Introducing the ZTAuth* Architecture

nicola-gallo

Published in ztauth

7 min read · 6 days ago

Chapter 3

In the previous chapter, I explained how to unlock Zero Trust Delegation using Permissions and Policies. In this chapter, I will focus on introducing the **ZTAuth* Architecture**.

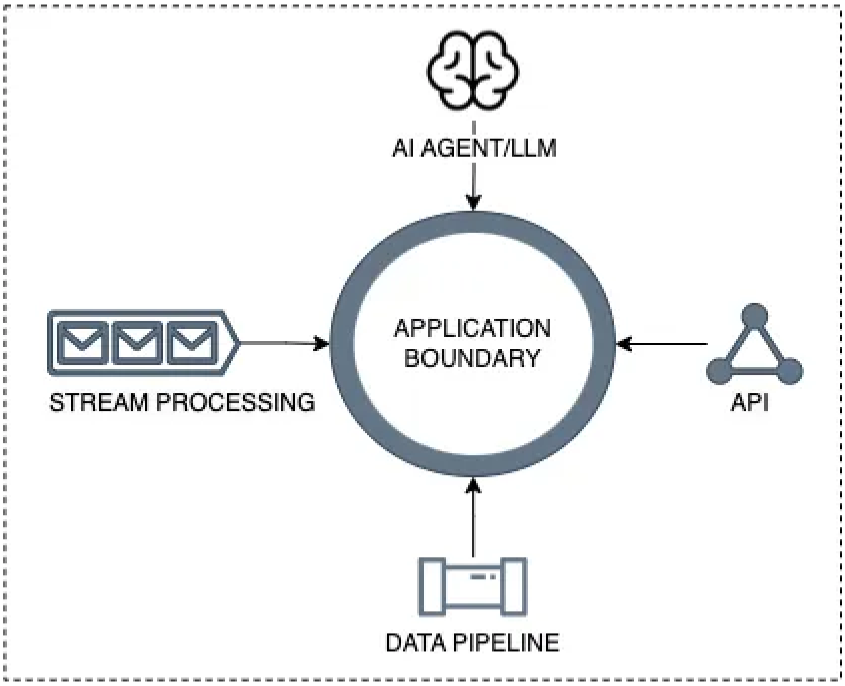
To explain what **ZTAuth*** is, let's use a comparison:

- **ZTNA (Zero Trust Network Access):** Ensures secure, identity-based access to networks or applications by applying least privilege at the network boundary.
- **ZTAuth* (Zero Trust Auth*):** Ensures secure, identity-based execution of actions on resources by enforcing least privilege at the application boundary. Built for eventual consistency, the security model is incrementally synchronized across applicative nodes in an immutable, versioned manner.

Asynchronous by Design: Built to Mirror Reality, Not Mask It — Robust Where Synchronous Fails.

> *If you're unfamiliar with ZTNA (Zero Trust Network Access), it is a security model based on the Zero Trust framework, which ensures that every access request is verified using identity, context, and strict security policies. Unlike traditional models, ZTNA treats all requests as untrusted, regardless of whether they come from inside or outside the network.*
>
> *ZTNA is widely adopted for scenarios such as enabling secure remote work, protecting SaaS applications, and replacing VPNs with more flexible, granular, and context-aware access controls.*

ZTAuth* is designed to operate with ***eventual consistency***, ensuring support for ***partially connecte***d environments or scenarios where network reliability is limited or intermittent. Changes are ***packaged*** and ***chained*** into ***versioned, immutable*** data structures that are asynchronously distributed incrementally.



APPLICATION BOUNDARY

Every **resource action** executed at the ***application boundary*** is verified against ***identity-based*** and strict ***security policies*** governed by an ***authorization schem***a. Here, the application boundary refers not only to APIs exposed to external consumers but also to interactions between machines or services within the application ecosystem. These interactions can occur using a variety of ***protocols*** and ***technologies,*** including ***synchronous*** requests, ***asynchronous*** messaging, and ***event-driven*** architectures. By encompassing all these layers, ***ZTAuth**** ensures that security policies are consistently enforced across diverse communication methods.

