(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113687918 A
(43)申请公布日 2021.11.23

(21)申请号 202111002602.7

(22)申请日 2021.08.30

(71)申请人 北京同创永益科技发展有限公司
地址 100082 北京市海淀区西直门北大街52、54、56号4层中栋0101-402

(72)发明人 朱柯 潘星文

(74)专利代理机构 北京市盛峰律师事务所 11337
代理人 于国强

(51)Int.Cl.
G06F 9/455(2006.01)
G06F 9/54(2006.01)

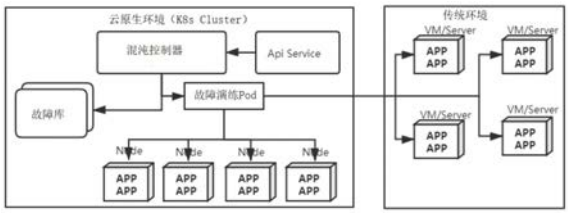
权利要求书2页 说明书5页 附图2页

(54)发明名称

一种兼容云原生和传统环境的可扩展的混沌工程实验架构

(57)摘要

本发明公开了一种兼容云原生和传统环境的可扩展的混沌工程实验架构，包括自定义资源、混沌控制器和混沌故障库；所述自定义资源包括混沌引擎自定义资源对象、混沌实验自定义资源对象、混沌结果自定义资源对象；混沌控制器用于管理自定义资源的生命周期；混沌故障库用于存储混沌工程实验所需所有故障的实现过程；混沌工程实验架构通过创建混沌引擎提交故障演练请求，获取相应的故障演练结果，并写入到K8s。优点是：采用K8s自定义资源的方式来定义故障，通过将故障演练实施内容封装到Docker镜像中，通过Docker镜像仓库统一管理，升级Docker镜像或者创建新的Docker镜像和K8s自定义资源即可升级或者扩展云原生混沌工程故障，提升云原生混沌工程故障演练的可扩展性。



CN 113687918 A

1.一种兼容云原生和传统环境的可扩展的混沌工程实验架构,其特征在于:包括自定义资源、混沌控制器和混沌故障库;

所述自定义资源包括,

混沌引擎自定义资源对象;用于为给定的应用程序创建混沌引擎,并使用appLabel标记;所述混沌引擎将一个或多个故障演练与应用程序绑定在一起;所述混沌引擎包括每个故障的具体定义以及故障对应的故障演练和与其相关的环境变量;

混沌实验自定义资源对象;用于创建混沌实验来保存和操作应用程序中实际故障演练过程;定义故障演练的类型和故障演练的关键参数;通过选择现成的故障或者扩展新的故障构建所需的混沌实验的混沌引擎;

混沌结果自定义资源对象;混沌控制器运行故障演练后获取的相应混故障演练结果,混沌结果将故障演练结果写入到K8s;

混沌控制器用于管理自定义资源的生命周期;

混沌故障库用于存储混沌工程实验所需所有故障的实现过程;

混沌工程实验架构通过创建混沌引擎提交故障演练请求,获取相应的故障演练结果,并写入到K8s;混沌工程实验架构支持云原生环境的故障演练以及传统环境的故障演练。

2.根据权利要求1所述的兼容云原生和传统环境的可扩展的混沌工程实验架构,其特征在于:混沌工程实验架构采用如下方式扩展混沌故障库,

S1、初始化系统到应用集群中,采用Operator扩展方式,创建相关的自定义资源以及自定义控制器;

S2、根据需求开发新的故障类型,并且创建新docker镜像,将docker镜像推送到docker仓库;定义一个新的混沌实验,且该混沌实验包含了新故障的信息;

S3、根据需要通过Portal、Kubectl或者K8s Api Serve创建混沌引擎来提交故障演练请求,并获取故障演练结果。

3.根据权利要求2所述的兼容云原生和传统环境的可扩展的混沌工程实验架构,其特征在于:步骤S2包括如下具体内容,

S21、开发人员根据需求开发新的故障;

S22、开发人员编写Dockerfile,并且使用docker命令将新的故障打包成docker镜像,上传到镜像仓库;

S23、开发人员定义新的混沌实验,使该新的混沌实验包含该新的故障的信息,并将该新的混沌实验通过K8s命令或者K8s Api Server写入到K8s中;

