Accelerate Cloud Migration by Distributing Context-Rich Identity to Modern Applications

Styra Declarative Authorization Service (DAS) for Cloud-Native Entitlements

Built with Open Policy Agent (OPA)

As organizations proceed down their digital transformation journeys to enable agility and business transformation utilizing cloud-native technologies, they are taking a pragmatic approach to achieving these goals. The transition from primarily an on-premise IT infrastructure to the cloud has proven to be a valuable asset for an organization during their digital transformation, but it comes with its own challenges and opportunities.

To take advantage of Platform-as-a-Service (PaaS) and Infrastructure-as-a-Service (IaaS) companies have adopted techniques like CI/CD pipelines, Kubernetes, Terraform, Services Meshes to provide the basic infrastructure to run their business solutions. These technologies are complex as they manage the storage, compute and networking needs in the cloud.

Companies need to understand who is managing and modifying their infrastructure and applications. With the introduction of the cloud-native techniques the attack vector has increased dramatically, only exacerbating the challenges that devOps teams are facing. To help with this problem, they need to treat their policies that define "who can do what to what?" in a fashion that allows the policies to be authored, edited, tested and versioned. As well as be able to indicate to the policy author what the impact will be on the system if a particular policy is modified.

To achieve this, Policy-as-Code (PaC) evolved to fill this need. The key element of this new capability is to transition all the policy best practices that are housed in years of corporate documents, PDFs, powerpoint decks and team members heads into a corporate artifact (policy) that can be authored, tested, verified and placed under source control so that the organization can evolve their policies over time, while maintaining control over what has been modified.

PaC has become the preferred approach to externalizing authorization requests across the entire cloud-native stack. To this point the main focus has been on establishing best practices using PaC to manage an organization's infrastructure. Companies are starting to become comfortable with the concept of PaC for their infrastructure, and many are starting to ask "Can we modernize our applications policy to take advantage of PaC?" and "Can we provide a common authorization server for our cloud-native and self-hosted applications?"

In addition, the pandemic has exacerbated the situation, where now there is an increased focus on allowing access to corporate resources from anywhere, anytime and for anyone. Remote workers are no longer rare, it is now the standard for many organizations to allow their internal and external users to access the company resources remotely, from almost anywhere, on various devices and at all hours of the day.

> "Can we modernize our applications policy to take advantage of PaC?"

"Can we have a centralized unifying policy solution that allows for the policy management for infrastructure and applications, built on industry best practices, that is verifiable and understandable by both our organization's Identity and Access Management teams and implementable by engineers with minimal expertise/experience?"

Given all these desires, the move to cloud-native, the need to provide fine-grained access control across infrastructure and applications and the problems that the increase in pandemic has caused, it is no surprise that engineering teams are looking for a solution that can help with all these corporate activities.

IAM teams are asked to manage existing legacy applications, integrate new SaaS solutions and to evolve into a microservices architecture to enable corporate agility and business transformation. Unfortunately, many of the systems that the IAM teams have control over were either written many years ago, in various languages, by various teams, and often with conflicting or often repeating access control models. One application may provide basic authentication with no authorization, another application may have some elementary authorization services that were hand crafted for the particular solutions needs. Other applications may implement a Roles-Based Access Control (RBAC) or more advanced applications may implement Attribute-Based Access Control (ABAC) facilities.

The challenges for the IAM teams are many. They are responsible for shepherding the transition to the cloud for many of the traditional IAM solutions, providing integration between various internal and external systems, and currently, focusing on the authentication of users into the corporation's resources. It is not surprising to see developers gravitate towards offerings that help them with these challenges, especially authentication. This is due to the fact that the process of logging a user into a website or mobile device has become well known, and the patterns for authentication and Single-Sign On (SSO) are easily available from many Identity or Access solutions like Okta, Auth0, OneLogin, Microsoft, etc. Standards like SAML, OIDC have evolved to provide the patterns to allow a user or service to login or authenticate to ascertain who a user claims to be.

