

Evolution of traditional Edge Infrastructures

Daniele Santoro (dsantoro@fbk.eu)
Andrea Detassis (andrea.detassis@thinkin.io)
Tommaso Schiavinotto (tommaso.schiavinotto@thinkin.io)

October, 2022


The TOFANE Project

Fair organizations are manifesting strong curiosity for innovative solutions that can exploit the processing of data collected in exhibition venues with the double goal of enriching the visitor experience and providing added value services for the exhibitors. One kind of data for which a lot of interest has been expressed is the localisation of visitors during the exhibition.

This convinced the Provincia Autonoma di Trento to fund a research project named TOFANE (L.P. 6/99 “legge provinciale sugli incentivi alle imprese” – art. 5). The goal is the creation of a state-of-the-art platform (ThinkIN for Fairs) for providing services for fairs based on localisation of visitors and assets. Such services should address both the visitors of a given event (navigation, personalized content delivery) and the fair organization (visitor flow analysis during the event, asset tracking in event setup and dismantling).

The project considers multiple aspects for providing fair organizers with all the tools needed to satisfy the requests they get from visitors and exhibitors. For this reason multiple partners are involved:

- ThinkIN s.r.l. contributes with its experience in localisation data collection, processing and presentation
- Dimension s.r.l. has a strong knowledge in the mobile centered application starting from the required backends to the UX experience of the final application
- The Robust and Secure Distributed Computing (RiSING), a unit of the Cyber Security Research Center at FBK, investigates methods and design and implements platforms that guarantee efficiency, robustness and security in distributed computing environments covering the whole cloud-to-edge continuum.

 In particular ThinkIN provides a software platform (ThinkIN platform) that has been developed with the goal of collecting localization data for assets and people. The platform is agnostic to the technology used for detecting the position of the asset or person. ThinkIN is successfully deployed for several companies in various scenarios (Retail, Industry, Healthcare), it has also been used for some fair, and the early exposure to this new scenario made it clear its potential in terms of services that could be provided.

The aim of the project is to move the platform over an improved infrastructure that would allow it to be secure, scalable, resilient, easy to maintain and would enable some flexibility to provide some services closer to the consumer. The goal of this article is to describe the chosen approach to address such challenges.

The traditional infrastructure

Before starting the project the ThinkIN infrastructure was based on a fixed number of bare metal servers hosting all the required components (DBs, storage, backend services, frontend applications) running on Docker Swarm.

The static infrastructure would not allow easily to vary the number of servers to face possible peak of usage. In particular, fair events last a few days; sizing the whole infrastructure to be able to face the peak of the usage during the fair causes a waste of resources for the rest of the time. One way to solve this problem was to create an ad hoc infrastructure for the events, requiring a lot of effort for maintaining it and keeping it up to date.

The lack of a serious orchestration and deployment process made it less than convenient for the creation/replacement of servers or the deployment of the infrastructure on-premises when required.

Although Docker Swarm allowed the platform to address some aspects of the orchestration, we found that zero-time updates of kernels or Docker itself was often not achievable because of disrupting changes in Docker Swarm protocol between versions.

Removing Docker Swarm would not allow us to ensure an encrypted connection between services; if the service is on-premises we could achieve such security layer only if the infrastructure where the platform is deployed provides some virtual private network, and the use of such facility might change, making the installation and maintenance even more difficult.

Finally, edge software was deployed on-premises on the venue (the machine might have been provided by the customer), with no central administration and difficulty on tracking every single edge node.

During the project we progressively moved to a new infrastructure addressing many shortcomings we had experienced.

The automated infrastructure

One of the principal challenges of the project is to design and implement a distributed infrastructure able to manage, in an efficient and reliable way, data and localisation services with

the main objective to save operational costs for fairs, taking into account also privacy obligations.

Given that the operational costs of a manually managed infrastructure are clearly higher with respect to infrastructures where processes are optimized and automated, the natural evolution is to adopt cloud technologies, already used for some backend services in the Cloud, at the edge of the infrastructure, directly on fair's pavilions.

With this idea in mind, the team decided to introduce two products of the HashiCorp Cloud Platform at the edge, namely: Consul and Nomad. These two tools are very interoperable each other and in particular Consul is used primarily for enabling service discovery and managing a service mesh, even though it also became an enabler for many other features, that run on top of it, while Nomad is used as a container scheduler and orchestrator and has been chosen due to its simplicity and flexibility.

