ALIBABA CLOUD

阿里云

应用高可用服务

故障演练

文档版本：20201014

阿里云

法律声明

阿里云提醒您在阅读或使用本文档之前仔细阅读、充分理解本法律声明各条款的内容。如果您阅读或使用本文档，您的阅读或使用行为将被视为对本声明全部内容的认可。

1. 您应当通过阿里云网站或阿里云提供的其他授权通道下载、获取本文档，且仅能用于自身的合法合规的业务活动。本文档的内容视为阿里云的保密信息，您应当严格遵守保密义务；未经阿里云事先书面同意，您不得向任何第三方披露本手册内容或提供给任何第三方使用。

2. 未经阿里云事先书面许可，任何单位、公司或个人不得擅自摘抄、翻译、复制本文档内容的部分或全部，不得以任何方式或途径进行传播和宣传。

3. 由于产品版本升级、调整或其他原因，本文档内容有可能变更。阿里云保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在阿里云授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过阿里云授权渠道下载、获取最新版的用户文档。

4. 本文档仅作为用户使用阿里云产品及服务的参考性指引，阿里云以产品及服务的“现状”、“有缺陷”和“当前功能”的状态提供本文档。阿里云在现有技术的基础上尽最大努力提供相应的介绍及操作指引，但阿里云在此明确声明对本文档内容的准确性、完整性、适用性、可靠性等不作任何明示或暗示的保证。任何单位、公司或个人因为下载、使用或信赖本文档而发生任何差错或经济损失的，阿里云不承担任何法律责任。在任何情况下，阿里云均不对任何间接性、后果性、惩戒性、偶然性、特殊性或刑罚性的损害，包括用户使用或信赖本文档而遭受的利润损失，承担责任（即使阿里云已被告知该等损失的可能性）。

5. 阿里云网站上所有内容，包括但不限于著作、产品、图片、档案、资讯、资料、网站架构、网站画面的安排、网页设计，均由阿里云和/或其关联公司依法拥有其知识产权，包括但不限于商标权、专利权、著作权、商业秘密等。非经阿里云和/或其关联公司书面同意，任何人不得擅自使用、修改、复制、公开传播、改变、散布、发行或公开发表阿里云网站、产品程序或内容。此外，未经阿里云事先书面同意，任何人不得为了任何营销、广告、促销或其他目的使用、公布或复制阿里云的名称（包括但不限于单独为或以组合形式包含“阿里云”、“Aliyun”、“万网”等阿里云和/或其关联公司品牌，上述品牌的附属标志及图案或任何类似公司名称、商号、商标、产品或服务名称、域名、图案标示、标志、标识或通过特定描述使第三方能够识别阿里云和/或其关联公司）。

6. 如若发现本文档存在任何错误，请与阿里云取得直接联系。

通用约定

格式	说明	样例
危险	该类警示信息将导致系统重大变更甚至故障，或者导致人身伤害等结果。	危险 重置操作将丢失用户配置数据。
警告	该类警示信息可能会导致系统重大变更甚至故障，或者导致人身伤害等结果。	警告 重启操作将导致业务中断，恢复业务时间约十分钟。
注意	用于警示信息、补充说明等，是用户必须了解的内容。	注意 权重设置为0，该服务器不会再接受新请求。
说明	用于补充说明、最佳实践、窍门等，不是用户必须了解的内容。	说明 您也可以通过按Ctrl+A选中全部文件。
>	多级菜单递进。	单击**设置**> **网络**> **设置网络类型**。
粗体	表示按键、菜单、页面名称等UI元素。	在**结果确认**页面，单击**确定**。
`Courier字体`	命令或代码。	执行 `cd /d C:/window` 命令，进入Windows系统文件夹。
斜体	表示参数、变量。	`bae log list --instanceid` *Instance_ID*
[] 或者 [a|b]	表示可选项，至多选择一个。	`ipconfig [-all|-t]`
{} 或者 {a|b}	表示必选项，至多选择一个。	`switch {active|stand}`

目录

1.故障演练概述

故障演练是一款遵循混沌工程实验原理并融合了阿里巴巴内部实践的产品，提供丰富故障场景实现，能够帮助分布式系统提升容错性和可恢复性。

流程

故障演练建立了一套标准的演练流程，包含准备阶段、执行阶段、检查阶段和恢复阶段。通过四阶段的流程，覆盖用户从计划到还原的完整演练过程，并通过可视化的方式清晰的呈现给用户。

故障演练

适用场景

故障演练可适用于以下典型场景：

- **衡量微服务的容错能力**
  通过模拟调用延迟、服务不可用、机器资源满载等，查看发生故障的节点或实例是否被自动隔离、下线，流量调度是否正确，预案是否有效，同时观察系统整体的QPS或RT是否受影响。在此基础上可以缓慢增加故障节点范围，验证上游服务限流降级、熔断等是否有效。最终故障节点增加到请求服务超时，估算系统容错红线，衡量系统容错能力。
- **验证容器编排配置是否合理**
  通过模拟杀服务Pod、杀节点、增大Pod资源负载，观察系统服务可用性，验证副本配置、资源限制配置以及Pod下部署的容器是否合理。
- **测试PaaS层是否健壮**
  通过模拟上层资源负载，验证调度系统的有效性；模拟依赖的分布式存储不可用，验证系统的容错能力；模拟调度节点不可用，测试调度任务是否自动迁移到可用节点；模拟主备节点故障，测试主备切换是否正常。
- **验证监控告警的时效性**
  通过对系统注入故障，验证监控指标是否准确，监控维度是否完善，告警阈值是否合理，告警是否快速，告警接收人是否正确，通知渠道是否可用等，提升监控告警的准确和时效性。
- **定位与解决问题的应急能力**
  通过故障突袭，随机对系统注入故障，考察相关人员对问题的应急能力，以及问题上报、处理流程是否合理，达到以战养战，锻炼人定位与解决问题的能力。

故障演练与AHAS服务体系

故障演练作为AHAS的一部分，与AHAS其他功能组成了一套完善的高可用保障服务，可以帮助用户实现包括架构、业务、人员的全面高可用提升。故障演练在其中承担着问题发现、问题验证、高可用经验沉淀的作用。

2.产品架构

本文向您介绍故障演练的产品架构，以下简称故障演练为 AHAS Chaos。
产品架构如下图所示：

AHAS Agent

AHAS Agent 安装在指定的目标机器上，用来执行服务端下发的故障注入命令以及采集演练相关的必要信息，比如 CPU、内存占用等，主要有以下特点：

- 快捷高效
  - 支持在控制台一键安装 AHAS Agent，如果是公网则需要用户手动安装。
- 安全可靠
  - Agent 本身对应用无入侵。除了演练的故障场景需要作用于应用进程的情况例外，例如 Java 类的故障场景。
  - 网络异常可自动重试。
  - 数据传输通道严格加密，保证数据安全。
- 完善的自我保护
  - 实时监控系统资源，不占用过多带宽。
  - 非用户手动卸载情况下的异常退出可以自我保活，保证 AHAS Agent 在主进程消失的情况下， 可以自动重启。

ChaosBlade

ChaosBlade 是 AHAS Agent 的核心组件，用来解析、校验和执行服务端下发的故障指令，并且已正式对外开源，具有以下特点：

- 简单易用
  - 清晰易懂的混沌工程实验模型，易于理解。
  - 自带完善的命令行工具，方便本地调试。
- 功能稳定
  - 在阿里内部被广泛使用，已通过真实环境的严格考验。
  - 活跃的开源社区确保了功能的不断迭代和缺陷的快速修复。
- 场景丰富
  - 基础资源：CPU、内存、网络、磁盘、进程等实验场景。
  - Java 应用：数据库、缓存、消息、JVM 本身、微服务等，还可以指定任意类方法注入各种复杂的实验场景。
  - 云原生场景：
    - 节点上 CPU、内存、网络、磁盘、进程实验场景
    - Pod 网络和 Pod 本身实验场景例如删除 Pod
    - 容器的实验场景例如删除容器、容器内 CPU、内存、网络、磁盘、进程等实验场景