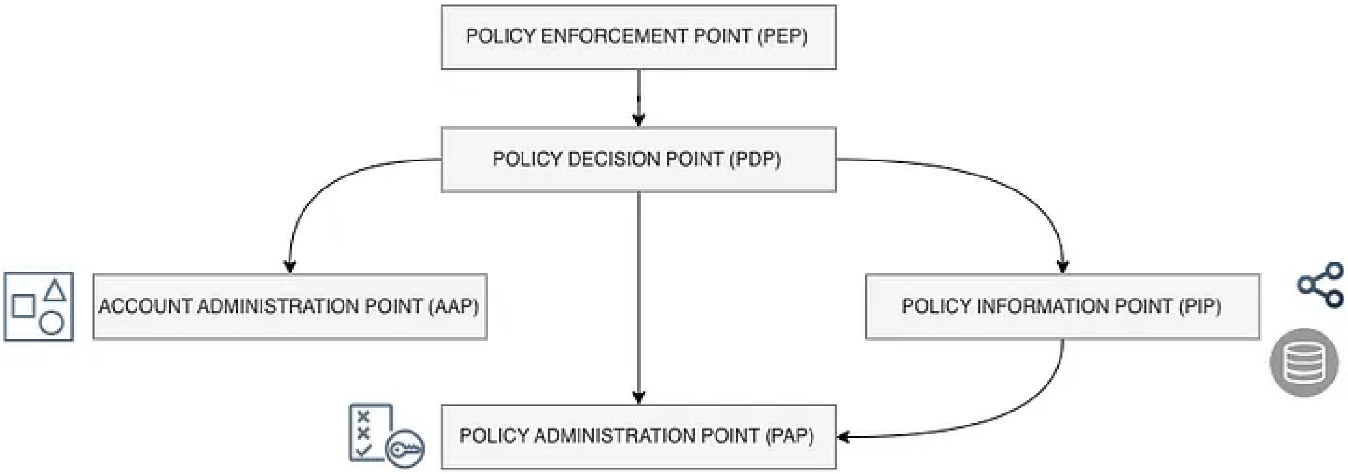
At its core, *ZTAuth** relies on *ZTAccount* (**Zero Trust Account**), which applies **Zero Trust** security principles to manage access within an application. Here's a detailed explanation:

- **Identity-Based Access Control:** ZTAccount ensures that access to resources, data, and features inside the application is based on verifying the identity of the user or system making the request. No action is trusted by default.
- **Granular Policies and Permissions:** Access is managed with detailed rules (policies) and specific rights (permissions) assigned to identites. These ensure that users or systems can only access what is strictly necessary for their role or task.
- **Secure Delegation and Elevation:** ZTAccount allows delegating permissions or temporarily elevating privileges in a controlled, auditable way, ensuring that access remains secure and traceable.
- **Enforcing Zero Trust Within Boundaries:** Unlike ZTNA, which secures access at the network level, ZTAccount operates within the application boundaries, applying security at the internal layer of the application itself.
- **Auth* Models:** ZTAccount allows configuring the Authentication Model, Authorization Model, and Trusted Delegation Models. *Administration Points* are used to manage and configure these models. These configurations are distributed as *snapshots* that are *transferable, verifiable, versionable,* and *immutable*. The snapshots are redistributed to nodes running the applications and synchronized as connectivity becomes available, following an *eventual consistency* approach. This ensures that the models are always up-to-date and can be used effectively to evaluate authentication and *authorization* in the *Policy Decision Point (PDP)*.

Before diving into the internal details of the **ZTAccount**, we need to define the high-level architecture components that should be included. These components are:

- **Policy Enforcement Point (PEP):** This is the component that enforces policies by allowing or denying access requests based on decisions from the Policy Decision Point.

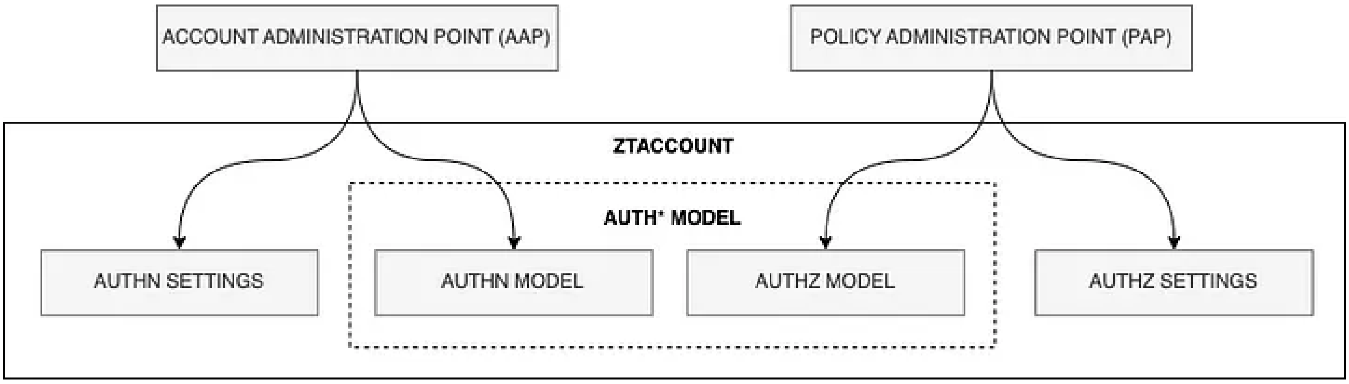
- **Policy Decision Point (PDP):** This is the component that evaluates policies and makes decisions about whether access should be granted or denied.
- **Account Administration Point (AAP):** This is the component used to manage accounts, including creating, updating, and configuring them. Configuration tasks may include setting up Identity Sources, Identity attributes, and other related configurations.
- **Policy Administration Point (PAP):** This is the component used to manage permissions, policies, trusted delegation, and trusted elevation. It handles the creation, updating, and deletion of these governance elements.
- **Policy Information Point (PIP):** This is the component that provides additional information required to evaluate policies, such as user attributes, resource details, or environmental conditions.