Our first target is of course the implementation of a Consul and Nomad cluster at the edge, so we started to design the edge infrastructure, trying to imagine simple use-cases running on two different pavilions that offer services of the *ThinkIN for Fair* platform to final users.

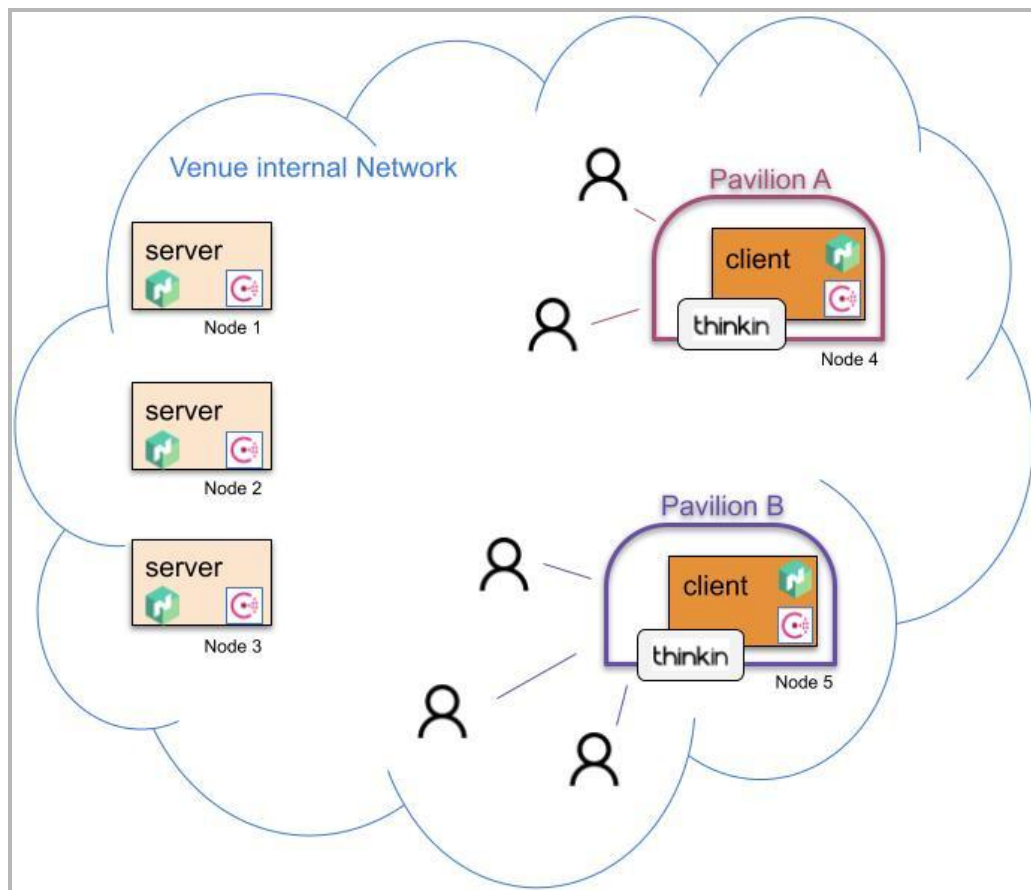


Fig. 1: Nomad/Consul architecture

The final architecture is illustrated in Fig. 1, it shows a typical venue in which we have two distinct pavilions in which we deploy two Consul and Nomad Data Plane nodes (clients) managed by three Control Plane nodes (servers) installed on the same venue network for high availability.

Regarding the location of cluster nodes, a typical fog/edge architecture suggests, of course, to physically place data plane nodes on every pavilion because their role is to serve users in proximity of them relying on the ThinkIN software layer.

Deployment automation and Infrastructure as Code

The bootstrap and installation of such infrastructure is performed using [Terraform](#) and [Ansible](#), two very interoperable tools which are nowadays a de-facto-standard for implementing automation and Infrastructure as Code principles.

Terraform, which has a huge compatibility with many cloud providers, is used for the node provisioning but mainly during the development and testing phase, conducted on the [Hetzner Cloud](#), because in real fair environments, most of the time, bare-metal nodes are already provisioned by the facility owner.

Ansible is the key of deployment and configuration management, indeed it is extensively used to build-up our platform on nodes, by default, on top of Ubuntu 22.04 LTS distribution. In particular we use a main Ansible Playbook which behaves differently relying on two different Ansible Roles: one for installation and configuring client nodes (hosting cluster data-plane) and one dedicated to server nodes (hosting cluster control-plane).

