混沌工程介绍与实践

> 文章地址：https://github.com/StabilityMan/StabilityGuide
> 作者：肖长军（穹谷，@xcaspar）

在分布式系统架构下，服务间的依赖日益复杂，很难评估单个服务故障对整个系统的影响，并且请求链路长，监控告警的不完善导致发现问题、定位问题难度增大，同时业务和技术迭代快，如何持续保障系统的稳定性和高可用性受到很大的挑战。我们知道发生故障的那一刻不是由你来选择的，而是那一刻来选择你，你能做的就是为之做好准备。所以构建稳定性系统很重要的一环是混沌工程，在可控范围或环境下，通过故障注入，来持续提升系统的稳定性和高可用能力。
本文会着重介绍什么是混沌工程，为什么需要混沌工程以及混沌工程相关工具与实践。如有遗漏或错误，欢迎补充指正。

目录

什么是混沌工程

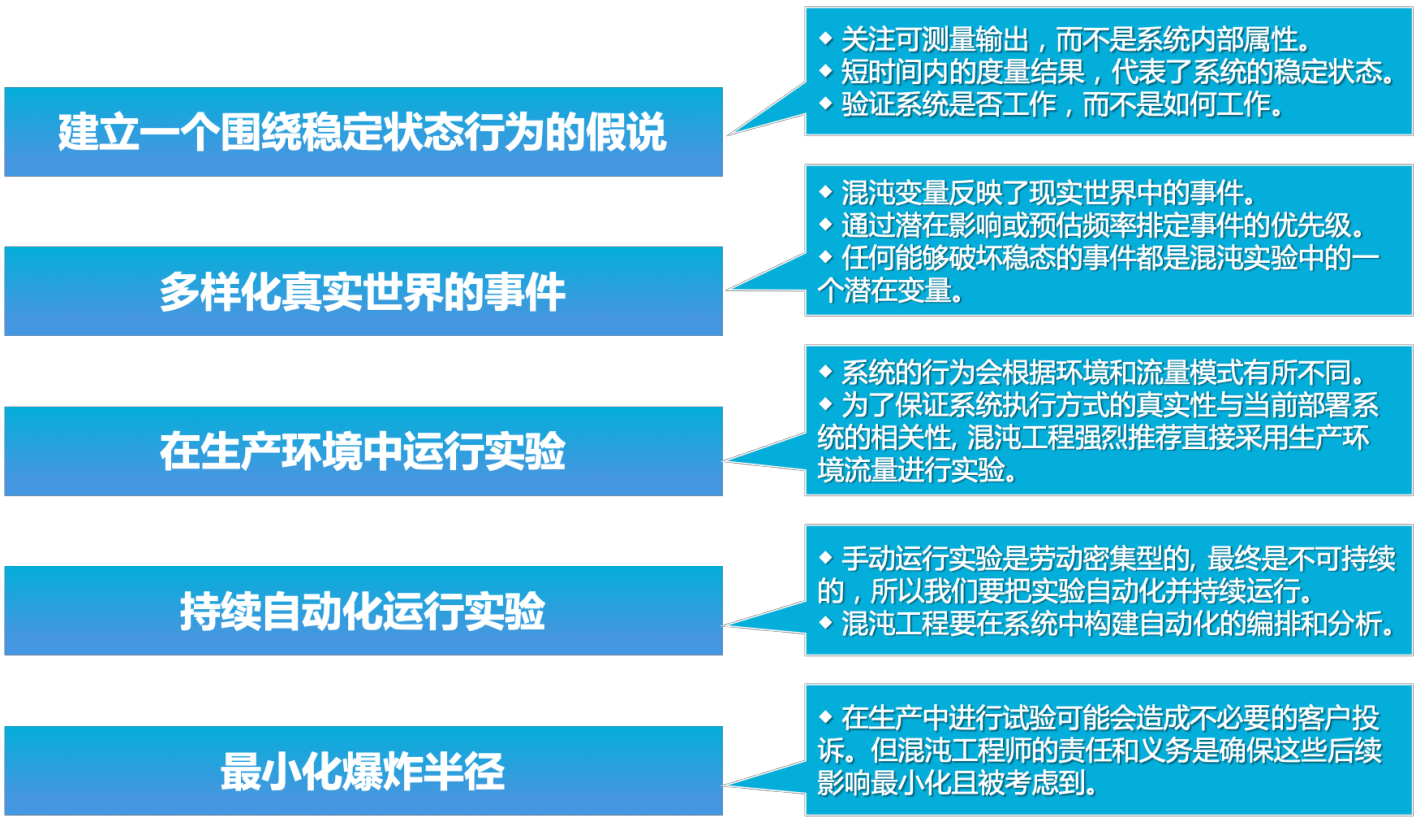
混沌工程是在 混沌工程理论 一文中提出，但在 2010 年 Netflix 从物理机基础设施迁移到 AWS 过程中，为保证 EC2 实例故障不会对业务造成影响，其团队开发出了杀 EC2 实例的工具，这也是混沌工程的雏形。在 2015 年社区发布《混沌工程理论》一文后，混沌工程开始快速发展。
混沌工程是在分布式系统上进行实验的学科，旨在提升系统容错性，建立系统抵御生产环境中发生不可预知问题的信心。”打不倒我的必使我强大“，尼采的这句话很好了诠释了混沌工程反脆弱的思想。

