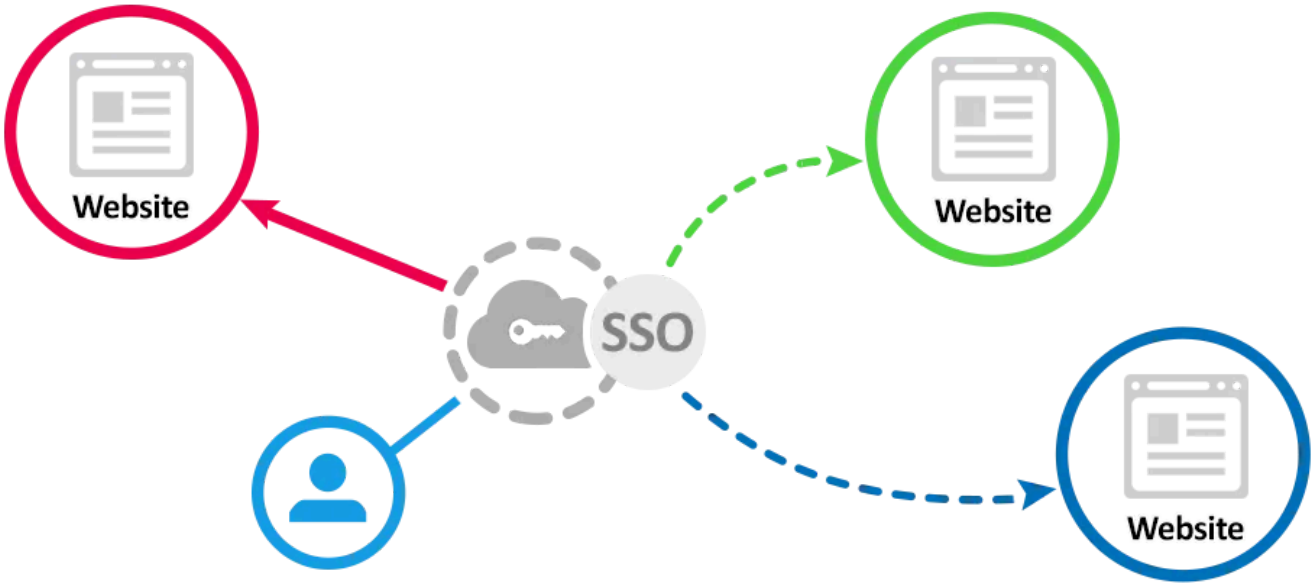
Identity Management: SAML vs. OAuth2 vs. OpenID Connect

Jad Karaki · Subscribe

6 min read · Apr 3, 2019

It all started with organisations needing a way to centralize their authentication systems for better management and security. That's where Single Sign On (SSO) came in. Single sign-on (SSO) is a centralized session and user authentication service in which one set of login credentials can be used to access multiple applications. Its beauty is in its simplicity; the service authenticates you on one platform, enabling you to transparently login to several internal services without having to log in and out each time.