S24、SRE工程师可以使用新的混沌实验实施故障演练。

4.根据权利要求2所述的兼容云原生和传统环境的可扩展的混沌工程实验架构,其特征在于:步骤S3具体包括如下内容,

S31、SRE工程师通过K8s Api Server或命令行的方式创建混沌引擎来提交新的故障演练请求,该故障演练请求包含了混沌实验以及混沌实验所需的环境变量信息;

S32、混沌控制器监测到故障演练请求后根据,该故障混沌实验的信息以及所需的环境变量信息从镜像仓库中拉取指定的镜像,创建故障演练Pod来实时具体的故障演练;

S33、判断该故障演练是针对云原生环境的故障演练还是针对传统环境的故障演练;

S34、若是针对云原生环境的故障演练,故障演练Pod通过调用K8s Api Server对云原

生环境实施故障演练，获取故障演练结果后进入S36；

S35、若是针对传统环境的故障演练，故障演练Pod通过远程调用的方式给指定的机器安装故障演练Agent，由Agent来是实施具体得演练，获取故障演练结果后进入S36；

S36、故障演练Pod将故障演练结果通过混沌结果自定义资源对象写入到K8s中；

S37、用户可通过K8s Api Server或者命令获取相应的故障演练结果。

5.根据权利要求2所述的兼容云原生和传统环境的可扩展的混沌工程实验架构，其特征在于：新的故障的信息包括该故障执行的docker镜像、故障的启动命令和参数、故障执行的环境变量以及执行该故障所需要的权限。

一种兼容云原生和传统环境的可扩展的混沌工程实验架构

技术领域

[0001] 本发明涉及云原生技术领域，尤其涉及一种兼容云原生和传统环境的可扩展的混沌工程实验架构。

背景技术

[0002] 不管在将软件投入生产之前进行多么困难的测试以发现错误，错误总是会发生-云和可用区域会出现问题，网络会崩溃，是的，错误会让人感觉它们的存在。容错性(Resilience/弹性)是指一个系统承受这些错误的能力。例如，一个高度容错性的系统，一个由松散耦合的微服务构建的系统，它本身可以很容易地重新启动和扩展，在不影响用户的情况下克服这些错误。混沌工程是在系统出现故障之前，将其注入系统的实践。混沌工程现在被认为是确保当今频繁变化和高度复杂的系统实现所需的容错性的基本方法。通过混沌工程，可以在引起用户问题之前发现和纠正未预料到的故障场景。

[0003] 除了传统环境，广泛的采用也使Kubernetes(简称K8s)云原生环境成为最重要的软件开发和操作平台之一。针对云原生环境的混沌工程实验可以弥补现有的针对云原生环境测试技术的不足，最大程度提高系统的可靠性。

[0004] 传统的混沌工程演练平台兼容性差，以及故障库的扩展性差。传统的混沌工程平台，针对传统环境和云原生环境是分离实施的，比如分成环境演练平台以及云原生演练平台，造成平台管理，维护工作难度大，另外还需要提前安装演练Agent，使用复杂。另外传统的混沌工程演练平台故障扩展困难，需要通过重新部署或者升级实验平台的方式，会造成混沌工程实验平台的停工。

发明内容

[0005] 本发明的目的在于提供一种兼容云原生和传统环境的可扩展的混沌工程实验架构，从而解决现有技术中存在的前述问题。

[0006] 为了实现上述目的，本发明采用的技术方案如下：

[0007] 一种兼容云原生和传统环境的可扩展的混沌工程实验架构，包括自定义资源、混沌控制器和混沌故障库；

[0008] 所述自定义资源包括，

[0009] 混沌引擎自定义资源对象；用于为给定的应用程序创建混沌引擎，并使用appLabel标记；所述混沌引擎将一个或多个故障演练与应用程序绑定在一起；所述混沌引擎包括每个故障的具体定义以及故障对应的故障演练和与其相关的环境变量；

[0010] 混沌实验自定义资源对象；用于创建混沌实验来保存和操作应用程序中实际故障演练过程；定义故障演练的类型和故障演练的关键参数；通过选择现成的故障或者扩展新的故障构建所需的混沌实验的混沌引擎；

[0011] 混沌结果自定义资源对象；混沌控制器运行故障演练后获取的相应混故障演练结果，混沌结果将故障演练结果写入到K8s；

[0012] 混沌控制器用于管理自定义资源的生命周期；

[0013] 混沌故障库用于存储混沌工程实验所需所有故障的实现过程；

[0014] 混沌工程实验架构通过创建混沌引擎提交故障演练请求，获取相应的故障演练结果，并写入到K8s；混沌工程实验架构支持云原生环境的故障演练以及传统环境的故障演练。