The challenging aspect of allowing users to login is to make sure they are who they claim to be, is that we have all been taught that simply using a username and password is insufficient to protect many corporate solutions. Multi-Factor Authentication (MFA) has become the rallying cry of the IAM space, and many companies focus on exactly that, how to enable an organization/engineer to incorporate strong MFA into existing or new

> Can we provide a common authorization server for our cloud-native and self-hosted applications?"

applications. It is no longer desirable to allow non-standard authentication mechanisms to be hand-crafted by engineering teams. The state of the art within authentication is that it is relatively easy for an engineer to become skilled in utilizing authentication solutions from 3rd party providers. Do you really want your billing system writing custom authentication flows to allow both your internal employees and external partners? What about the flows that users need to obtain their MFA? Or to revoke them in the case of a data breach? Or to understand which authentication flow would be most appropriate given contextual information concerning the users location, device, or previous behavioral patterns.

Given all these constraints and challenges it is not surprising that IAM teams are looking for ways to reduce their burden. As mentioned above, authentication is a well known and solved problem. IAM teams now have standards based tools and frameworks they can utilize to assure with a high degree of certainty that the user authenticating is who they claim to be.

Authentication, check that one understood, standardized, productized, and done.

But, the next step, authorization, understanding and controlling what actions or resources a user or service can access is not at the same level of maturity that authentication is at. That is not to say that authorization is not well known or understood, only that there is not an easy path for IAM engineers to either modernize an existing solution to use industry best-practices for authorization, or to create from the ground up, a new solution based on industry standards and policy libraries.

It is not surprising that the benefits and controls that organizations have been able to achieve with PaC within their infrastructure (Kubernetes, Terraform, etc) are starting to be noticed by organizations and specifically by the IAM teams and the legions of engineers who are tasked with implementing authorization.

Application Entitlements — or sometimes just referred to as Entitlements — is the capability to externalize an organization's application authorization decisions to a centralized PaC authorization service.

There are a many advantages of pursuing a PaC model for Entitlements that organizations are looking at:

1. Introducing PaC which allows users to codify policy decisions in software. This is beneficial because PaC decouples decision logic from business logic in services.
2. Treating PaC allows for automated decision-making, giving developers and engineers the independence to manage feature defining work without sacrificing compliance.
3. Defining policy at the IAM corporate level by a centralized IAM governance team and made available to the organization's application developers, these policy definitions can be left to a smaller part of the organization, which is more in-tune with compliance and regulation versus leaving it to developers in the organization to define these policies

4. Within applications, there are a lot of policy decisions. These application developers know about their application and service but they don't always know security. So, decoupling allows the application veloper and platform engineers to delegate the decision and policy to security and compliance teams in the organization.
5. Many applications have different authorization needs, and a one size fits all is not appropriate. Maybe the application would be best suited for a RBAC model, or maybe an ABAC model. Utilizing a centralized PaC solution, applications can evolve over time to take advantage of more contextual information to help assist an application transition from no authorization to RBAC and then to ABAC.
6. Being able to slide across the RBAC - ABAC spectrum with little or no application modification, enables agility and increased control with minimal developer involvement.
7. Entitlements can be created, deployed and tested without an application re-deploy.
8. Pre-defined Authorization Policy packs can be developed providing best practices for RBAC/ABAC facilities.
9. IAM teams can establish the guardrails (policies) for the organization authorization needs. For example, a general corporate rule may enforce "Access denied on Saturdays and Sundays", as well as very complex rules based on corporate governance and auditing needs.
10. Application teams can review the authorization policies that the IAM team have established and can request an exception for a rule that may cause issues for one of the organization's applications.
11. Entitlements stored as PaC can be controlled in part by existing systems of record like LDAP and ActiveDirectory.
12. Policy Decisions can be audited and analyzed centrally.
13. Improve compliance with centralized audit across multiple lines of business and multiple environments.
14. Security Assurance by unifying best practices with Policy packs.
15. Policy treated as code and versioned.
    A. Allows you to swap in new app code or features.
    B. Allows you to swap in new authz rules.
16. Styra Declarative Authorization Service (DAS) and Open Policy Agent (OPA) provide guardrails to manage authorization policy for both apps and infrastructure—which in turn simplifies the services themselves.
17. Developer churn is a concern of many organizations and being able to centralize the policy decisions and policy specifications as code helps protect an organization from brain drain. Being able to construct authorization policy with pre-canned policies allows non-developers to set the guidelines or guardrails for the organizational policies. There is no need to educate the IAM team on the PaC developer language.
18. Remote teams can focus on the code while IAM teams focus on the policy.
19. No need to educate out-sourced teams with corporate IP.