AHAS Service

AHAS Chaos 的后端服务，用来下发演练命令，管理演练数据，主要的功能特点如下：

- 功能完善

- 通过流程编排来组织故障演练场景。
- 资源管理帮助您更方便的了解演练的机器资源使用情况。

- 稳定可靠
  - 分布式的部署方式和完善的监控系统保证了演练功能的高可用性。
  - 在服务不可用情况下，您可以通过手动执行命令来恢复演练。

3.产品优势

3.1. 功能优势

本文向您介绍故障演练产品的功能优势，故障演练以下简称 AHAS Chaos。

灵活的流程编排

- AHAS Chaos 将故障演练的环节分为了准备、注入、检查以及恢复四个阶段，每个阶段除了系统初始化完成的必要节点之外，您也可以根据需要添加所需的流程节点。
- AHAS Chaos 支持一次演练包含多个定义的故障场景，同时您可以定制这些场景的运行方式，选择依次进行故障注入或同时注入多个场景，通过不同的策略配置来达到不同的故障注入效果。

丰富的故障场景

丰富的故障场景也是 AHAS Chaos 的一大特色，包括了以下场景：

- 常见的基础设施资源例如CPU、内存、磁盘等。
- 应用级别的故障注入，目前只支持 Java 应用，后续将陆续推出对于 NodeJs 和 C++ 的应用故障注入。
- 云原生领域的演练场景。

无论您是需要设置集群级别的大规模故障还是应用级别的请求级别细粒度故障，都可以在 AHAS Chaos 找到适合的场景，下图是 AHAS Chaos 提供的部分故障场景。

多样的专家经验

AHAS Chaos 将阿里内部多年的故障演练经验浓缩成了专家经验，专家经验具有以下优点：

- 专家经验都来自于阿里内部经常演练的场景，保证了演练场景的真实性以及实用性。
- 专家经验不但包括了可执行的演练流程，而且还描述了专家经验试图解决的问题以及针对的系统架构弱点。
- 专家经验极大的提升了演练创建的效率，您可以基于专家经验配置好的流程一键生成自己的演练。

安全的演练防护

在保护您的演练安全性上 AHAS Chaos 也做了非常多的防护措施。

- 在演练的任意一个环节，您都可以随时终止演练，每一个终止操作都会自动恢复注入的场景。
- 您可以一键终止所有正在运行当中的演练。
- 您可以配置演练自动的恢复时间，防止因演练时间过长而忘记恢复演练引发的不必要问题。
- 您可以通过全局恢复功能来配置自动恢复的策略，当某个指标符合某个要求时自动恢复演练。

深度集成的阿里云产品

AHAS Chaos 和阿里云的许多产品如 ARMS、SLS、EDAS、OTS 以及架构感知服务等做了深度集成，通过授权您可以实现以下功能。

- 对依赖的阿里云组件进行故障注入。
- 基于接入的阿里云监控系统数据如 ARMS 来丰富演练检查和恢复的手段。
- 通过 RAM 服务来授权不同账号的演练权限，提升演练的安全性。

3.2. 围绕混沌工程的平台实践

本文主要介绍 AHAS Chaos 是如何围绕混沌工程来打造故障演练服务，您可以了解到混沌工程的基本知识和 AHAS Chaos 的优势。

混沌工程和故障演练

首先您需要了解混沌工程和故障演练的关系。以下是混沌工程官方定义：混沌工程是在分布式系统上进行实验的学科，目的是建立对系统抵御生产环境中失控条件的能力以及信心。因此混沌工程是一门学科，它提供了基本的理论指导，而故障演练是混沌工程的具体实践，通过向目标系统注入真实可能发生的故障来考量系统的稳定性。

混沌工程和 AHAS Chaos

AHAS Chaos 是以混沌工程为理论指导的故障演练平台，目标是成为混沌工程领域的最佳平台实践，因此在功能设计上紧紧围绕了混沌工程的基本原则。

原则1 建立一个围绕稳定状态行为的假说

- 稳定状态规定了在演练开始前需要定义好观测的指标，这些指标需要可以真实的反应系统的健康程度，并且您还需要知道当前的故障场景对指标可能造成的影响。
- 假说就是在演练之前您需要知道演练可能会对系统有什么影响，并且通过演练去验证。

以上两点强调了系统指标的重要性，在实验期间您需要持续监控这些指标，并且根据指标做出相应的措施，比如进一步扩大演练范围或者停止演练等。

AHAS Chaos 提供了全局监控配置，可以帮助您在演练期间实时监测系统状态，及时处理预期之外的情况。AHAS Chaos 除了提供一些基础的系统指标如CPU、内存、网络、磁盘之外，还集成了阿里云的 ARMS，如果您的应用已经接入 ARMS 就可以在 AHAS Chaos 里直接观测到对应的指标，后续也将会集成更多的监控系统，功能如下图所示。详情请参见创建演练。

原则2 多样化真实世界的事件

在故障演练开始之前，您需要思考以下问题，在现实生活中，系统已经发生过哪些问题？可能会发生什么样的新问题？有哪些问题是正在解决中的？这些问题可以帮助您在设计演练流程时选择和系统业务有关的更优场景。

为了帮助您选择合适的演练场景，AHAS Chaos 在功能设计上做了以下两点：

- 丰富的演练场景
  首先在全面性上 AHAS Chaos 的演练场景覆盖了从基础设施、应用层到容器服务、云原生等多维度的领域，在场景真实性上 AHAS Chaos 也做了严格的筛选，无论是基础设施还是应用层针对的都是通用的组件，是被大多数的系统和业务所依赖的，比如 CPU、磁盘、网络、MySql 等，此外 AHAS Chaos 还提供了阿里云组件的故障注入能力。
- 演练经验
  如果说演练场景是基于组件维度的，那么演练经验就向您提供了基于架构维度和故障效果维度的场景筛选，您无需手动配置流程，直接使用系统推荐的模板就可以生成演练，功能如下图所示。

原则3 在生产环境中运行实验

混沌工程推荐故障演练是在生产环境中进行，主要的原因有以下两点：

- 系统的行为会根据环境和流量模式的变化，例如系统依赖的组件在测试环境和生产环境会有比较大的差异。
- 系统的监控和人员的应急响应在测试环境和生产环境也会有比较大的差异。

AHAS Chaos 无法知道您当前的演练环境是否属于生产环境，为了更好的演练我们建议您将生产环境的演练和测试环境演练进行区分，您可以通过 Chaos 提供的 Namespace 功能将演练分为测试环境和线上环境，针对不同的环境指定响应的演练策略。

原则4 持续自动化运行实验

由于系统、业务以及依赖的环境是在不断的变化，故障演练也需要持续的进行下去，因此自动化的故障演练是必不可少的环节。

演练平台必须高效安全，为此 Chaos 提供了以下能力：

- 提供了演练流程编排能力和多种演练运行方式。
- 可以根据您配置的恢复参数来自动恢复演练。
- 您还可以通过配置定时任务来自动化运行演练。

一个完整的演练配置如下图所示。

原则5 最小化爆炸半径

最小化爆炸半径意味着混沌工程的影响范围必须得到控制，逐渐扩大故障范围，要保证演练是可控的，因此在演练当中需要您时刻关注在稳态假设中配置好的系统指标，如果影响范围超出了预期，请立刻终止演练，并且修复问题。如果影响范围没有达到预期，您也需要考虑下场景选择的合理性，演练的范围大小，或者其他的问题。

AHAS Chaos 从多方面来控制演练的爆炸半径：

- 机器和场景选择方面
  AHAS Chaos 已经将安装了演练探针的机器分成了主机和容器服务 K8S（Kubernetes）集群两种维度，方便您控制机器范围，同时在演练场景上，AHAS Chaos 针对主机和 K8S 可以选择的场景也做了区分，防止因为机器或者场景的错误选择而导致演练失败。
- 一键停止演练按钮
  AHAS Chaos 提供了多个演练停止入口来让您立刻终止演练，并且演练终止都会自动恢复已经注入的故障。详情请参见停止演练。
  - 在演练列表页面，您可以一键停止所有的演练任务。
  - 在演练执行界面，您可以立刻终止演练。
  - 在演练流程节点上，您也可以单击停止来终止演练。
- 基于指标自动恢复演练
  AHAS Chaos 不但提供了多种手动演练入口，还可以配置自动恢复策略，请参见以下两种策略。
  - 基于监控指标的自动恢复，如果指标触发了恢复规则，系统会自动执行演练恢复操作。
  - 基于演练时长的自动恢复。

更多信息

关于 AHAS Chaos 的更多信息，请参见Chaos Engineering 的历史、原则以及实践和混沌工程实践经验：如何让系统在生产环境中稳定可靠。

4.开始演练

4.1. 演练概述

基于阿里巴巴多年业务的真实线上故障库的积累，AHAS 故障演练模块为您预定义了丰富的测试任务，检验应用的高可用能力。目前，仅支持对部署在阿里云 ECS 实例上的应用进行故障演练。
一次完整的故障演练包括以下四个阶段：

