# Multi-tenant Data Management in Collaborative Zero Defect Manufacturing

Francisco Fraile
*CIGIP*
*Universitat Politècnica de València*
Valencia, Spain
ffraile@cigip.upv.es

Leticia Montalvillo
*Industrial Cybersecurity*
*IKERLAN*
Arrasate, Spain
lmontalvillo@ikerlan.es

María Ángeles Rodríguez
*CIGIP*
*Universitat Politècnica de València*
Valencia, Spain
marodriguez@cigip.upv.es

Héctor Navarro
*CIGIP*
*Universitat Politècnica de València*
Valencia, Spain
hecnabae@cigip.upv.es

Ángel Ortiz
*CIGIP*
*Universitat Politècnica de València*
Valencia, Spain
aortiz@cigip.upv.es

***Abstract*—This research paper describes different patterns and best practices to effectively implement multi-tenancy of production sensor data in collaborative applications. The paper explains the design considerations taken to support multi-tenancy in the Zero Defects Manufacturing Platform (ZDMP), using concrete collaborative use cases as an example. The main objective is to provide an overview of multi-tenancy as an enabler of collaborative use cases in digital manufacturing platforms, describe the different design patterns, the main trade-offs, and best practices.**

***Keywords*— Multi-tenancy, zero defects manufacturing**

## I. INTRODUCTION

One of the keys to implement effective zero defects manufacturing systems is the ability to integrate detection, prediction, correction, and prevention strategies into a holistic approach to eliminate product and process defects [1]. Data collected in the shop-floor is the cornerstone of this vision. Thanks to the latest advancements in sensor technology, Machine to Machine (M2M) communications, and Big Data, manufacturing companies are able to capture information about products and processes with unprecedent level of detail. From these data, machine learning, and artificial intelligence unveil hidden defects and learn courses of action that can potentially lead the company towards zero defects manufacturing. The benefits obtained depend to a great extent on the quality and the quantity of collected data. The accuracy of the detection and prediction capabilities of the system improve as the company enhances its capacity to collect and store data.

However, a manufacturing company can only go so far in this journey towards zero defects without collaboration, for two main reasons. One is the high operational costs of managing Big Data. Certainly, companies are reluctant to let critical operational data leave the boundaries of their IT infrastructure. The first option is therefore to invest in the required infrastructure and services to manage collected data privately, within the organization boundaries. However, this alternative comes at the expense of additional operational costs. If companies decide to externalize infrastructure and services, thanks to economies of scale, digital platforms are able to offer managed services at a very competitive cost, without the know-how required for deployment, maintenance, and management.

The other reason is that, nowadays supply chains are so entangled that there is a high degree of correlation between the defects that occur within the boundaries of collaborators and the production data collected either downstream or upstream the supply chain. Indeed, the failures detected in a company can be rooted back to the production process or the production environment of one of the providers. Conversely, the defects detected on a product manufactured by a provider may be traced back to the production process of a company. Machine learning is able to capture these relationships and provide actionable feedback as long as it receives the right input data.

To benefit from these synergies, collaborators need to integrate their production data, either on premise (i.e. managed by one of the partners) or in cloud (i.e. managed by an external service provider). Ideally, each collaborator must be able to exercise their rights over their data: manage who can access what data in which way. Multi-tenancy is a software pattern to accomplish this and is considered a key enabler of cloud computing. Multi-tenant architectures allow different organizations (or tenants) to securely share data services and underlying infrastructure. Each collaborator (or tenant) is able to manage access to their data using Rule Based Access Control (RBAC) policies.

The Zero Defects Manufacturing Platform (ZDMP) [2] is a digital manufacturing platform specifically designed to support zero defect manufacturing strategies in companies of any sector. This paper presents the position of ZDMP towards multi-tenancy and discusses the main trade-offs of the different alternatives for zero defects manufacturing use cases. The rest of the paper is structured as follows. Next section contains a brief description of multi-tenancy. Section III describes the key technological enablers and the approach in ZDMP, and Section IV discusses configuration options and best practices to support multi-tenancy in different use cases. Finally, Section V includes a discussion on the different approaches to enable collaboration in ZDMP.

