning** all your on-prem applications, then local LDAP/Active Directory should stay in place
  - Remember the part where it was stated that **on-prem LDAP architecture** was extremely complex?
  - You can’t just “lift and shift” this to the cloud - highly likely your legacy applications don’t support modern authentication/identity protocols
  - **Preferred solution: Manage user store on-prem, synchronize with IDaaS product like Okta or Azure AD to seamlessly provide enhanced features like Federated Identity, SSO, User Provisioning, etc**