- 安装探针：对指定机器进行演练，需要在机器上面安装故障演练探针，探针的作用是下发故障演练执行命令。
- 创建演练：配置演练基本信息、演练对象和演练全局参数。可同时选择多个故障类型。
- 执行演练：将故障注入机器，可通过演练时需曲线、演练参数、演练日志等检查故障注入的效果是否符合预期。
- 停止阶段：清除故障。当故障演练自动结束、您主动终止或者演练中的任何环节出现异常后，系统都会进入恢复阶段，自动清除相应的故障，使故障演练对象恢复演练前的状态。

安装探针	>	创建演练	>	执行演练	>	停止演练

4.2. 接入演练应用

故障演练支持接入容器服务Kubernetes应用以及Java、GO、PHP等语言的应用，本文介绍如何在故障演练中接入新应用。

接入Java应用

通过配置JVM启动参数来指定应用和应用分组，用于精确划分该机器所归属的应用或应用分组（与安装探针时指定的应用或应用分组不冲突）。

1. 配置JVM启动参数。
   AppName只能包含字母、数字和特殊字符_和-。

   ```
   -Dproject.name=应用名 -Dproject.group.name=应用分组名
   ```

   请根据实际情况替换以上的值，上述配置默认值如下：
   - `project.name`：默认值`ahas-default-app`。
   - `project.group.name`：默认值`ahas-default-app-group`。

   > 注意 在已部署故障演练探针的机器上，修改JVM启动参数并重启，应用会自动识别并生效，无需重新部署故障演练探针。

2. 启动应用。
3. 登录AHAS控制台，在左侧导航栏单击**探针管理**，进入探针管理页面，单击右上角**安装故障演练探针**。
4. 在**选择环境**页面单击**阿里云ECS**。
5. 在**安装应用高可用探针**页面安装探针，在目标主机右侧**操作**列单击**单击安装**。
6. 填写应用、应用分组信息。
   - 已有应用，则选择应用名称与应用分组，单击**安装**。
   - 新增应用，则填写应用名称与应用分组，单击**安装**。

7. 单击**下一步**，查看已安装的探针。
8. 单击**完成**。

接入Kubernetes容器化应用

通过Pod标签识别其所归属的应用或应用分组，配置如下：

1. 配置Pod标签。
   标签的值只能包含字母、数字和特殊字符_和-。
   在模板（YAML格式）中将以下lables配置添加到spec > template > labels层级下：

   ```
   labels:
   ahas.aliyun/app-instance: 应用名
   ahas.aliyun/app-group: 应用分组名
   ```

   根据实际情况替换以上的值，如果不配置以上的值，会再次识别是否包含`app-group-name`（容器服务应用配置）、`edas.oam.acname`（EDAS应用配置）、`app`、`k8s-app`标签配置作为应用名，应用分组名格式默认为：应用名-group。
   根据标签识别应用，标签优先级如下：ahas.aliyun/app-instance > app-group-name > edas.oam.acname > app > k8s-app。
   如果都不存在以上标签，则应用和应用分组的默认值如下：
   - 应用：ahas-default-app。
   - 应用分组：ahas-default-app-group。

   > 注意 在已部署故障演练探针的集群，修改Pod标签即可生效，无需重新部署故障演练探针。

2. 登录AHAS控制台，在左侧导航栏单击**探针管理**，进入探针管理页面，单击右上角**安装故障演练探针**。
3. 在**选择环境**页面单击**容器服务**，创建ack-ahas-pilot。详情请参见架构感知监控。

接入其他应用

通过主机部署，接入其他应用操作步骤如下。

1. 登录AHAS控制台，在左侧导航栏单击**探针管理**，进入探针管理页面，单击右上角**安装故障演练探针**。
2. 在**选择环境**页面单击**阿里云ECS**。
3. 在**安装应用高可用探针**页面安装探针，在目标主机右侧**操作**列单击**单击安装**。
4. 填写应用、应用分组信息。
   - 已有应用，则选择应用名称与应用分组，单击**安装**。
   - 新增应用，则填写应用名称与应用分组，单击**安装**。
5. 单击**下一步**，查看已安装的探针。
6. 单击**完成**。

4.3. 安装故障演练探针

对指定机器进行演练时，需要在机器上面安装故障演练探针，探针的作用是下发故障演练执行命令。

背景信息

一次完整的故障演练包括以下四个阶段：

安装探针	>	创建演练	>	执行演练	>	停止演练

操作步骤

1. 登录 AHAS 控制台。
2. 在左侧导航栏选择**探针管理**，在**探针管理**页面右上角单击**安装故障演练探针**。
3. 在**选择环境**页面选择安装探针的环境，具体安装步骤请参见架构感知接入概述。

> **说明** 架构感知探针已经包含了故障演练的功能，如果已经安装架构感知探针，则无需重复安装故障演练探针。

后续步骤

- 创建演练
- 执行演练
- 停止演练

4.4. 配置RAM权限

AHAS支持对云服务器ECS（Elastic Compute Service）和容器服务ACK（Alibaba Cloud Container Service for Kubernetes）进行演练，为了控制被演练对象的范围，AHAS故障演练支持对RAM子账号进行授权配置。

配置方法

1. 登录RAM访问控制。
2. 在左侧导航栏中选择**权限管理 > 权限策略管理**。
3. 在**权限策略管理**页面，单击**创建权限策略**。
4. 在**新建自定义权限策略**页面，填写**策略名称**，在**配置模式**下选择**脚本配置**。
5. 在**策略内容**中填写脚本。以如下配置为例进行说明。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ahas:*",
      "Resource": [
        "acs:ecs:*:*:*",
        "acs:cs:*:*:*"
      ]
    }
  ],
  "Version": "1"
}
```

   - Action：配置为ahas:* 拥有资源的全部权限。

- 创建演练权限：配置为 `ahas:CreateExperiment` ，授予该权限后，可以对资源进行演练配置操作（含更新操作）。
- 执行演练权限：配置为 `ahas:ExecuteExperiment` ，授予该权限后，可对资源进行演练执行操作（演练可执行的条件为“子账号拥有该演练中所有资源的执行权限”）。

- Resource：目前仅支持ECS和ACK两种资源，请根据实际情况配置允许演练的资源，具体语法请参见创建自定义策略说明。

6. 授权新创建的权限策略给用户，详情请参见为RAM用户授权。

应用权限配置方法

应用对应的权限配置方法请参见以下操作步骤。

1. 登录RAM访问控制。
2. 在左侧导航栏中选择**权限管理 > 权限策略管理**。
3. 在**权限策略管理**页面，单击**创建权限策略**。
4. 在**新建自定义权限策略**页面，填写**策略名称**，在**配置模式**下选择**脚本配置**。
5. 在**策略内容**中填写脚本。

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ahas:*",
      "Resource": "acs:ahas:*:*:*"
    }
  ],
  "Version": "1"
}
```

- Action：配置为ahas:* 拥有资源的全部权限。
  - 创建演练权限：配置为 `ahas:CreateExperiment` ，授予该权限后，可以对资源进行演练配置操作（含更新操作）。
  - 执行演练权限：配置为 `ahas:ExecuteExperiment` ，授予该权限后，可对资源进行演练执行操作（演练可执行的条件为“子账号拥有该演练中所有资源的执行权限”）。
- Resource：支持按应用和应用分组进行资源划分，以不同维度配置授权。
- Resource表达式格式：

```
acs:ahas:region:accountId:environment/appInstance/appGroup/ip
```

- Resource表达式说明：
  - accountId：用户ID。
  - environment：环境。指AHAS平台上的环境，取值是环境名，例如默认的default。
  - appInstance：应用。用户部署故障演练探针时确定，如果用户部署时不指定，会归属为默认应用（ahas-default-app）。

- appGroup：应用分组。用户部署故障演练探针时确定，如果用户部署时不指定，会归属为默认应用分组（ahas-default-app-group）。
- ip：资源IP，如ecs ip，pod ip或node ip等。

示例

示例1

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ahas:CreateExperiment",
      "Resource": "acs:ahas:*:*:online/business/business_host/*"
    }
  ],
  "Version": "1"
}
```

如上示例1所示，持有该权限的人对**online环境 -> business应用 -> business_host分组**内所有资源拥有配置及更新权限。

示例2

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ahas:ExecuteExperiment",
      "Resource": "acs:ahas:*:*:default/order/order_1/10.1.0.1"
    }
  ],
  "Version": "1"
}
```