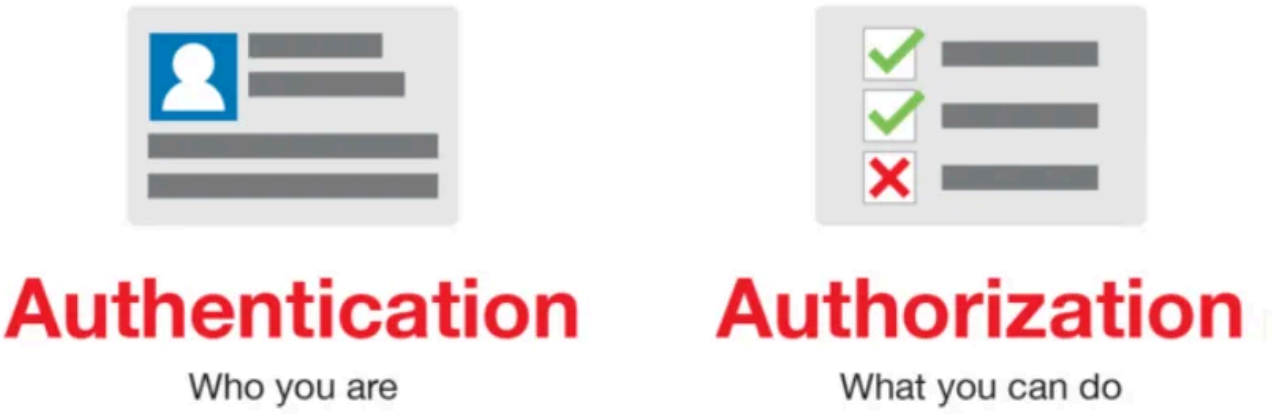
Next came third party app developers wanting to use internal APIs to integrate it with their products and solutions, this was clearly a challenge. Then came social networks and made things even more complicated, we currently have thousands of apps that support authentication through social networks like Facebook, Google, Twitter, LinkedIn, etc. The problem in these architectures is the challenge of keeping things as simple as possible and increasing security at the same time. The Solution? Federated Identities.

Currently, the three majors protocols for federated identity are: SAML, OAuth2 & OpenID Connect. Before diving deep into these three protocols, let's discuss some common concepts people tend to confuse.

Authentication vs Authorization

