Q/JHK

聚好看科技股份有限公司企业标准

Q/JHK G06035—2019

|  |  |
| --- | --- |
|  |  |

故障注入管理办法

2019 - 03 - 29发布

2019 - 03 - 29实施

聚好看科技股份有限公司   发布

前 言

本程序由聚好看科技股份有限公司提出；

本程序由聚好看科技股份有限运维部负责起草；

本程序由聚好看科技股份有限公司运维部负责解释；

本程序起草人：李永福

本程序于2019年首次发布，本次是第1次新规制定/修订。

本程序自批准之日起实施。

审核：  时间：

批准：  时间：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **修改履历** | | | | |
| 文件名称： 故障注入管理办法 | | | 文件编号：Q/JHK  G06035-2019 | |
| 版本号 | 发布日 | 制定、修改内容 | 起草部门 | 备注 |
| V1.0 | 2019/3/29 | 新规制定 | 运维部 |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 次

[1 主题内容与适用范围 3](#_Toc882104)

[2 术语和定义 3](#_Toc882105)

[3 缩略语 3](#_Toc882106)

[4 职责分工 3](#_Toc882107)

[4.1 混沌工程师 3](#_Toc882108)

[4.2 运维工程师 3](#_Toc882109)

[4.3 运维部负责人 3](#_Toc882110)

[4.5 产品线研发负责人 4](#_Toc882111)

[5 故障注入流程 4](#_Toc882112)

[5.1 研究准备故障注入技术 4](#_Toc882113)

[5.2 定义系统稳态 5](#_Toc882114)

[5.3 评估系统整体健康情况 5](#_Toc882115)

[5.4 设定假设 6](#_Toc882116)

[5.5 开始故障注入 6](#_Toc882117)

[5.6 故障注入复盘总结 7](#_Toc882118)

[5.7 故障注入自动化 7](#_Toc882119)

[6 裁剪指南 7](#_Toc882120)

[7 文件相关性 7](#_Toc882121)

[7.1 上级文件 7](#_Toc882122)

[7.2 相关文件 7](#_Toc882123)

[7.3 下级文件 7](#_Toc882124)

[7.4 相关记录 7](#_Toc882125)

[8 度量指标 7](#_Toc882126)

[9 检查与考核 7](#_Toc882127)

故障注入管理办法

1 **主题内容与适用范围**

故障注入，又称混沌工程，是一个强大的实践, 它已经在世界上一些规模最大的业务系统上改变了软件是如何设计和工程化的。 相较于其他方法解决了速度和灵活性, 混沌工程专门处理这些分布式系统中的系统不确定性。

混沌工程不是制造问题，而是揭示问题，它为我们大规模的创新和给予客户他们应得的高质量的体验提供了信心.

混沌工程的核心目标是：“Resiliency through orchestrated chaos”

本管理程序明确了聚好看科技股份有限公司（以下简称公司）故障注入的活动和工作内容，用以指导和规范公司故障注入的实施过程。

2 **术语和定义**

2.1 故障注入

即混沌工程，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

2.2 监控系统

用于收集系统Metrics，如国内用的多的是小米开发的open-falcon。随着容器的普及的，能监控容器内部的状况和容器内部服务的信息的Prometheus因时而生。普罗米修斯(prometheus)是google内部监控报警系统的开源版本，是google sre 思想在其内部不断完善的产物,它的存在是为了更快和高效的发现问题，快速的接入速度，简单灵活的配置都很好的解决了这一切