如上示例2所示，持有该权限的人对**default环境 -> order应用 -> order_1分组 -> 10.1.0.1设备**拥有演练执行权限。

4.5. 创建演练

创建演练包括配置演练基本信息、演练对象和演练全局参数。

背景信息

一次完整的故障演练包括以下四个阶段：

安装探针	>	创建演练	>	执行演练	>	停止演练

操作步骤

1. 登录AHAS控制台，在左侧导航栏中选择**故障演练 > 演练列表**。
2. 在**故障演练**页面左上角，单击**新建演练**。
3. 在**新建演练**页面，选择**从空白创建**或其他演练模板。
   选择其他演练模板会从经验库里直接生成演练的配置基本信息。
4. 在**演练配置**页面，填写**演练名称**、**演练描述**和**演练标签**。

配置项	配置说明
演练名称	填写演练名称。
演练描述	为该演练添加描述，包括演练原因、演练场景以及演练可能造成的影响等信息。
添加标签	自定义演练标签，便于演练的查询和统计。 ○ 输入标签内容时，系统会显示已有的标签。 ○ 一个演练场景中，最多可输入五个标签。 ○ 标签将自动保存，下次可通过搜索或直接从下拉列表中选择已有的标签。

5. 配置演练对象。
   i. 在**演练对象**页面设置**分组名称**，选择**演练对象**。
      - 若**演练对象**选择**应用**，则需要选择**演练应用**、**应用分组**、**机器列表**等。
      - 若**演练对象**选择**非应用**，则需要选择**机器类型**和**机器列表**。

      > 说明 **机器列表**中可以选择一台或多台机器。

   ii. 单击 **+ 添加演练内容**，然后按需选择演练场景。故障演练支持的场景请参见演练场景说明。
   iii. 单击**保存**，然后单击**下一步**。
6. 配置全局配置。在**全局配置**页面完成以下配置。

   全局配置

   以下配置以脚本方式制造CPU满载的演练场景为例。
   i. 选择**演练流程**为**顺序执行**。
   ii. 在**全局监控节点**区域单击**+ 新增节点**，在**新增节点**对话框中选择需查看的指标，然后单击**确定**。全局监控节点与演练内容对应，若演练场景为**脚本方式制造CPU满载**，则全局监控节点选择**CPU指标**。
   iii. 在**全局监控配置**面板中选择**指标**。
   iv. 在**恢复策略**区域单击**+ 新增策略**，在**新增节点**对话框中选择**CPU指标**，然后单击**确定**。
   v. 在**策略配置**面板中，配置恢复规则和恢复策略。例如配置恢复策略为：当满足70% 的机器满足system.cpu.util（综合利用率）等于100% 且持续30s时，即可解除CPU满载，恢复初始CPU使用率。具体配置如下图所示。
   vi. 设置自动恢复时间。
7. 单击**下一步**。

后续步骤

演练创建成功后，您可以：

- 执行演练
  在演练列表中的某个演练的**操作**列，单击**演练**，执行演练。请参见执行演练。
- 查看并编辑演练详情
  在演练列表，单击某个演练名称，可查看演练内容。单击页面右上角**编辑**，可修改演练内容。
- 拷贝演练
  在演练列表中的某个演练的**操作**列，单击**拷贝**，拷贝一个同名的演练，您可以在此基础上编辑演练内容。
- 删除演练
  在演练列表中的某个演练的**操作**列，单击**删除**。

4.6. 执行演练

在执行故障演练过程中，您可以实时查看演练进度、每个演练活动的运行状态及结果，同时也能够随时结束演练，进行恢复阶段的活动，清除故障演练影响。

背景信息

一次完整的故障演练包括以下四个阶段：

安装探针	>	创建演练	>	执行演练	>	停止演练

操作步骤

演练任务创建完成后，您可以直接执行演练。

1. 在左侧导航栏中选择**故障演练 > 演练列表**。
2. 在**演练列表**中单击目标演练任务右侧**操作**栏的**演练**，然后在**开始执行演练**对话框中单击**确定**。
   可以看出故障开始注入之后，目标机器的CPU指标开始增加，说明故障已经生效。

   

   您可以随时关注演练进度、演练时长、活动运行的结果等。演练执行界面分为以下几个区域。

参数	描述
基本信息区域	包括了演练进度以及开始时间等信息。
指标展示区域	如果在演练参数设置里面配置了全局监控节点，则可以看到实时的系统指标数据。 ○ 指标数据将定时更新，您可以单击右上角的刷新图标手动请求数据。 ○ 如果演练尚未结束，时间范围为演练开始到当前时间，如果演练已经结束，时间范围为演练开始到演练结束时间。指标采集存在约1分钟延迟。 ○ K8s类场景时候用户配置的机器地址为Master地址，非故障实际生效的节点IP，此处展示的IP地址是实际生效的节点IP地址。例如用户配置了192.168.1.1，此为故障下发地址，实际影响的是K8s集群上192.168.1.2,192.168.1.3两个节点地址，那么监控数据会展示192.168.1.2,192.168.1.3。

参数	描述
保护策略区域	若配置了保护策略，则可以看到正在运行的保护策略列表。 ○ 演练执行后，保护策略开始执行。 ○ 若您手动终止演练，那么保护策略也会终止。
流程执行区域	流程执行区域展示了当前演练的节点运行情况以及当前节点每台机器的执行情况。
节点操作说明	根据节点的通用配置，在节点上会有以下操作。 ○ ：说明当前节点已经运行完毕，需要用户手动单击才能进入到下一个节点。 ○ ：说明当前节点已经运行完毕，如果您无需再运行下一个节点，则单击终止整个演练任务。 ○ ：说明当前节点运行失败，单击重试。
节点详情	单击任何一个节点，会在右侧展示节点的详情，包含以下信息。 ○ 机器信息：可以看到每一台机器的运行情况，如果当前机器执行错误，可以单击机器IP来查看具体的错误信息。 ○ 参数：可以看到演练节点的配置参数。 ○ 日志：可以看到演练运行过程中当前节点的执行日志。

后续步骤

停止演练

常见问题

请参见故障演练常见问题。

4.7. 停止演练

除了设置故障演练自动结束时间外，还可以手动停止演练。停止演练后，系统会进入恢复阶段，自动清除相应的故障，使故障演练对象恢复演练前的状态。

背景信息

一次完整的故障演练包括以下四个阶段：

安装探针	>	创建演练	>	执行演练	>	停止演练

自动停止

在创建演练过程中需设置自动恢复时间和保护策略，起到自动停止演练的作用。具体步骤请参见创建演练。

- 当演练时长超过设置的自动恢复时间，则会终止演练任务。

- 若为演练任务配置了保护策略，触发保护策略则会终止演练任务。

手动停止

您可以通过以下三种方式手动停止演练。

- 在左侧导航栏选择**故障演练 > 演练列表**，然后在**演练列表**中单击目标演练任务右侧**操作**列的**停止**。
- 在演练详情页右上角单击**终止**。

- 在演练详情页**执行情况**区域执行节点后单击 。

4.8. 接入ARMS监控

故障演练时通过接入应用实时监控服务ARMS（Application Real-Time Monitoring Service）可以对演练过程的指标进行监控，包含JVM内存、JVM线程数、JVMGC相关、网络出入口流量、磁盘、CPU等指标。本文介绍如何在故障演练中接入ARMS并监控演练指标的操作步骤。

前提条件

- 已分别在故障演练和ARMS中接入应用，且两个应用的名称要保持一致。接入应用的操作请参见接入演练应用和创建应用监控任务。
- 已安装好探针，详情请参见安装故障演练探针。

操作步骤

1. 登录AHAS控制台，在左侧导航栏选择**故障演练 > 演练列表**。
2. 单击**新建演练**，选择**从空白创建**，进入**演练配置**页面。
3. 在**演练配置**页面，填写**演练名称**、**演练描述**和**演练标签**。

配置项	配置说明
演练名称	填写演练名称。
演练描述	为该演练添加描述，包括演练原因、演练场景以及演练可能造成的影响等信息。
添加标签	自定义演练标签，便于演练的查询和统计。 ○ 输入标签内容时，系统会显示已有的标签。 ○ 一个演练场景中，最多可输入五个标签。 ○ 标签将自动保存，下次可通过搜索或直接从下拉列表中选择已有的标签。

