STRATA

Identity Orchestration and Multi-Cloud

What is Identity Orchestration and Why You Need It to Succeed with Multi-Cloud?

What is Identity Orchestration?

Multi-cloud and hybrid adoption create new challenges with identity fragmentation for enterprises. For example, as companies move to the cloud, they pick up new identity silos that come with each cloud platform. Due to technical and organizational reasons, it is impossible to consolidate identity into a single identity system. Consequently, multi-cloud now means multi-identity.

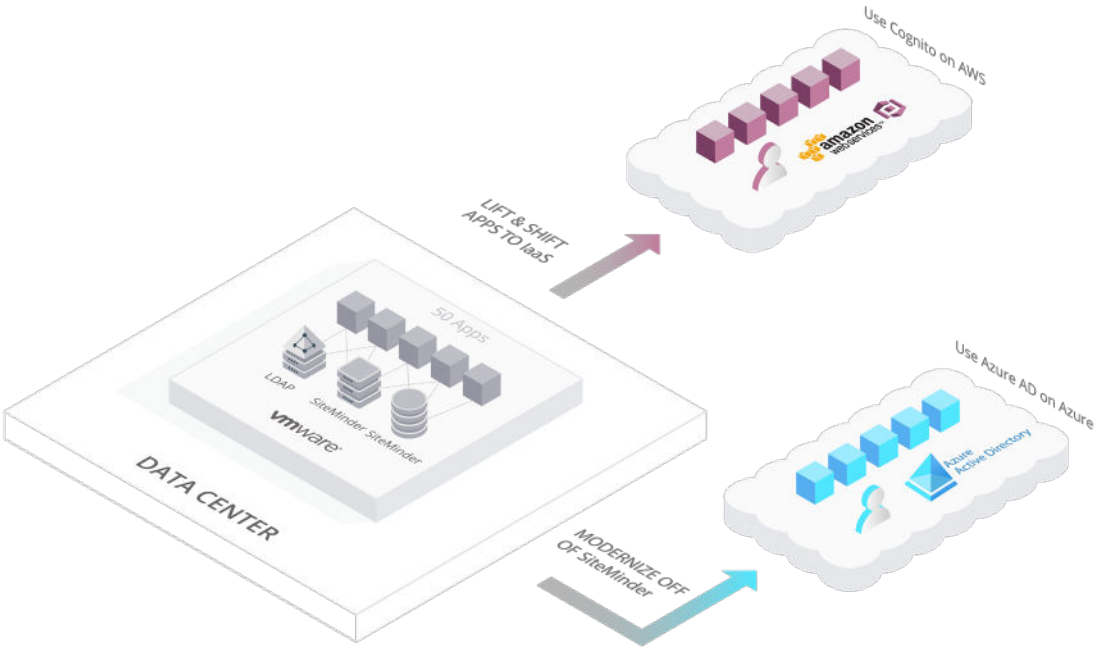
Identity Orchestration is a New Breed of Identity

Managing identity in the multi-cloud world requires a distributed identity model. Users must have consistent access to apps running on-prem or on multiple cloud platforms, whether from the cloud or within the enterprise network. Identity orchestration is the next generation of identity management software that makes this possible.

> ***Identity orchestration software creates a logical identity fabric that ensures identities and user access policies are consistent across disparate identity systems and multiple locations, both in the cloud and on-premises.***

Consider what is required to transition 50 Web apps from on-premises to the cloud. These legacy on-premises apps run on VMware, protected by SiteMinder legacy identity infrastructure. The objective is to lift and shift these apps to run on Azure and AWS and use modern identity capabilities from Azure AD or AWS Cognito.

The apps' location will change, but our fictional company doesn't want to change the user experience or recreate access policies in multiple identity systems. In other words, they want to move apps to multiple cloud platforms with the least amount of disruption, in the fastest, most cost-effective way, while never compromising on security.



Managing consistent identity and policies across multiple identity systems.