POLICY ENFORCEMENT POINT (PEP)
POLICY DECISION POINT (PDP)
ACCOUNT ADMINISTRATION POINT (AAP)
POLICY INFORMATION POINT (PIP)
POLICY ADMINISTRATION POINT (PAP)

PEP, PDP, AAP, PAP, PIP

We have covered the theory; now it's time to define the **ZTAuth* Architecture**.

Let's begin designing the architecture by focusing on the *administration layer,* which is used to create, update, delete, and configure resources.

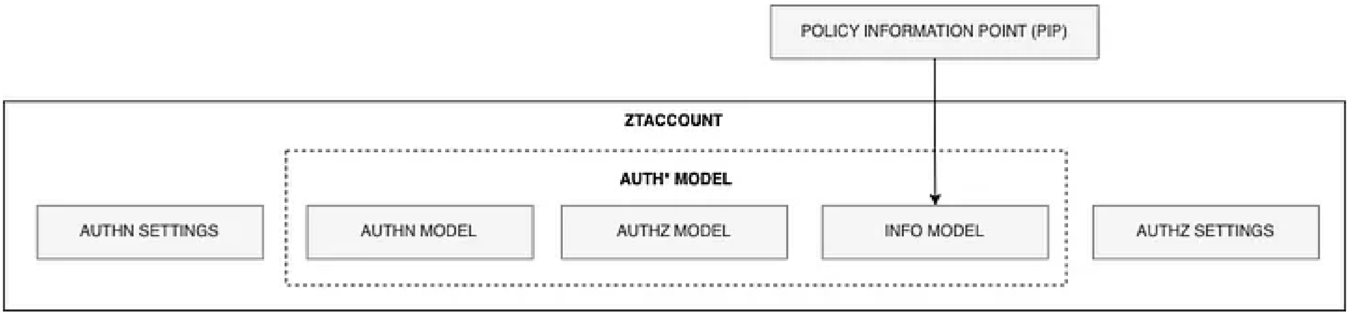


ZERO TRUST ACCOUNT

Administration is managed through two administration points:

- **Account Administration Point (AAP):** Responsible for managing the account lifecycle, including creating, updating, and deleting ***ztaccounts***. It also configures ***authentication settings***, such as identity sources, and defines ***authentication models***, including users and roles.
- **Policy Administration Point (PAP):** Focuses on configuring ***authorization settings***, such as data sources for the Policy Information Point (PIP), and managing ***authorization models***, including permissions, policies, trusted delegations, and trusted elevations.

Now it is time to focus on the **Policy Information Point** (**PIP**), which manages the information used by the **Policy Decision Point** (**PDP**) during evaluations.



ZERO TRUST ACCOUNT

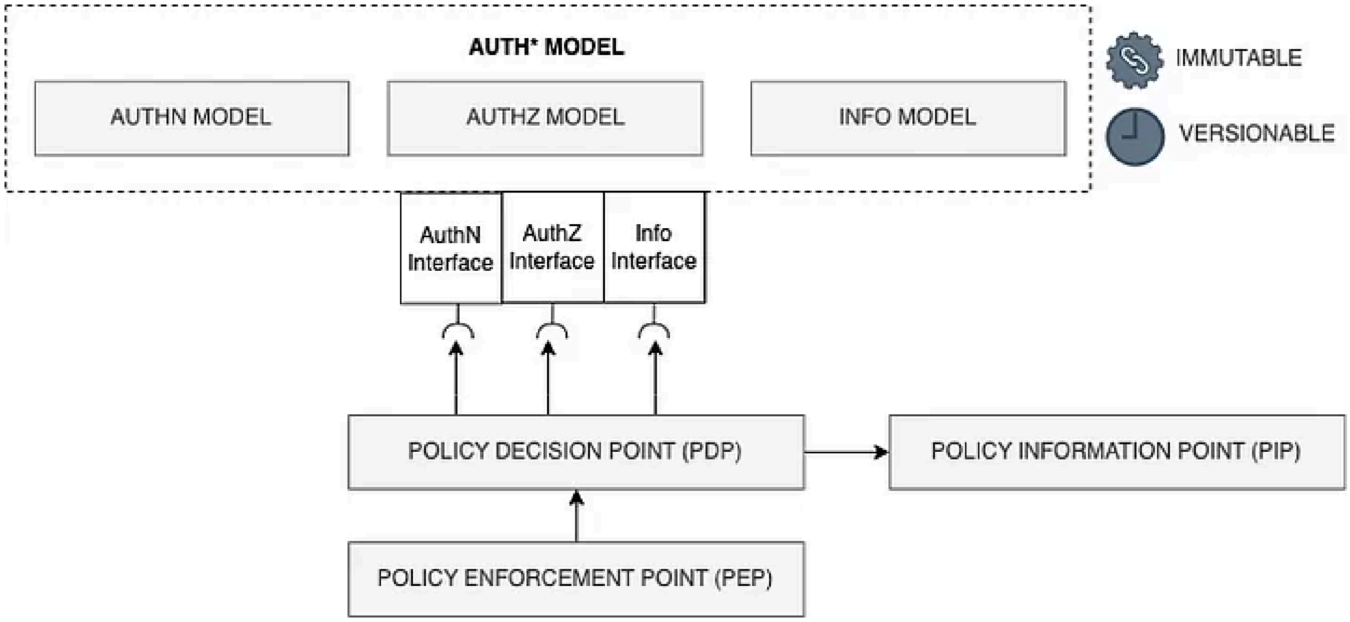
The information is categorised into two types:

