此外，数据平面插件的扩展性和安全性也得到了社区的广泛重视。从数据平面的角度看，Envoy 得到了包括 Google、IBM、Cisco、微软、阿里云等在内的企业共建以及主流云平台的采纳，进而成为事实标准。在 Envoy 为插件机制提供了良好的可扩展性的基础之上，目前业界正在探索如何用 WASM 技术实现各种插件之间的隔离，避免因为某一插件的软件缺陷，而导致整个数据平面不可用的情况。WASM 技术的优势除了提供沙箱功能之外，还能支持多编程语言，从而最大程度地让掌握不同编程语言的开发者可以使用自己所熟悉的技能去扩展 Envoy 的能力。

在安全性方面，Service Mesh 非常利于实现云原生时代的零信任架构，包括 Pod Identity、基于 mTLS 的链路层加密、RBAC（Role Based Access Control，基于角色的访问控制）、基于 Identity 的微隔离环境（动态选取一组节点组成安全域）等内容。

4.6.3 应用场景案例：阿里巴巴 Service Mesh 实践

阿里巴巴坚信 Service Mesh 是未来的关键技术，因而坚定地对其进行探索，结合“借力开源、反哺开源”的发展策略，选择基于开源的 Istio 和 Envoy 不断进行技术延展。新技术的发展不可避免地需要应对过去发展时所积累的历史包袱。为此，阿里巴巴在 Service Mesh 的探索之路上经历了不同的发展阶段，并采用了不同的技术架构。长远来看，阿里巴巴希望借助大规模场景持续打磨 Service Mesh 技术，在实现反哺开源的同时，用开源、开放的标准技术取代阿里巴巴内部的私有技术，最终用一套开源技术服务阿里巴巴与蚂蚁集团和阿里云客户，实现“三位一体”。

在相当长的一段时间内，新技术的不成熟很可能是一种常态，关键在于如何实现新技术架构的平滑演进，避免出现推倒重来的情况。基于这一方面的考量，阿里巴巴对 Service Mesh 的实践经历了“起步”“三位一体”和“规模化落地”三大阶段，并在不同阶段采用了不同的软件架构或部署方案，如图 4-20 所示。

在起步阶段，Istio 控制平面的 Pilot 组件会存放在一个单独的容器中，并当作一个独立的进程和 Sidecar（指 Envoy）部署在一个 Pod 中。这种方式可以将开源的 Envoy 和 Pilot 的改动降到最小，从而加速它们的落地，也便于我们基于开源的版本做能力增强和反哺。这一方案的缺点在于，每个 Pod 中都包含一个 Pilot 进程，这会增大应用所在机器上的资源消耗。但在服务规模并不大的情形下，增加的资源消耗相对较小，此时可以忽视这一缺点。