Entitlements Management Overview

Today, many organizations are accustomed to using a centralized Entitlements management system for their custom applications. Instead of hard-coding into every custom software application the rules and regulations about what users can perform which actions (and when), developers will instead integrate those custom apps with a separate system that handles all of those rules and regulations on behalf of the application.

Decoupling those rules, regulations, and policies (i.e. Entitlements) from the application and using a dedicated, logically centralized Entitlements system brings a number of benefits:

- Entitlements can be updated without re-deploying any applications.
- Entitlements can be read and written by people who are not developers, e.g. business, security, and compliance.
- Entitlements can be controlled in large part by existing systems of record like LDAP and ActiveDirectory while still preserving tight controls over the load and access to those systems.
- Decisions can be audited and analyzed centrally.
- Governance over Entitlements, how they are changed, and their roll-out are handled uniformly across all applications.

That is, many organizations today manage Entitlements for their applications using a centralized service that looks something like the following diagram.

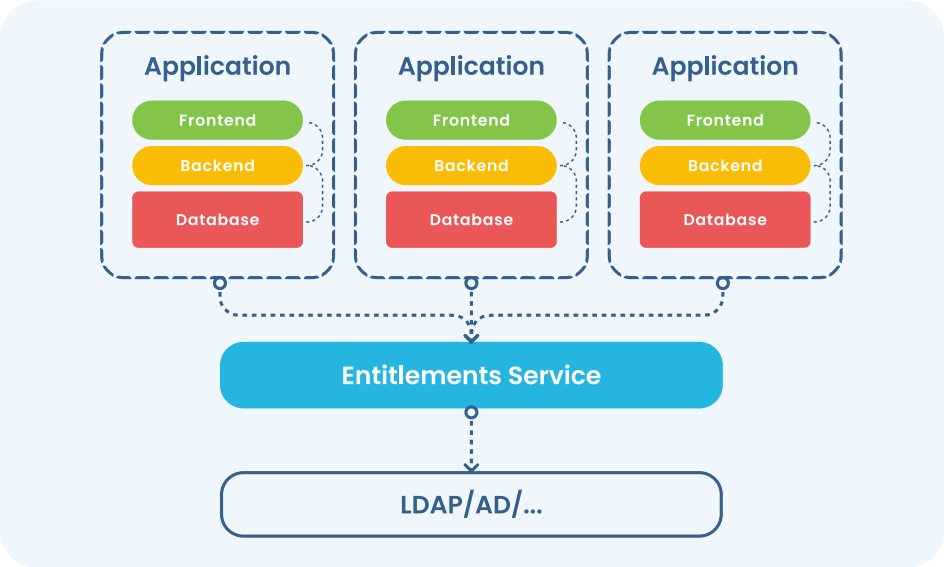

Figure 1. Entitlements Management

Cloud's Impact on Entitlements Management

At the same time, many of the same organizations that find so much value in an Entitlements system are also migrating their custom software applications to the cloud and modernizing those applications so that they are built and run in a cloud-native way. The naive approach to moving applications to the cloud while retaining a centralized approach to Entitlements management is to continue integrating cloud-applications with the on-premise Entitlements service, as shown below.