- **Offline Information:** Data that can be managed and stored offline, integrated into an ***Auth* model***, and distributed across applicative nodes.
- **Online Information:** Data that requires real-time communication with the PIP. This type of information cannot be stored or distributed, may introduce latency, and is unavailable in disconnected scenarios.

Let's now focus on the Auth* models. These models must be synchronized to application nodes as incremental snapshots based on time-based checkpoints. To ensure seamless functionality, they require the following characteristics:

- **Transferable and Verifiable:** Operates smoothly across systems and environments, with verifiable origins certified by the central PDP.

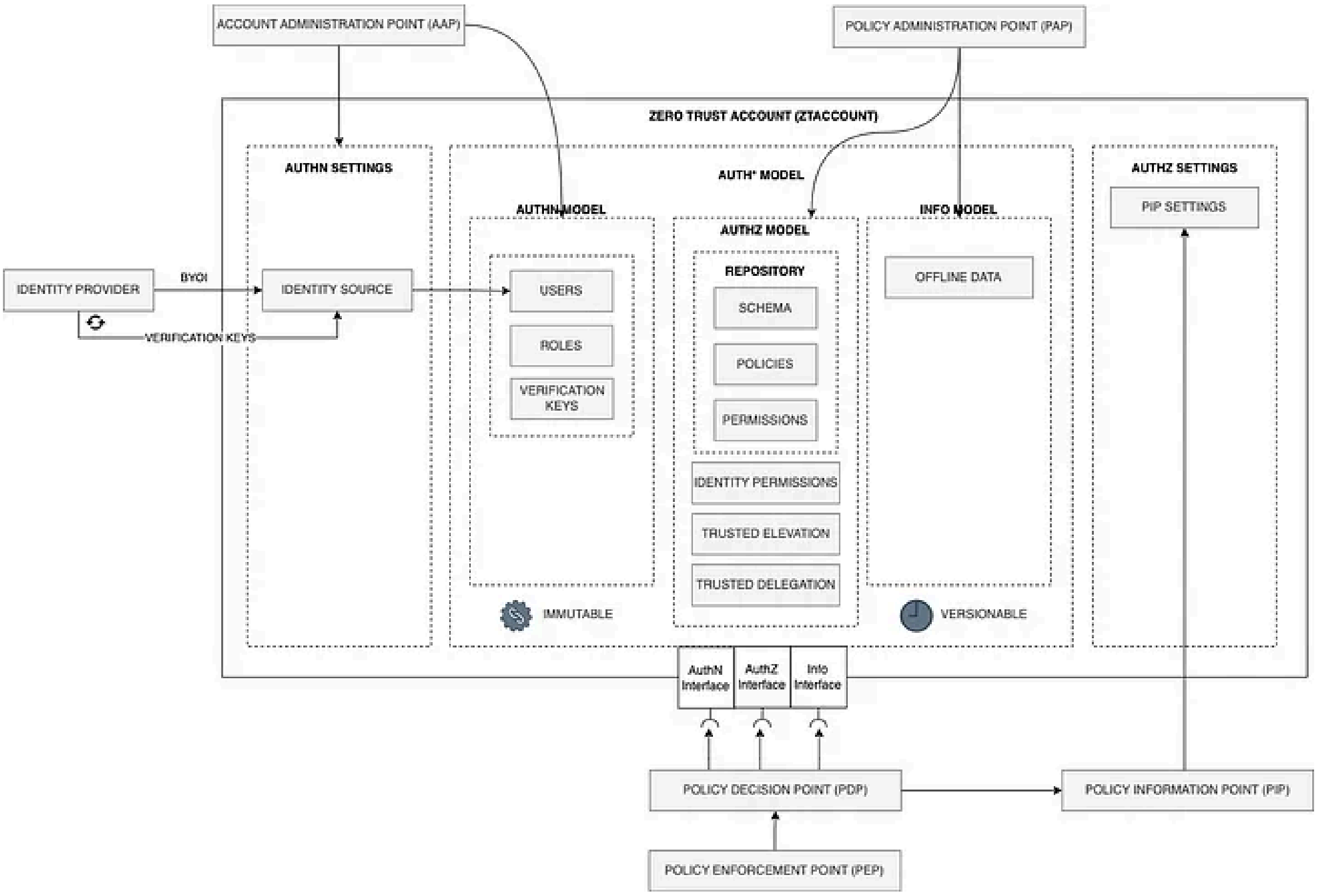
- **Versionable and Immutable:** Ensures integrity, auditability, and backward compatibility for secure and reliable operations.
- **Resilient to Disconnection:** Supports eventual consistency, allowing functionality in partially connected or disconnected environments.



