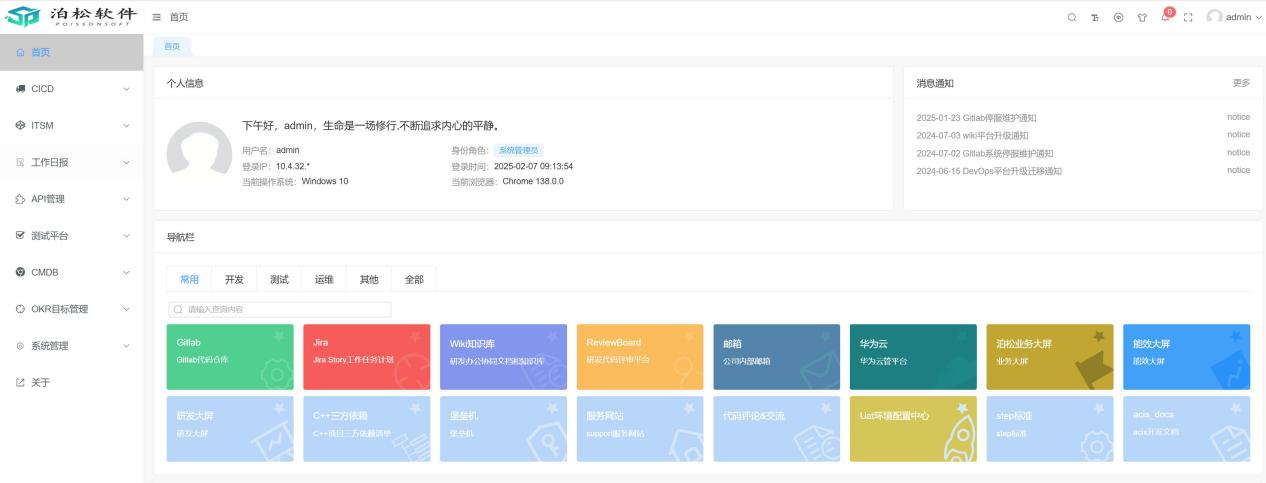
# Magicsim云仿真平台任务调度系统产品使用手册

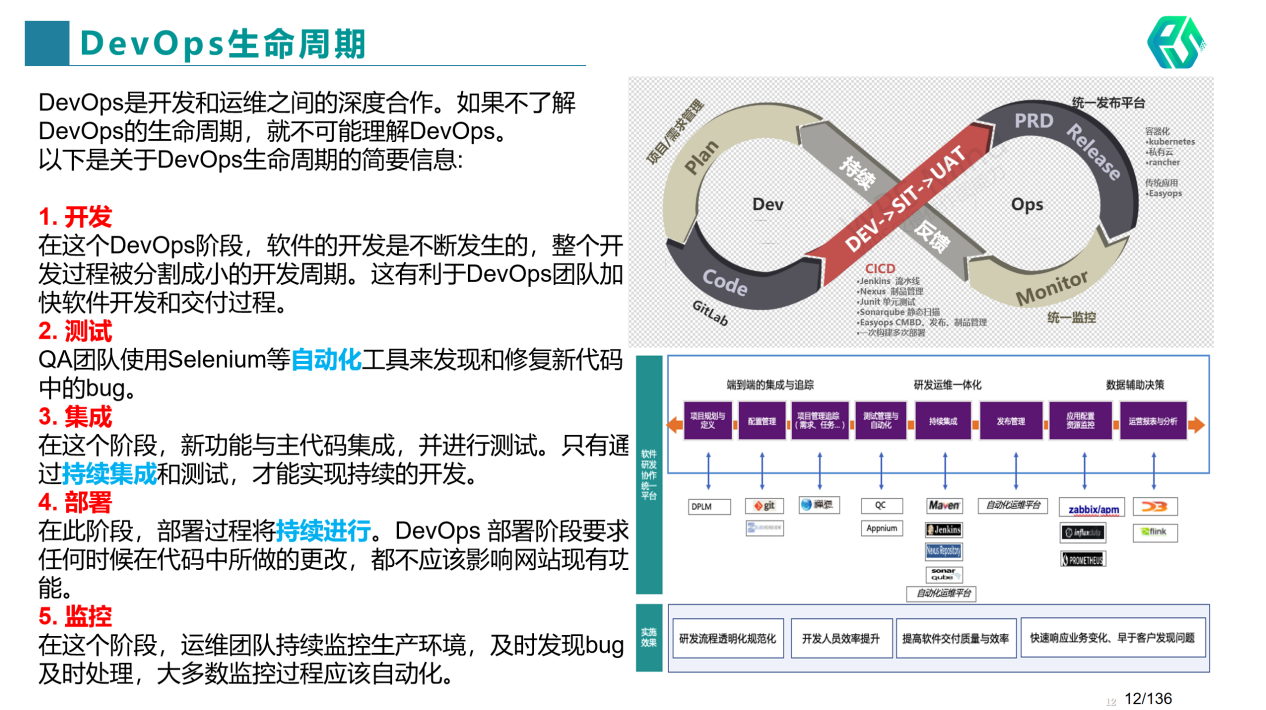


**Magicsim云仿真平台任务调度系统**包含CICD、ITSM、CMDB资产管理、ORK目标管理等基础功能，支持集成其他应用，如：Gitlab、K8S、Nacos、PM等。



**Magicsim云仿真平台任务调度系统介绍**

Magicsim云仿真平台任务调度系统是一款基于云架构的智能化研发工具链平台，为企业提供 DevOps、微服务治理、云管理的 IT 服务。平台是基于 Kubernetes 的容器编排和管理能力，整合 DevOps 工具链、微服务和持续集成及部署，来帮助企业实现敏捷化的应用交付和自动化的运营管理。



**devops平台**

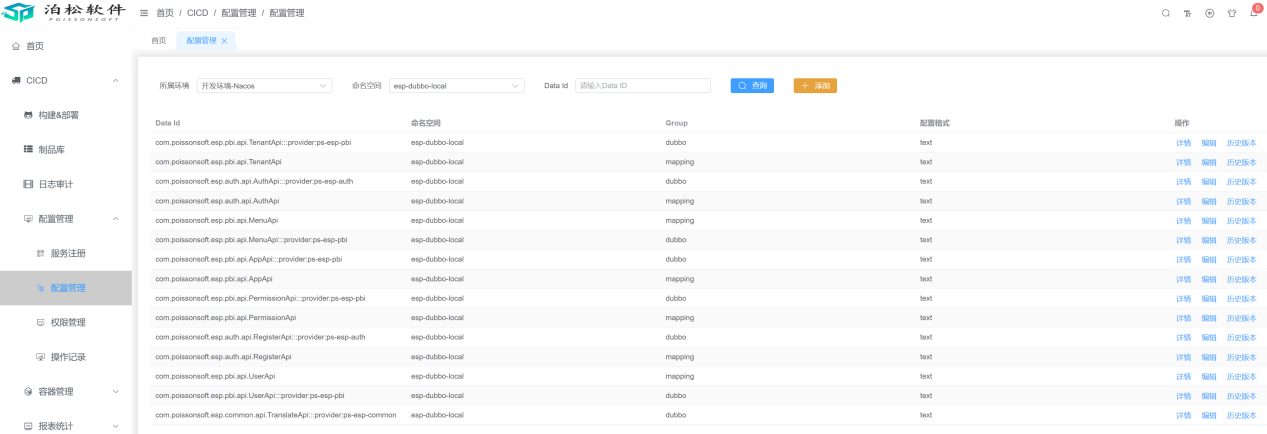
传统企业采用瀑布式应用开发模式的时候，从需求到交付的周期往往需要数月之久，在当前要求应用开发节奏越来越快的背景下，这个矛盾越来越严重。另外一方面应用传统烟囱式架构，伴随企业数字化转型过程中软件规模和种类的增多，各应用系统之间完全孤立，业务和数据孤岛的问题层出不穷，后期想进行业务和数据打通的难度和成本都会非常高。

为此，平台提供了企业级一站式 DevOps 解决方案，致力于成为数字企业的研发效能引擎！整个解决方案覆盖了项目应用管理、代码仓库管理、敏捷项目协作管理、CICD、自动化测试以及运维监控，有效提高企业的管理、研发、运维和运营效率，从而帮助企业在应对外界需求快速变化的情况下降低成本。

平台提供了遵循 GitFlow 规范的代码托管机制；采用 Pipeline 模式实现了代码编译、测试、打包等一系列的构建流程；采用 helm标准规范管理Kubernetes包，实现基础设施的可编程性，从而遵循基础设施即代码的设计理念，实现企业的复杂 IT 软件、项目以及分布式服务、微服务等的一键编排部署和准确高效的监控运维。

**微服务治理平台**

基于 Nacos 的 Mesh 方案完全透明支持 Dubbo 和 SpringCloud 等平台框架，无侵入、无开发语言限制的完整服务治理，整体功能上覆盖微服务总览、注册中心、配置中心 ServiceMesh 的同时，还从多维度立体覆盖了微服务的运维监控及云资源监控。