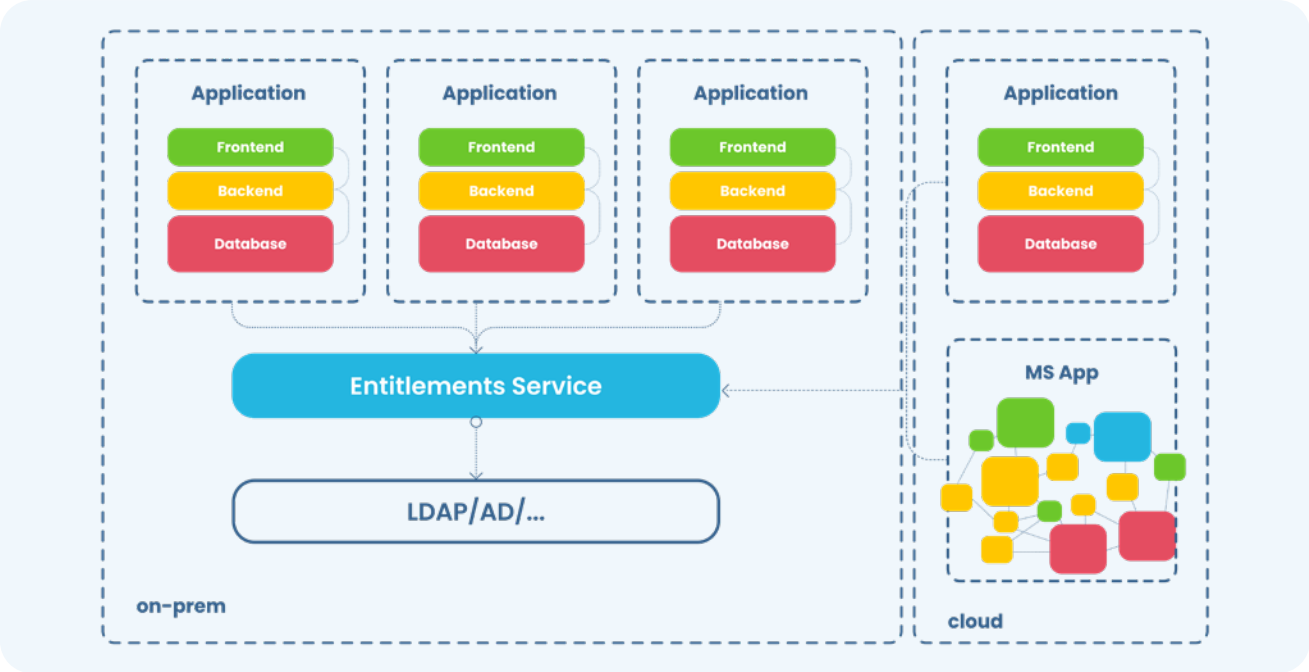

Figure 2. Naive Entitlements Management for Cloud Applications

This approach presents several challenges:

Single Point of Failure
The on-premise Entitlements service is now a single point-of-failure for all cloud-based applications. A SPOF is an anti-pattern for the cloud, where all systems are architected to be resilient to failures through replication and redundancy. Generally, the cloud aims for isolation between clouds/regions/availability-zones/etc. so that if any instance of the application fails for any reason, traffic can be diverted to another instance.

Access
The on-prem, centralized Entitlements service is typically a well-protected (sometimes called tier0) service. Enabling cloud applications built, deployed, and run on many different clouds, regions, azs, and by many different teams to connect to the on-prem Entitlements service can take tremendous amounts of coordination, technology, and approvals from security, compliance, operations, etc.

Performance and Availability
The Entitlements service is invoked whenever mission critical actions are taken within an application. If every time one of those mission critical applications must wait for a request that may connect to a service on the other side of the globe, the performance of that application can degrade to become unusable. Centralizing the management of Entitlements should not come at the expense of making the application unusably slow. Similarly, network problems can make the Entitlements service temporarily unreachable, leading to an

application that isn't just slow but is unresponsive because the Entitlements service is unavailable for serving requests.