When it comes to security and access, authentication & authorization are two terms that people tend to overlook and are often mistaken to mean the same thing. Well, authentication is verifying your identity whereas authorization is verifying what you have access to.

Authentication is validating the login credentials before giving the user access to the system. When it comes to security, it's recommended to use at least two authentication factors before granting the user access to anything. (2FA, MFA, digital & physical tokens, etc.).



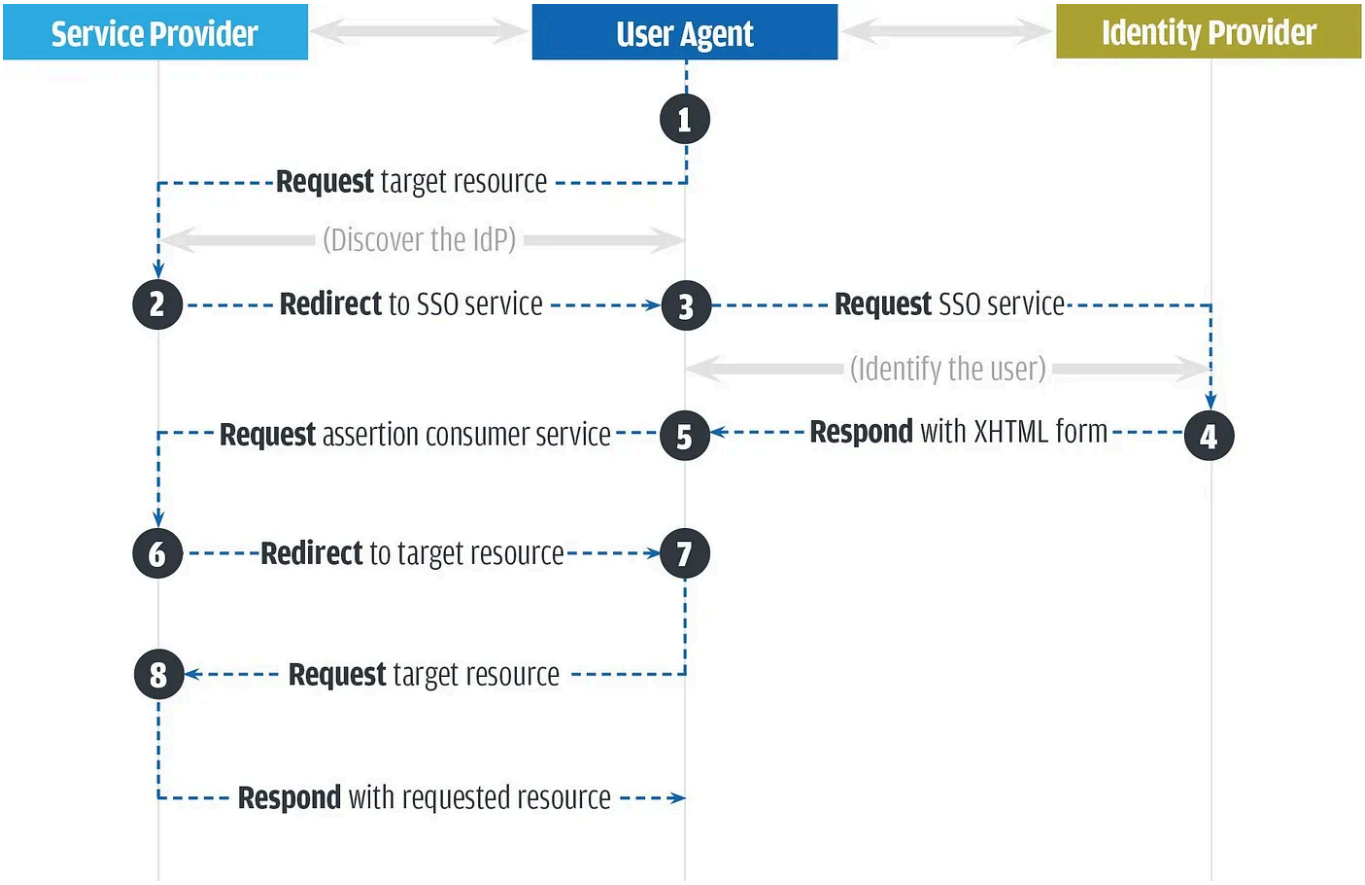
Authorization occurs after the authentication process by verifying your rights before granting you access to the required resources such as databases, files, repositories, etc. A practical example would be once an employee's logins have been verified, is to see to which floors he has access.