4. 在**配置**页签，设置演练对象的参数。

演练对象应用.png

   i. 设置**分组名称**。

   ii. **演练对象**选择**应用**。

iii. 选择**演练应用**、**应用分组**和**机器列表**等。

> **说明** 机器列表中可以选择一台或多台机器。

iv. 单击**添加演练内容**，然后按需选择演练场景。故障演练支持的场景请参见演练场景说明。

v. 单击**保存**，可添加多个演练分组。然后单击**下一步**。

5. 配置全局配置。

i. 选择**演练流程**。

ii. 单击**全局监控节点**的**新增节点**，选择需要监控的节点，单击**确定**。然后选择参数对应的指标。如查询JVM内存信息（ARMS），可选择新生代、老年代等相关指标。

iii. 在**恢复策略**区域单击**新增策略**，选择指标。

iv. 在**策略配置**面板中，配置恢复**规则**和**恢复策略**。

v. 设置**自动恢复时间**和**定时运行时间**。

6. 单击**下一步**，演练创建成功。

执行结果

在演练详情页，单击**演练**，执行演练后，可以观察监控的指标。

演练指标.png

5.演练场景

5.1. 演练场景说明

故障演练场景是演练任务的核心。AHAS 提供基础资源类场景和 Kubernetes 类场景，帮助分布式系统提升容错性和可恢复性。
每一个执行阶段的演练场景都对应一个恢复阶段的演练任务。恢复阶段的演练任务目的是清除故障演练的影响，使应用或服务恢复正常，通常不需要配置参数。本文不再介绍恢复阶段的演练活动。

基础资源类场景

场景名称	特性
Node 演练场景	Kubernetes 集群中 Node 资源故障场景，目前包含基础资源中的 CPU、网络和进程。每个 Node 场景下都包含通用的 Node 筛选参数，用于查找目标 Node。
Pod 演练场景	Kubernetes 集群中 Pod 资源故障场景，包含杀 Pod 和 Pod 网络异常场景，每个 Pod 场景下都包含通用的 Pod 筛选参数，用于查找目标 Pod。
Container 演练场景	Kubernetes 集群中 Pod 资源下的容器故障场景，目前包含杀容器以及容器内故障场景。每个容器故障场景下都包含通用的容器筛选参数，用于查找目标容器。

Java

K8s 类场景

5.2. 基础资源类场景

5.2.1. CPU 类场景

CPU 类场景可以指定 CPU 使用率，也可以指定 CPU 满载的使用核数。帮助您在 CPU 在特定负载的环境下，验证应用或服务的性能指标以及监控告警、流量调度、弹性伸缩等能力。

参数说明

参数名称	参数说明
CPU 使用率百分比	指定 CPU 使用率，取值区间为 [0,100]。填写时不用填写百分号，例如填写 80，则指定 CPU 的整体使用率达到 80%。
CPU 使用率满载的核数	指定使用率满载的 CPU 核数，可取值区间为 [0，应用总核数]，默认值为 0，表示全部核满载。
CPU 使用率满载的具体核位置	指定特定的 CPU 核，可以指定多个核索引，索引从 0 开始，多个核索引之间可以使用逗号分隔。如果是连续的索引，可以使用短线连接符连接，指定后该核使用率满载。

示例

指定全部核数 CPU 满载的演练场景配置如下：

5.2.2. 网络类场景

网络类场景包含网络延迟、网络丢包和篡改域名解析等场景。帮助您在网络异常的情况下验证应用或服务的容错能力。

网络延迟

网络延迟场景可以指定网络延迟因素（例如网卡、本地端口、远程端口、目标 IP 等）和延迟时间，对应用或服务注入网络调用延迟故障。验证网络延迟情况下系统的容错能力。

参数说明

参数名称	参数说明
网卡名称	具体的网卡设备，为必选项，例如 eth0。
本地服务端口	本地服务监听的端口，外部流量不允许通过此端口进来。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
远端服务端口	调用远程服务的端口，本地流量不允许通过此端口出去。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
排除端口	无需注入网络延迟调用故障的端口，与本地服务端口和远程服务端口功能互斥。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
远端服务 IP	指定演练对象访问的远端服务 IP。可以通过子网掩码来指定一个网段的 IP 地址，例如若填写 192.168.1.0/24.，则 192.168.1.0~192.168.1.255 都生效。
延迟时间	指定网络延迟的时间，单位是毫秒，必填项。
延迟上下浮动时间	网络延迟时间的上下浮动范围，单位是毫秒。例如配置 100，则表示所配置的延迟时间上下浮动 100 毫秒。

示例：

应用 A 调用下游服务（服务端口是 7001）延迟 5 秒，延迟时间上下浮动 500 毫秒，网络调用的网卡是 eth0。演练场景配置如下：

网络丢包

网络丢包场景是指通过指定网络丢包因素（例如网卡、本地端口、远程端口、目标 IP 等）和丢包百分比，对应用或服务注入网络丢包故障。验证网络丢包情况下系统的容错能力。

参数说明

参数名称	参数说明
网卡名称	具体的网卡设备，为必选项，例如 eth0。

参数名称	参数说明
本地服务端口	本地服务监听的端口，外部流量不允许通过此端口进来。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
远端服务端口	调用远程服务的端口，本地流量不允许通过此端口出去。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
排除端口	无需注入网络延迟调用故障的端口，与本地服务端口和远程服务端口功能互斥。可以指定多个，使用逗号分隔，使用连接符表示范围。例如 80,8000-8080。
远端服务IP	指定演练对象访问的远端服务 IP。可以通过子网掩码来指定一个网段的 IP 地址，例如若填写 192.168.1.0/24.，则 192.168.1.0~192.168.1.255 都生效。
丢包百分比	网络调用丢包率，为必填项。取值为正整数，取值区间为 [0,100]，填写时不加百分号。

> **说明**
> - 排除端口不能与本地服务端口、远端服务端口同时使用。
> - 若只配置网络名称，不配置本地服务端口、远端服务端口或排除端口，则执行演练时会影响该网卡全部端口。

示例：

应用 B 调用下游服务丢包率为 50%，调用端口为 7001，网络调用的网卡是 eth0。则演练场景配置如下：

篡改域名解析

篡改域名解析场景是指通过修改目标主机的 *hosts* 文件，篡改域名地址映射，使域名访问异常。用于验证域名解析错误的情况下，应用或服务的容错能力。

参数说明

参数名称	参数说明
被篡改的域名	指定需篡改的域名，必填项，例如 www.aliyun.com。 **说明** 域名不添加 http:// 或 https://，并且域名后面不添加端口号。
域名映射的 IP	指定域名映射的 IP，必填项，例如 10.0.0.1。

示例：

为应用 A 配置篡改域名解析演练场景，示例如下：

5.3. Java 类

5.3.1. 虚拟机场景

本文列出了虚拟机常见的故障演练场景。

JvmCodeCache 满

制造 JVM CodeCache 区域满的故障，CodeCache 区域满会直接导致 JIT 编译关闭，从而引起业务系统性能下降。一般用于验证业务系统在高并发且系统性能下降的情况下，是否能够通过限流、快速扩容等手段保证业务连续性。

参数说明如下：

参数名称	是否必选	默认值	参数说明
进程 ID	必选其一	无	Java 进程的 ID。
进程关键字		无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
开启 Debug	否	否	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

Java 产生 OutOfMemoryError 异常

填充 JVM 指定的内存区域，导致指定内存区域空间不足，引发 OOM 异常。

参数说明如下：

参数名称	是否必选	默认值	参数说明
内存区域	是	无	指定填充的JVM内存区域，可选项：新生代、老生代（MetaSpace）、堆外内存。
暴力模式	否	False	是否开启暴力模式，如果是暴力模式，在OOM发生之后也不会释放之前创建的内存，可能会引起应用进程无响应。
时间间隔	否	无	两次 OOM 异常间的时间间隔，只有在非暴力模式才生效，可以减缓 GC 的频率，避免因频繁的 FullGC 导致进程无响应。
数据块大小（MB）	否	无	每次填充内存的数据块大小，仅当内存区域选择新生代或堆外内存时生效。
进程 ID		无	Java 进程的 ID。

参数名称	是否必选	默认值	参数说明
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
开启 Debug	否	否	选择是否开启debug日志，用于排查演练执行过程中遇到的问题。开启 debug 后，请到 ~/logs/chaosblade/chaosblade.log 路径下查看日志。

