Key Challenges in Scaling Up OPA for Enterprise

Styra DAS vs. DIY OPA

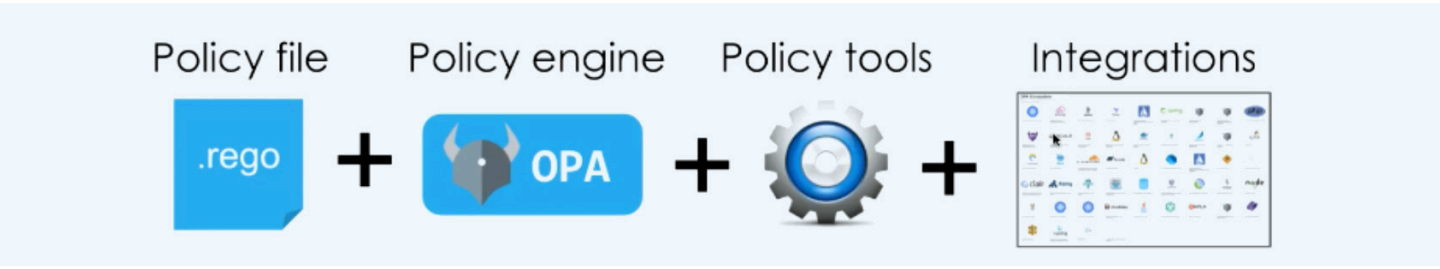
Digital transformations are driving more organizations to the cloud as they look to improve scalability, increase efficiency and reduce overhead costs. As a result, enterprises have also redefined how they build and run software. Modern cloud-native applications today are oftentimes comprised of many different microservices, housed in containers and hosted on dynamic scaling platforms like Kubernetes. However, with these migrations comes the the need to bring legacy authorization practices into the cloud or seek unified authorization across the cloud-native stack.

Before Open Policy Agent (OPA), teams generally relied on their own authorization subsystem, including writing their own data storage/retrieval subsystem. We created OPA, an open source, general purpose policy engine, to decouple policy decision-making from policy enforcement. With OPA, you can manage decisions for Kubernetes, Microservices, automated policy guardrails across public cloud configuration, and more, thanks to its single unified policy language.

But OPA was intended to help a single developer solve their policy problems, not necessarily an enterprise organization. As enterprises look to adopt cloud-native architecture, the need to have a unified platform that is supported by the founders and maintainers of OPA is essential to keep up at scale.

OPA focuses on four pieces of the policy problem:

- **Policy file:** How do you write and express policy?
- **Policy engine:** How do you provide a policy engine that when handed a policy file can make decisions using that file?
- **Policy tools:** What tools do you need to debug policy or check performance?
- **Integrations:** Before you can use OPA to solve a problem, you have to integrate it with some external piece of software.



As enterprises look to adopt cloud-native architecture, the need to have a unified platform that is supported by the founders and maintainers of OPA is essential to keep up at scale.

When you start rolling out OPA to a team or across an enterprise, you encounter challenges that lie outside of OPA's well-defined scope. Challenges that require either a unified platform like Styra Declarative Authorization Service (DAS) or a DIY approach to leverage OPA at scale. So as organizations look to scale the benefits of OPA, Styra DAS can help them operationalize OPA for the enterprise without having to spend the time and resources building from scratch.

Requirements to Deploy OPA at Scale
Based on thousands of OPA instances and production deployments across enterprises, the most critical deployment considerations can be summed up in four stages:

1. **Design:** Gather requirements (policy, data, architecture) and decide how to use OPA.
2. **Integrate:** Deploy OPAs and integrate with relevant software to decouple decisions.
3. **Control:** Connect all OPAs to a control plane for visibility.
4. **Adopt:** Facilitate using OPA throughout your team(s): onboarding, policy authoring, change management.

Design

As an organization starts to think about implementing OPA at a larger scale, they must first gather requirements and decide how they intend to use OPA. Oftentimes the easiest place to start is with enforcement. For example, what technologies does your organization currently own and what integrations are you looking to implement?

Once enforcement points have been determined, you can look at which policies need to be enforced. Documenting these policies will translate into the policy-as-code you build with OPA.