为什么需要混沌工程

分布式系统日益复杂，而且在系统逐渐云化的背景下，系统的稳定性受到很大的挑战。这里从四个角色来说明混沌工程的重要性。 - 对于架构师来说，可以验证系统架构的容错能力，比如验证现在提倡的面向失败设计的系统； - 对于开发和运维，可以提高故障的应急效率，实现故障告警、定位、恢复的有效和高效性。 - 对于测试来说，可以弥补传统测试方法留下的空白，之前的测试方法基本上是从用户的角度去做，而混沌工

程是从系统的角度进行测试，降低故障复发率。 - 对于产品和设计，通过混沌事件查看产品的表现，提升客户使用体验。所以说混沌工程面向的不仅仅是开发、测试，拥有最好的客户体验是每个人的目标 所以实施混沌工程，可以提早发现生产环境上的问题，并且可以以战养战，提升故障应急效率和可以使用体验，逐渐建设高可用的韧性系统。

混沌工程实施原则



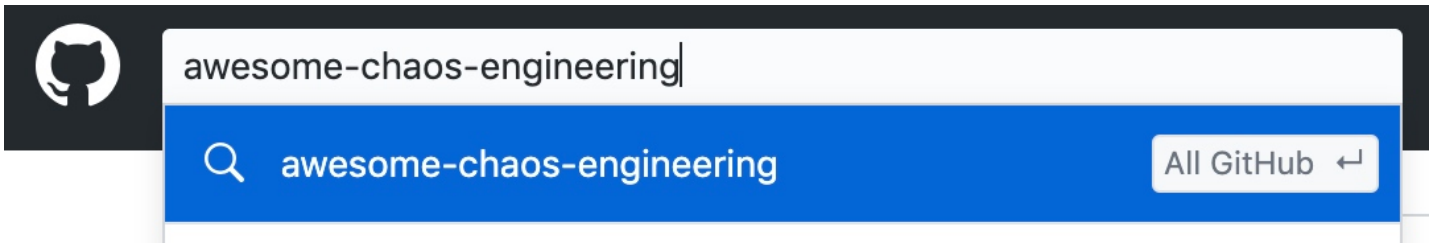
- 第一条：”建立一个围绕稳定状态行为的假说“，其包含两个含义，一个是定义能直接反应业务服务的监控指标，需要注意的是这里的监控指标并不是系统资源指标，比如CPU、内存等，这里的监控指标是能直接衡量系统服务质量的业务监控。举个例子，一个调用延迟故障，请求的 RT 会变长，对上层交易量造成下跌的影响，那么这里交易量就可以作为一个监控指标。这条原则的另一个含义是故障触发时，对系统行为作出假设以及监控指标的预期变化。
- 第二条指模拟生产环境中真实的或有理论依据的故障场景，比如依赖的服务调用延迟、超时、异常等。
- 第三条建议在生产环境中运行实验，但也不是说必须在生产环境中执行，只是实验环境越真实，混沌工程越有价值，但如果知道系统在某个故障场景下不具备容灾能力，不可以执行此混沌实验，避免资损发生。
- 第四条，持续的执行才能持续的降低故障复发率和提前发现故障，所以需要持续的自动化运行试验。
- 最后一个，混沌工程很重要的一点是控制爆炸半径，也就是试验影响面，防止预期外的资损发生，可以通过环境隔离或者故障注入工具提供的配置粒度来控制。