AUTH* MODEL and PDP

After synchronization, the model must be accessible to the PDP through an **interface**. This connection can be implemented in various ways, such as file-based reads, APIs, or other methods. The term **interface** here represents an abstraction, providing flexibility to accommodate diverse scenarios without relying on a specific implementation.

Now we have all the components and can add the details from the previous chapters. Below is the final architecture.



ZTAUTH* ARCHITECTURE

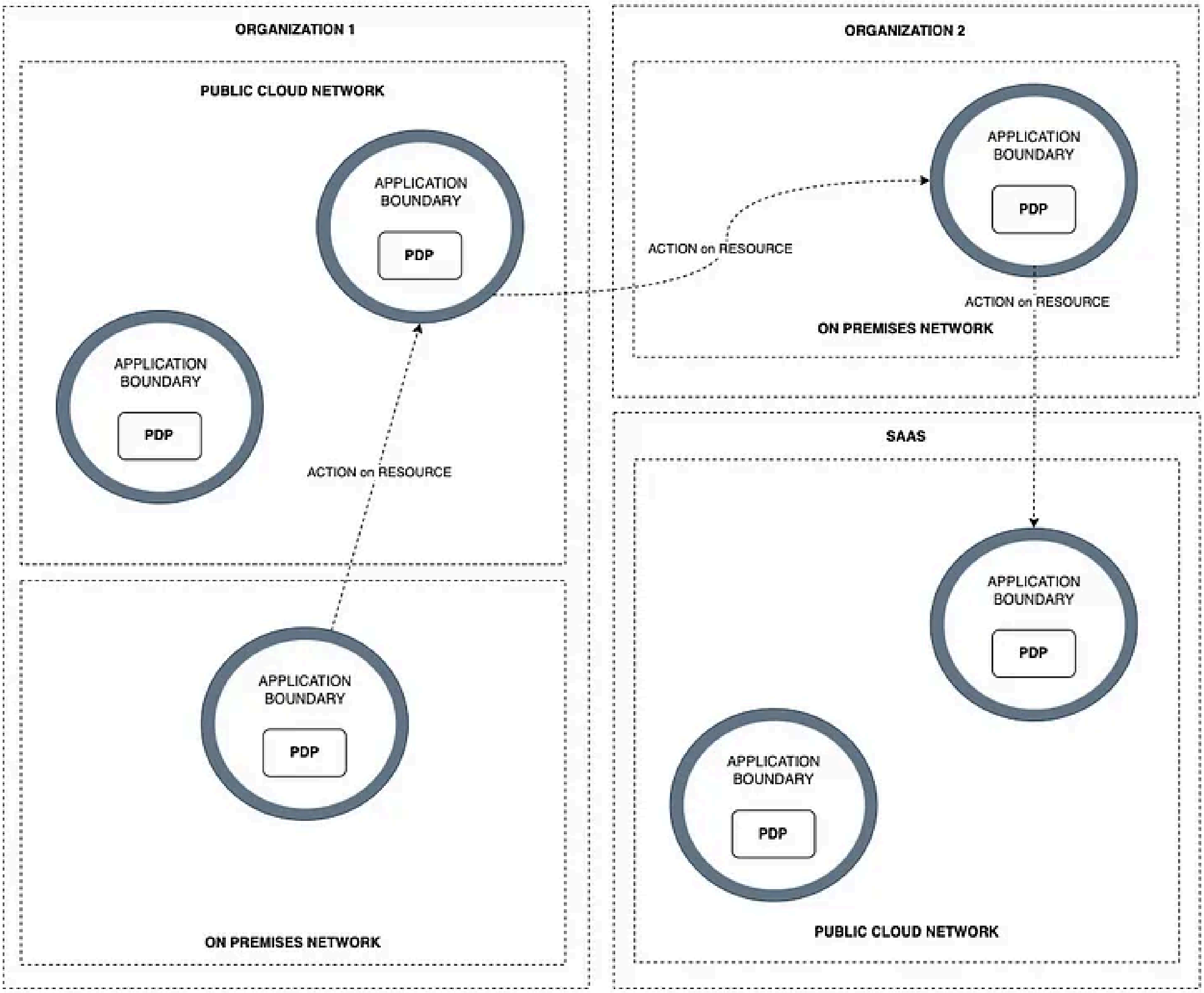
> *Note: I have moved Trusted Delegation and Trusted Elevation into the Authz Model, as at this stage of understanding the architecture, there is no clear reason to keep them outside the Authz Model.*

The ***ZAuth* Architecture*** is now complete. To summarize, a software solution can have ***multiple PDP deployments,*** each linked to a ***ztaccount***. However, it is also possible for a PDP to be associated with more than one ***ztaccount*** if needed.