Microservice complications
As developers begin moving away from monolithic architectures to embrace microservice architectures, the demands on the Entitlements service can increase exponentially depending on how the developers break the application into pieces. Without a solution to handling Entitlements in the context of microservices, either the organization may lose control of centralized Entitlements and they end up hardcoded into applications or progress toward cloud-native, microservice-based applications is hampered by the need for centralized Entitlements.

A Cloud-Native Approach to Entitlements

The key to solving the Entitlements management problem for on-premise and cloud applications simultaneously is to treat the Entitlements Service seen in the diagrams above as a physically distributed but logically centralized system.

Physically distributed
Instead of having 1 on-premise Entitlements service, have N replicas of the Entitlements service that you run in any cloud/region/availabiility-zone/cluster that has applications that require Entitlements. This enables every instance of an application to have an Entitlements service that runs as physically close to the application as you need it to. This solves the challenges outlined earlier: Single point of failure, Performance, Availability, and Access. If you design that Entitlements service so that its capable of running Entitlements as a microservice sidecar, you solve the challenges around microservices as well.

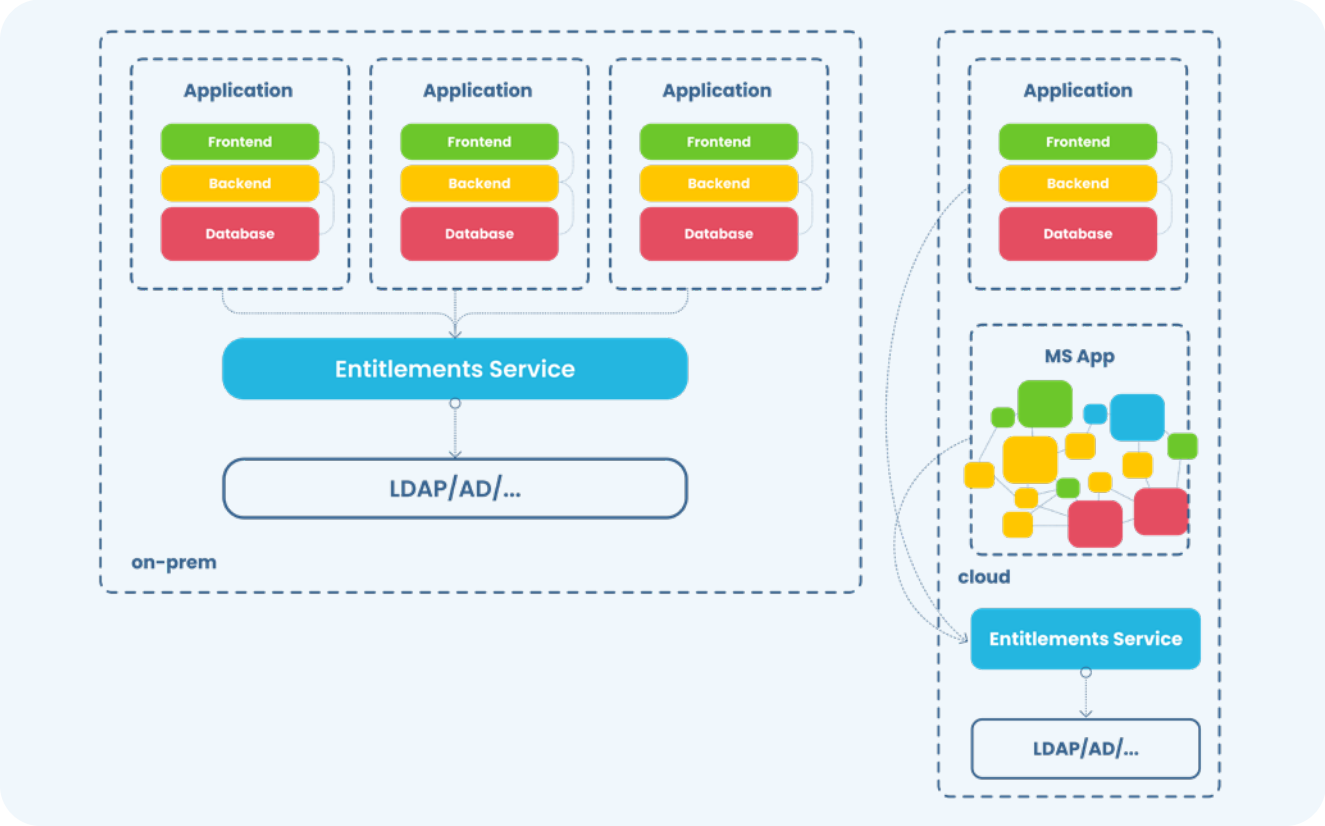

Figure 3. Physically distributed Entitlements Service

Logically centralized

N copies of the Entitlements service means you have challenges managing those Entitlements:

- How do you port your on-premise Entitlement service so it runs successfully on commodity hardware in the cloud? Or how do you build an entirely new, cloud-native Entitlements service?
- How do you keep the N Entitlements services synchronized?
- How do you log and audit the decisions that get made?
- How do you safely roll out updates?
- How do you connect your systems of record (e.g. LDAP, ActiveDirectory) to those N copies?

To complement those N physically distributed copies of your Entitlements service, you need a centralized management service that gives you a single pane of glass, a centralized logging/audit console, a roll-out feature that helps you safely update and distribute those updates, and a mechanism that ingests your systems of record, whatever they might be, and replicates them as needed.

Styra DAS for Cloud-Native Entitlements

Building your own physically distributed and logically centralized Entitlements service requires solving a good number of distributed systems, data management, and policy authoring problems. Alternatively, Styra provides an out-of-the-box solution that is built on the Open Policy Agent (OPA), ensuring production-grade management capabilities built atop proven, industry-standard policy technology.

Styra DAS for Cloud-Native Entitlements is one of several use cases in the space of authorization and policy solved by Styra DAS. As shown in the following diagram, there are two main components to this solution.