Both authorization & authentication are crucial for security and access management and although they both have different concepts behind them, they are a critical to the infrastructure of a system and understanding them is key for identity and access management.

SAML

Security Assertion Markup Language (SAML) is an XML-based open standard used for single sign on (SSO) implementations. SAML 2.0 was released in 2005 and is the current version of the standard.

SAML is used for both authentication & authorization between two parties: a Service Provider (Office365, Salesforce, G Suite, etc.) & an Identity Provider (Okta, OneLogin, Ping Identity, etc.). The Service Provider (SP) agrees to trust the Identity Provider (IdP) in the authentication process. This is done through a SAML XML document sent by the IdP containing the user authorization & authentication and then redirected to the service provider.



Let's consider this example:

- Identity Provider (IdP): Okta
- Service Provider (SP): Salesforce application

1. User tries to login to his company's Salesforce application from Chrome.
2. Salesforce app responds by generating a SAML request

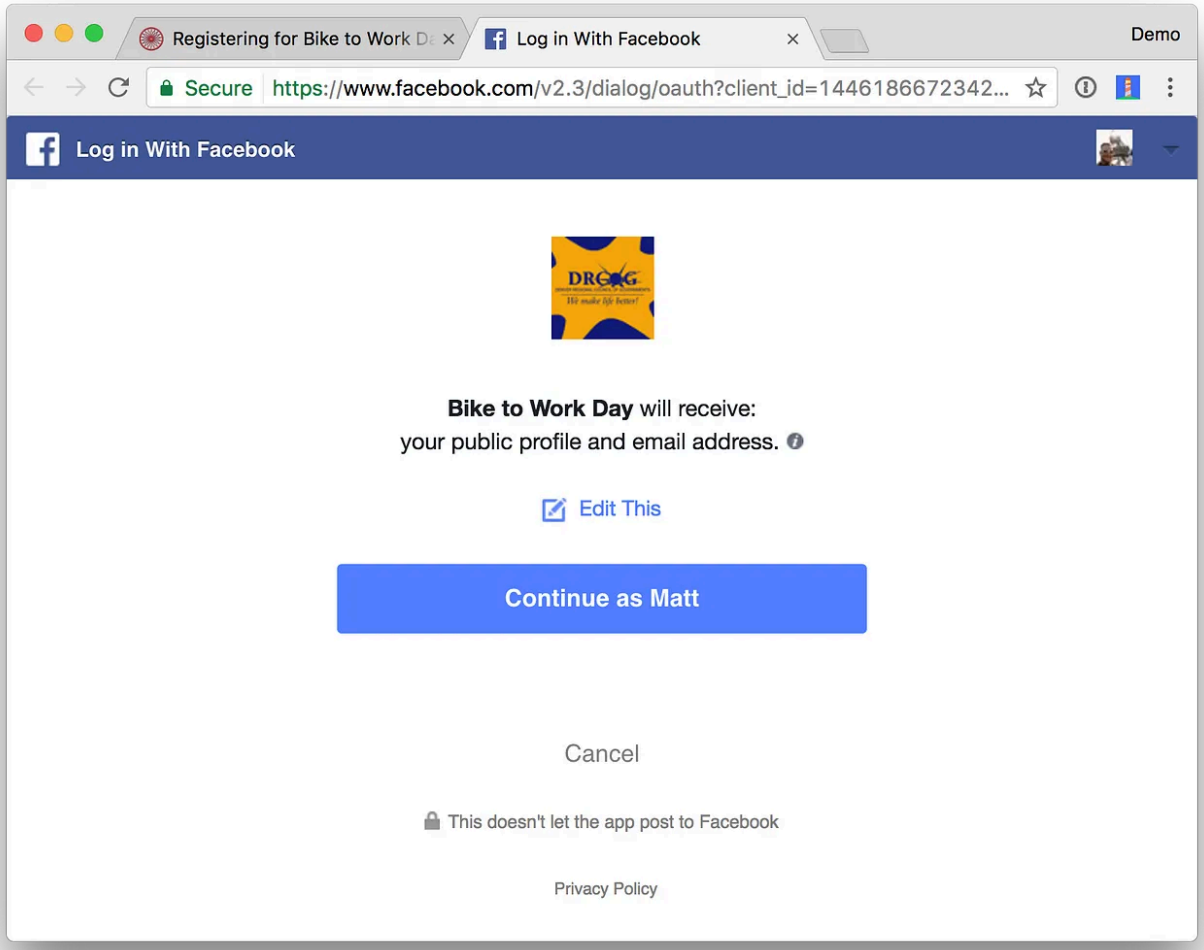
3. Chrome redirects the user to an SSO URL, Okta parses the SAML request, authenticates the user (this could be via username and password, two-factor authentication or MFA is user is not on company's internal network; if the user is already authenticated on Okta, this step will be skipped) and generates a SAML response.
4. Okta resends the encoded SAML response to Chrome
5. Chrome redirects the SAML response to the Salesforce app
6. If the verification is successful, the user will be logged in to the Salesforce application and granted access to all the various resources.

OAuth2

> *"How can I allow an app to access my data without necessarily giving it my password?"*

OAuth2 is an open standard used for authorization, it allows apps to provide application with 'delegated authorization'. Unlike other frameworks that provide authentication, OAuth only authorizes devices, API, servers with access tokens rather than credentials and it works over HTTPS.

If you've ever seen one of the dialogs below, that's what we're talking about. This is an application asking if it can access data on your behalf. This is OAuth.