3 **缩略语**

FIT - Failure Injection Testing 故障注入测试

MTTF - Mean Time To Failure 平均故障时间

[MTTD](https://fanyi.baidu.com/#en/zh/MTTD) - Mean Time to Detect 平均探测时间

MTTR - Mean Time To Repair平均故障修复时间

MTBF - Mean Time Between Failure 平均故障时间

4 **职责分工**

4.1 混沌工程师

4.1.1 负责研究和制定混沌技术方案并在实验环境验证；

4.1.2 对故障注入实施过程提供技术支持。

4.2 运维工程师

4.2.1 端到端负责制定具体业务的故障注入实施方案，准备故障注入申请，输出故障注入总结；

4.2.2 将发现的隐患及时输出至产品研发，并跟踪闭环。

4.3 运维部负责人

4.3.1 对故障注入申请和变更实施方案进行审核确认

4.5 开发负责人（SE）

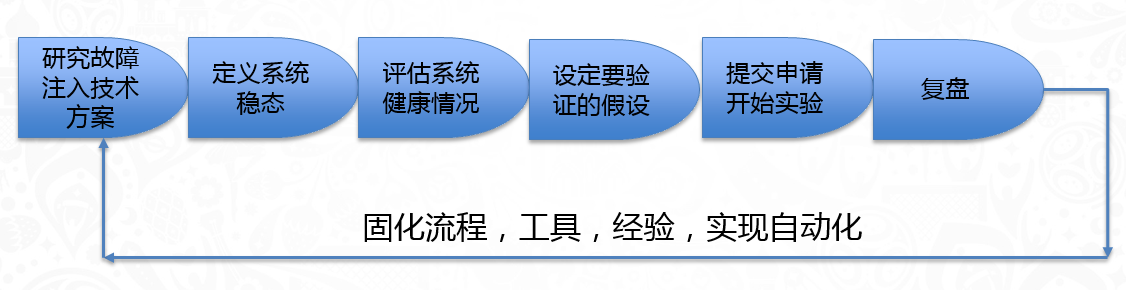
4.5.1 对业务注入的故障提供紧急恢复支持；

4.5.2 负责对故障注入发现的问题进行闭环解决。

|  |  |
| --- | --- |
| **角色** | **职责描述** |
| 混沌工程师（chaos engineer） | 1、负责研究和制定混沌技术方案并在实验环境验证；  2、对故障注入实施过程提供技术支持。 |
| 运维工程师（Operation engineer） | 1、负责制定具体业务的故障注入实施方案，准备故障注入申请，输出故障注入总结；  2、将发现的隐患及时输出至产品研发，并跟踪闭环 |
| 运维部负责人（Manager of Operation department） | 1. 对故障注入申请和变更实施方案进行审核确认 2. 知会故障注入过程设计的相关干系人并获得审批 |
| 开发负责人（System Engineer） | 1、评审故障注入的实施过程  2、对注入故障的紧急恢复提供支持；  3、负责对故障注入发现的问题进行闭环解决 |

5 **故障注入流程**

故障注入基本流程



5.1 研究准备故障注入技术

实施故障注入方法很多，可以通过手动命令触发，也可以利用工具，一些常见的开源平台已经有相应的开源工具来触发故障

混沌工程师负责输出故障注入技术方案，学习研究业界开源工具，定制化开源工具，并输出使用使用指南。

对于分布式系统来说，常见的故障可以分为程序内的故障和应用程序外的故障，应用程序内的故障，指的是代码内部预置可引发程序故障的API接口、配置项或者集成第三方故障库，然后人工调用API触发故障；程序外的故障指的的是程序依赖的运行环境如机器资源等发生故障导致程序故障；

现代微服务和容器架构下，程序内置故障API或者引发程序运行环境故障如容器异常，都可实现同样的故障注入效果;对于无服务器部署如lamdba，需要直接调用故障API来触发故障。

程序外的故障大致可分为：

|  |  |  |  |
| --- | --- | --- | --- |
| **层级** | **类型** | | |
| **基础设施层** | 应用  运行  资源 | CPU | 增加CPU利用率 |
| Memory | 增加内存利用率 |
| IO | 增加I/O读写压力 |
| Disk | 增加磁盘利用率 |
| 应用  运行  状态 | 主机、虚拟机、操作系统、容器等 | 终止、重启、关机、暂停，下电主机操作系统 |
| Time travel | 修改主机系统时间 |
| Process killer | 杀掉进程 |
| 网络 | Blackhole | Drop所有网络流量 |
| Latency | 增加出口流量延时. |
| Packet loss | 增加出口流量丢包. |
| DNS | 阻止访问DNS服务器 |
| **平台层** | Openstack,Vmware,Java,Kafka,Kubernetes,Docker,Hadoop,Spark等 | | |
| **应用层**  **(业务请求)** | 模拟攻击特定customerID或特定请求，观察该用户的情况 | | |
| 模拟攻击特定的endpoints，观察系统情况等 | | |