Finally, you must determine what data needs to go into those policies. OPA works by making decisions about structured data (eg. JSON) — specifically, whether data conforms to or violates a policy. You will need to feed OPA data that defines parameters, such as:

- Roles vs. attributes
- Where is that data located
- Interfaces for getting updates
- Consistency guarantees

Instead of taking the DIY approach with docs, blogs and trial-and-error, you can quickly prototype with sample apps, data load for multiple enforcement points, and leverage the pre-built policies with Styra DAS.

Deploy & Integrate OPA

Now at the software level, you're almost ready to deploy, configure, and harden your OPA configuration to ensure OPA instances can only communicate with the appropriate APIs. But if you're building out OPA to scale across your team, you'll need to first think about:

- **Integrations:** How many different OPA integrations do you need? Do they already exist in the community? Do you need to create your own?
- **Enforcement:** What class of actions does the integration provide control over? Any limitations because of eventual consistency and the fail-over model? What compensating controls do you need?
- **Performance:** What are the performance baselines? How do you make your implementation meet or exceed those expectations?
- **Security:** How do you securely configure the Service-to-OPA integration and OPA's management APIs?
- **Versioning:** As the underlying Service deprecates configuration options / APIs and exposes new functionality, who will adapt the integration as needed? How do you find out about releases and pull them into your processes?
- **Roll-out:** How do you roll-out that OPA integration to your software stack and to the teams that will be using that integration?

Working with a control plane like Styra DAS provides native support for the most popular OPA integrations and keeps them updated to support new versions of software. The list of integrations expands all the time, and includes Kubernetes Admission Control, microservices, cloud configuration and more.

By implementing Styra DAS, appropriate policy structure, quickstart, and pre-built rules (where applicable) are already included so you know which controls an integration provides. Additionally, regular performance benchmarking and updates to out-of-the-box integration configurations help to maximize performance and security.

To roll-out the OPA integration to your stack, teams can download the appropriate integration from DAS either through the GUI or the API and automate the injection of that integration into their deployments.

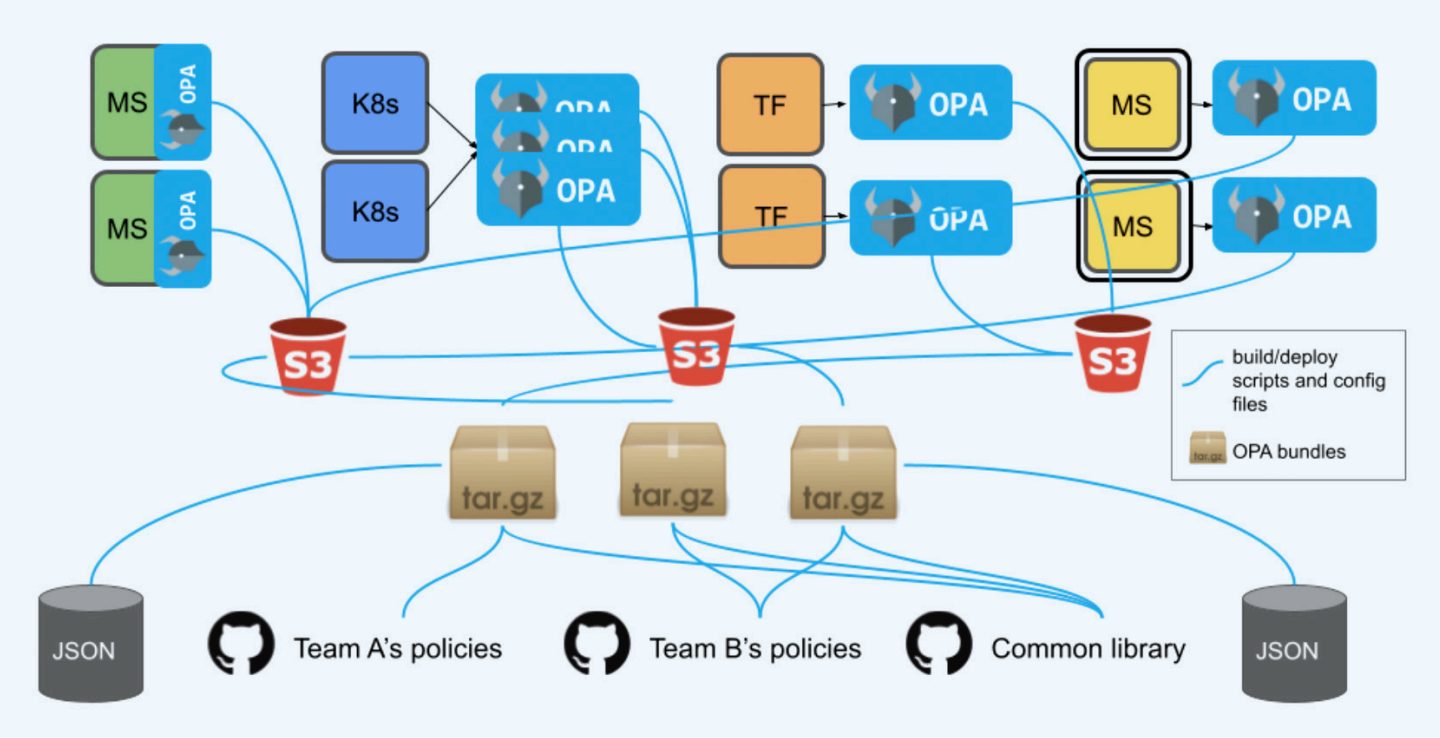
Styra DAS provides native support for the most popular OPA integrations and keeps them updated to support new versions of software.

