Agilicus Secure Municipal Cloud Platform

Overview

The Agilicus Secure Municipal Cloud Platform provides a Platform-as-a-Service model, taking simple applications and making them widely available on any device, any location, without the need for VPN. It provides seamless and simple authentication and identity, and world-class security.

It reduces the effort and work in launching a new application, handling the hosting and capacity as well as the heavy-lifting of making something run securely, simply, and reliably at scale.

Key Features

- SSL/TLS certificate request/revocation/rotation/management
- User identity
- Safely expose Active Directory (or Active Directory Federation) without exposing it to Internet
- Seamlessly federate user identity with Social login
- Audit all connections and actions
- Seamless, no impact, in service upgrade
- Strong encryption on all components
- Full data sovereignty
- Disaster recovery, High-Availability across multiple data centres
- High end user performance
- Self-served, self-enabled for upgrades, user-management, new application onboarding

Key Benefits

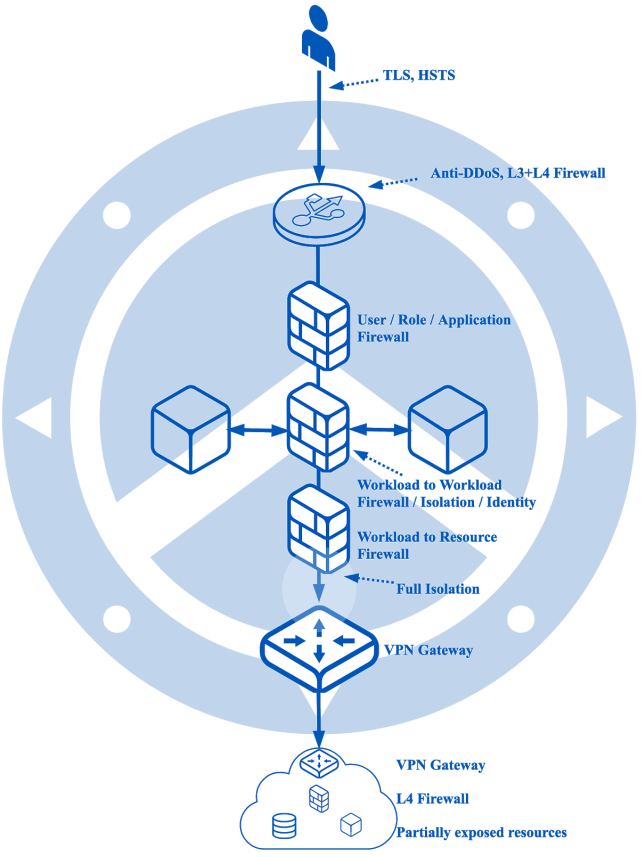
- Digitally enable an external, contractor-based work-force with any device
- Comply with data sovereignty requirements

- Enable 2-factor authentication to all end-users to reduce phishing
- Dramatically reduce cost of launching and managing applications
- No ongoing firewall maintenance or reverse proxy management
- Managed Security Operations Centre for applications
- Simple staging environment: try new versions with live users, toggle upgrades with no effort

Key Principles

Defense In Depth

The Agilicus Secure Municipal Cloud is based around the key principle of *Defense in Depth*. In this model we make the assumption that any one layer may be (or already is) compromised. Rather than build a single infinitely strong bastion firewall, we implement a set of layers each seeking to delay, restrict, hamper an attacker.



Each layer uses strong encryption for privacy and identity management. We use standards including TLS 1.3, SPIFFE, JWT and associated encryption and secure hash standards. This is done to ensure that there is no lateral (east-west) traversal from one component to another, reducing the risk of privilege-escalation.

Example risks that might otherwise be overlooked include supply-chain risks in the individual application back-ends (e.g. malware that ships with them, or activates after the fact, perhaps coming from upstream open-source).