These are the technical challenges to address for this scenario:

Challenge #1: Multiple identity systems to manage

The company uses SiteMinder to protect its apps on-premises. It plans to use AWS Cognito to provide identity for its AWS-hosted apps and Azure AD to provide identity for Azure-hosted apps. This means that in addition to managing the various systems that provide user attributes and MFA, they now also have three different identity systems to manage.

Challenge #2: Apps require rewrites when modernizing identity systems

The company's apps are hard-wired into SiteMinder using different types of HTTP headers and authentication mechanisms. Historically, moving these apps has meant rewiring from one identity system to another manually and at considerable cost and time.

The company wants to use standards like SAML and OIDC, but none of the apps support these protocols and, therefore, require refactoring to accept SAML or OIDC for SSO. Furthermore, to use SAML and OIDC, each identity must have a local account to match up the SSO session with the identity profile. These local accounts don't exist in Azure or AWS. In the past, this meant that custom synchronization code must be developed (and maintained).

Challenge #3: All users' passwords need resets during big bang migration of identities

The company's identities on-premises are stored in LDAP, and they must be replicated from LDAP to Azure AD and AWS. Because of password hashing, it's impossible to 'copy and paste' users from one directory to another because authentications will fail when compared to a hash. Historically this has meant forcing users to do a password reset workflow as part of a 'big bang' migration.

Challenge #4: Legacy and modern identity systems need consistent policy enforcement

The company has long used SiteMinder's sophisticated policy capabilities, including contextual access, Active Responses, and even the retrieval of attributes at runtime to personalize user experience at sign-on. Azure AD does not support the same functional capabilities as SiteMinder. Historically this has meant compromising and down-grading access policies to fit the lowest common denominator of policy, thus weakening security protections.

Challenge #5: Working within the company's complex environment

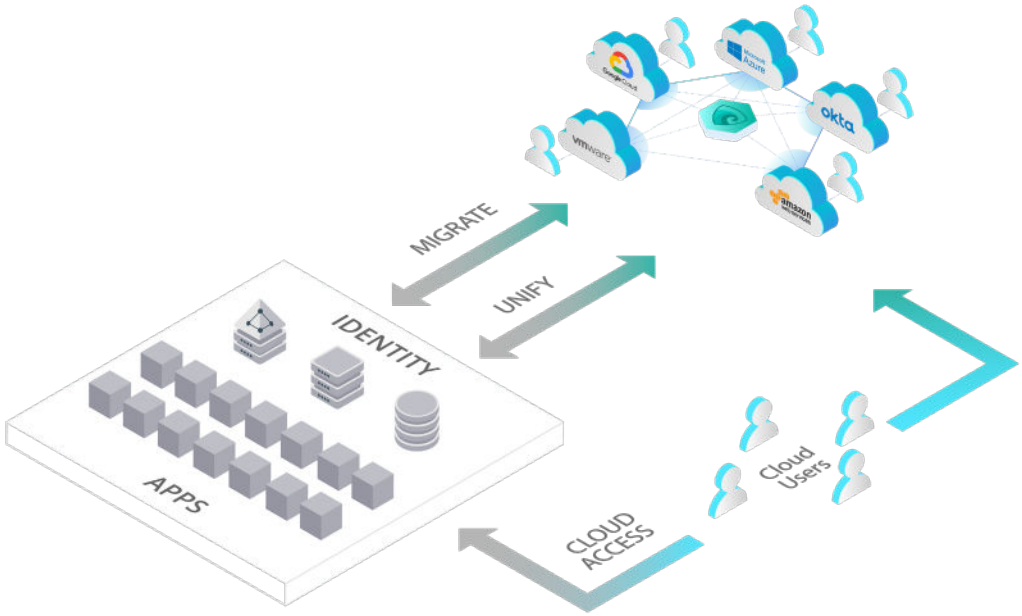
The company has a robust enterprise-grade networking infrastructure developed over many years that provides layers of defenses to keep bad guys out and selectively allow good users in through VPNs. The VPN approach has collapsed under the rapid rise in the number of employees working remotely. The company needs to extend access to its on-premises apps securely at the edge of its data center. Historically this has meant radical reconfigurations of enterprise networks at great expense and time delays.