## II. MULTI-TENANCY

Multi-tenancy refers to a software architecture in which a single instance of software serves multiple tenants. A tenant is a group of users who share a common access with specific privileges to the software instance [3]. Regarding data, if a service is multi-tenant, then each tenant can manage authorization permissions within the boundaries of their organization, but not across different organizations. From a usage perspective, each organization is an independent data silo: users of different organizations may use the same application and use their respective credentials to authenticate, but the data is completely segmented based on the access right that they have in the organization they log in to.

However, from a technical point of view, multi-tenancy is a cross-cutting concern that impacts several layers. For instance, some common patterns that can be used to manage data in a multi-tenant application deployed with a microservice architecture [3] are:

- Separate data services per tenant: Applications may manage a separate instance of a database service per tenant, achieving a high level of data isolation, at the expense of complexity, and scalability.
- Separate data service resource per tenant: Applications may use a single instance of a database service, but tenants may use independent resources within the service (e.g. different databases or different file storage service buckets). This pattern achieves higher isolation than the alternative below, but it may hinder collaboration between companies.
- Separate data elements per tenant: Applications may use a single data resource shared among all tenants (e.g. shared database). In this case, it is easier to share data between tenants, but there is less isolation. Also, it requires a robust access policy model to adequately manage access permissions.

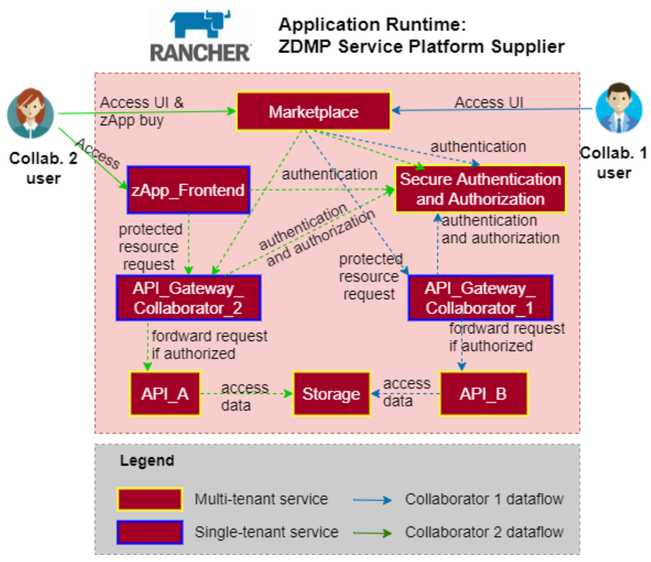
These patterns provide different trade-offs (between isolation, implementation complexity, operational efficiency) that need to be taken into account for each particular use case. Additionally, there are different ways to implement multi-tenancy with respect to the authentication (verify user's identity ) and authorization (verify users' permissions) services, as well as, in the virtualization service. All these considerations need to be taken carefully into account in any implementation.

## III. MULTI-TENANCY IN ZDMP

ZDMP is a service Platform to support zero defects manufacturing. It provides a range of services that allows companies to integrate AI and analytic services into their business processes, i.e., Industrial IoT to connect to physical assets, digital twins to model the hierarchical levels of the factory, Big Data, analytic services, and AI to implement simulation, learning and reasoning capabilities. Additionally, ZDMP provides a development environment that facilitates the creation of new verticals for specific sectors or use cases. A marketplace allows companies to discover and install these applications. There are different deployment patterns available for interested companies. Any company that wants to use a ZDMP application (or zApp in short) may connect to the services hosted by a ZDMP Service Platform provider (i.e., Platform as a Service). Some use cases may require that some of the services are deployed on-premise, using the ZDMP edge platform (distributed computing). The main objective of the edge platform is to reduce latency and/or data throughput between the edge tier and the cloud platform tier for demanding use cases. Also, a company may deploy the entire platform on-premise, provisioning and managing the required infrastructure, as well as, the platform services.

Regarding collaborative zero defects manufacturing, companies may collaborate between them using multi-tenant applications. Supported by the ZDMP core services, these applications allow users to manage how supply chain collaborators can access their data. If the Platform is installed on-premise, then the ZDMP Service Platform provider is also one of the collaborators of the supply chain. For instance, a large company in the automotive sector may be the ZDMP Service Platform Provider, and collaborate with tier providers, logistic service providers, or customers through applications. Other companies may use the applications of a ZDMP Service Platform which is hosted by a company which core business is to manage the platform services. Additionally, applications may be federated [4], so that they can offer applications from one platform marketplace into another, and interconnect data services.