Control

OPA provides decision APIs so external software can ask for policy decisions but OPA is not the source of truth for policy. Even if OPA is integrated with software, you still need to load OPA with all of the policies and data that it needs to make decisions.

Using Styra DAS provides both visibility and control over OPA. DAS provides a management or control plane for OPA.

If you were to build a control plane for OPA from scratch, it might look something like this:



This means that you would need to build and deploy scripts for all of the connections between:

- OPAs (which support the integrations) and S3 buckets
- Tar.gz and Git repos (where to pull policies from)
- S3 buckets and Tar.gz

This DIY control plane provides the basic plumbing that delivers policy bundles to OPA. However, it does not provide:

- Visibility into policy deployment, decisions, analytics, especially for non-developers
- Governance over who can change policy in what ways (outside of git merge rights)
- Data-fetching or data-versioning

With Styra DAS, you can simplify tasks such as:

- Operations and Audit: The ability to review the contents of the proper DAS System folder and inspect Git configurations.
- Analytics: Get easy insight into analytics with the Styra DAS dashboard.
- Troubleshooting: The "Replay decision" button allows you to identify which decisions may change when you modify policies and/or data.
- Compliance: The ability to give personnel given the appropriate roles in Styra DAS to determine who can change security controls.

Adopt

So now you're ready to start rolling out OPA to a team. However, as you look to scale OPA across your team, both now and in the future, you may run into:

- **Joiners:** How do new teams and services adopt OPA, get it deployed, integrated, etc?
- **Authors:** How do you help teams understand how to write policy, how to collaborate?
- **Policy rollout:** How do you put safeguards around updates to a tier 0 service?
- **Leavers:** How do you ensure your OPA investment survives beyond its champion's tenure?

Policy Authoring:
With Styra DAS, instead of relying on tribal knowledge, documentation, or wikis, teams can quickly install, configure and harden instructions along with quickstarts built into Styra DAS, and get started quickly with:

- GUI for authoring policy: Casual authors avoid the common syntactic mistakes of using a text editor, but nevertheless get the benefits of policy-as-code.
- Pre-built, parameterized rules: No need for authors to know how to implement rules — they just need to search for the one they need.
- Out of the box compliance packs.
- Policy overrides: No need for policy authors to reinvent the wheel.

Policy Rollout with OPA
With OPA, there is no need for all members of a team to learn Rego. In fact, Styra DAS lets teams prove the value of policy-as-code, even when auditors, security, or compliance teams aren't coders using detailed decision logs that show the input and output of OPA policy decisions.

Styra DAS also allows you to know the impact of your policy changes, long before distributing them. This includes live testing for pre-built rules, state-based testing and back-testing, ensuring that you know what impact policy will have on your current software systems.

Ready to get started? Give it a try today with Styra DAS Free!

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

styra

©2022 Styra, Inc. All rights reserved. **styra_wp_opa-enterprise_040422**