In addition to the technical challenges described, the company is under pressure to cut costs in the near term to pay for the increase in spending to support remote workers. Timelines are severely compressed. What would historically take several years to accomplish with custom code and complex migration projects must now be done in months (or faster) to keep pace with market changes.

Lastly, a word about lock-in. With the rise of the dominant cloud platforms from Amazon, Microsoft, and Google, has come a greater risk of getting locked into these vendors. It won't come as a surprise that identity is one of the biggest sources of lock-in. Companies don't want to get locked into any platform, and the fact that they move to multiple clouds is a reflection of this. However, if not done correctly, an organization can find themselves moving from its legacy on-premises lock-in to cloud lock-in.

Identity Orchestration is Here to Solve These Problems by Providing an Identity Fabric

An Identity Orchestrator is a lightweight service that deploys in the cloud or on-premises. Each Identity Orchestrator runs as a service on a Linux server and reads its configuration from either a local or centralized YAML configuration file.



Modernizing apps without rewiring identity

Identity Orchestrators use Connectors, Workflows, and App Gateways to orchestrate behavior across identity systems and create an abstraction layer that applications use to integrate with any identity system without changing application code or modifying configurations. Identity Orchestrators can move policies, configurations, and identities across any identity system. They are also used to route login requests to different identity providers or lookup and retrieve user attributes, groups, and other identity data from various identity stores.

The following foundational requirements define identity Orchestration:

1. **Natively distributed**. Identity Orchestration must be engineered from the start to solve the distributed problems of multi-cloud and multi-identity use cases. Identity Orchestration will not succeed by retrofitting a centralized identity system to perform in a distributed way. This has been proven in compute where we have seen the rapid rise of Kubernetes due to its ability to natively manage distributed workloads using its distributed architecture.

2. **Consistent identities.** Identity Orchestration must enable consistent identities across multiple clouds and identity systems by programmatically automating identities into the various identity providers and creating a composite identity profile by building attributes from several identity providers in real-time.

3. **Consistent policies.** Identity Orchestration enables consistent user access policies by using an identity abstraction layer to normalize the definition of user access policies in a common syntax. It also must normalize enforcement of user access with meta-policies to fill functional gaps in policy that commonly occurs between different identity systems.

4. **Identity abstraction layer.** Identity Orchestration relies on a normalized identity fabric that abstracts the various underlying identity infrastructures that an organization uses. This abstraction layer unifies different identity systems' APIs, data models, user access policies, and feature sets into a consistent identity fabric, saving effort learning multiple APIs and identity systems.

5. **Distributed deployments.** Identity Orchestration must be fit to deploy in many different configurations that reflect the often complex environments in today's environments. This means running on-premises and in the cloud. Identity Orchestration must also be highly performant, meaning it must handle all kinds of traffic and run as either a proxy or adjacent to apps through a sidecar model, all of which allow it to deliver elastic scale and rock-solid resilience.

These foundational requirements are critical and define what Identity Orchestration must be capable of to support today's multi-cloud and multi-identity use cases.