Core ZDMP services have been mentioned in the paragraph above. These services provide the runtime to all other services as well as key platform features like multi-tenancy. They are depicted in the Fig. 1.



RANCHER
Application Runtime:
ZDMP Service Platform Supplier
Collab. 2 user
Collab. 1 user
Access UI & zApp buy
Marketplace
Access UI
Access
authentication
zApp_Frontend
authentication
Secure Authentication and Authorization
protected resource request
authentication and authorization
authentication and authorization
protected resource request
API_Gateway_ Collaborator_2
API_Gateway_ Collaborator_1
forward request if authorized
forward request if authorized
API_A
access data
Storage
access data
API_B
Legend
Multi-tenant service
Single-tenant service
Collaborator 1 dataflow
Collaborator 2 dataflow

Fig. 1. Core ZDMP services running in ZDMP Service Platform (Rancher) and dataflows for Collaborator 1 and Collaborator 2.

The ZDMP platform is deployed thanks to the Application Runtime component (Rancher – a Kubernetes management platform that deploys the Kubernetes cluster running all the ZDMP services as docker containers [5]). So far, the approach is to have one cluster in which all components (from all ZDMP Service Platform provider collaborators) are deployed. Besides, as stated above, the distributed component can deploy smaller clusters at the edge tier of collaborators for specific use cases. Multi-tenancy is rooted in the Authorization and Authentication component, which is implemented with Keycloak [6]. Keycloak is an open-source Identity Access Management software that provides authentication and authorization mechanisms for all users across all collaborators, as well as, for any other components deployed in the application runtime component. The Application Programming Interface Gateway (API-GW) is an intermediary software component that acts as a single-point of entry for a set of services and APIs (Application Programming Interfaces). Furthermore, it enforces access control on API endpoints, by connecting to the authentication and authorization component to enforce access policies (authorization based on scopes) [7]. The Storage component provides different storage mechanisms for securely storing platform data. These services, together with the Portal, which provides centralised access to all platform components, and the Marketplace, are known as the ZDMP core services.

ZDMP applications and components can be either multi-tenant or single tenant. The Marketplace is inherently multi-tenant since all the ZDMP Service Platform Supplier's collaborators, i.e. *Collaborator1* and *Collaborator2* can access it. A given zApp can be single-tenant if only one client has the rights to access to it (or multiple-tenant if multiple clients can have access to it). An example of a single-tenant application is an application deployed at the edge.

Regarding data, the ZDMP Platform provides several options. The Storage component is a service that allows to create and access data-lakes and manage large volumes of data. It also provides central file storage and database services for internal components. All these services are also multi-tenant, but it is not the only option available. Applications can also deploy single-tenant or multi-tenant internal data services, either in the platform, or at the edge.

Now, with all these options available, developers have a wide variety of options to implement the right multi-tenancy pattern for their collaborative zero defects manufacturing use case: Given an application that is multi-tenant, the data can be stored in separate data services, in separate databases, or in a shared database. The following section introduces some Keycloak key concepts needed to support the comparison analysis on the different alternatives to handle multi-tenancy and describe the implications and possible impact in other related ZDMP components (Application Runtime, API-GW, Storage).

### A. *Keycloak Concepts and considerations*

There are some Keycloak core concepts that need to be described in order to understand the different options available to implement multi-tenancy.

Firstly, users are entities that can log into the system. They can be assigned group membership and have specific roles assigned to them. Secondly, roles identify a type or category of user, like "admin", "user", "manager", or "employee". Applications often assign access and permissions to specific roles rather than individual users, following the Role-Based Access Control (RBAC). Thirdly, groups manage set of users. Roles can be mapped to groups and attributes can be defined for a given group. Users that become members of a group inherit the attributes and role mappings that group defines. Fourthly, a realm is an object of Keycloak that manages a set of users, credentials, roles, and groups (entities that need to be protected). A user belongs to and logs into a realm. Realms are isolated from one another and can only manage and authenticate the users that they control. Finally, clients are entities that can request Keycloak to authenticate a user. Most often, clients are applications and services that want to use Keycloak to secure themselves and provide a single sign-on solution.