常见的工具有：

|  |  |
| --- | --- |
| **层级** | **可用工具** |
| **基础设施** | **ChaosMonkey**随机杀死实例，需结合spinnaker(CI/CDpingt)使用，Spinnaker支持AWS, Google Compute Engine, Azure, Kubernetes, Cloud Foundry；  **GomJabbar**设计用于在私有云架构中执行攻击。攻击通过yaml配置文件定义，并以普通shell命令（例如sudo service$module stop）执行。它还集成了Ansible和Rundeck  **Gremlin**在您的私有云架构中找出弱点，避免引发问题。Gremlin使混沌工程变得简单、安全和安全，提高了私有云的稳定性和弹性;Free版本可以机型Shutdown and CPU攻击，专业版本可以进行InfrastructureLayer的所有攻击.  **Muxy**是一个用go编写的开源工具，它允许您在系统的传输层、TCP层或HTTP协议层篡改网络流量。它可以模拟真实的网络连接问题，也可以通过Plugo接口使用插件进行扩展  **OS-faults**提供了一个统一的抽象API，用于对OpenStack云执行破坏性操作，可以针对service,container,node进行start/kill/terminate等操作；  Blockade 可用于模拟测试分布式系统的网络状况，基于docker;  **ChaosToolkit**是个开源的工具集，可以进行多种混沌实验，支持AWS,kubernates等 |
| **应用平台** | **PowerfulSeal**用于kubernetes集群，它可以杀死目标pod，启动或停止VM；  **Kubernetes Pod Chaos Monkey**是一个shell脚本，功能是随机删除pods；  **Hadoop Killer** 可以随机删除机器上运行的Java进程，通过简单的YAML配置；  **Trogdor**是针对Kafka的故障注入工具；  **Byte-Monkey**进行JVM的字节码级错误注入。它通过动态检测应用程序代码来故意引入异常和延迟等错误；  **Pumba**是一个功能强大的混沌测试工具，用于在Docker中注入混沌。它可以使用高度可配置的选择规则来终止，暂停，停止和删除Docker容器。它还可以通过延迟，数据包丢失，速率限制等执行网络仿真;  **Kube-monkey**是Chaos Monkey的开源实现，用于Kubernetes集群并用Go编写。与最初的Chaos Monkey一样，Kube-monkey只执行一项任务：它随机删除群集中的Kubernetes pod，作为在系统中注入失败并测试剩余pod的稳定性的一种方法。它基于伪随机规则，在工作日的预定时间运行，然后构建计划。基于生成的计划随机选择pod目标，该目标将在同一天的随机时间被攻击和杀死，该时间范围是可配置的。 |
|  | More |

5.2 定义系统稳态

故障注入前需要定义一个系统的“**稳态**”，这样可以衡量故障注入的效果。该稳态可由日常的各种监控指标来体现，如运维监控系统统计的指标，同时更建议收集业务指标，如Netflix在混沌实验期间监控用户成功播放视频的点击次数，这是他们的核心指标或者订单数量；业务指标可更直观体现终端用户的使用情况。

参考：

|  |  |  |
| --- | --- | --- |
| 指标分类 | 指标 | 说明 |
| 基础设施监控指标 | Resource: CPU, IO, Disk & Memory | 应用运行环境 |
| State: Shutdown, Processes, Clock Time |
| Network: DNS, Latency, Packet Loss |
| 告警和on-call指标 | Total alert counts by service per week | 告警和on-call数量 |
| Time to resolution for alerts per service |
| Noisy alerts by service per week (self-resolving) |
| Top 20 most frequent alerts per week for each service. |
| Incidents | Total count of incidents per week by SEV level | 事故 |
| Total count of critail incidents per week by service |
| MTTD, MTTR and MTBF for SEVs by service |
| 应用指标 | Events | 应用内部 |
| Stack traces |
| Context |
| 业务指标 | 可参考运营相关指标定义 | 面向终端用户 |

