Achieving ATOs Through Declarative Architecture and Shared Common Platforms

ATARC

ATARC cATO Working Group

Acknowledgments

ATARC would like to take this opportunity to recognize the following cATO Working Group members for their contributions:

Trevor Bryant, CISA
Jack Nelson, Caveonix
Anton Hoffman, Caveonix
Chetin Durak, Caveonix
Andrew Perkinson, NASA
Shane Williams, DLA
Darren Death, EXIM, ATARC cATO Working Group Government Chair
Brian Hajost, Steelcloud, ATARC cATO Working Group Industry Chair
Douglas L. Johnson, Jr., ATARC cATO Working Group Member

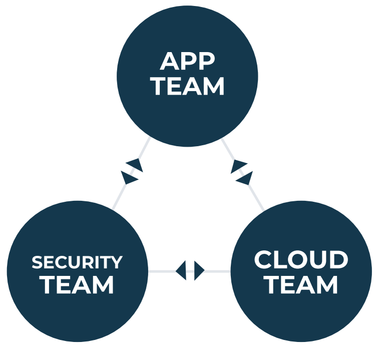
Overview

For many government agencies and cloud service providers, authorizing information systems is a time-intensive, documentation-heavy process that often functions as a bottleneck for innovation and capability delivery. Core to this problem is the traditional practice of paper-based compliance to all-encompassing security and privacy standards that were originally designed with single digit, static, physically hosted information systems per organization in mind that are run in parallel to system engineering practices. In the modern environment where organizations authorize dozens, or even hundreds, of dynamic, cloud-native information systems where the average lifespan of a container is measured in seconds, the traditional means of paper-based assessment and authorization simply cannot keep up.

The key to reducing redundancy, increasing trust, and accelerating modernization initiatives lies in leveraging transparent, consistent, and common security control baselines that can be assured to be compliant. Merging these with sound systems architecture and declarative Infrastructure as Code (IaC) supported by common platforms enables the application teams to focus on development; security teams to have a better understanding of the boundaries and functions of the assets they're working to protect; and cloud teams to dedicate their attention to their hosting responsibilities. Furthermore, it increases the capability to conduct automated, real-time monitoring and continuous assessment of the environment for the needs of all parties involved.

Gaining Trust Through Sound Architecture

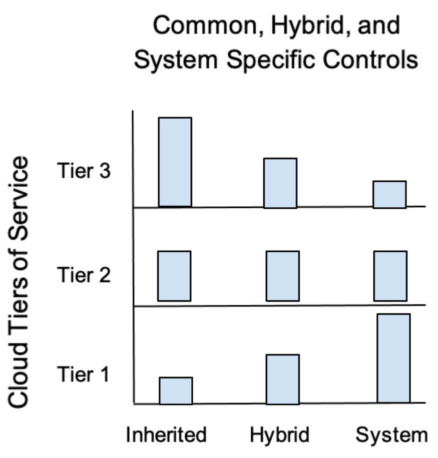
Common trust is a cornerstone to achieve infrastructure and application authorizations. Building that trust is not simply about the accuracy of documentation; it's about demonstrating consistency, transparency, and repeatability in how that infrastructure or application is designed, operated, and continuously monitored.



In typical situations this responsibility is shared between the application team, the security team, and the cloud operations team. That trust relationship is expressed in the application of each party's responsibilities and displayed through the common control packages that each party maintains for the collective. When these component packages express actual practice so that these three groups are in sync, the risk management process becomes more efficient, and that efficiency is reflected up the chain to the Authorizing Official (AO) with more rapidly produced and more accurate ATO and cATO checkpoint reports.

Encouraging Adoption of Shared Configurations for Reciprocity

Every agency faces resource constraints—whether time, staffing, or budget. One of the most impactful ways to streamline authorizations and system onboarding is to adopt standardized, pre-authorized configurations with an automation-first strategy. Shared platforms and deployment templates reduce the burden on security and operations teams by making trusted, vetted infrastructure available out-of-the-box along with clear, predefined lines of responsibility. Pre-authorized configurations add to this by ensuring repeatability and setting clear expectations for how to operate in the deployed environment.