A graphical depiction of these concepts is shown in the Fig. 2, one supplier company which provides ZDMP Marketplace has one realm (*ZDMP Service Platform Supplier Realm*) with two clients (*API-1 and webapp*). Keycloak holds the information about these applications (URL, allowed origins, allowed redirects, etc). Then, in that realm there is a set of registered users. Users within a realm can have access to that realm's clients (if they have the rights to access them). Additionally, users can be assigned to multiple groups. Moreover, roles can be created within a realm, and user groups can be attached to multiple roles, as well as users can be directly assigned with roles (roles serve as access control, and grant access to certain clients).

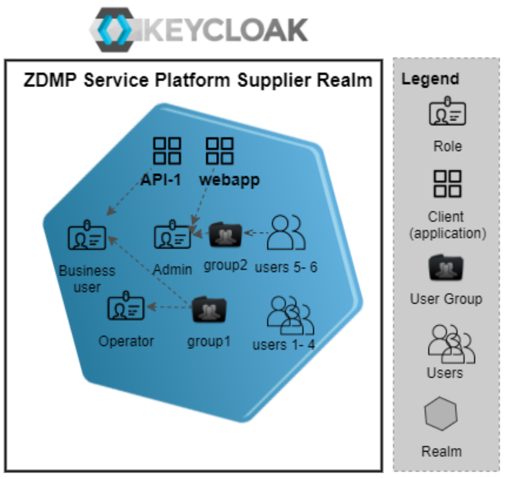


Fig. 2. Depicting Keycloak concepts.: realm, clients, roles, user groups and users

Now, after introducing these key concepts, let us discuss some technical considerations and restrictions important regarding multi-tenancy. First, the Application Runtime instance (Rancher) can only be connected to one realm. This connection is to provide users access to the Rancher user interface (UI) and clients to deploy components or applications. Therefore, there must be a specific realm for the ZDMP Service Platform Provider, which is used to manage the platform deployment and provide access to the runtime management UI. Second, regarding Keycloak, any user and client need to be registered in a realm to secure communications. However, there is no restriction on the level of isolation provided by the platform security mechanisms. Broadly, it is possible to go for a multi-realm option (one realm per client), or one realm where all users from all clients are mixed. The API-GW component and can connect to just one realm. That is, enforcement is provided as per realm basis. Finally, ZDMP applications: ZDMP applications need to connect to the authentication component first to get authenticated (e.g. user login). It is easier to connect to one realm, but it is also feasible to connect to different realms. Within each realm, independent API-GW instances will act as independent policy enforcement points to secure connections and provide an adequate level of isolation.

Taking these considerations into account, multi-tenancy in collaborative scenarios can be achieved in different ways, described in the following sections.

## IV. Multi-tenancy Implementation Options

### A. *Multi-realm implementation*

The Multi-realm implementation implies the deployment of one realm for each of the organizations registered within ZDMP. Therefore, in this case, Keycloak would have one separate realm for each supply chain collaborator. One of the key concepts of multi-tenancy architecture is the complete isolation from one tenant to the other. Keycloak logically splits multi-tenant apps into realms, meaning that each collaborator will have a separate configuration (users, groups, roles), as shown in Fig. 3.

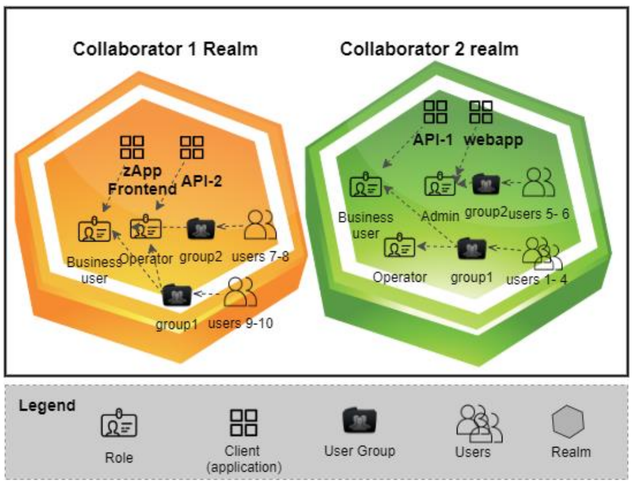


Fig. 3. Multi-realm approach: each collaborator has its own isolated realm where applications (clients), groups, roles and users are registered

As an example, the use of this approach in a collaborative zero defects manufacturing use case using this approach is shown in Fig. 4. There are two differentiated realms, each holds info only of a unique company (i.e., roles, users' data, permission) and provides separation of the data. Also, there are two API-GW deployments, each connecting to a single realm. Each API-GW may hold the configuration of proprietary applications (single-tenant applications) with private databases, like *App_1* and *App_2* of *Collaborator1*, or shared applications (multi-tenant) with shared data services like *App_N* of *Collaborator2*. Note that shared database services need to appropriately separate the data for each tenant, applying any of the multi-tenancy patterns introduced in Section II.