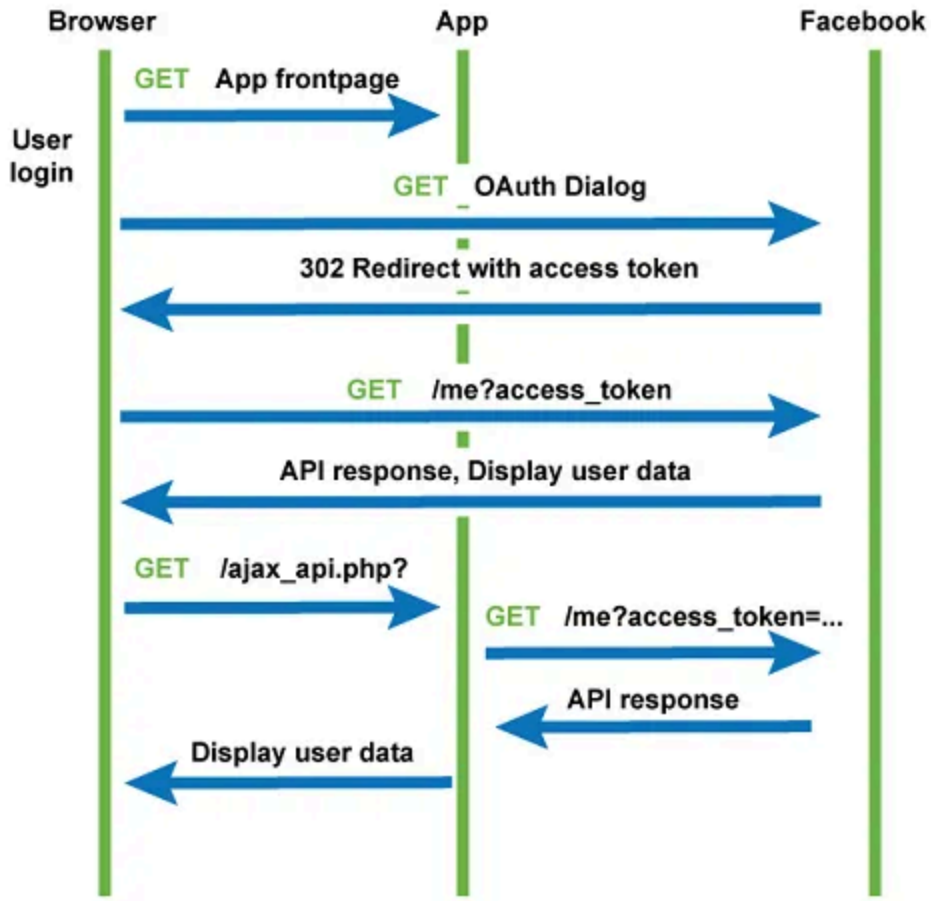
You can think of this like hotel key cards, but for apps. If you have a hotel key card, you can get access to your room. How do you get a hotel key card? You have to do an authentication process at the front desk to get it. After authenticating and obtaining the key card, you can access resources across the hotel.

OAuth defines four roles:

- Resource Owner: Generally the user himself
- Client: Application requesting access to a resource server

- Resource Server: Server hosting protected data (for example Facebook hosting your profile and personal information)
- Authorization Server: Server issuing access token to the client. This token will be used for the client to request the resource server.

Below is a diagram of the OAuth2 flow.



Let's consider this example:

1. Spotify wants to access your friends list from your facebook account.
2. You are redirected by Spotify to the authorization server (facebook in this case)