One example of this is the cloud operations team offering multiple tiers of service models . In this case there will be a common control package for each tier of service that builds on the base shared security model to incorporate additional controls, relieving the application team of additional burden by increasing fixed configuration settings. In the diagram below, Tier 1 represents a low level of service, for example, the cloud team simply creates the account for the system boundary and the application team takes it from there. This might be more suitable for a highly customized application. Tier 3 is a very high level of service where more and more controls are implemented by the cloud service provider. It might be suitable, for example, to deploy a simple packaged application. As the conceptual diagram shows, the set of system specific controls reduces as the level of service increases.

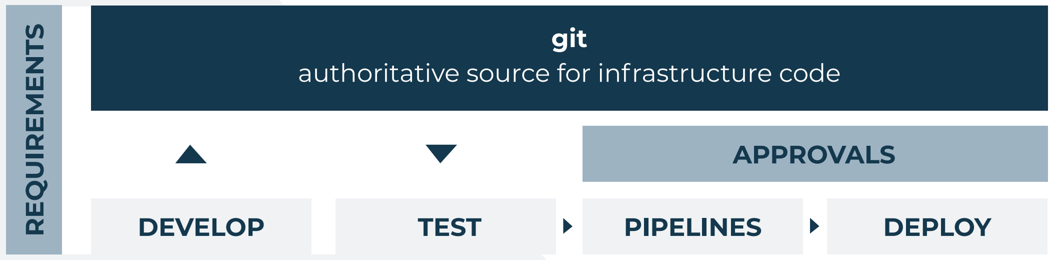
The Role of Architecture in a Modern Software Delivery Lifecycle

To support accelerated authorization, the architecture must do more than meet functional requirements—it must also address the operational, security, and compliance needs of the organization across diverse environments. Key elements of a shared common platform, to achieve sound architecture, must include:

- **Predictability:** Platforms must behave reliably whether deployed in development, test, or production, or other operational environments.
- **Scalability:** Infrastructure must be flexible to meet mission demand without introducing new security vulnerabilities, and while managing additional risks.
- **Interoperability:** Architectures should support multi, hybrid, or private environments and interconnected services.
- **Resilience:** Platforms should be fault-tolerant and compliant even under stress or failure scenarios.

A sound architecture must meet the needs of the mission and ensure variability and designs for consistency—across teams, clouds, and contracts.

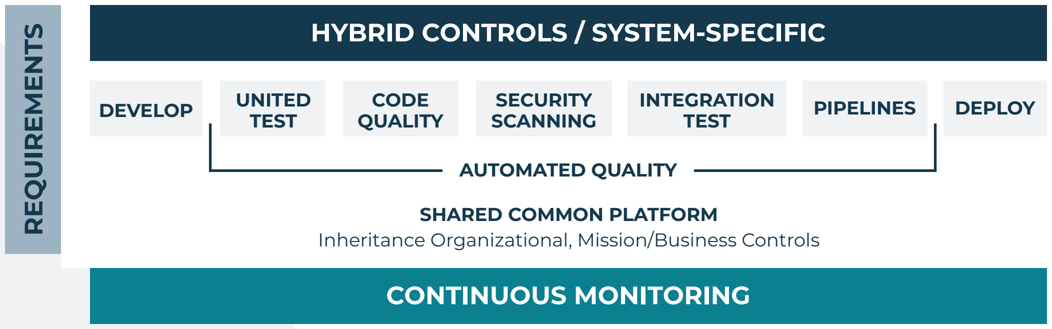
This architecture outlines the initial flow of activities from requirements gathering through production hosting in a software development context. Each phase supports the delivery of secure and reliable code used to build the initial infrastructure.



Expanding the infrastructure in this model is just an extension of the initial infrastructure code. This new infrastructure should include the requirements from each stakeholder. Here, the technical and process-oriented organizational and mission/business controls are implemented, as supported by the organization's Common Control Providers (CCP), into the shared common platform.

The key architectural concepts include:

- **Requirements feeding downstream activity:** This includes not only business functionality but also security, compliance, and performance requirements.
- **Development tightly integrated with version control and CI/CD tooling:** The IaC and pipelines live here.
- **Testing in parallel with development, not afterward:** Both functional and security-related checks are automated early.
- QA/UAT allows for formal testing with users, ensuring that required outcomes are met before going live.
- Production hosting closes the loop with operational controls, observability, and continuous monitoring.