混沌工程实施步骤

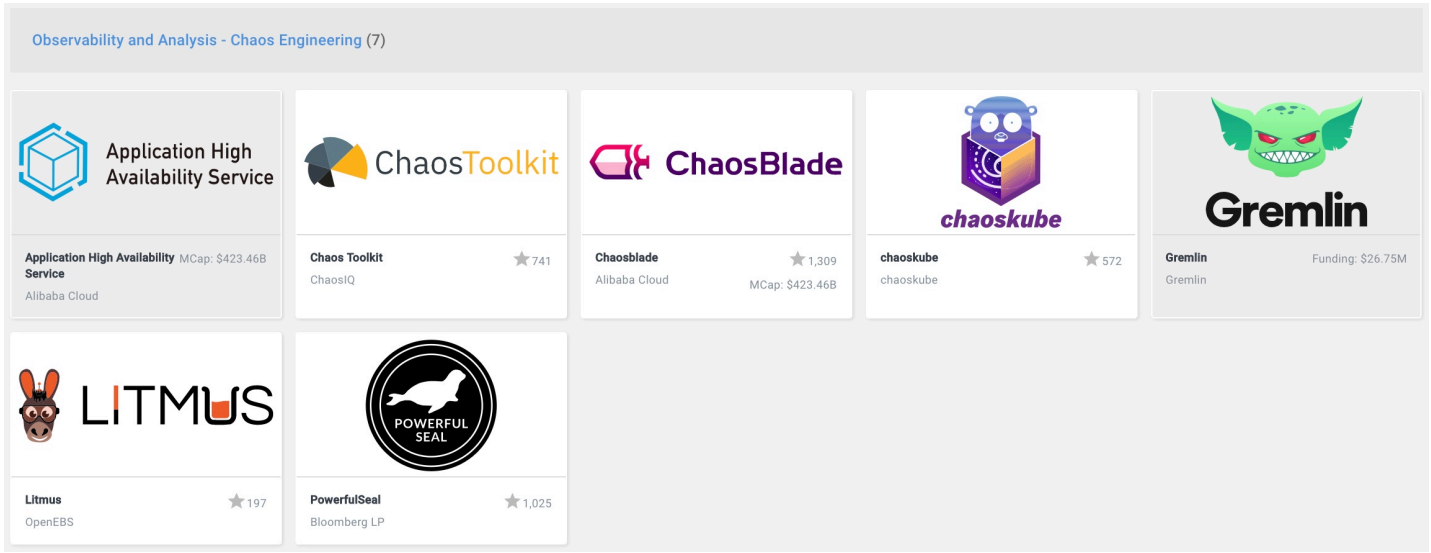
- 制订混沌实验计划

- 定义系统稳态指标
- 做出系统容错行为假设
- 执行混沌实验
- 检查系统稳态指标
- 记录&恢复混沌实验
- 修复发现的问题
- 自动化持续进行验证

推荐工具产品



大家可以从工具的场景丰富度、类型、易用性等方面来选择一款合适的工具，awesome-chaos-engineering Github 项目收纳了一些开源的混沌工程工具，在 CNCF Landscape 中混沌工程作为单独的一个领域存在，并且收纳了一些主流的工具，包含阿里巴巴开源的 ChaosBlade 工具和 AHAS 阿里云产品。