In this bootstrapping phase, the support from the [Open Source Ansible Community](#) is extremely important given that it is based on extensively tested and secure public playbooks. Thanks to the [Ansible Galaxy](#) utility, we pull code of three additional roles for deploying respectively [Consul](#), [Nomad](#) and [Vault](#) (discussed in next sections) on cluster nodes, in addition to other minor requirements related to these main components.

In this way we are able to support, with minimal effort and cost, the rapid provisioning and deployment of different environment inventories varying from production bare-metal servers or cloudlets at the edge to development and testing virtual machines in the cloud.

Finally, [Ansible Vault](#) is our choice to ensure that sensitive content related to specific environments such as passwords, keys, configuration variables and bootstrapping tokens or TLS certificates, is protected and encrypted rather than visible in plain text from the Git repository where we store code for building the infrastructure.

Enabled Use-case(s)

Now that the infrastructure building blocks are there, the target is to test and demonstrate, through simple PoCs, that the innovative applications and services offered by the *ThinkIN for Fairs* platform are supported by the underlying infrastructure and then validated with final users.

The applicative use-cases that will be offered from the platform highlight the advantages of the combination between an automated infrastructure, cloud-native applications and an indoor/outdoor localisation system.

For example, a first use-case is about "*Low latency real time position*". In this use-case premium users looking to navigate to a specific stand may be connected via WiFi instead of mobile networks (eg: 3G/4G, LTE) reducing the latency by receiving position updates from the "nearest" edge server which is running locally the *ThinkIN* position service.

Another interesting use-case is the "*Real Time position sharing*" in which in order to meet, or to be coordinated, two or more individuals might want to mutually share their position or, as a slightly sophisticated privacy-oriented version, in which the pavilion is always shown, while the detailed position is shown only when they are in the same pavilion.

Finally two additional innovative use-cases enabled by the platform are "*Delimited zone violation*" and "*Crowd management*". The first is interesting if kids are left in a supervised play area and parents or caretakers might be notified if the kids leave the area, while the second may be useful if a guided group is visiting the venue: in this case the coordinator wants to be alerted when someone, kids or elders, gets far from the group.

Of course all the above use-cases introduce additional technical challenges, for instance in some cases users need to be transparently connected, for example via websockets, to the closest server. Here we identify two technical requirements the underlying infrastructure should support: the first is about controlling the routing of the connection and the second is about identifying the destination node.

In order to support the application layer to provide these features, two additional backend services have been installed, they are Traefik e Prometheus.

[Traefik](#) is used as the entrypoint for all the connections towards the ThinkIN Platform, in this way we are able to programmatically control where the connection should be directed, for example using specific HTTP query string parameters. In the same context [Prometheus](#) helps in keeping both applications and infrastructure monitored allowing the platform to make the right decisions, for example, regarding which is the "best" or "closest" server offering a specific service to the user.

At this point infrastructure automation is in place enabling new features and use-cases but, given that a core target of the project is to treat data distribution by taking into account also privacy obligations, we cannot sacrifice confidentiality, integrity and authentication. For this reason an additional step is necessary: introduce strong security at the edge.

Enforce security

We should stress one more time that this is not a traditional "*on-premises*" context in which infrastructure administrators have the property and the full control of the physical network. Indeed the platform is usually hosted on the venue where the event will be organized and this implies, even further, that strong security at the edge must be in place, for both communication and processes.

With the main objective of increasing the overall level of security and thanks to the guidelines of [ENISA](#) and [NIST](#) the team decided to apply, as far as possible, the Zero Trust Architecture (ZTA) paradigm targeting, in particular, three different aspects of the platform:

- The Secrets layer
- The Network layer
- The Application layer

To accomplish the task of implementing ZTA in the platform we based our journey on top of two different references:

- The [Zero Trust Architecture](#) publication from NIST and in particular on the chapter 2.1 in which guidelines and tenets of a ZTA are summarized
- The set of [Requirements and recommendations](#) for operating a secure Consul deployment from HashiCorp, because with respect to network communications, Consul is the key component of the platform.

Zero-Trust at the Secrets layer

If your infrastructure relies on Consul and Nomad and you want to implement the ZTA paradigm, introducing the [HashiCorp Vault](#) is a must for two main reasons: i) the features it offers and ii) the integration with other HashiCorp components.