Java 应用内部制造 CPU 满载

Java 应用进程内制造 CPU 满载，表现为 Java 应用本身原因导致 CPU 高负载。

参数名称	是否必选	默认值	参数说明
指定 CPU 满载的个数	否	无	指定 CPU 满载的个数，默认系统当前全部核数。
进程 ID	必选其一	无	Java 进程的ID。
进程关键字		无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
开启 Debug	否	否	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

5.3.2. 代码逻辑场景

本文列出了故障演练支持的的代码逻辑场景。

篡改 Java 方法返回值

修改 Java 指定方法的返回值，返回指定的值。

参数名称	是否必选	默认值	参数说明
类名	是	无	完整的类名，包含包名。例如：com.alibaba.service.XxxService。如果模拟接口故障，需填写接口的实现类。
方法名	是	无	方法名，例如：方法 getUser(Long userId)，则填写 getUser。如果存在多个重载方法，如：getUser(String name) 和 getUser(Long userId)，则对两个方法均生效。

参数名称	是否必选	默认值	参数说明
返回值	是	无	指定方法的期望返回值，仅支持基础类型、基础类型的包装类、String 类型。如果返回空值，填写 null。
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
进程 ID		无	进程的 ID。
受影响的请求数	否	0	限制最多发生故障的请求总数，每生效一次故障计数加 1，累计发生故障请求数超出设定值后，请求则不再发生故障。填写数值小于等于 0 时，则表示不限制。
受影响的请求占比（%）	否	0	限制发生故障的请求数占所有应该发生故障请求数的百分比，也可代表每次请求发生故障的概率。填写小于或等于 0，则表示 100% 发生故障。 说明 仅填写百分比数字部分即可，即 80%，填写 80。
请求过滤规则	否	无	通过脚本方式自定义规则，通过自定义规则决策是否对请求产生故障。自定义规则生效前提为需满足其它设定条件。
过滤规则执行阶段	否	无	自定义过滤规则执行的阶段，可选择 Java 方法调用前执行或 Java 方法调用后执行。
开启 Debug	否	False	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

Java 方法调用延迟

模拟 Java 指定方法调用延迟。

参数名称	是否必选	默认值	参数说明
类名	是	无	完整的类名，包含包名。例如：com.alibaba.service.XxxService。如果模拟接口故障，需填写接口的实现类。
方法名	是	无	方法名，例如：方法 getUser(Long userId)，则填写getUser。如果存在多个重载方法，如：getUser(String name) 和 getUser(Long userId)，则对两个方法均生效。
延迟时间	是	无	延迟的时长，单位 ms。

参数名称	是否必选	默认值	参数说明
延迟波动范围	是	无	延迟时间的上下波动范围，例如：延迟时间为 2 ms，波动范围为 1 ms，则实际延迟时间为[1-3] ms。
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
进程 ID		无	进程的 ID。
受影响的请求数	否	0	限制最多发生故障的请求总数，每生效一次故障计数加 1，累计发生故障请求数超出设定值后，请求则不再发生故障。填写数值小于等于 0 时，则表示不限制。
受影响的请求占比（%）	否	0	限制发生故障的请求数占所有应该发生故障请求数的百分比，也可代表每次请求发生故障的概率。填写小于或等于 0，则表示100%发生故障。 说明 仅填写百分比数字部分即可，即80%，填写 80。
请求过滤规则	否	无	通过脚本方式自定义规则，通过自定义规则决策是否对请求产生故障。自定义规则生效前提为需满足其它设定条件。
过滤规则执行阶段	否	无	自定义过滤规则执行的阶段，可选择 Java 方法调用前执行或 Java 方法调用后执行。
开启 Debug	否	False	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

Java 方法抛出异常

模拟 Java 指定方法调用时抛出指定异常。

参数名称	是否必选	默认值	参数说明
类名	是	无	完整的类名，包含包名。例如：com.alibaba.service.XxxService。如果模拟接口故障，需填写接口的实现类。
方法名	是	无	方法名，例如：方法 getUser(Long userId)，则填写 getUser。如果存在多个重载方法，如：getUser(String name) 和 getUser(Long userId)，则对两个方法均生效。
异常	是	无	抛出的异常的完成类名，抛出的异常类必须是 java.lang.Exception 的子类。例如：java.io.IOException。

参数名称	是否必选	默认值	参数说明
异常信息	否	chaosblade-mock-exception	指定抛出异常时的异常信息，默认为 chaosblade-mock-exception。 说明 异常信息中请勿包含空格。
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
进程 ID		无	进程的 ID。
受影响的请求数	否	0	限制最多发生故障的请求总数，每生效一次故障计数加 1，累计发生故障请求数超出设定值后，请求则不再发生故障。填写数值小于等于 0 时，则表示不限制。
受影响的请求占比（%）	否	0	限制发生故障的请求数占所有应该发生故障请求数的百分比，也可代表每次请求发生故障的概率。填写小于或等于 0，则表示 100% 发生故障。 说明 仅填写百分比数字部分即可，即 80%，填写 80。
请求过滤规则	否	无	通过脚本方式自定义规则，通过自定义规则决策是否对请求产生故障。自定义规则生效前提为需满足其它设定条件。
过滤规则执行阶段	否	无	自定义过滤规则执行的阶段，可选择 Java 方法调用前执行或 Java 方法调用后执行。
开启 Debug	否	False	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

Java 方法抛出第一个异常

模拟 Java 指定方法调用时抛出方法声明中的第一个异常（按照声明顺序）。

参数说明如下：

参数名称	是否必选	默认值	参数说明
类名	是	无	完整的类名，包含包名。例如：com.alibaba.service.XxxService。如果模拟接口故障，需填写接口的实现类。

参数名称	是否必选	默认值	参数说明
方法名	是	无	方法名，例如：方法 getUser(Long userId)，则填写 getUser。如果存在多个重载方法，如：getUser(String name) 和 getUser(Long userId)，则对两个方法均生效。
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
进程 ID		无	进程的 ID。
受影响的请求数	否	0	限制最多发生故障的请求总数，每生效一次故障计数加 1，累计发生故障请求数超出设定值后，请求则不再发生故障。填写数值小于等于 0 时，则表示不限制。
受影响的请求占比（%）	否	0	限制发生故障的请求数占所有应该发生故障请求数的百分比，也可代表每次请求发生故障的概率。填写小于或等于 0，则表示 100%发生故障。 说明 仅填写百分比数字部分即可，即 80%，填写 80。
请求过滤规则	否	无	通过脚本方式自定义规则，通过自定义规则决策是否对请求产生故障。自定义规则生效前提为需满足其它设定条件。
过滤规则执行阶段	否	无	自定义过滤规则执行的阶段，可选择 Java 方法调用前执行或 Java 方法调用后执行。
开启 Debug	否	false	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 *~/logs/chaosblade/chaosblade.log* 路径下查看日志。

5.3.3. JVM 注入动态脚本

向指定的 Java 方法注入一段动态代码，您可通过代码方式实施任意故障场景，例如篡改方法入参、篡改方法返回值等。

参数说明如下：

参数名称	是否必选	默认值	参数说明
脚本类型	否	java	动态脚本的语言类型，可选项：Java、Groovy。
脚本名称	否	无	动态脚本的名称。
脚本内容	是	无	动态脚本的代码。

参数名称	是否必选	默认值	参数说明
类名	是	无	完整的类名，包含报名。例如：com.alibaba.service.XxxService。如果模拟接口故障，需填写接口的实现类。
方法名	是	无	方法名，例如：方法 getUser(Long userId)，则填写 getUser。如果存在多个重载方法，如：getUser(String name) 和 getUser(Long userId)，则对两个方法均生效。
返回值	是	无	指定方法的期望返回值，仅支持基础类型、基础类型的包装类、String 类型。如果返回空值，填写 null。
进程关键字	必选其一	无	用于识别唯一的关键字，可以通过该关键字查找到唯一进程，使用 `ps -ef | grep <key>` 来尝试查找进程，能找到唯一进程则正确。
进程 ID		无	进程的 ID。
受影响的请求数	否	0	限制最多发生故障的请求总数，每生效一次故障计数加 1，累计发生故障请求数超出设定值后，请求则不再发生故障。填写数值小于等于 0 时，则表示不限制。
受影响的请求占比（%）	否	0	限制发生故障的请求数占所有应该发生故障请求数的百分比，也可代表每次请求发生故障的概率。填写小于或等于 0，则表示 100% 发生故障。 **说明** 仅填写百分比数字部分即可，即 80%，填写 80。
请求过滤规则	否	无	通过脚本方式自定义规则，通过自定义规则决策是否对请求产生故障。自定义规则生效前提为需满足其它设定条件。
过滤规则执行阶段	否	无	自定义过滤规则执行的阶段，可选择 Java 方法调用前执行或 Java 方法调用后执行。
开启 Debug	否	false	选择是否开启 Debug 日志，用于排查演练执行过程中遇到的问题。开启 Debug 后，请到 `~/logs/chaosblade/chaosblade.log` 路径下查看日志。