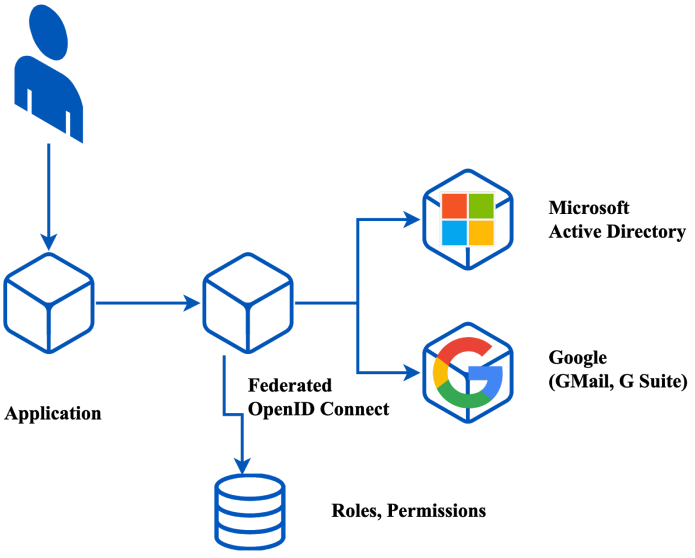
The heart of the security infrastructure is shown below. Each data-flow hand-off uses:

- Mutual TLS (mTLS) using X.509 PKI. This guarantees that each end is who it says it is (and that no-one can intercept it)

- Application Rules & Roles & Identity-based firewall. E.g. “User jane can do a POST to the /records endpoint for her own user-id”.
- Header and URL normalisation (reducing XSS and CSRF and other web-based attacks)
- Workload-To-Workload identity-based firewall (ensuring that only the appropriate application reaches the subset of exposed resources it is entitled to communicate with)
- Strong IPSEC VPN using mutual certificate-based authentication (not IP-based).

Strong Identity, Multi-Factor, Single-Sign-On, Without Passwords

The key to strong security is to make it simple enough that the end-users can use it. We have built a system that federates multiple identity providers (Active Directory, Social). This removes the need for new identities or new passwords. In fact, the Agilicus system does not store nor have access to the end-user passwords. This is part of the defense in depth: if our system is breached, the user’s password is not available; we never see it.



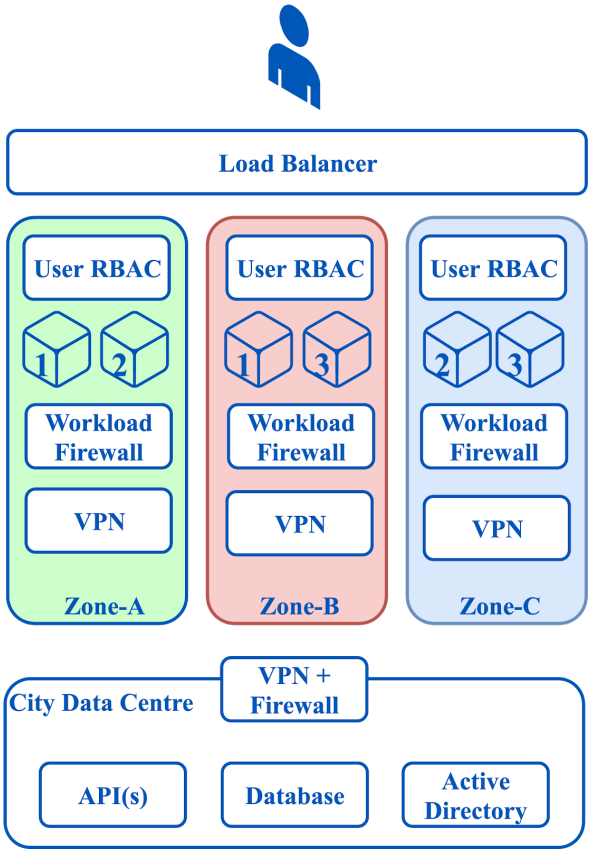
The system uses a standard called OpenID Connect. This allows the user to simply select which Identity provider to use. Their authentication is then done entirely with that provider, returning an ID Token. If their Identity provider uses 2-Factor Authentication (e.g. as Google does), this step is performed automatically. There is no separate password, password reset, etc., required.

The Identity is enriched with an Authorisation layer, providing role-based and group-based access control. E.g. I can say “alice@gmail is an admin of Inventory, a viewer of Training. There is no difference in capability regardless of the upstream identity (e.g. Active Directory and Gmail provide the same capability and security).