Vault is an identity-based secret and encryption management system that you can use as a secure source of data for any sensitive information: from simple application passwords to most complex service certificates used for TLS encryption and authentication, but it also offers advanced features like management of credentials and certificates rotation.

In the TOFANE project, Vault is installed in a HA (High Available) configuration and it is composed of three nodes and it is used mainly as PKI secrets engine for X.509 certificates in order to secure all the infrastructure and application communications. This means that the storing, generation and automatic renewal of TLS certificates and keys is completely managed by Vault, allowing the use of short lived certificates that, of course, decrease the attack surface.

In addition to PKI management, Vault is the reference point for any secrets or confidential information that applications running in the cluster may need to use. They can rely on many ways of interfacing with it, for instance CLI, API and even dedicated programming libraries.

Zero-Trust at the Network layer

At the infrastructure layer, we want to implement the paradigm of "TLS everywhere", meaning that all the networking communications, involving the backend components of the cluster, must be protected using TLS. This means that the communication between all agents running on different nodes and in some cases also on the same node uses TLS and short-lived certificates (with a TTL of maximum 24 hours) to encrypt the traffic making it very hard for an attacker to exploit content hijacking.

So we protect communication between, for example, Consul agents and Vault, or Consul agents on one node and other Consul agents on other nodes but how can we ensure that a given agent has not been compromised?

This interesting and advanced configuration is possible by relying on mTLS for authentication other than for encryption ensuring that the parties (eg: Consul or Nomad agents) at each end of a network connection are who they claim to be by verifying that they both have the correct private key. Moreover other information within their respective TLS certificates, like Common Names, Alternative Names or IP Sans, provides additional verification.

Zero-Trust at the Application layer

We like to think of these security aspects as if they were layers of a Matrioska: enforcing security on the external surfaces is of course more important but also internal ones must be protected if we want to fully apply ZTA principles.

If the infrastructure layer is the most external one, the one which has a portion of its surface exposed, the application layer is somehow protected but we can do more by applying ZTA principles on the Consul Connect service-mesh, which is the network used by applications in the cluster.

So at the application layer, meaning service-to-service or container-to-container communication, we configured two important security features: i) Access Control List (ACL) and ii) mTLS encryption and authentication across application components.

With the first we are essentially able to control which services are allowed to communicate with other services and by applying a "Deny All" default policy, only explicit rules allow two or more application services to communicate.

The second feature is similar to the one described at the network layer but applied for traffic of applications that are in the service mesh. In this way, and thanks to a sidecar proxy based on [Envoy](#), application components, which are usually part of a microservice architecture, can not only communicate securely on an encrypted channel, but also verify the authenticity of the other parties.

If your target is to optimally manage services and resources, without sacrificing reliability, there is an additional step to push the evolution forwards: the erogation of services could be dynamically balanced between cloud and edge, introducing the concept of Hybrid infrastructure which spans over the Cloud to Edge continuum.

With automation and security in mind we envision innovative, robust and secure hybrid architectures to support additional use-cases.

The hybrid infrastructure

The use of a Nomad, Consul and Vault cluster on top of a secure communication layer and a proper access control configuration allows the creation, in the future, of a hybrid infrastructure where some cloud based nodes and edge nodes deployed at fair venues belong to the same cluster. A unique cluster would also allow to have a single management entry point for all the remote locations and a unique control plane for multiple clusters would allow, even further, more complex hybrid topologies.

Based on the knowledge built during the project the HashiCorp tools should allow such a configuration by leveraging the Cluster Federation features which permit the partitioning of cluster resources not only by data centers but also by regions.

Furthermore, this approach enables some possible scenarios where computations can be moved towards the edge or on cloud based on requirements of locality or better resource exploitation; edge resource limitations might be alleviated by moving part of computation on cloud during peak periods. One can think of exploiting unused resources in a venue when required by another venue in need that belongs to the same customer.

Of course additional aspects regarding bootstrapping, automating and securing the whole hybrid infrastructure will become more important, as will those related to networking but we have built solid pillars to start with new interesting experimentations.

Thanks to the above technological evolution and innovations developed during the project and in line with ThinkIN and Dimension market positioning, the next steps will target the engineering of project results into a full fledged product that can enter the market with a clear proposition and take advantage of the distributed and secure infrastructure as a distinguishing element compared to other solutions approaching the market.