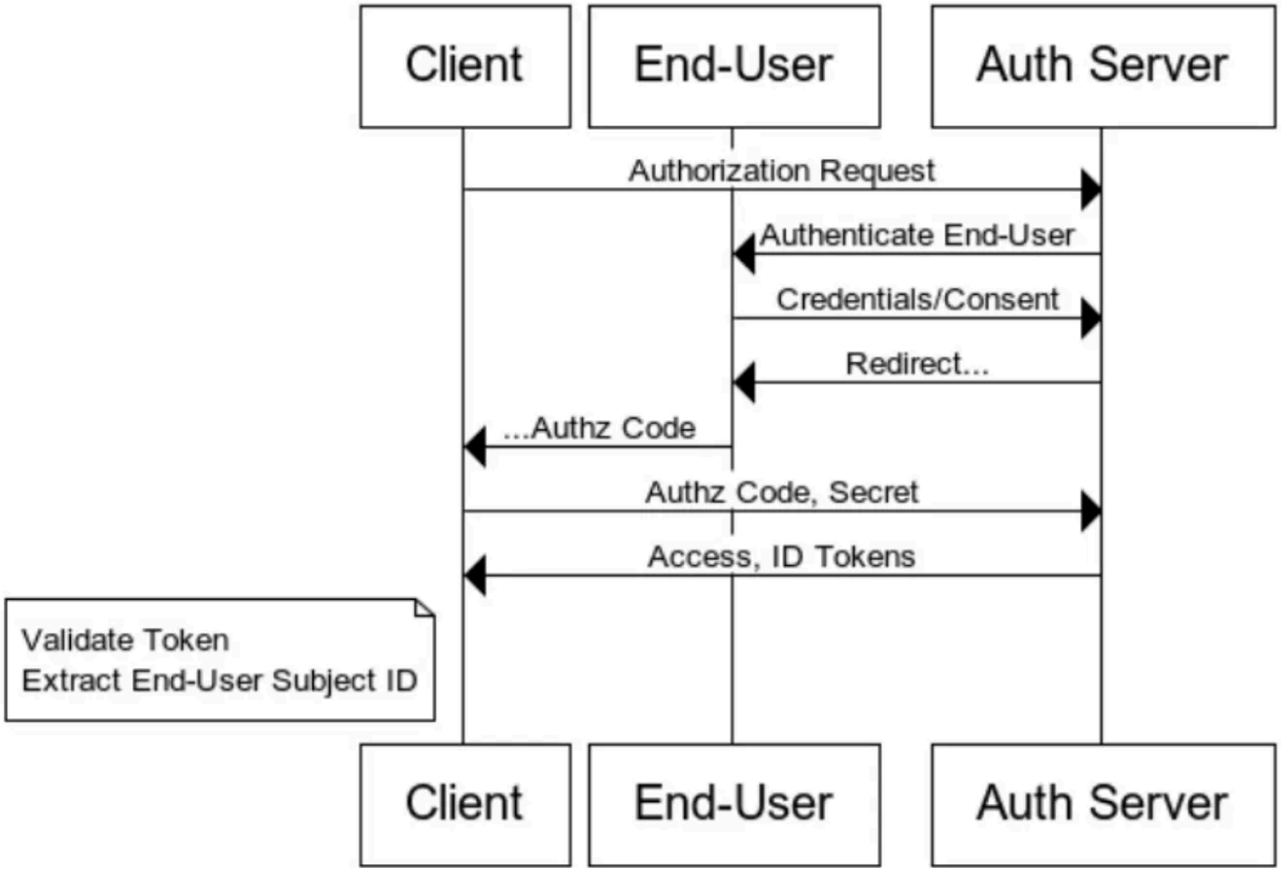
3. If you authorize access, the authorization server sends an authorization code to the client (Spotify) in the callback response.
4. Then, this code is exchanged against an access token between the Facebook and Spotify.
5. Now Spotify is able to use this access token to query the resource server (Facebook) and retrieves your friends list.

One thing to note is that the user never gets to see the access token, it will stored in the session. The authorization server also sends other information such as the token lifetime and a refresh token.

OpenID Connect

OpenID Connect is simple identity layer on top of the OAuth 2.0 protocol that extends OAuth2 and allows for 'Federated Authentication'.

The OpenID Connect process flow is similar to the OAuth2 authorization flow with the major difference being a 'id-token' that allows the user authentication.



Note that Federated Authentication is a completely different from Delegated Authorization. Let's take again the example of Facebook and Spotify

- Federated Authentication is logging to Spotify using your facebook credentials.
- Delegated Authorization is the ability of an external app to access resources. In this case,, Spotify trying to access your facebook friends list to import it into Spotify.

To wrap things up, here's a table summarising all three protocols/frameworks.