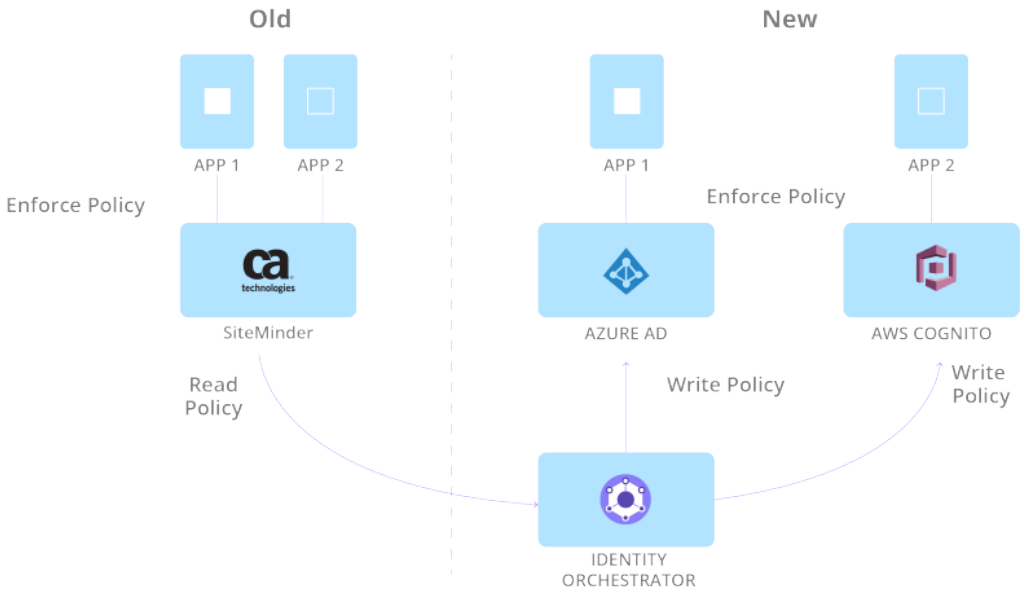


FIGURE 1 Managing consistent identity and policies across multiple identity systems.

Let's apply Identity Orchestration to the scenario on page 6 to understand how organizations can move to multi-cloud while managing identities and access policies consistently.

Use case #1: Managing consistent identity and policies across multiple identity systems.

To extend the policy defined originally in SiteMinder, Strata Identity Discovery extracts policies from SiteMinder. Then the Identity Orchestrator creates the system-specific access policy in Azure AD and AWS using their respective APIs. Next, Identity Orchestrator is configured to present the identity to Azure AD and AWS Cognito with appropriate attributes so that Azure AD and AWS Cognito can enforce consistent access.

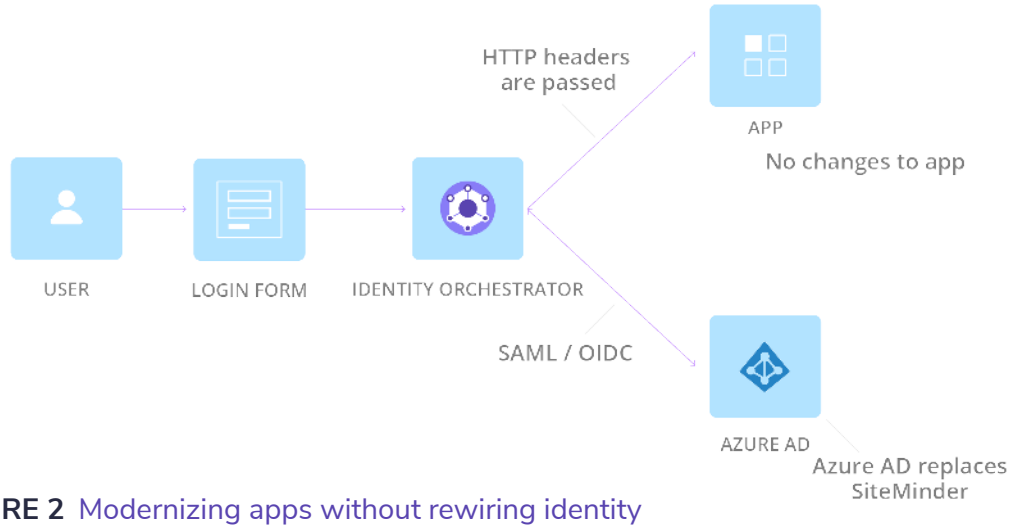


FIGURE 2 Modernizing apps without rewiring identity

Use case #2: Modernizing apps without rewiring identity

Sitting between the apps and identity systems, the Identity Orchestrator determines where to route a user for authentication, using whatever standard protocol that identity system expects. The Identity Orchestrator also retrieves attributes from additional identity stores. The app behaves as though still integrated with the legacy identity system when, in fact, the Identity Orchestrator transparently transforms SAML or OIDC authentication sessions, and attribute claims the proprietary sessions and headers the legacy app expects. Deployed as a proxy or sidecar, the Identity Orchestrator deploys with no change to apps or identity systems.