## DevOps 的工作原理：DevOps 生命周期

DevOps 生命周期（以线性方式描述时，也称为持续交付管道）是一系列自动化的迭代式开发流程（也称工作流程），在更大的自动化迭代式开发生命周期内执行，旨在优化高质量软件的快速交付。 工作流程的名称和数量因人而异，但通常可归结为以下六类：

* 规划（或称为构思） 在这个工作流程中，团队会根据划分了优先级的最终用户反馈和案例研究，以及来自所有内部利益相关方的意见和建议，确定下一个发行版中新特性和新功能的范围。 规划阶段的目标是通过生成在交付后可实现预期成果和价值的功能的待办工作列表，最大程度地提高产品的业务价值。
* 开发 这是编程步骤，开发人员根据待办工作列表中的用户情景和工作项，测试、编码和构建新功能和增强功能。 常用的实践组合包括测试驱动的开发 (TDD)、配对编程和同级代码评审等。 开发人员通常使用本地工作站来执行代码编写和测试的“内部循环”，然后再将代码发送到持续交付管道。
* 集成（或称为构建，或持续集成和持续交付 (CI/CD)） 如上所述，在这个工作流程中，新代码集成到现有代码库中，然后测试并打包到可执行文件中以进行部署。 常见的自动化活动包括将代码变更合并到“主”副本中，从源代码存储库中检出代码，自动执行编译、单元测试然后打包为可执行文件。 最佳实践是将 CI 阶段的输出存储在二进制存储库中，用于下一阶段。
* 部署（通常称为持续部署） 将（来自集成的）运行时构建输出部署到运行时环境 - 通常是执行运行时测试的开发环境，旨在确保质量、合规性与安全性。 如果发现错误或缺陷，开发人员有机会在任何最终用户看到问题之前拦截并修复任何问题。 通常存在开发、测试和生产环境，每个环境都需要逐步“严格”的质量关口。 部署到生产环境的最佳实践通常是首先部署到最终用户的子集，然后在产品趋于稳定后最终向所有用户部署。
* 运营 如果将功能交付到生产环境被描述为“第 1 天”， 那么这些功能在生产环境中运行时，即标志着“第 2 天”运营的开始。 监控功能的性能、行为和可用性可确保这些功能为最终用户增添价值。 运营旨在确保功能平稳运行，不会发生服务中断 - 必须确保网络、存储、平台、计算和安全资源一切正常！ 如果出错，运营可确保发现事件，提醒适当的人员，确定问题，并应用修复措施。
* 学习（有时称为持续反馈）  收集来自最终用户和客户对特性、功能、性能和业务价值的反馈，根据这些反馈，规划下一个发行版中的新功能和增强功能。 这还包括来自运营活动的任何学习和待办工作项，旨在帮助开发人员主动避免将来再次发生过去的任何事件。 这就是规划阶段的“总结”，我们“不断改进”！

这些工作流程之间还另有三个重要的持续工作流程：

* 持续测试：在典型的 DevOps 生命周期中，集成和部署之间有一个单独的“测试”阶段。 而 DevOps 更进一步，可以在各个工作流程中执行测试的某些要素，例如在规划中执行行为驱动的开发；在开发中执行单元测试与合同测试；在集成中执行静态代码扫描、CVE 扫描和 linting；在部署中执行烟雾测试、渗透测试、配置测试；在运营中执行混沌测试、合规性测试；在学习中执行 A/B 测试。 测试是一种强大的风险和漏洞发现方法，为 IT 提供接受、缓解或补救风险的机会。
* 安全性：瀑布方法和敏捷实现是在交付或部署后“附加”安全工作流程，而 DevOps 力求从一开始（规划）就融入安全措施，因为此时安全问题最容易解决，而且解决成本最低，然后在开发周期的其余阶段持续实施安全措施。 这种安全性方法称为左移（请参阅图 1 以便更轻松地理解这个概念）。 一些企业的左移工作不甚理想，这也导致了 DevSecOps 的兴起。
* 合规性：合规性（治理和风险）也在开发生命周期的早期和整个过程中得到最有效的满足。 受监管行业通常必须强制性满足特定级别的可观察性、可跟踪性以及可访问性，以表明自己如何在运行时运营环境中交付和管理功能。 这需要在持续交付管道和运行时环境中规划、制定、测试和执行策略。 合规性措施的可审计性对于向第三方审计机构证明合规性而言极其重要。

