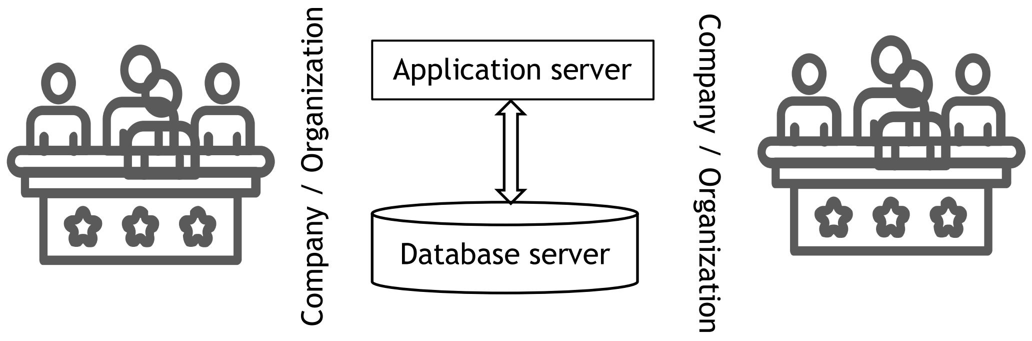
Abstract

According to Wikipedia, 'The term "software multitenancy" refers to a software architecture in which a single instance of software runs on a server and serves multiple tenants.' This term is widely referred and perceived differently in the industry by SIs (Service Integrators) and clients. Many of them restrict multi-tenancy to the departments of verticals in their organization.

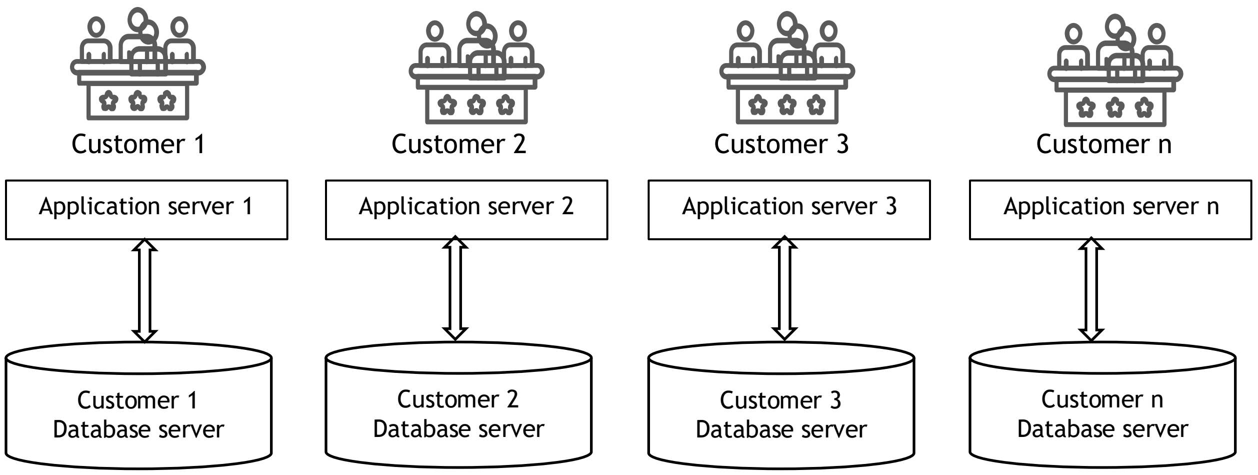
Let's just take a layman's approach to understand it before we discuss it technically. Imagine a gated community available for tenancy. The facilities/options offered are apartments, villas, duplex, community hall, swimming pool etc. Most of the facilities can be offered in common while some people can opt select facilities privately - One can opt for a private swimming pool, servant quarters, balcony, car park etc.

When rules & regulations are created for governance, they are applicable to all of them. For instance, when a visitor checks in at security gate, he is expected to park his vehicle, if any, at visitor's parking. The common gymnasium may have different restrictions of using the equipment for different tenants, depending on membership types. However, some tenants who would've opted/subscribed for additional services may use their private facilities as per their requirements. Further, when such some tenants plan to privately do changes (customizations) to their wardrobes or balcony or servant quarters, that shouldn't impact facilities or services being used by other tenants.