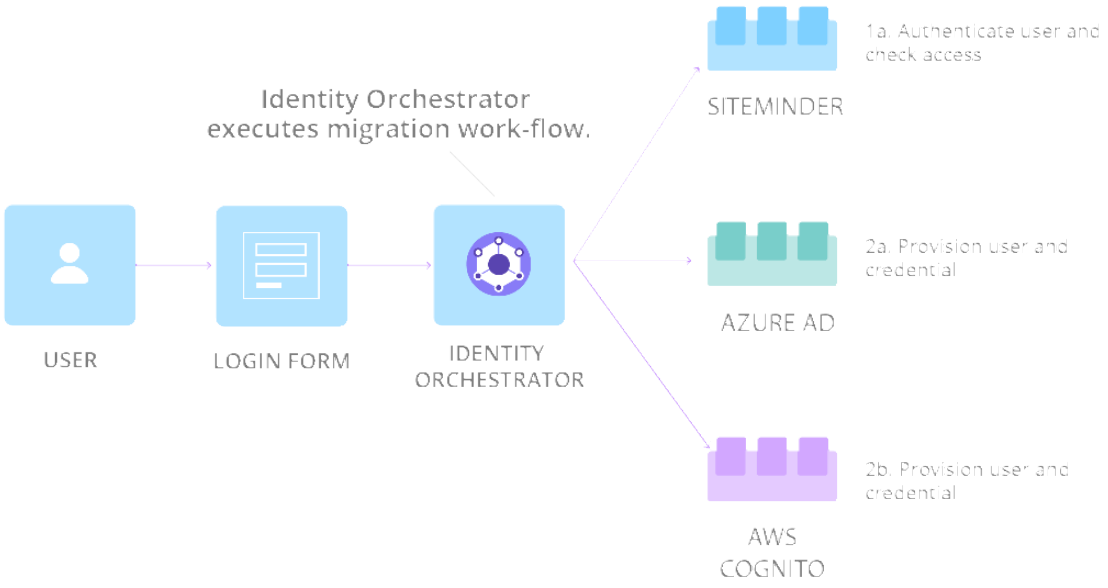


FIGURE 3 Avoiding big bang migration of identities with incremental migrations

Use case #3: Avoiding big bang migration of identities

For incremental user migrations, the Identity Orchestrator proxies the login page and requests authentication from the legacy identity system. Then, the Identity Orchestrator fetches additional attributes and creates a user accounts Azure AD, assigning groups or roles and determining how to apply app access policies. This process takes place over several months where the majority of active users are migrated.

The accounts that have not logged in during the period are considered dormant accounts and are bulk-migrated and put into a password reset process to revalidate the user identity. This eliminates the risk of dormant accounts being used by hackers to gain illicit access.