Each ***application boundary***, even if it is a single microservice, has a ***PDP deployed.*** Communication between application boundaries happens when one boundary requests an action on a resource from another application boundary. Once the request is received, it is securely executed using identity-based policies and enforcing least privilege at the application boundary.

> *Using this approach, it is easy to see how the federation of organizations across different networks could change significantly.*

By reviewing the diagram, the comparison with ZTNA should now be clearer.



APPLICATION BOUNDARY AND PDP

This post is provided as Creative commons **Attribution-ShareAlike 4.0 International** license and attributed to **Nitro Agility Srl** as the sole author and responsible entity, except for referenced content, patterns, or other widely recognized knowledge attributed to their respective authors or sources. All content, concepts, and implementation were conceived, designed, and executed by Nitro Agility Srl. Any disputes or claims regarding this post should be addressed exclusively to Nitro

Agility Srl. Nitro Agility Srl assumes no responsibility for any errors or omissions in referenced content, which remain the responsibility of their respective authors or sources.

Zero Trust Cybersecurity Architecture Policy As Code Integration

ZTAuth*

Following

Published in ztauth

4 Followers · Last published 6 days ago

ZTAuth*: Redefining AuthN, AuthZ, and Trusted Delegation with Transferable, Immutable, and Resilient Models for a Zero Trust World

Edit profile

Written by nicola-gallo

9 Followers · 4 Following

A tech entrepreneur specializing in cloud and enterprise strategic technology, with a focus on distributed systems.

No responses yet

What are your thoughts?

Respond

More from nicola-gallo and ztauth

In ztauth by nicola-gallo

ZTAuth*: A Paradigm Shift in AuthN, AuthZ, and Trusted Delegations

Introduction

Nov 24 37 1

In ztauth by nicola-gallo

Resources, Actions and Accounts in the context of Autonomous and Disconnected Challenges