Now that you are clear on the fundamentals, understanding them while discussing technical solution becomes easier. Traditionally, the ITSM solutions have been implemented considering Application server on the one hardware while Database server on a different hardware instance to assist scalability. Web server can be on the same or different hardware than Application server, usually. We will assume the presence of web server as one of above cases, as we go over technical architecture.



When the solution needs to be extended to other external customers, the approach usually considered is to replicate the instance using multiple instances. Multiple instances are considered because some tools or solutions cannot incorporate different workflows or SLAs or templates used by different departments or external customers on the same instance. Essentially, even many SaaS vendors spin off new instances for different clients. This is an expensive approach as it comprises of procuring additional hardware (if not hardware, VMs or containers), software, integration and maintenance, as shown in below diagram -



Imagine implementing the solution from the scratch on separate hardware or replicating the VM instances for different clients. It does get expensive. The customers expect minimum cost towards implementation. As you may have guessed, most likely the costs are passed to the new customers or need to be absorbed.

Problem

These days, all the service management tools following best practices are expected to meet the minimum criteria of ‘department-wise’ or ‘business-wise’ multi-tenancy. Most of the solutions limit multitenancy to data. However, being a multitenant solution, it should also offer tenant specific features, especially like workflows and most importantly, customizations. For example, SLA for an incident of severity “S2” in HR department may differ than with an incident of same severity in IT department.

The complexity increases when the departments in larger organization are spread across different countries and continents and when they would need specific features. The solution also needs to consider the different time-zones as an additional factor for every tenant.

The litmus test to pass is when a large organization who works as MSP (Managed Service Provider) intends to implement common service management solution to cater to its external customers (or internal employees as well). A common application instance and multiple database instances to store every client’s data separately can be considered. Although this is relatively less expensive compared to approach considering multiple instances explained above, it still does not solve the problem of being truly multitenant solution. This approach needs to consider critical features like SLA, workflows and customizations as common to all customers. For example, what if SLA or workflow for one customer for severity “S1” is different than for another customer with same severity? What if this MSP organization has its suppliers/vendors to extend support to its customers?

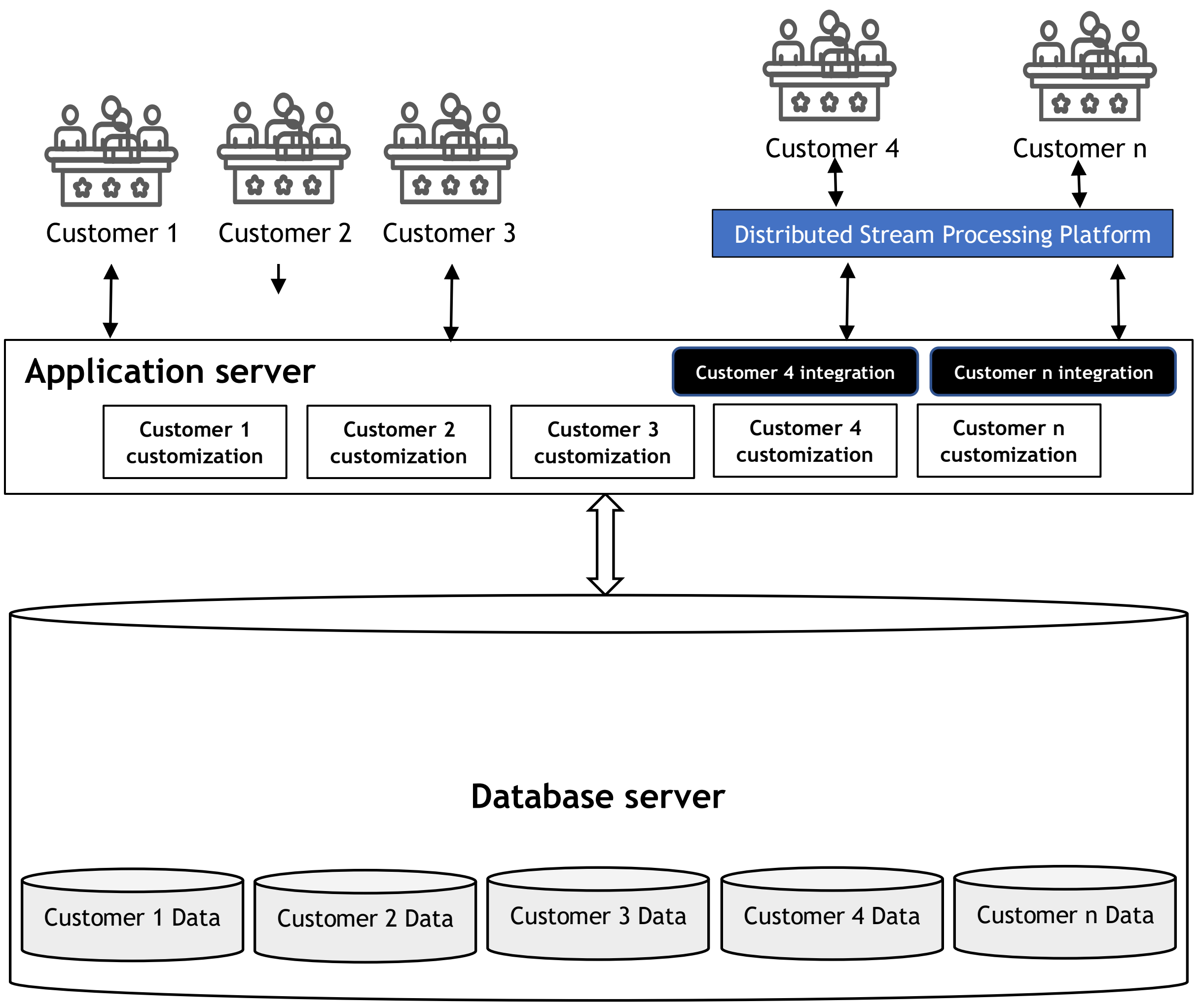
According to predictions for EMEA by Forrester for 2021, Digital leaders will drive double-digit growth in 2021. Research (Forbes, Harvard Business Review & McKinsey) indicates that 70% of Digital Transformation initiatives will not reach stated goals, equating to $900 billion waste. Hence, the importance of ensuring the solution as “truly” multitenant becomes even more critical. The service management solution needs to be truly multitenant to align and accommodate its external customers, suppliers and employees and yet be cost effective.