下文重点介绍 ChaosBlade 及其相关实践。

ChaosBlade

ChaosBlade 中文名混沌之刃，是一款混沌实验实施工具，支持丰富的实验场景，比如应用、容器、基础资源等。工具使用简单，扩展方便，其遵循社区提出的混沌实验模型。Github 地址：https://github.com/chaosblade-io/chaosblade

功能和特点

场景丰富度高 ChaosBlade 支持的混沌实验场景不仅覆盖基础资源，如 CPU 满载、磁盘 IO 高、网络延迟

等，还包括运行在 JVM 上的应用实验场景，如 Dubbo 调用超时和调用异常、指定方法延迟或抛异常以及返回特定值等，同时涉及容器相关的实验，如杀容器、杀 Pod。后续会持续的增加实验场景。

使用简洁，易于理解 ChaosBlade 通过 CLI 方式执行，具有友好的命令提示功能，可以简单快速的上手使用。命令的书写遵循阿里巴巴集团内多年故障测试和演练实践抽象出的故障注入模型，层次清晰，易于阅读和理解，降低了混沌工程实施的门槛。

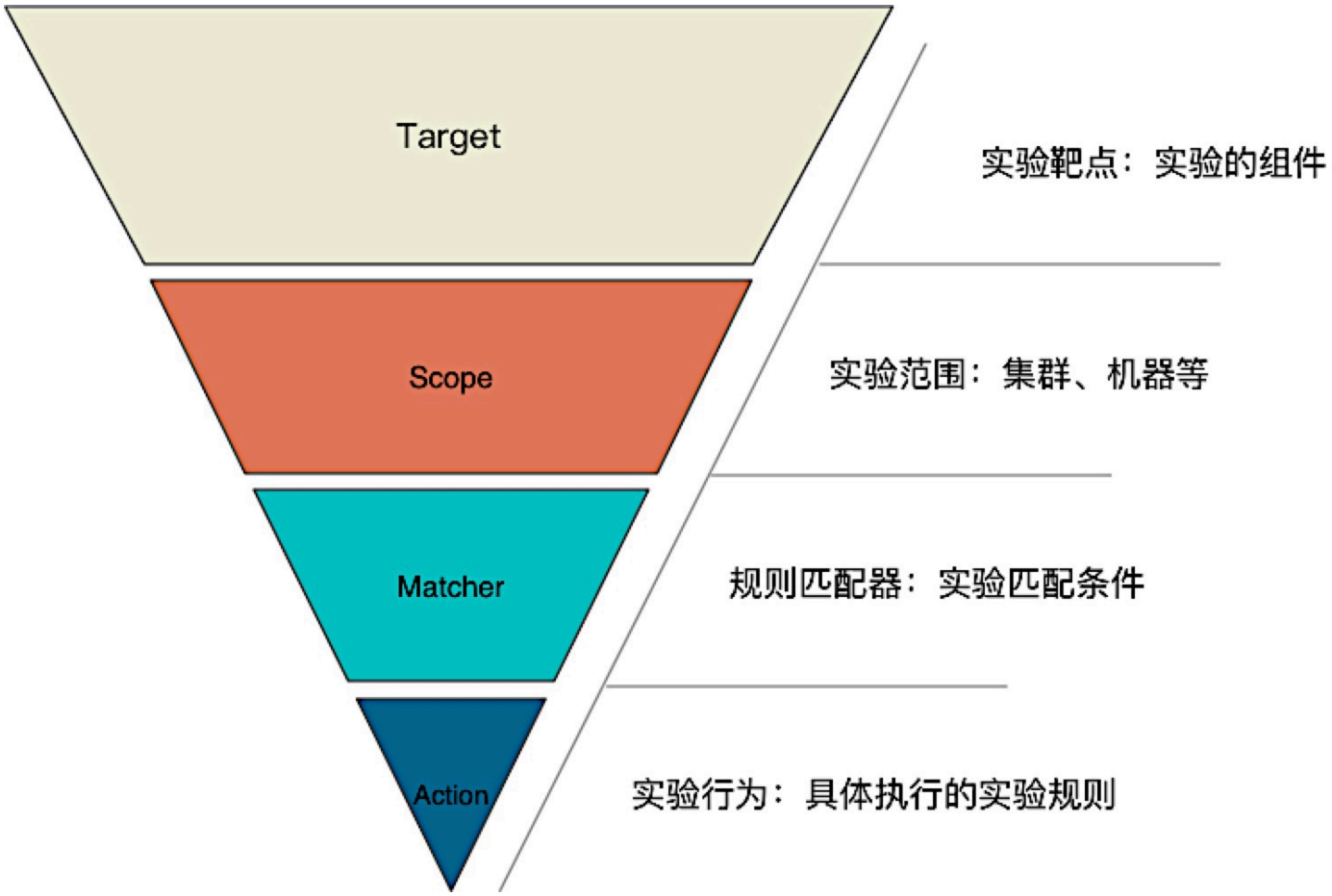
动态加载，无侵入 ChaosBlade采用动态故障注入的方式，执行混沌实验时用户系统不需要做任何系统改造或发布，开箱即用。