脚本编写说明（以 Java 为例）

被故障注入代码：

```
package com.alibaba.service;

import com.aliaba.model.UserDO;

public class UserService {

   private UserMapper userMapper;

   public UserDO getUserById(Long userId) {
      UserDO user = userMapper.findUser(userId);
      return user;
   }
}
```

注入的脚本代码：示例脚本演示通过自定义脚本方式篡改方法返回值。

```
// 需要创建类，并import需要的类。
// 如果import自定义类，需要保证该类在目标应用系统中存在。
import java.util.Map;
import com.aliaba.model.UserDO;

public class UserServiceInterceptor {

  // 必须包含该方法，且该方法的定义不可改变(返回值、类名、参数均不可改变)。
  // 参数context包含的内容参见《脚本入参说明》。
  public Object run(Map<String, Object> context) {
    //获取getUserById方法的实际入参
    //Map的key是getUserById方法的参数列表的索引位置，value是参数值
    Map<Integer, Object> arguments = context.get("params");

    //获取getUserById方法的一个参数，即userId
    Long userId = (Long)arguments.get(0);

    //构建篡改后的方法返回值
    UserDO mockUser = new UserDO();
    mockUser.setUserId(userId); //使用真实的userId
    mockUser.setUserName("mock_user_name"); //构造错误的userName

    //返回篡改后的对象
    return mockUser;
  }

}
```

脚本入参说明：

> **说明** 列表中的对象可通过 conext.get(key) 获得。例如：key 为 object，即通过 context.get("object") 获得。

key	描述	返回值类型
params	方法的参数列表	java.util.Map<Integer,Object>（Map 的 key为拦截的方法的参数列表中参数的索引值，value为参数取值）
object	调用方法的应用对象	实际拦截的类的实例
method	当前调用方法的实例	java.lang.reflect.Method

return	实际的返回值（仅“选择生效阶段”填写true时有效）	实际拦截的方法的返回

日志输出：

使用 slf4j 打印日志。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;

Logger logger = LoggerFactory.getLogger("logger_name");
```

5.4. Kubernetes 类场景

5.4.1. Node 演练场景

Kubernetes 集群中 Node 资源故障场景，包含 CPU、网络和进程等基础资源类演练场景。每个 Node 场景下都包含通用的 Node 筛选参数，用于查找目标 Node。

通用参数说明

参数名称	参数说明
节点名称	节点资源名，选择多个节点资源时资源名之间使用逗号分隔。
节点资源标签	节点资源标签，多个标签之间使用逗号分隔，标签之间是与的关系。
限制节点数量	根据限制条件筛选后，最终故障注入故障的节点数量。默认值为 0，表示不限制。

基础资源类演练场景

- CPU 类演练场景，请参见CPU 类场景
- 网络类演练场景，请参见网络类场景

5.4.2. Pod 演练场景

Kubernetes 集群中 Pod 资源故障场景，包含杀 Pod 和 Pod 网络异常场景。每个 Pod 场景下都包含通用的 Pod 筛选参数，用于查找目标 Pod。

通用参数

参数名称	参数说明
Pod 资源名称	Pod 资源名称，多个资源名称之间使用逗号分隔，表示选择多个 Pod
Pod 资源标签	Pod 资源标签，例如app=frontend，多个标签之间使用逗号分隔，标签之间是与的关系

参数名称	参数说明
命名空间	namespace，用于限定 Pod 的查找范围
限制 Pod 数量	限制根据条件筛选后，最终故障注入故障的 Pod 数量，0 和默认值表示不限制，如果填写的数大于根据条件筛选的数量，则最终为条件筛选出的数量

杀 Pod 场景

网络延时场景

- 网络延迟场景，请参见网络延迟
- 网络丢包场景，请参见网络丢包
- 篡改域名解析场景，参见篡改域名解析

5.4.3. Container 演练场景

Kubernetes 集群中 Pod 资源下的容器故障场景，包含杀容器以及容器内故障场景。每个容器故障场景下都包含通用的容器筛选参数，用于查找目标容器。

通用参数

参数名称	参数说明
Pod 资源名称	Pod 资源名称，多个资源名称之间使用逗号分隔，表示选择多个 Pod
Pod 资源标签	Pod 资源标签，例如app=frontend，多个标签之间使用逗号分隔，标签之间是与的关系
命名空间	namespace，用于限定 Pod 的查找范围
容器ID	容器ID，多个ID之间使用逗号分隔
容器名称	容器资源名称，多个资源名称之间使用逗号分隔
限制 Pod 数量	限制根据条件筛选后，最终故障注入故障的容器数量，0 和默认值表示不限制，如果填写的数大于根据条件筛选的数量，则最终为条件筛选出的数量

网络延时场景

- 网络延迟场景，请参见网络延迟
- 网络丢包场景，请参见网络丢包
- 篡改域名解析场景，参见篡改域名解析

6.强弱依赖治理

6.1. 强弱依赖治理概述

随着分布式微服务的发展，系统正在变得越来越复杂，一个普通的应用也可能依赖了很多其他的服务。在没有明确强弱依赖关系的前提下，系统很难进行限流降级、优化改造等操作。强弱依赖治理就是通过科学的手段持续稳定地得到应用间依赖关系、流量、强弱等数据，提前发现因为依赖问题可能导致的故障，避免依赖故障影响用户体验，积累数据持续推进系统稳定性提升。

什么是强弱依赖？

异常发生时，不影响核心业务流程，不影响系统可用性的依赖称作弱依赖，反之为强依赖。

以商品详情页为例。

商品详情页

商品详情页后台系统架构如下。

- 如果商品详情页对下游依赖是强依赖，例如当下游依赖**库存**、**优惠**、**物流**出现故障的时候，将导致业务流程无法推进，会出现类似如下的说明，严重影响用户体验。

  强依赖

- 如果商品详情页对下游依赖是弱依赖，例如当下游依赖**库存**、**优惠**、**物流**出现故障的时候，将导致商品详情页部分内容加载不全，出现如下图的情况，但核心业务流程仍可继续推进，用户体验并未受到太多影响。

  商品详情页2

通过以上对比可以看出，当下游依赖出现问题时，当前系统会受到一些影响，严重影响用户体验的是强依赖，影响较小的则是弱依赖。

强弱依赖治理

强弱依赖治理就是提前发现因为依赖问题可能导致的故障，避免依赖故障影响用户体验，积累数据持续推进系统稳定性提升。

强弱依赖模型，包含依赖关系、流量、强弱。依赖治理就是通过科学的手段持续稳定地拿到应用间依赖关系、流量、强弱等数据，将此数据用于系统改造、故障决策等场景。

强弱依赖治理方案

强弱依赖治理每次选中1个应用进行治理，以30天为治理周期。

强弱依赖治理主要包含以下步骤：

1. 应用接入（需要安装探针）。
2. 依赖分析。
3. 依赖预判。
4. 依赖验证（通过演练进行验证）。
5. 方案归档。

强弱依赖治理的应用

强弱依赖治理主要可以被应用到以下场景：

- **系统改造验收**：对于分布式系统，至少在运行态中，不会因为依赖的系统后台出现故障，引起当前应用出现系统级可用性的故障，例如进程挂掉、频繁FullGC、负载飙高等，何时何地都应具备快速止血的能力。
- **限流降级参考**：对于弱依赖，一般都要配置限流或是自动降级策略。比起通过经验值来设定限流值，还是通过实际的依赖验证来进行微调更为可靠。例如对于下游出现的超时情况，就可以通过实验得出基于线程池限流的数据。
- **应用启动顺序**：理想情况下，应用启动更应该做到零强依赖启动，但实际情况是无法做到零强依赖。因此应用启动的依赖顺序也需要实时关注，特别是新IDC、机房建站时，会有着蜘蛛网一样的依赖关系，此时最好是通过系统方式获得依赖顺序。
- **故障根源定位**：后台系统的故障，往往通过上一级的业务故障表现出来。故障处理讲究的是争分夺秒，良好的强弱依赖，对于系统自动化诊断有非常大的助力作用。
- **依赖容量评估**：正常调用链路下的系统容量需要评估。例如当某个弱依赖挂掉时，需要注意整体的容量是否有变化。

强弱依赖治理的演进

强弱依赖治理分为三阶段：启动强弱依赖、低流量强弱依赖、高流量强弱依赖。

6.2. 强弱依赖治理方案

强弱依赖治理每次选中1个应用进行治理，以30天为治理周期。本文介绍强弱依赖治理的操作步骤。

前提条件

需要安装探针才能识别到应用，如未安装请参见管理探针。

创建强弱依赖治理

1. 登录AHAS控制台，在左侧导航栏选择**故障演练 > 业务场景演练**。
2. 进入**业务场景演练**页面，在左侧导航栏选择**强弱依赖治理**。
3. 单击左上角的**创建治理方案**。进入第一步**应用接入**。

参数	描述
方案名称	输入治理方案的名称。
治理应用	选择要治理的应用，每个应用在同一个时段只能参与一个治理方案。若未找到应用，可能是由于暂未接入所需治理的应用，需要单击**新应用接入**，详情请参见应用接入。
机器分组	选择好要治理的应用，系统会自动检测应用所在机器分组及探针状态。若探针存在异常的情况，可根据提示查看如何解决。详情请参见探针安装常见问题。

4. 单击**下一步**，进入**依赖分析**，系统会自动对应用进行依赖分析。在分析过程中，您可以单击**启动**或**暂停**来控制分析时间。

> **说明** 需要保持5分钟正常的业务流量用于系统采集业务数据、分析依赖关系。随着数据的采集会逐步加载分析对象及其依赖对象，可同时选中多个分析对象进行分析。

依赖分析

5. 单击**下一步**，进入**依赖预判**，设置强弱依赖关系。
   您可以根据业务实际情况，标注分析对象与依赖对象的强弱依赖关系，作为业务预判。

   依赖预判

6. 单击**下一步**，进入**依赖验证**，单击**去验证**。

   依赖验证

   根据上一步的**依赖预判**，系统自动生成验证用例（演练配置），您可以单击**去验证**触发验证执行，通过观察实际的监控数据、业务现象，判断真实业务的强弱依赖关系。当依赖预判与验证结果一致时为**符合预期**，否则为**不符合业务预期**。

   > **说明**
   > - 随着业务的变化，部分依赖关系会失效。您可以根据业务的实际变化，多次触发依赖分析，更新依赖预判。
   > - 每次单击**去验证**时，系统会自动检测**第二步：依赖分析**是否有更新；其次，检测验证用例（演练配置）参数是否有更新；如有变化，系统会自动提示您。
   > - 在验证列表中：
   >   - **结论**存在-或**不符合预期**，则方案的**治理结果**为**不符合预期**。
   >   - **结论**都是**符合预期**或**已失效**，则方案的**治理结果**为**符合预期**。

7. 单击**方案归档**，结束1个应用的强弱依赖治理。归档后将无法进行再次分析、验证，但治理报告可正常下载。未手工归档的方案，系统将在30天内自动归档。

   > **说明** 强弱依赖治理方案以30天为治理周期。

7.最佳实践

7.1. 强弱依赖治理最佳实践

本文以对一个部署在Kubernetes上的微服务应用进行强弱依赖治理为例，介绍通过场景化演练来发现依赖问题、暴露风险的整个过程。

背景信息

本文示例的应用是一个开源电商Demo，提供了商品选购、购物车、下单等功能。由于应用程序不存在对外部云服务的依赖，并且提供了预构建镜像，所以该应用可以直接部署在任意Kubernetes集群上。Demo应用架构如下图所示。

该微服务Demo由9个服务构成：

- nacos-server：注册中心。
- frontend：前端。
- recommendationservice：推荐系统，负责对首页的产品列表进行排序。
- cartservice：购物车。
- cart-redis：购物车信息储存的数据库。
- checkoutservice：下单。
- checkout-mysql：下单信息储存的数据库。
- productservice：商品。
- product-mysql：商品数据库。

本文Demo信息：

- Demo地址：http://115.29.178.169:8080/
- 开源地址：https://github.com/cosmic-jc/alibabacloud-microservice-demo

部署应用

部署应用支持以下2种方法：

- 部署方法1：使用预构建镜像快速部署。
  运行prebuild中的所有YAML文件，使用预构建的镜像快速部署。