5.3 评估系统整体健康情况

根据checklist进行点检确认系统是否符合健康，是否适合进行故障注入

|  |  |  |  |
| --- | --- | --- | --- |
| **序号** | **点检项** | **是否通过** | **确认人** |
| 1 | 当前业务系统是否健壮，是否支持弹性伸缩 | √ 通过；  🞎 不通过 |  |
| 2 | 当前业务系统无故障时长 | √ 通过； 🞎 不通过 |  |
| 3 | 当前业务系统是否适合注入故障 | √ 通过； 🞎 不通过 |  |
| 4 | 最近有无重大事件 | √ 通过； 🞎 不通过 |  |
| 5 | 运营是否知会 | √ 通过； 🞎 不通过 |  |
| 6 | More | √ 通过； 🞎 不通过 |  |

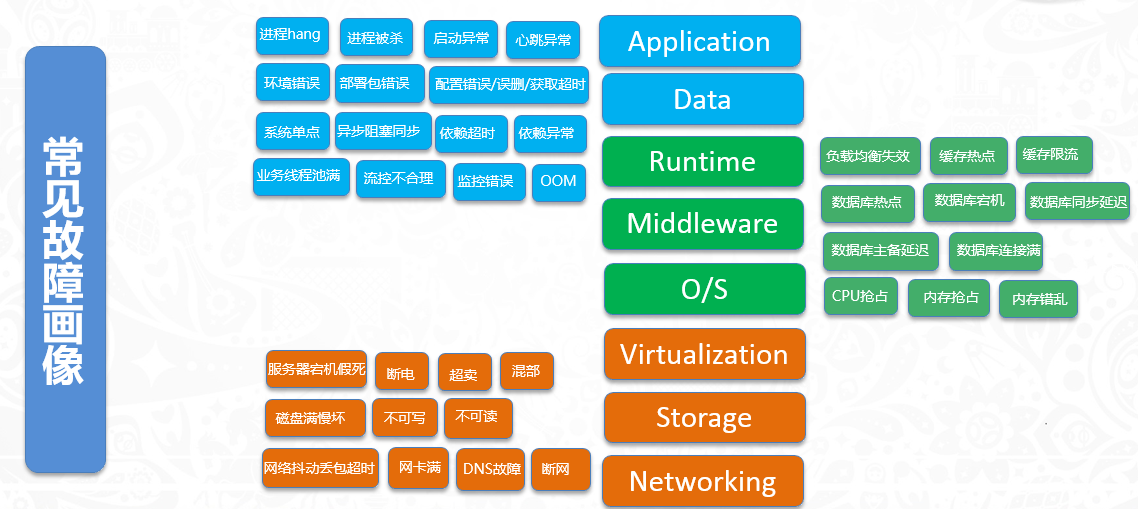
5.4 设定假设

故障注入相当于在生产环境进行各种各样的实验，实验需要有个假设，该假设是在熟悉生产环境的情况下作出的合理的假想，假设的前提是生产环境足够弹性，可以满足假设的各种“折腾”；如果当前已经识别系统弹性不足，那需要立刻停止故障注入；

假设可由运维工程师根据各自系统情况选择，目标是尽可能的验证系统的抗风险性，如业务量上升30%，CPU会耗尽；网络延迟情况下业务会响应缓慢；DNS不可用情况下业务会中断等

如果假设无效，必须尽可能调查根因或改进假设；

选择的假设尽量模拟现实中可能发生的故障；



5.5 开始故障注入

故障注入作为一种变更，需要提交故障注入申请表，和其他变更需要的材料；

故障注入申请表，包含故障注入的目标业务，故障注入类型，故障注入技术方案，要验证的假设，故障影响范围，故障恢复方案，故障注入开始时间、恢复时间、故障注入之前的状态、之后的状态，故障注入的结果等；运维负责人，开发负责人；

参考申请模板：



5.6 故障注入复盘总结

故障注入实施完成后，需在一周内对故障注入过程进行总结，输出报告，便于后续改进优化。

5.7 故障注入自动化

首次故障注入可按季度或月度执行；

首次使用的新技术需分业务逐步覆盖，持续丰富故障注入技术；

年度进行总结，逐步将成功的故障注入技术进行工具化，平台化；

6 **裁剪指南**

无。

7 **文件相关性**

7.1 上级文件

无。

7.2 相关文件

《故障注入申请表模板》

《系统软件变更管理办法》

《网上问题管理指南》

7.3 下级文件

无。

7.4 相关记录

无。

8 **度量指标**

无。

9 **检查与考核**

本程序的执行情况由公司运维部进行检查与考核。