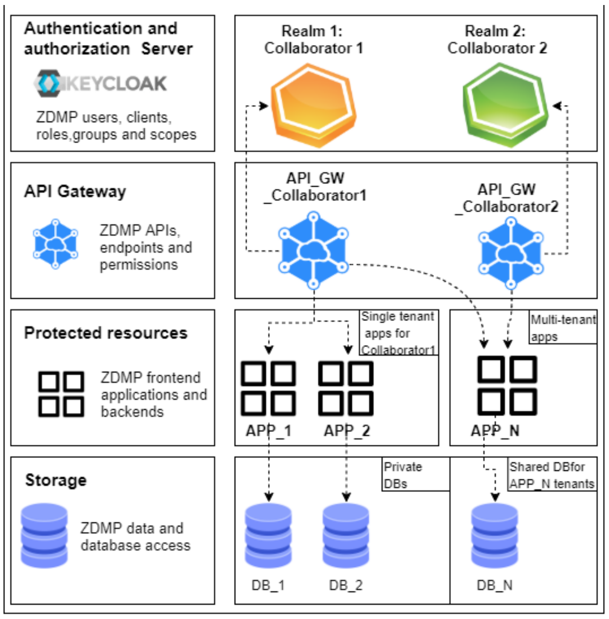

Authentication and authorization Server
KEYCLOAK
ZDMP users, clients, roles,groups and scopes
Realm 1: Collaborator 1
Realm 2: Collaborator 2
API Gateway
ZDMP APIs, endpoints and permissions
API_GW _Collaborator1
API_GW _Collaborator2
Protected resources
ZDMP frontend applications and backends
Single tenant apps for Collaborator1
Multi-tenant apps
APP_1
APP_2
APP_N
Storage
ZDMP data and database access
Private DBs
Shared DBfor APP_N tenants
DB_1
DB_2
DB_N

Fig. 4. Multi-tenant approach in ZDMP platform: multi-realm alternative

## B. Single-realm implementation

This implementation implies having one realm shared by all the collaborators. Here, multi-tenancy is achieved by organizing users into groups and providing access to client applications based on these groups. In this case, the ZDMP Service Platform Supplier Realm should have one Keycloak server and one shared realm to manage user rules and policies, for all companies, whether they are supply chain collaborators or not, as shown in Fig. 5.

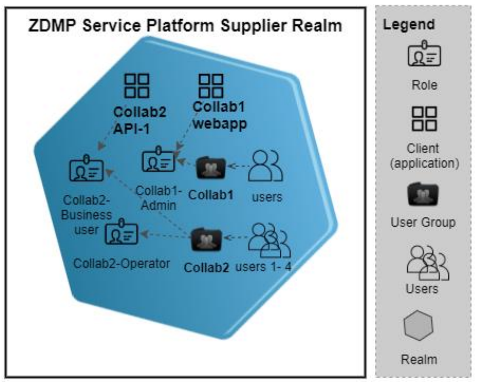


Fig. 5. Single-realm approach: collaborators 1 and 2 share the same realm and naming conventions for groups, roles and clients need to be declared to separate them apart.