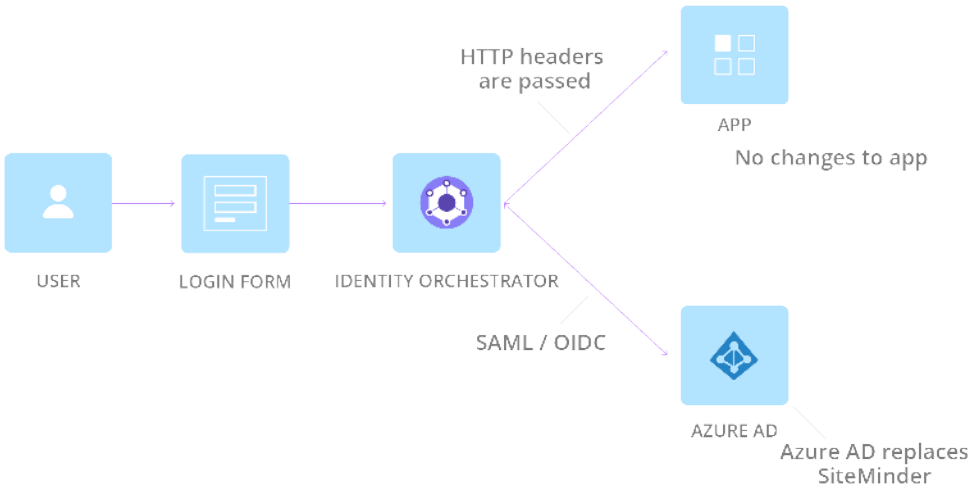


FIGURE 4 Extending SAML to apps that don't support SAML.

Use case #4: Extending SAML to apps that don't support SAML

For legacy apps that don't support SAML or OIDC, the Identity Orchestrator handles the processing of federated identity. Upon success, it passes the user principal (User ID) to the web app the way it already consumes identity, through HTTP headers.

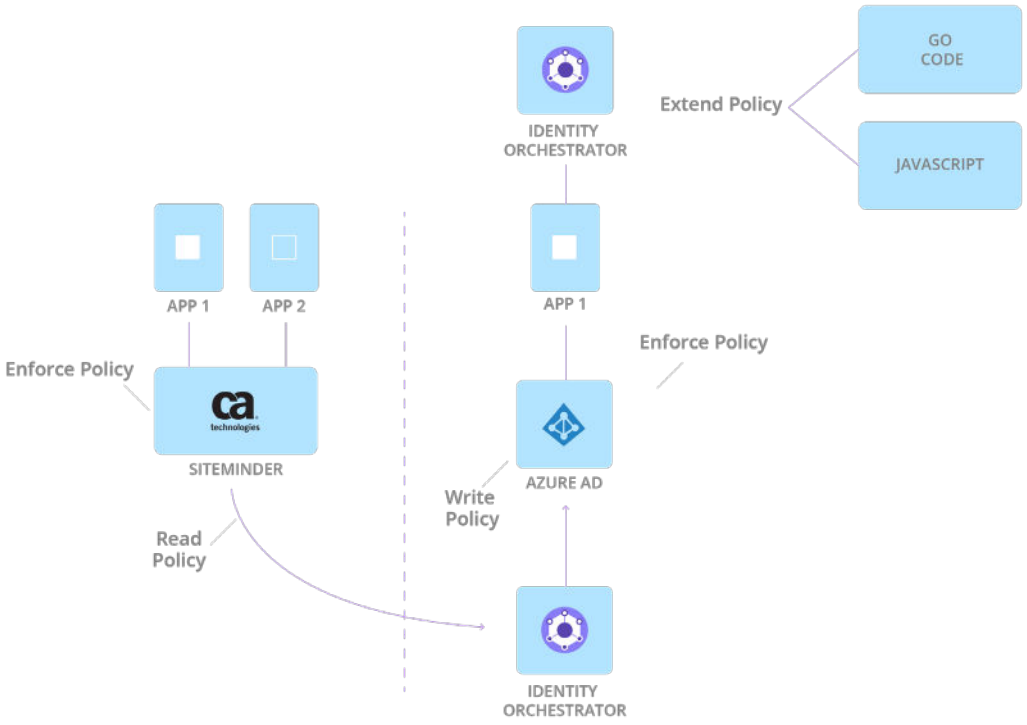


FIGURE 5 Consistent policy enforcement between legacy and modern identity systems.

Use case #5: Enabling consistent policy enforcement between legacy and modern identity systems.