	SAML 2.0	OAuth2	OpenID Connect
What is it?	Open standard for authorization and authentication	Open standard for authorization	Open standard for authentication
History	Developed by OASIS in 2001	Developed by Twitter and Google in 2006	Developed by the OpenID Foundation in 2014
Primary use case	SSO for enterprise apps	API authorization	SSO for consumer apps
Format	XML	JSON	JSON

Feel free to comment on this article, would gladly discuss any related matter down below.

Identity Management | Cybersecurity | Identity And Access | Cloud Computing

Written by Jad Karaki

247 Followers · 42 Following

Demystifying Cybersecurity

Subscribe

Responses (8)

john lafata

What are your thoughts?

Lakshit Nagar
Jun 23, 2020

Great article. DEMYSTIFIED :-)

1 Reply

Salil Belapurkar
Jun 7, 2020

As you mentioned that OIDC is another layer on top of oAuth ; it means OIDC is performing Authentication as well as authorization.

I think you should correct the comparison matrix for OIDC where you have mentioned that its Authentication.

Please correct me if my understanding is incorrect

1 Reply

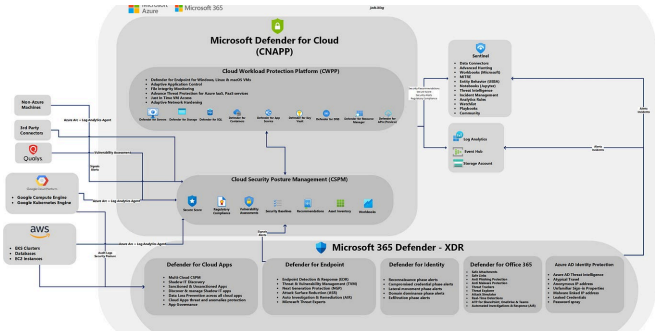
Akhilesh varma Gokaraju
Oct 31, 2019

Really nice article, you have summarized the concepts in a clear and concise manner

1 Reply

See all responses

More from Jad Karaki



Jad Karaki

Microsoft Defender for Cloud Apps—Architecture Diagram

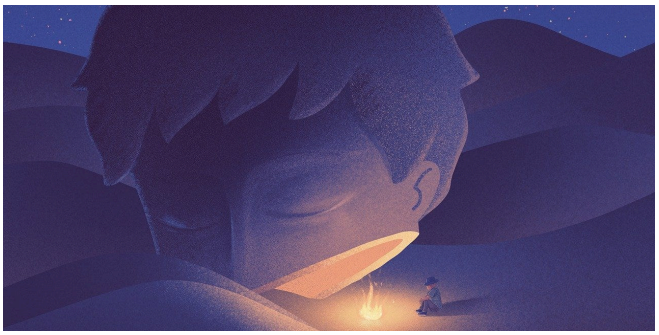
Apr 27, 2023 12

In The Startup by Jano le Roux

The Toothbrush That Cured My Bad Breath—And Made Me Believe in...

I finally discovered the Apple of toothbrushes.

Apr 1 3.8K 93

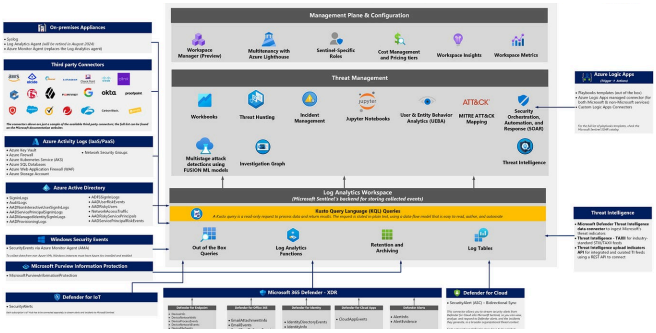


In The Startup by Jano le Roux

How This 17-Year-Old Quietly Built a $1.12M/Month AI App

I stumbled upon his exact strategy from A to Z and it's brilliant.