For most applications there is no code change: this can be done either with built-in configuration, with a transparent proxy in front, or with the injection of a single line of JavaScript. All of the security is handled in the Platform framework.

Cloud-Native Containerisation and Orchestration

The platform provides a cloud-native orchestration, based around container technologies. At its heart we use Kubernetes for orchestration, and Istio as a service mesh. Our platform runs on top of Google Cloud in a multi-zonal regional deployment.



We have 1+1 resiliency for all customer applications, meaning each application is always running at least two copies, in two separate data centres for maximal redundancy and disaster recovery.

The Agilicus Secure Hosting Platform is built around modern cloud-native principles:

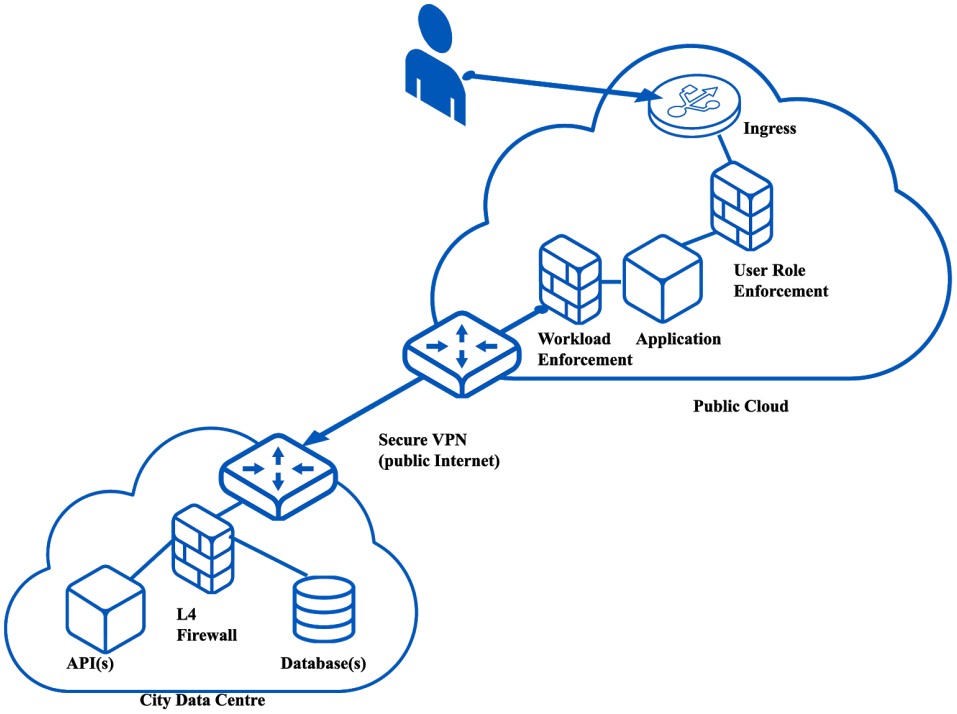
- Micro service API's
- Stateless, horizontal-scaled components
- Self-healing
- Live, incremental, non service-affecting upgrades

Hybrid Cloud Data Flow

A hybrid cloud is a computing environment that combines a public cloud and a private data centre by allowing data and applications to be shared between them.

Traditionally this environment runs the same work-load cross both the public and private data centres, acting as an expansion pool. In the Agilicus model we keep critical data and physical interconnects in the on-premise and move the user logic and interface into the public cloud. This helps guarantee data privacy and ownership.

The high level architecture is shown below. There are two main locations, labelled ‘City Data Centre’ and ‘Public Cloud’.