场景扩展方便 所有的 ChaosBlade 实验执行器同样遵循上述提到的故障注入模型，使实验场景模型统一，便于开发和维护。模型本身通俗易懂，学习成本低，可以依据模型方便快捷的扩展更多的混沌实验场景。

使用方式

在 ChaosBlade Release 页面下载最新版本的包，解压即用。如创建一个 CPU 满载实验，命令为：`blade create cpu fullload` 具体使用方式可详见：ChaosBlade 新手指南

混沌实验模型

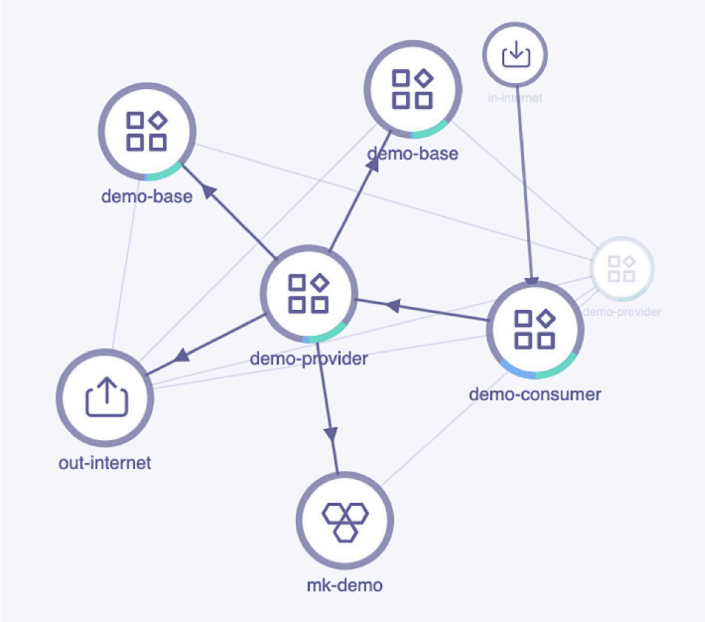


该模型分四次，层层递进，很清晰的表达出对什么组件做实验，实验范围是什么，实验触发的匹配规则有哪些，执行什么实验。该模型简洁、通用，语言领域无关、易于实现。阿里集团内的 C++、NodeJS、Dart 应用以及容器平台的实验场景都基于此模型实现。此模型具有很重要的意义，依据此模型可以更精准的描述、