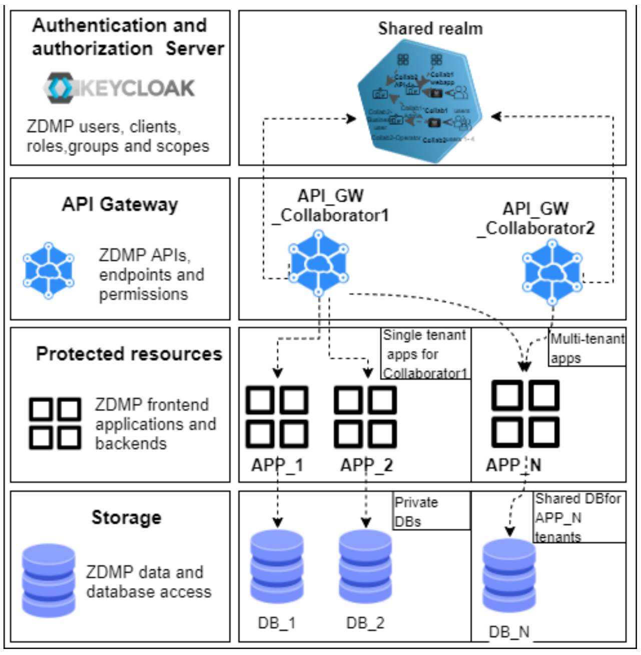

Authentication and authorization Server
KEYCLOAK
ZDMP users, clients, roles,groups and scopes
Shared realm
API Gateway
ZDMP APIs, endpoints and permissions
API_GW _Collaborator1
API_GW _Collaborator2
Protected resources
ZDMP frontend applications and backends
Single tenant apps for Collaborator1
Multi-tenant apps
APP_1
APP_2
APP_N
Storage
ZDMP data and database access
Private DBs
Shared DBfor APP_N tenants
DB_1
DB_2
DB_N

Fig. 6. Multi-tenant approach in ZDMP platform: single-realm alternative.

In ZDMP, the use of this approach is shown in Fig. 6. There is one shared realm for all collaborators, and having multi-tenants in one realm requires to work around tenant's separation using groups. Also, there are independent API-GW deployments for each collaborator, one for *Collaborator1* and one for *Collaborator2*, each connected to the same realm. API-GW may be holding the configuration of proprietary applications (single tenants) with private databases like *App_1* and *App_2* of *Collaborator1*, or shared applications (multi-tenant) with shared data services like *App_N* of *Collaborator2*. Again, shared database services need to apply one of the multi-tenancy patterns described above.

## C. Multi-Keycloak implementation

It is also possible to have multiple Keycloak servers. In this configuration, each collaborator has an independent Keycloak server. In ZDMP platform the use of this approach is very similar to Fig 4. Instead of having one keycloak server with different realms (as in Fig. 4), in a multi-keycloak approach each collaborator has its own keycloak server instance where their realm is kept. The platform provider's

Keycloak server manages a realm with the Keycloak administrator account of the different collaborators, so that rancher (that can only connect to one realm) can authenticate administrator users, who can in turn monitor the applications deployed applications in their corresponding realms. Each collaborator can manage their user accounts and realms independently in their own Keycloak server. Multi-tenancy applications can integrate clients to connect to different customer realms (e.g. *Collaborator1* and *Collaborator2*). And single-tenant applications (proprietary applications) connect to a single realm.

### D. Hybrid implementation

This implementation implements a combination of the first and the second alternative, leveraging Keycloak identity brokering and delegated authentication. In this configuration, a central authentication server manages all user accounts, and the rancher realm. Each collaborator manages independently their respecting realms, delegating authentication to the central platform authentication server. This resolves the limitation that rancher must connect to only one realm, but at the same time providing the required security and isolation between collaborators, as there is one realm per collaborator in the different use cases, and an independent policy enforcement point. The main advantage is that all collaborators can use a shared authentication server with a single credential per user to access either single tenant applications or multi-tenant applications, as shown in Fig.8. The shared central authentication server would again need to differentiate tenants using groups and allow/deny access based on groups.