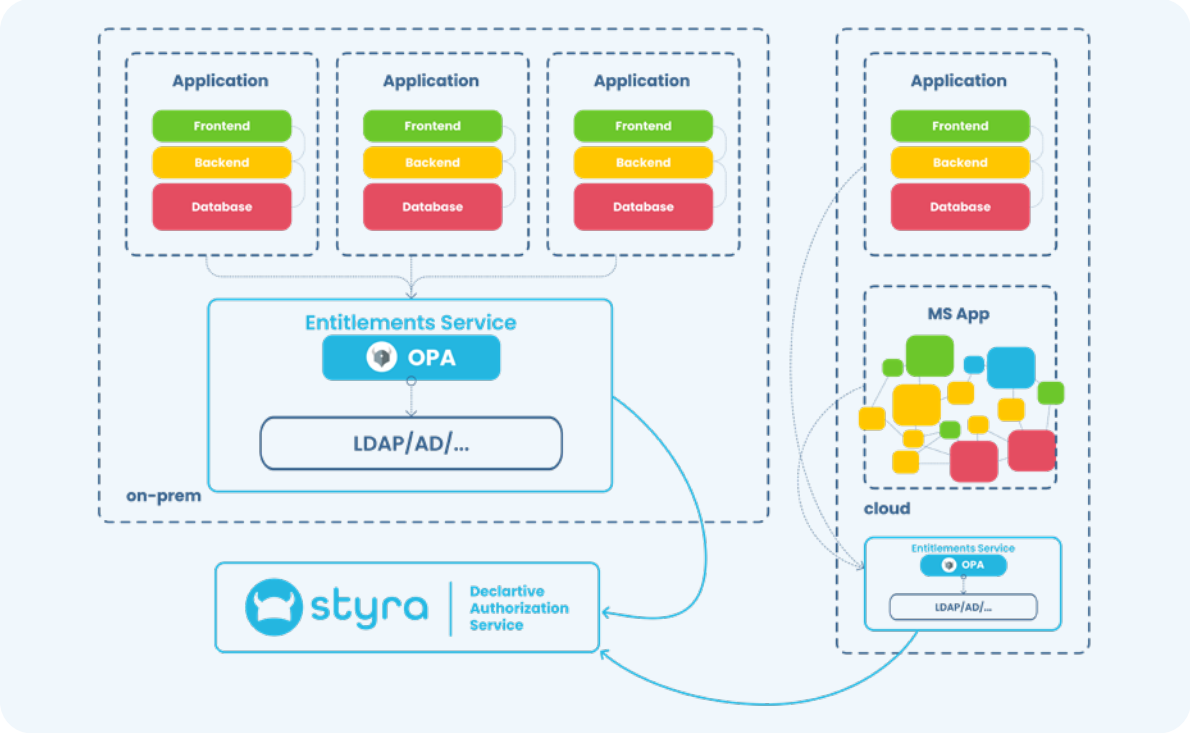

Figure 4. Styra DAS for Cloud-Native Entitlements

- OPA is used as a building block for implementing the physically distributed, N copies of the Entitlements Service shown earlier. You run that OPA-based Entitlements service in any cloud, region, availability zone, or cluster that has applications needing Entitlements.
- Those separate Entitlements Services are all managed centrally by the Styra DAS - designed to help you author policy, to ingest data from different systems of record, to distribute those policies and data as they are updated, and generally to give you visibility and control over all of the decisions, the policies, the OPAs, and the data that make up this distributed system.

OPA and Styra DAS have different responsibilities in the overall functionality of the physically distributed and logically centralized Entitlements service.

OPA

- **Local decision making.** OPA makes decisions locally for the applications talking to it using the policy and data it has replicated locally. To get a decision, an application talks to the local OPA, and OPA makes that decision using only the information it has without connecting back to the DAS.
- **Central decision logging.** OPA periodically uploads its decisions to a dedicated log for archival, search, and analysis purposes.
- **Flexible policy language.** The logic for making decisions is encoded in OPA's open-source, CNCF-owned policy language that is flexible enough to incorporate RBAC, ABAC, Access Control Lists (ACLs), etc.
- **Context-aware decisions.** OPA is loaded with the data from your systems of record that is necessary for making decisions. Whatever those systems-of-record may be.

Styra DAS for Cloud-Native Entitlements