更好的理解、更方便沉淀实验场景以及发掘更多的场景。依据此模型实现的工具更加规范、简洁。实验模型介绍可详见：混沌实验模型介绍。

混沌工程实践案例

案例 Demo 拓扑图



拓扑图来自于-阿里云 AHAS 产品架构感知功能

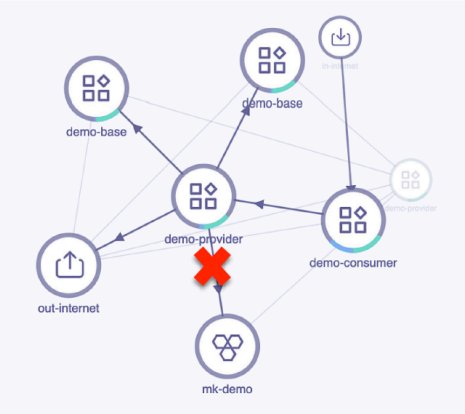
此拓扑图来自于阿里云 AHAS 产品架构感知功能，可自动感知架构拓扑，并且可以展示进程、网络、节点等数据。这个分布式服务 Demo 分三级调用，consumer 调用 provider，provider 调用 base，同时 provider 还调用 mk-demo 数据库，provider 和 base 服务具有两个实例，在 AHAS 架构拓扑图上，我们点击一个实例节点，可以到非常清晰的调用关系。我们后面结合这个 Demo 去讲解实践。

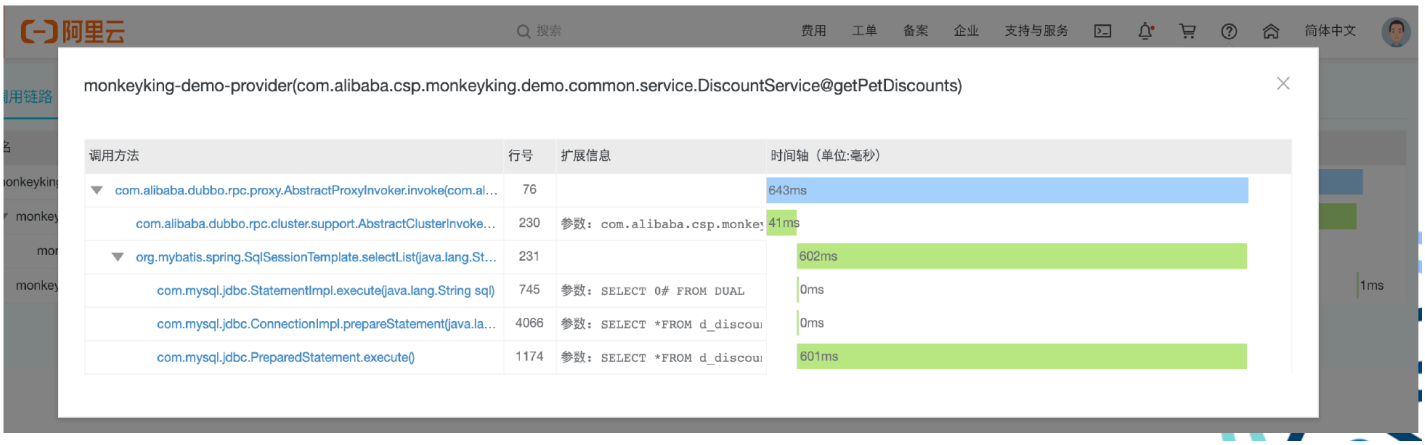
验证监控告警

案例一：验证监控告警

场景：数据库调用延迟
监控指标：慢 SQL 数，告警信息
期望假设：慢 SQL 数增加，钉钉群收到慢 SQL 告警
混沌实验：对 demo-provider 注入调用 mk-demo 数据库延迟故障
监控指标：慢 SQL 数增加，钉钉群收到告警，符合预期
问题排查：通过 ARMS 慢调用链路排查