  ```
  cd prebuild/
  for i in *.yaml; do kubectl apply -f $i; done
  ```

- 部署方法2：手动编译并部署。
  i. 在根目录下运行 `mvn clean install` 。
  ii. 进入每个子模块目录*src/{submodule}*，运行 `./build.sh` ，编译并构建镜像文件。
  iii. 修改YAML文件中镜像。
  iv. 部署应用。

创建治理方案

一个治理方案针对一个应用，例如需要对电商Demo的前端页面进行治理，则选择frontend应用。在主机模式下，接入探针时需填写应用名；在K8s环境下，您需要通过Label来识别应用名。

1. 登录AHAS控制台，然后在页面左上角选择地域。
2. 在左侧导航栏选择故障演练 > 业务场景演练 > 强弱依赖治理，选择创建治理方案。
3. 输入方案名称，单击新应用接入，选择Kubernetes，接入新应用。
   i. 登录容器服务管理控制台。
   ii. 在左侧导航栏选择市场 > 应用目录，单击ack-ahas-pilot，根据具体情况修改参数，单击创建，详情请参见架构感知监控。

   强弱依赖治理最佳实践2.png

   接入探针后，Kubernetes中打标签为 `app=<name>` 的Pod将显示在AHAS控制台故障演练的治理应用中。

   frontend
4. 单击下一步，进入依赖分析，注入流量。
   由于依赖关系的准确识别是需要流量的，如果在流量不足的测试环境中接入，则需要您手动提供流量。推荐使用PTS等工具创建压测提供流量，本文示例将压测API中的链接地址更换为frontend的公网访问地址，可自定义流量注入链路。
   i. 登录PTS控制台。
   ii. 在左侧导航栏选择创建压测 > 创建PTS场景，在场景配置中创建压测场景。详情请参见创建压测场景。

   注入流量
5. 在依赖分析页面，系统自动识别依赖关系。
   进入依赖分析页面，在应用存在正常流量的情况下，系统会自动分析建立依赖关系。如果超过一定时长依赖关系无变化，则表示在当前流量关系下，依赖已完全展示。本文示例识别出frontend有5个依赖。

   依赖识别
6. 单击下一步，进入依赖预判，进行业务依赖判断。
   业务依赖是指对识别到的依赖进行强弱关系的预判，依赖预判不能脱离业务的特性，系统将依赖分为强依赖与弱依赖。如本示例下图所示。

   业务依赖

   通过以上业务依赖的预判可以得出以下结论：
   - 前端对商品推荐服务预判为弱依赖，表示当推荐服务发生故障时前端正常访问不应该受阻。
   - 在购物链路中，商品服务product对商品数据库product-mysql预判为强依赖。表示如果扣减库存失败，则应该阻断下单流程，否则可能导致超卖的错误。
7. 单击下一步，进入依赖验证。选择要验证的依赖，单击去验证。
   在对业务分析进行依赖预判后，应通过故障注入的方式验证真实依赖关系是否与预判相符，例如注入依赖的服务间的网络延迟故障。强弱依赖的验证可以有多种指标，例如监控与日志的报警，请求的返回状态码等等。
   本文示例预期frontend与cartservice为弱依赖，单击去验证，开始故障演练。

   依赖验证

   验证结果是前端页面无法显示（或HTTP返回结果非200），如下所示。

   依赖验证2
8. 演练完成后，单击恢复或终止。在结果反馈页面，记录结论和验证结果。返回到依赖验证页面继续验证其他依赖。

9. 单击**方案归档**，结束1个应用的强弱依赖治理。归档后将无法进行再次分析、验证，但治理报告可正常下载。未手工归档的方案，系统将在30天内自动归档。

总结

强弱依赖治理提供了一系列自动化功能，包括应用接入、依赖智能识别、对应演练自动创建、依赖验证等。为了实现全面准确的依赖分析，您需要提供全面覆盖依赖的流量。

利用强弱依赖治理梳理出应用的依赖关系的强弱，能够提前发现风险、发现不合理的依赖关系，进而提升应用的高可用能力。

演示操作

以下通过3个视频，演示下强弱依赖治理的整个过程。

1. 准备工作
2. 操作步骤
3. 总结