Chapter 1

Nov 28 10

In ztauth by nicola-gallo

Unlocking Zero Trust Delegation through Permissions and Policies

Chapter 2

Nov 29 8

See all from nicola-gallo

See all from ztauth

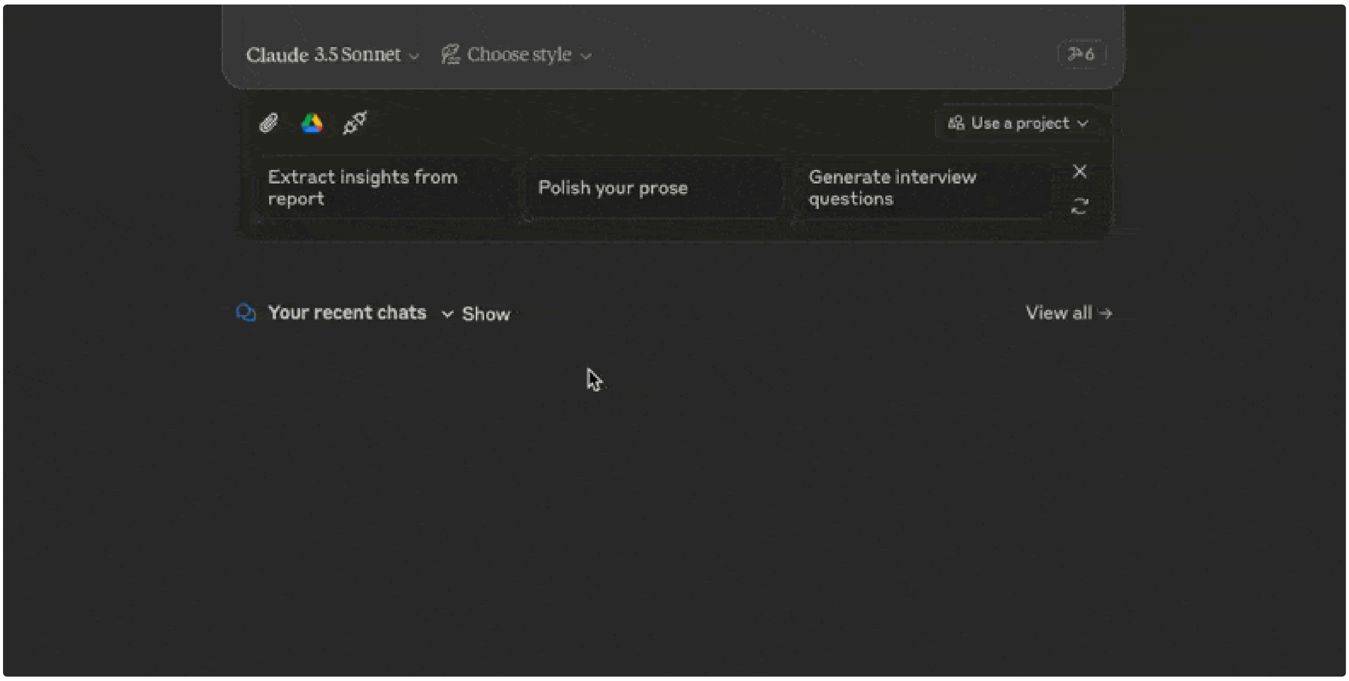
Recommended from Medium

In Level Up Coding by Kawin Promsopa

5 Kubernetes CLI Tools You Must Try in 2024

Essential Tools for Debugging, Monitoring, and Managing Kubernetes

✦ Nov 25 222 3



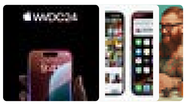
In SecurityBreak by Thomas Roccia

Building a Threat Intelligence GenAI Reporter with ORKL and Claude

Using Model Context Protocol (MCP)

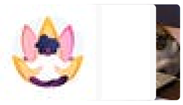
Nov 27 32 2

Lists



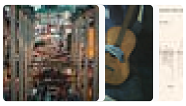
Tech & Tools

21 stories · 363 saves



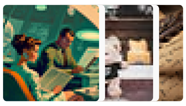
Stories to Help You Grow as a Software Developer

19 stories · 1509 saves



My Kind Of Medium (All-Time Faves)

102 stories · 600 saves



Medium's Huge List of Publications Accepting Submissions

377 stories · 4070 saves

Dr. Jess Steier

I'm a Public Health Scientist. Here's What Research Really Shows About Raw Milk.

Do a quick search for 'raw milk and lactose intolerance' on social media, and you'll find thousands of posts claiming unpasteurized milk is...

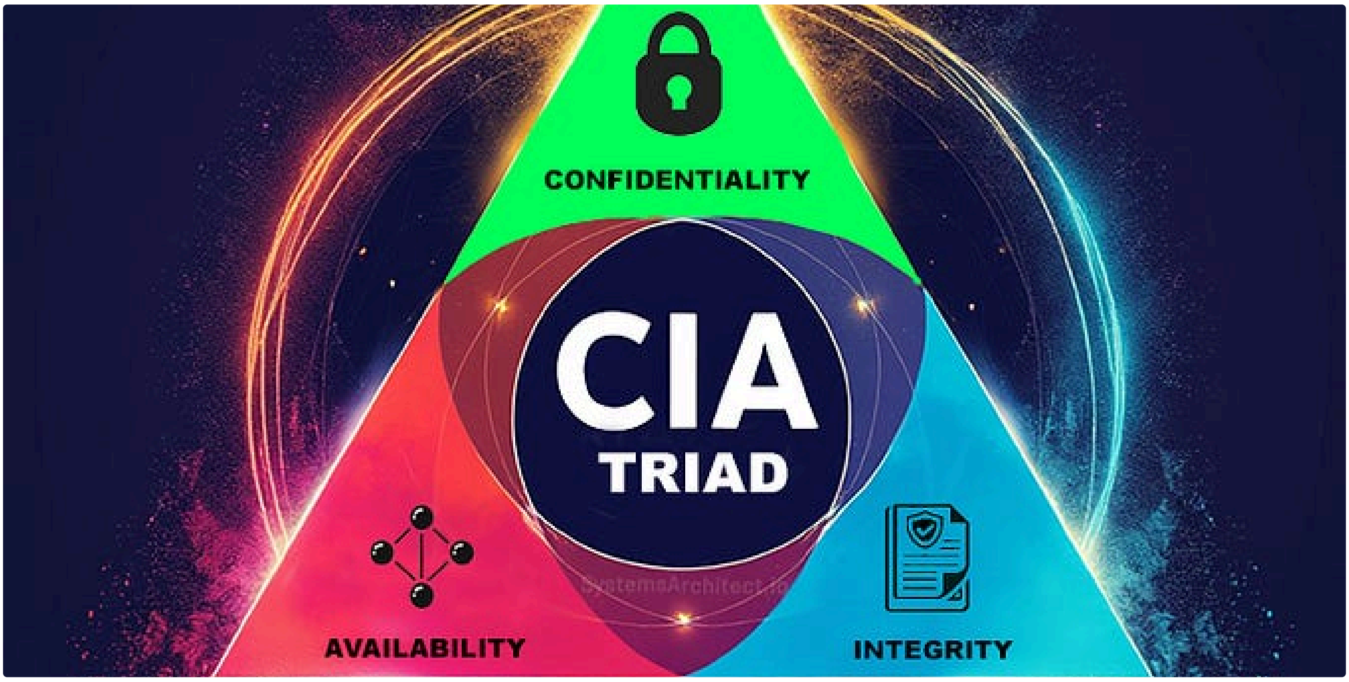
3d ago 6.1K 147

David Rodenas PhD

Improve Your Testing #15: Make Tests Human

Break free from technical testing patterns and learn the human-first approach that will make your tests crystal clear for years to come