- **Policy management lifecycle.** Styra DAS provides policy-lifecycle management capabilities: authoring policy, testing, deploying, monitoring - designed specifically to manage OPA and OPA policies throughout their entire lifecycle. Different policy-authoring interfaces enable different personas to write policies and help others in the organization write policies as well.
- **Data management.** Entitlements typically rely on data that come from one or more systems of record that are often tier0 services like LDAP or ActiveDirectory. Synchronizing this data with Styra DAS is all that is required for Styra DAS to then carefully distribute the relevant data to the OPAs that need it via encrypted channels and keep those OPAs up to date as the systems of record change.
- **Enterprise-grade governance.** Often Entitlements are the responsibility of multiple stakeholders throughout the organization. Ensuring there are proper controls over who can change policies, when they can be deployed, what tests must pass, what impact threshold must be met before release, etc.
- **Built by developers for developers.** Styra DAS is 100% API driven, born in the cloud, and available as a SaaS offering or on-premise. It integrates with existing tools like git, prometheus, Kubernetes, S3, Datadog, Slack, etc.

Styra DAS for Cloud-Native Entitlements allows organizations to extend existing systems-of-record for IAM to modern cloud-native applications, making it easy to ingest (data sources), apply authorization (policy) and then enforce authorization decisions across bespoke deployment models. Entitlements answer the questions of "Who can access what under which context?".

For more information on how to get started with Styra DAS for Cloud-Native Entitlements, set up a demo with our stellar solutions architects.

About Styra

We are reinventing policy and authorization for cloud-native. Today's cloud app infrastructure has evolved. Access, security, and compliance must also evolve. It's time for a new paradigm. It's time for authorization-as-code.

Learn more at www.styra.com