The Identity Orchestrator bridges functional gaps in user access policies between the complex policies enforced by legacy identity systems (such as contextual access and custom calculated attributes) and the simpler policies enforced by cloud identity provider (such as app-based MFA). To do this, the Identity Orchestrator uses JavaScript and Go Service Extensions that extend the built-in policies of Azure AD to provide nearly infinite flexibility to enforce consistent user access policies for any app.

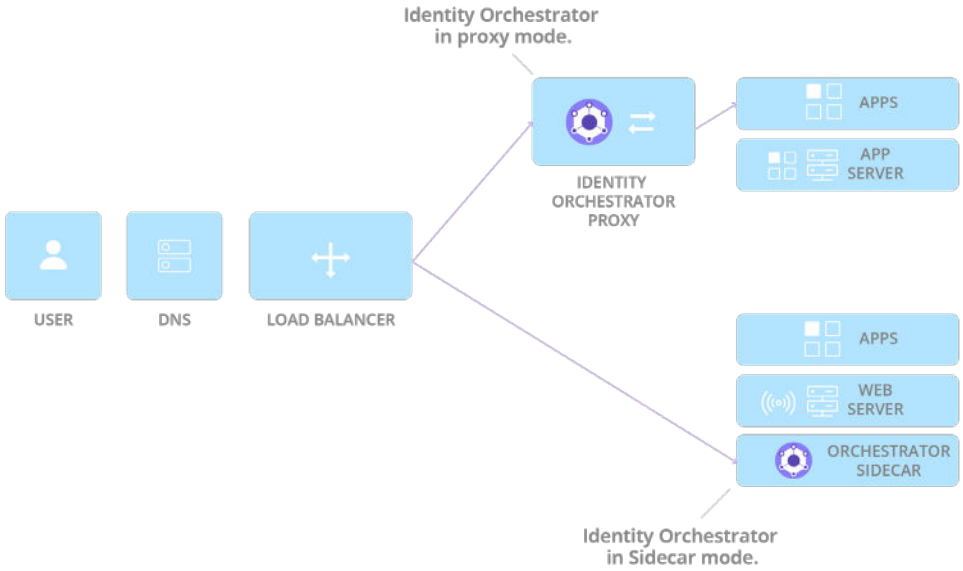


FIGURE 6 Flexibility for complex networking environments

Use case #6: Working within complex networking environments.

The Identity Orchestrator deploys seamlessly into any networking topology, proxying the upstream apps and working without changes to load balancers, firewalls, and the web-tier hosting the apps. To protect against 'side door' access, the Identity Orchestrator deploys as a 'sidecar' on web servers (such as IIS or Apache) and app servers. The Identity Orchestrator acts as a Zero Trust app gateway, securely extending access to apps whether those apps are on-premises or run in the cloud.

The Benefits of Maverics Identity Orchestrator

Strata's Maverics Identity Orchestrator enables organizations to confidently succeed with hybrid and multi-cloud environments using multiple identity systems, knowing there is a way to consistently manage access to apps running on multiple cloud platforms. This solution saves money and time by deprovisioning legacy infrastructures, avoiding costs to refactor or rewire apps to work with modern identity, and turning complex migration projects into rapid modernization projects that happen in weeks instead of years.

Strata improves agility by decoupling apps and identity and works with any identity system that is needed, with a simple change in configuration. It also improves security by moving your applications off end of life, outdated legacy identity systems, deploying modern security tools like MFA, identifying dormant accounts, and deploying zero trust for user access to apps.

Conclusion

The bottom line is that multi-cloud means multi-identity. This creates complex challenges to migrate and maintain identity and policies consistently across multiple clouds, unify on-premises identities, and modernize legacy identity. Deploying Strata Identity Orchestrator saves time and money while improving agility and security in these most crucial scenarios.

Take the next step by seeing a demo of Strata's Maverics software and requesting an Express Proof of Concept (POC) by visiting us here: **www.strata.io/demo**