Infrastructure as Code (IaC) in Declarative Statements

Reusable infrastructure code is essential for ensuring consistency when provisioning new infrastructure, regardless of the operating environment. By developing and using tools like Ansible playbooks and Terraform templates, teams can achieve a repeatable automation strategy that is key to success. Declarative Infrastructure as Code (IaC) allows developers to define system components as reusable templates, specifying the desired state of the platform environment. This approach is fundamentally different from the more traditional imperative operations, which focus on the step-by-step process of building the system to achieve its desired state. Embracing a declarative IaC strategy not only enhances consistency and reliability but also simplifies the management of infrastructure across diverse environments, whereas an imperative operation often leads to configuration drift.

Harkening back to the tiered service offering model from before, the use of declarative statements is critical to achieve success. The resulting benefits include:

- Accelerate ATO by reducing duplicative review of well-understood controls.
- Enable reciprocity of authorizing packages across departments, fostering trust between systems and reducing overhead.
- Promote collaboration by making configurations transparent, reusable, and community maintained.

This is especially relevant for hybrid environments, where common control baselines can be tailored and extended to local deployments, such as ships, aircraft, or satellites.

Companies like Netflix and Airbnb have successfully implemented declarative IaC to manage their vast and complex infrastructures. Netflix uses Spinnaker, an open-source multi-cloud continuous delivery platform, to deploy and manage its services across multiple cloud providers . This ensures that their infrastructure remains consistent and reliable, even as they scale globally. Similarly, Airbnb leverages Terraform to manage its infrastructure, allowing them to quickly and efficiently provision resources while maintaining a high level of consistency and control .

By adopting a declarative IaC approach, organizations can achieve similar success, ensuring that their infrastructure is always in the desired state, no matter the scale or complexity. It doesn't just have to be large, multi-cloud environments; the same results can be achieved for much smaller environments. This not only boosts efficiency but also fosters innovation and agility within the organization.

Imperative vs. Declarative Operations

Aspect	Imperative Operation	Declarative Operation
Definition	Tells the system how to achieve a desired state (step-by-step)	Describes what the desired state is; the system figures out how to reach it.
Example	Install package A, then modify file B, then start service C.	"The system should have package A installed and service C running."
Human Dependency	Manual commands or scripts often required to update or maintain state.	Automated reconciliation through declarative tools (e.g., IaC, GitOps).
Repeatability	Error-prone; results may vary depending on sequence or timing.	Consistent outcomes across environments, every time.
Auditability	Changes tracked in logs or scripts, often separately.	Version-controlled desired state stored in code repositories.
Risk of Drift	High, manual or ad-hoc changes lead to configuration drift.	Low, declarative systems continuously reconcile to the desired state.
Maintainability	Complex to troubleshoot and scale; fragile under change.	Easier to scale, adapt, and validate against policy or security baselines.

With declarative IaC:

– **Consistency is enforced across environments**—every deployment results in the same, desired configuration.
– **Repeatability is built-in**—automated pipelines reduce human error while allowing teams to maintain oversight.
– **Auditability improves**—version-controlled IaC provides a living record of system configuration, enabling more effective monitoring, controls mapping, and regulatory documentation.
– **Declarative IaC doesn’t remove humans from the loop**—it empowers them to focus on high-value, meaningful decisions rather than manual, error-prone tasks.

Conclusion:
Building Sustainable Trust and Agility

As the landscape of system authorization evolves, the integration of sound architecture principles with declarative IaC represents a potentially transformative approach to compliance management. By establishing standardized, pre-authorized configurations and leveraging transparent common control baselines, developers, security teams, and cloud providers can significantly reduce the traditional bottlenecks that have hindered innovation and service delivery. This collaborative model - where each team understands their responsibilities within clearly defined boundaries - creates a solid foundation of trust that streamlines the authorization process while maintaining robust security standards.

The future of efficient system authorization lies in embracing the shared responsibility model that is supported by automation-first strategies. When organizations implement tiered service offerings with declarative IaC, they not only accelerate the authorization process but simultaneously enable reciprocity across the Community. This approach transforms the traditionally paper-based compliance activity into an opportunity for innovation and collaboration and allows teams to focus on their core missions instead of potentially redundant compliance tasks. By continuing to develop these practices, we can create a more agile, secure, and efficient collaborative environment that supports both stringent security requirements and the growing need for rapid innovation in our increasingly dynamic digital ecosystems.

ATARC

ATARC cATO Working Group