备注：以上告警和链路跟踪来自于阿里云 *ARMS* 产品



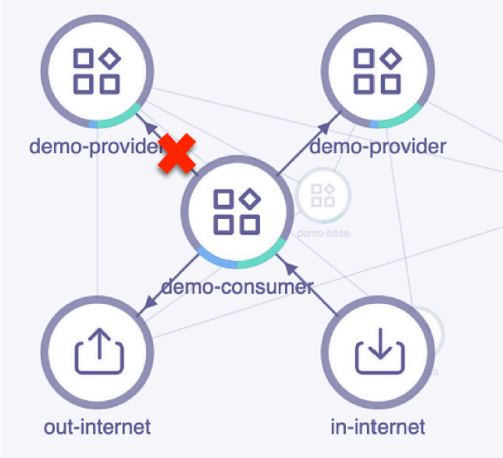


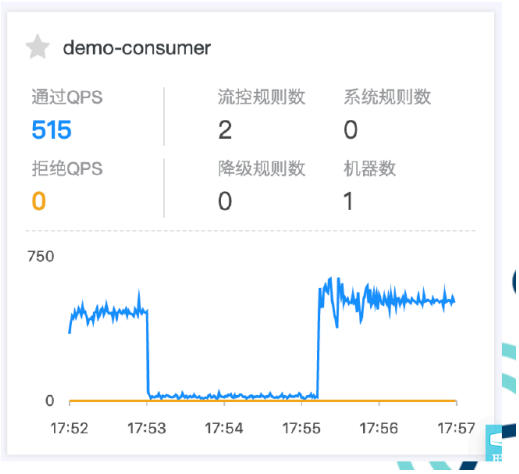
案例一，我们验证系统的监控告警性有效性。按照前面提到的混沌工程实施步骤，那么这个案例执行的实验场景是数据库调用延迟，我们先定义监控指标：慢 SQL 数和告警信息，做出期望假设：慢 SQL 数增加，钉钉群收到慢 SQL 告警。接下来执行实验。我们直接使用 ChaosBlade 工具执行，可以看下左下角，我们对 demo-provider 注入调用 mysql 查询时，若数据库是 demo 且表名是 d_discount，则对 50% 的查询操作延迟 600 毫秒。我们使用阿里云产品 ARMS 做监控告警。大家可以看到，当执行完混沌实验后，很快钉钉群里就收到了报警。所以我们对比下之前定义的监控指标，是符合预期的。但需要注意的是这次符合预期并不代表以后也符合，所以需要通过混沌工程持续性的验证。出现慢 SQL，可通过 ARMS 的 链路追踪 来排查定位，可以很清楚的看出哪条语句执行慢。

案例二

案例二：验证异常实例隔离

场景：下游一个服务实例出现延迟
监控指标：QPS，稳态在 510 左右
容错假设：QPS 会出现几秒的下跌，但很快恢复；系统会自动隔离或下线出问题服务实例，防止请求路由到此实例
混沌实验：对 demo-provider-1 注入延迟故障
监控指标：QPS 下跌到 40，不会自动恢复，不符合预期
业务方应急处理：下线出问题的实例，QPS 恢复
问题记录：系统缺失服务质量检查，不能对异常服务实例做隔离





前面讲了一个符合预期的案例，我们再来看一个不符合预期的。此案例是验证系统异常实例隔离的能力，我们的 Demo 中 consumer 调用 provider 服务，provider 服务具有两个实例，我们对其中一个注入延迟故障，监控指标是 consumer 的 QPS，稳态在 510 左右。我们做的容错假设是系统会自动隔离或下线出问题的服务实例，防止请求路由的此实例，所有 QPS 会有短暂的下跌，但很快会恢复。这个案例，我们使用阿里云 AHAS 混沌实验平台来执行，我们对 demo-provider-1 注入延迟故障，基于此平台可以很方便的执行混沌实验。执行混沌实验后，QPS 下跌到 40 左右，很长时间没有自动恢复，所以不符合预期，我们通过人工的方式对该异常的实例做下线处理，很快就看到，consumer 的 QPS 恢复正常。所以我们通过混沌工程发现了系统问题，我们后面需要做就是记录此问题，并且推动修复，后续做持续性的验证。

相关文章交流群

- ChaosBlade 钉钉讨论群号：23177705
- 相关资料：awesome-chaosblade 项目 后续的分享和讨论都会在上述钉钉群中进行，欢迎加入。我们还会不定期的给 ChaosBlade 社区贡献者发放纪念品，欢迎加入到 ChaosBlade 社区中，加入方式：star、issue、pr 等均可。

加入我们

【稳定大于一切】打造国内稳定性领域知识库，**让无法解决的问题少一点点，让世界的确定性多一点点。**

- GitHub 地址
- 钉钉群号：23179349
- 如果阅读本文有所收获，欢迎分享给身边的朋友，期待更多同学的加入！