Dec 3, 2024 6.7K 180



Jad Karaki

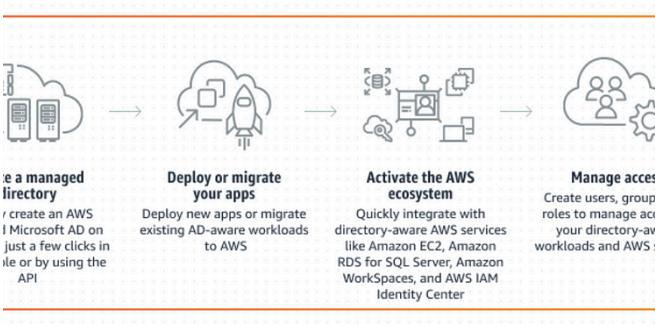
Microsoft Sentinel—Architecture Diagram

Here's a high-level diagram that illustrates Microsoft Sentinel's architecture and how it...

Jul 4, 2023 1

See all from Jad Karaki

Recommended from Medium

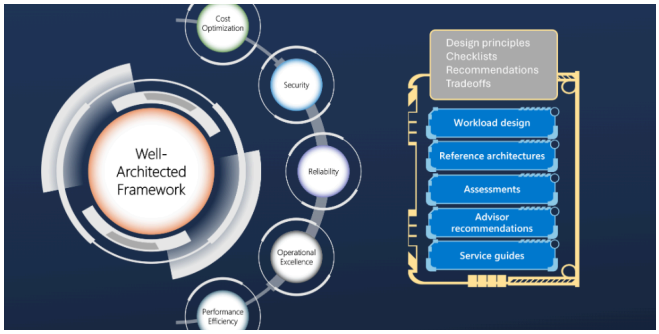


In AWS in Plain English by Alice the Architect

Understanding Microsoft Active Directory and its Integration with...

Seamlessly integrate Microsoft Active Directory with AWS for secure, centralized...

Nov 6, 2024 5

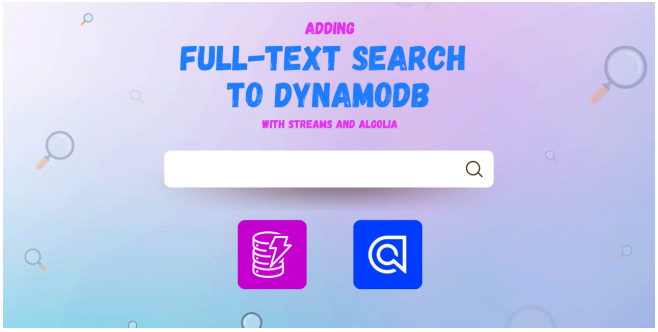


In Azure Cloud Techinical by Nseth

Azure Well Architected Framework

The Azure Well-Architected Framework is a set of best practices designed to guide...

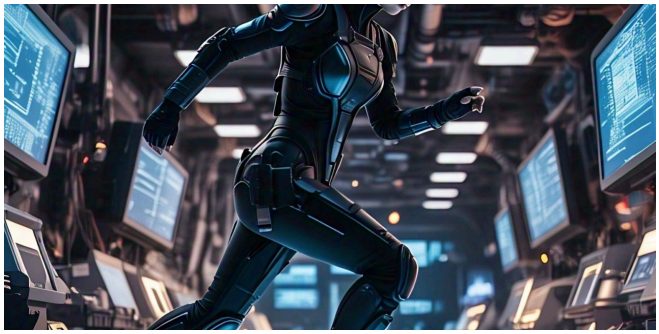
Nov 19, 2024 55



In Tech Odyssey by Uriel Bitton

Adding Full-Text Search To DynamoDB With Streams And...

A simple step-by-step guide on making your DynamoDB data searchable



Bl@ckC!pH3r

Lateral Movement: How to Detect and Stop it in Your Network

Apr 11

In Technology Hi... by Niluka Sripali Monnankula...

Wrapping It Up: Why SAML SSO Matters

A summary of why SAML SSO is a game-changer for businesses and users.

Mar 21 10

Apr 6 9

Aditya Bhuyan

How RBAC and ABAC Enhance Security and Compliance

Mar 7

See more recommendations