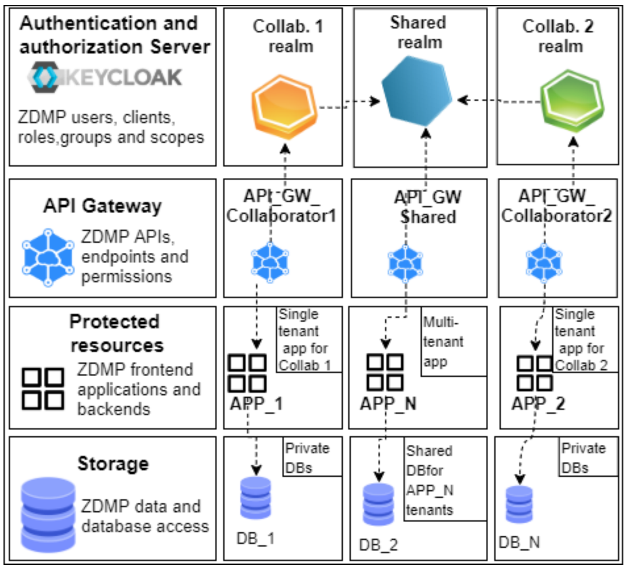

Authentication and authorization Server
KEYCLOAK
ZDMP users, clients, roles,groups and scopes
Collab. 1 realm
Shared realm
Collab. 2 realm
API Gateway
ZDMP APIs, endpoints and permissions
API_GW_ Collaborator1
API_GW Shared
API_GW_ Collaborator2
Protected resources
ZDMP frontend applications and backends
Single tenant app for Collab 1
APP_1
Multi-tenant app
APP_N
Single tenant app for Collab 2
APP_2
Storage
ZDMP data and database access
Private DBs
DB_1
Shared DBfor APP_N tenants
DB_2
Private DBs
DB_N

Fig. 7. Hybrid approach in ZDMP platform.

## V. Guidelines for Collaborative ZDM Use Cases

Based on this implementation patterns, this section discusses the different approaches to enable collaboration in ZDMP. The multi-realm implementation provides strong security control, because there is a clear separation of users and more control for access permissions since users and applications from different companies are separated. Further, the hybrid implementation provides a similar level of isolation between collaborators, with the advantage that all the user accounts can be managed centrally, which is beneficial also for usability. However, the approach chosen for ZDMP is the multi-realm approach, one realm per customer and one Keycloak server. This configuration provides security-by design as companies have separated realm for their users, roles, clients etc. and this prevents unauthorized access and data leakages upon misconfigurations. Each company can manage their own realm independently in both alternatives, and they decide how to share data with their collaborators in multi-tenant applications. Multi-tenant applications on the other hand can use any of the patterns available to achieve the required level of isolation for each specific use case. This set up is ideal for any collaborative use case, but comes at the expense of complexity, since collaborative zero defects manufacturing applications need to manage clients and connections to different realms within their code. To alleviate this, the ZDMP Portal component provides convenient session information to allow applications to connect to the right realm.

In the single-realm configuration, there is less privacy control for collaborators which are not the ZDMP Service Platform Provider, since applications need to connect to just one realm. However, it is still possible to achieve adequate levels of isolation between data. This set up is of course easier to manage and applications are also easier to develop. This set up may be convenient in collaborative use cases where there is a high level of trust between collaborators, and the ZDMP Service Platform Provider is the main agent of the supply chain. This set up is not convenient when the ZDMP provider is a PaaS service provider, since there is no adequate level of separation to host different supply chains.

## Acknowledgment

This work was supported in part by the European Commission under the Grant Agreement 825631. The author María Ángeles Rodríguez was supported by the Generalitat Valenciana (Conselleria de Educación, Investigación, Cultura y Deporte) under Grant-Agreement ACIF/2019/021.

## References

[1] Psarommatis, F., May, G., Dreyfus, P. A., & Kiritsis, D. (2020). Zero defect manufacturing: state-of-the-art review, shortcomings and future directions in research. International Journal of Production Research, 58(1), 1-17.

[2] Campbell, S., Cáceres, S., Pagalday, G., Poler, R., & Gonçalves, R. (2020). A European Manufacturing Platform for Zero-Defects. Proceedings of the 10th International Conference on Interoperability for Enterprise Systems and Applications (I-ESA 2020).

[3] Pachghare, V. (2016). Microservices Architecture for Cloud Computing. Journal of Information Technology and Sciences, vol. 2.

[4] Jacoby, M., Antonić, A., Kreiner, K., Łapacz, R., & Pielorz, J. (2016, November). Semantic interoperability as key to iot platform federation. In International Workshop on Interoperability and Open-Source Solutions (pp. 3-19). Springer, Cham.

[5] Buchanan, S., Rangama, J., & Bellavance, N. (2020). Inside Kubernetes. In Introducing Azure Kubernetes Service (pp. 35-50). Apress, Berkeley, CA.

[6] Keycloak (Accesed December 9, 2020). URL http://www.keycloak.org

[7] Christie, M. A., Bhandar, A., Nakandala, S., Marru, S., Abeysinghe, E., Pamidighantam, S., & Pierce, M. E. (2017). Using keycloak for Gateway Authentication and Authorization. Presented at Gateways 2017, University of Michigan, Ann Arbor, MI, October 23-25, 2017.