6d ago 584 13

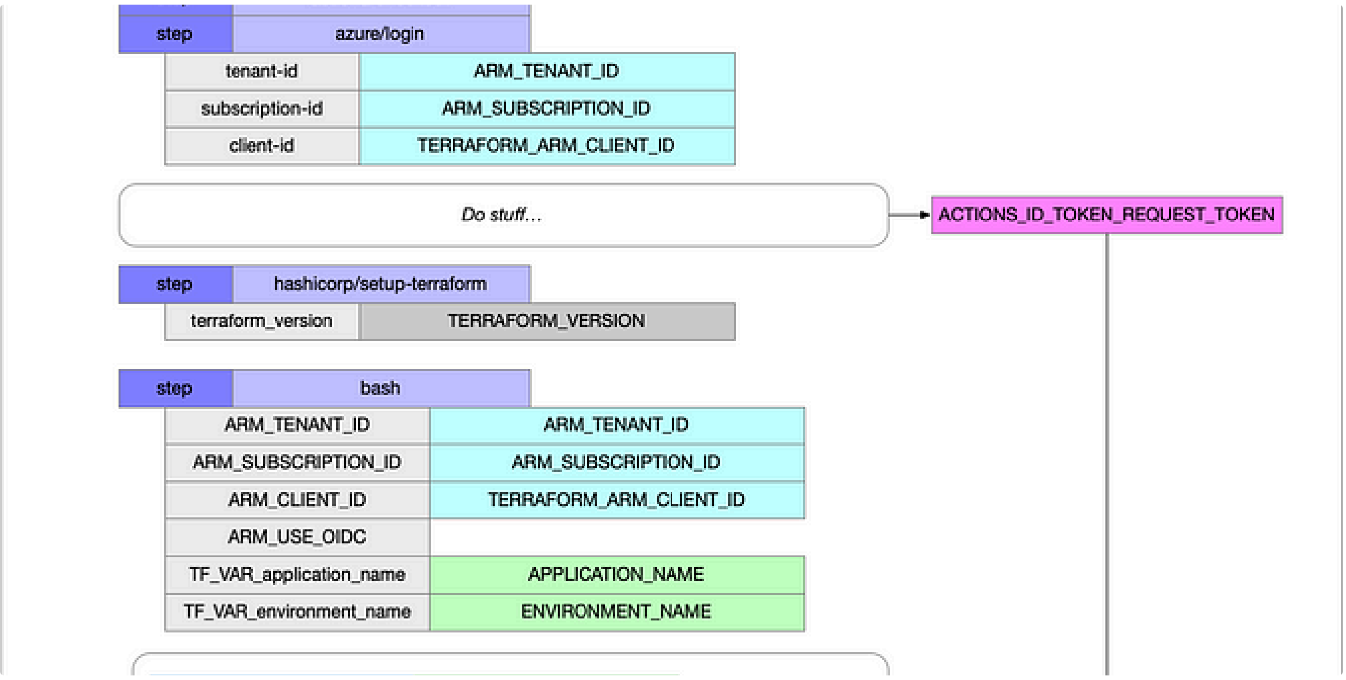


Chris St. John

CIA Triad in Cloud Security (Part 1: Confidentiality)

Confidentiality controls for a more secure cloud platform using the CIA Triad model (checklist)

Nov 22 128



step	azure/login
tenant-id	ARM_TENANT_ID
subscription-id	ARM_SUBSCRIPTION_ID
client-id	TERRAFORM_ARM_CLIENT_ID

Do stuff... → ACTIONS_ID_TOKEN_REQUEST_TOKEN

step	hashicorp/setup-terraform
terraform_version	TERRAFORM_VERSION

step	bash
ARM_TENANT_ID	ARM_TENANT_ID
ARM_SUBSCRIPTION_ID	ARM_SUBSCRIPTION_ID
ARM_CLIENT_ID	TERRAFORM_ARM_CLIENT_ID
ARM_USE_OIDC	
TF_VAR_application_name	APPLICATION_NAME
TF_VAR_environment_name	ENVIRONMENT_NAME

In Azure Terraformer by Mark Tinderholt

Making GitHub Actions Workflows More Modular and Efficient: A Deep Dive into My Setup

✦ 3d ago 130 2

See more recommendations