SaaS Adapter Microservice

Software-as-a-Service as a mechanism for offering applications comes with a lot of advantages: customers do not need to install and maintain software, and the supplier has full control over usage and thus can very precisely charge for the offering. Often, a SaaS solution provides functionality that is ready to plug into a microservices architecture. However, the vendor control itself can create issues.

How do you balance the simplicity of using a common API with the difficulty of making changes when the dependent API changes?

SaaS providers try very hard to make integration as simple as possible, which makes it very tempting to just a simple JSON call in the place where the SaaS service is needed. However, when a particular SaaS application is used in multiple places, changing to an alternative solution or upgrading to newer editions of the vendor's API become problematic.

Therefore,

Wrap SaaS calls in a SaaS Adapter Microservice, so that the vendor-specific implementation details are hidden and centralized.

There is a tension between vendors and customers of SaaS solutions: a main reason for selecting a SaaS solution is flexibility. Customers can quickly integrate without having to do complex application installations, and good SaaS solutions scale as needed. Switching between SaaS providers is often easy, as vendors compete on, among others, ease of integration. This low friction is problematic for the vendor, because customers can walk away too easily and this makes the enterprise volatile. Therefore, vendors try to create lock-in by expanding their offering, enhancing existing interfaces with proprietary additions, etcetera.

When implementing a microservices architecture, it is important to control the amount of vendor lock-in. By wrapping the integration in a vendor-specific implementation behind a (hopefully) vender-independent interface and deploying this as a separate microservice, the touchpoints with the vendor's system can be controlled. It becomes easy to check the use and scope of proprietary extensions, and in the case that switching vendors becomes necessary or preferable, only the adapter microservice needs to be changed.