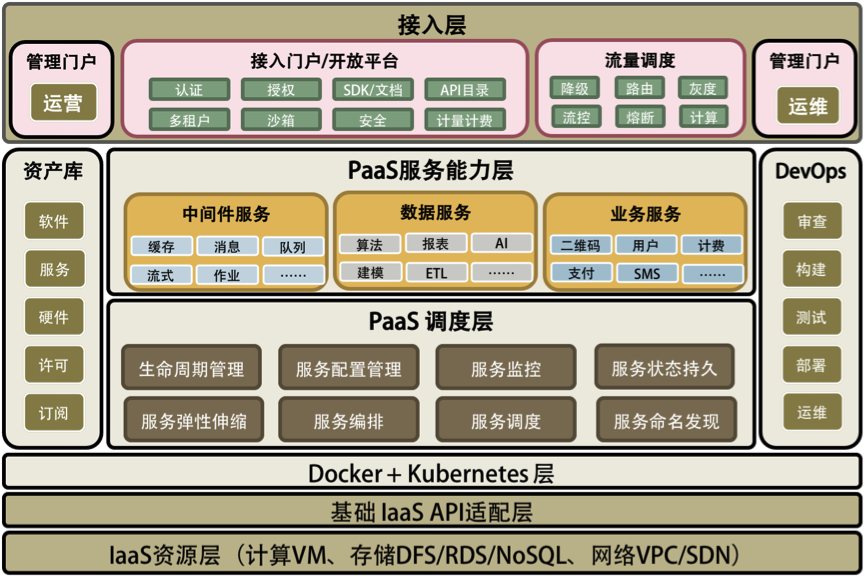
PAAS平台的本质

1.服务化事PaaS的本质，软件模块重用，服务治理，对外提供能力事PaaS的本质

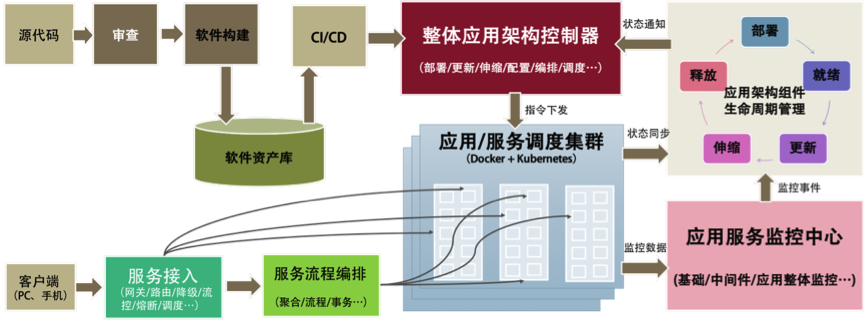
2.分布式事PaaS的根本特性，多租户隔离、高可用、服务编排是PaaS的基本特性

3.自动化事PaaS的灵魂，自动化部署安装运维，自动化伸缩调度是PaaS的关键



PaaS平台包含能力

1. PaaS调度层，主要是 PaaS 的自动化和分布式对于高可用高性能的管理
2. PaaS能力服务层，要是 PaaS 真正提供给用户的服务和能力
3. PaaS的流量调度，主要是与流量调度相关的东西，包括对高并发的管理。
4. PaaS的运营管理，软件资源库、软件接入、认证和开放平台门户。
5. PaaS的运维管理，主要是 DevOps 相关的东西。



分布式系统存在的问题

* 分布式系统的硬件故障发生率更高，故障发生是常态，需要尽可能地将运维流程自动化。
* 需要良好地设计服务，避免某服务的单点故障对依赖它的其他服务造成大面积影响。
* 为了容量的可伸缩性，服务的拆分、自治和无状态变得更加重要，可能需要对老的软件逻辑做大的修改。
* 老的服务可能是异构的，此时需要让它们使用标准的协议，以便可以被调度、编排，且互相之间可以通信。
* 服务软件故障的处理也变得复杂，需要优化的流程，以加快故障的恢复。
* 了管理各个服务的容量，让分布式系统发挥出最佳性能，需要有流量调度技术。
* 分布式存储会让事务处理变得复杂；在事务遇到故障无法被自动恢复的情况下，手动恢复流程也会变得复杂。
* 测试和查错的复杂度增大。
* 系统的吞吐量会变大，但响应时间会变长。

分布式系统问题的解决方案

* 需要有完善的监控系统，以便对服务运行状态有全面的了解。设计服务时要分析其依赖链；当非关键服务故障时，其他服务要自动降级功能，避免调用该服务。
* 重构老的软件，使其能被服务化；可以参考 SOA 和微服务的设计方式，目标是微服务化；使用 Docker 和 Kubernetes 来调度服务。
* 为老的服务编写接口逻辑来使用标准协议，或在必要时重构老的服务以使得它们有这些功能。自动构建服务的依赖地图，并引入好的处理流程，让团队能以最快速度定位和恢复故障，详见《故障处理最佳实践：应对故障》一文。
* 使用一个 API Gateway，它具备服务流向控制、流量控制和管理的功能。事务处理建议在存储层实现；根据业务需求，或者降级使用更简单、吞吐量更大的最终一致性方案，或者通过二阶段提交、Paxos、Raft、NWR 等方案之一，使用吞吐量小的强一致性方案。
* 通过更真实地模拟生产环境，乃至在生产环境中做灰度发布，从而增加测试强度；同时做充分的单元测试和集成测试以发现和消除缺陷；
* 最后，在服务故障发生时，相关的多个团队同时上线自查服务状态，以最快地定位故障原因。
* 通过异步调用来减少对短响应时间的依赖；对关键服务提供专属硬件资源，并优化软件逻辑以缩短响应时间。