Solution

A truly multitenant solution can be achieved by not merely providing configurations that are tenant specific but also customizations that can be tenant specific. Depending on the customer requirements on data privacy, this can be achieved in 2 different ways -

Multitenant approach for customers having logical data partition but in the same database

This is a recommended solution for a truly multitenant solution that can be considered cost effective having only one Application server instance and another Database server instance. Note that this architecture does not just address tenant specific templates, SLAs and workflows, it also addresses tenant specific customizations. Different customers can request customizations specific to their business requirements and when deployed, can be used by them explicitly. The below architecture depicts the working model -

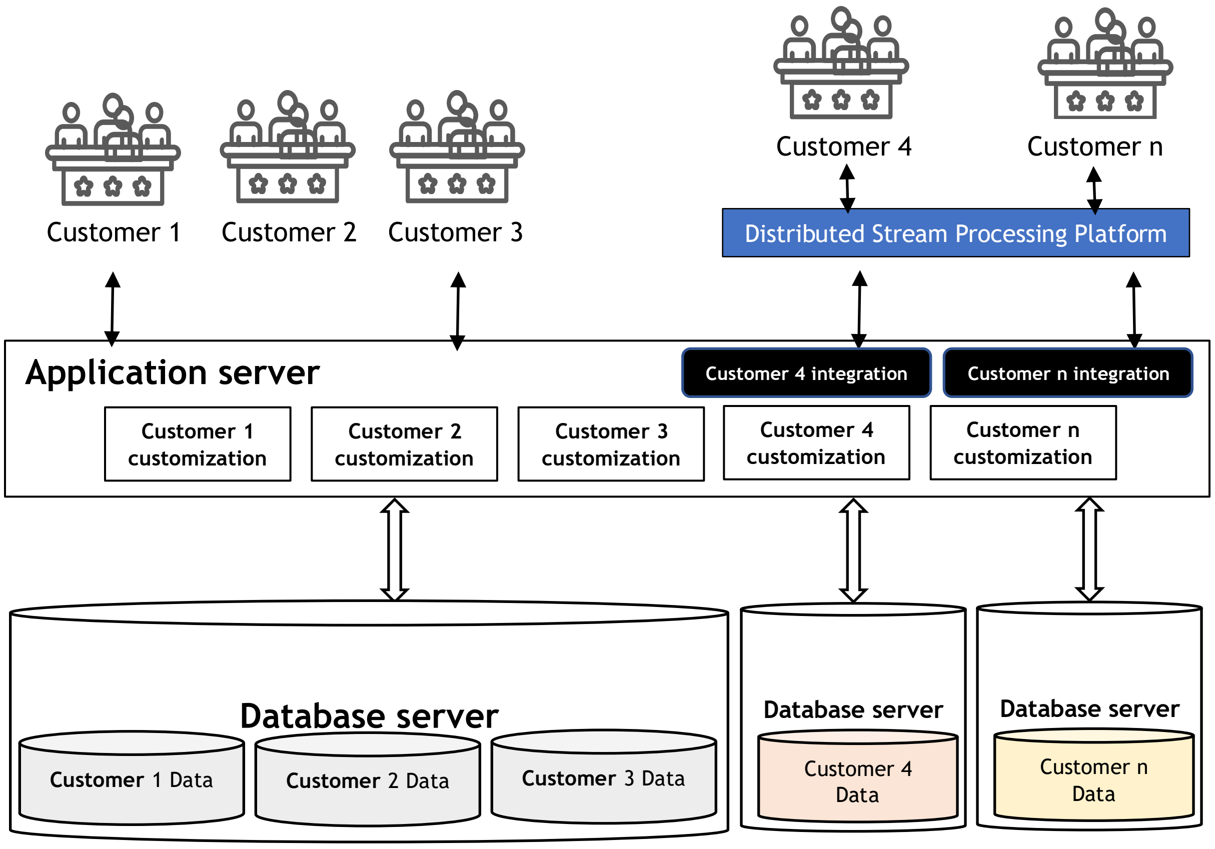


In the above scenario, all the tenants use same URL to access your multitenant system. In that case, the usernames must be unique across the system. That is, once the username is taken by a tenant, another tenant may not be able to use the same username, the way no two users can have same email ids in the email server system. What if some tenants insist to consider usernames that may have been taken by other tenants? What if they prefer their users to use AD or LDAP, the usernames of which may have been already offered to users of other tenants? Well, in that case, different URLs can be considered for every tenant, which will help the tenants to use usernames created in their AD or LDAP. For example, URL to log into the system for one tenant can be https://servicemgmt/client1/ServiceDesk while for another tenant, it can be https://servicemgmt/client2/ServiceDesk. Post authentication, the tenant specific SLAs, workflows, customizations, integrations etc. will be applicable to respective tenant.

Multitenant approach for customers expecting physical data partitions

Occasionally, few customers expect tight data privacy as a part of security policies that they may have signed up with their end customers. The cost goes up since a separate database server needs to be created either on a separate VM or hardware, based on the business requirements or security policy.