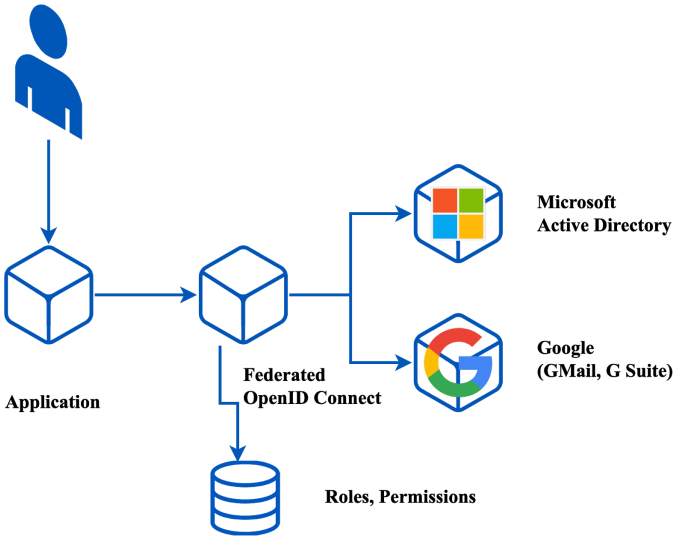


The system provides each application with aunique DNS domain name (e.g. `app.cloud.city.ca`). The user opens this as a web page, which presents to the user a login workflow (detailed in **Identity** section), resulting in an access token. This access token is presented on all subsequent web requests.

An identity-aware, role-aware, application-aware firewall is presented between the Internet and the application backend. This implements a set of rules per user , giving Role-Based Access Control (RBAC).

Identity

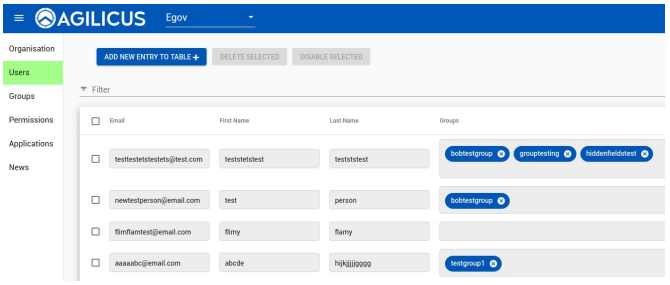
A key aspect of the Agilicus Secure Municipal Cloud Platform is identity. Identity is used in two ways: to identify a user (and thus apply their authorisation), and, to identify workload components and how they are allowed to talk to each other.



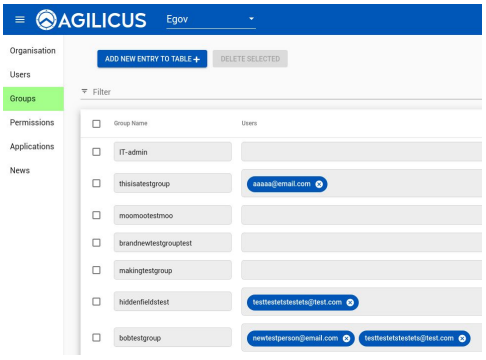
User Identity

User identity is obtained from a federated identity provider (IdP) with a number of upstream providers which can be trusted (nominally Active Directory and Social logins). The federated identity provider provides an OpenID Connect workflow towards the application, and returns an ID Token. It also includes an authorisation service which can be used to exchange the ID Token for an Access Token (which encodes what services the user can access, and in what modes).

The administrator of each organisation is given 3 primary web pages to manage their user base. The first allows selecting identities (e.g. enabling the upstream identity to be used in the system), and optionally assigning the user to one or more groups.



The second allows assigning users into groups (which themselves are identities). The concept of a group allows simplifying permissions (e.g. create an application-admins group, assign permissions to it, and then as new users are on boarded simply assign them to the group if needed).



The third (permissions) allows assigning permissions to groups/users. E.g. here we can select that ‘application-A’ allows group ‘B’ to be ‘owner’, ‘group C’ to be ‘editor’ and individual user ‘alice’ to be ‘viewer’.

This enriched set of user roles/permissions/groups is attached to the identity returned by the upstream IdP. No local password (authentication) is used.

Workload Identity

In the same way we use user-identity to drive the application/role firewall between the user and the application backend, we use workload identity to identify traffic originating from the application backend towards either east-west services, or, towards backend exposed API or databases. This is implemented with an enriched web token (JWT) on a WebSocket connection, upgraded from the original TCP. It implements an identity standard based on X.509 and JWT called SPIFFE. This allows TCP connections to be confirmed from source to destination without having to rely on layer 3 or layer 4 coordinates, simplifying firewall setup. It also renders spoofing impossible.

Audit

All connections and actions are audited. On the user side, the audit trail includes the user identity and request type. On the workload side the audit trail includes the specific workload and destination.