The architecture depicted below provides true multitenancy using one Application server and multiple Database servers, if some of the customers prefer or demand to have separate databases. The mechanism and ability to define critical features and customize tenant-wise remains the same as earlier multitenant architecture; except that the data related to some customers will get stored in the relevant database. In this case, some additional efforts are considered to develop filters for data transfers to specific databases based on the customers.



While you've understood the challenges and as well as an approach of making your system truly multitenant, you are now onboarding several tenants over the period. How can you be sure the system will continue to be truly multitenant? With us alongside you, you indeed will be. tussom has been able to solve the above problem and offers a truly multitenant solution that can cater to its external customers, suppliers/vendors, and employees. All the features, especially SLA, templates, workflows, automations, customizations, integrations etc. can be offered as tenant specific on one single no-code instance (cloud as SaaS or an on-premise solution) and tenants can be configured accordingly. It has been implemented in large datacenters, media giant, IT sectors etc., helping to cut their costs significantly and improve productivity.

Conclusion

In today's era, having a truly multitenant solution is a must for every organization that understands digital transformation to grow and multiply its customers while keeping its budgets in control. By having a true multitenant solution,

- MSP organizations can focus on the business instead of mulling over accommodating or integrating service management tools of its customers or suppliers.
- The administrators can also configure any number of its customers and vendors seamlessly as it scales up by deploying workflows, templates specific to every tenant in minutes.
- The implementers can consider developing client or tenant specific customizations or integrations.
- The CXOs will be able to make tenant specific decisions, plan, budget etc. using consolidated real-time data on their fingertips.
- The managers can get role-based dashboards to manage their respective teams and customers.
- The agents can focus and resolve the tickets assigned to them based on their groups and permissions as the respective workflows/approvals get triggered based on customers.