[0015] 优选的，混沌工程实验架构采用如下方式扩展混沌故障库，

[0016] S1、初始化系统到应用集群中，采用Operator扩展方式，创建相关的自定义资源以及自定义控制器；

[0017] S2、根据需求开发新的故障类型，并且创建新docker镜像，将docker镜像推送到docker仓库；定义一个新的混沌实验，且该混沌实验包含了新故障的信息；

[0018] S3、根据需要通过Portal、Kubectl或者K8s Api Serve创建混沌引擎来提交故障演练请求，并获取故障演练结果。

[0019] 优选的，步骤S2包括如下具体内容，

[0020] S21、开发人员根据需求开发新的故障；

[0021] S22、开发人员编写Dockerfile，并且使用docker命令将新的故障打包成docker镜像，上传到镜像仓库；

[0022] S23、开发人员定义新的混沌实验，使该新的混沌实验包含该新的故障的信息，并将该新的混沌实验通过K8s命令或者K8s Api Server写入到K8s中；

[0023] S24、SRE工程师可以使用新的混沌实验实施故障演练。

[0024] 优选的，步骤S3具体包括如下内容，

[0025] S31、SRE工程师通过K8s Api Server或命令行的方式创建混沌引擎来提交新的故障演练请求，该故障演练请求包含了混沌实验以及混沌实验所需的环境变量信息；

[0026] S32、混沌控制器监测到故障演练请求后根据，该故障混沌实验的信息以及所需的环境变量信息从镜像仓库中拉取指定的镜像，创建故障演练Pod来实时具体的故障演练；

[0027] S33、判断该故障演练是针对云原生环境的故障演练还是针对传统环境的故障演练；

[0028] S34、若是针对云原生环境的故障演练，故障演练Pod通过调用K8s Api Server对云原生环境实施故障演练，获取故障演练结果后进入S36；

[0029] S35、若是针对传统环境的故障演练，故障演练Pod通过远程调用的方式给指定的机器安装故障演练Agent，由Agent来是实施具体得演练，获取故障演练结果后进入S36；

[0030] S36、故障演练Pod将故障演练结果通过混沌结果自定义资源对象写入到K8s中；

[0031] S37、用户可通过K8s Api Server或者命令获取相应的故障演练结果。

[0032] 优选的，新的故障的信息包括该故障执行的docker镜像、故障的启动命令和参数、故障执行的环境变量以及执行该故障所需要的权限。

[0033] 本发明的有益效果是：1、本发明将故障演练和故障库分离管理，采用K8s自定义资源的方式来定义故障，通过将故障演练实施内容封装到Docker镜像中，通过Docker镜像仓库统一管理，升级Docker镜像，或者创建新的Docker镜像和K8s自定义资源即可升级或者扩展云原生混沌工程故障。扩展过程完全不影响故障演练平台的功能，从而大幅提升云原生混沌工程故障演练的可扩展性。2、针对不同的演练对象，采用K8s Api或者RPC的方式兼容

云原生环境以及传统换镜架，大幅提升平台的兼容性。

附图说明

[0034] 图1是本发明实施例中混沌工程实验架构的结构示意图；

[0035] 图2是本发明实施例中故障扩展示意图；

[0036] 图3是本发明实施例中同时兼容传统环境和云原生环境的故障演练流程图。

具体实施方式

[0037] 为了使本发明的目的、技术方案及优点更加清楚明白，以下结合附图，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施方式仅仅用以解释本发明，并不用于限定本发明。

[0038] 实施例一

[0039] 如图1所示，本实施例中，提供了一种兼容云原生和传统环境的可扩展的混沌工程实验架构，包括自定义资源、混沌控制器和混沌故障库；

[0040] 所述自定义资源包括，

[0041] 混沌引擎自定义资源对象；用于为给定的应用程序创建混沌引擎，并使用appLabel标记；所述混沌引擎将一个或多个故障演练与应用程序绑定在一起；所述混沌引擎包括每个故障的具体定义以及故障对应的故障演练和与其相关的环境变量；

[0042] 混沌实验自定义资源对象；用于创建混沌实验来保存和操作应用程序中实际故障演练过程；定义故障演练的类型和故障演练的关键参数；通过选择现成的故障或者扩展新的故障构建所需的混沌实验的混沌引擎；混沌实验包含故障的标签、执行该故障所需要的的权限、执行该故障的docker镜像，执行该故障的命令行，执行该故障的参数，执行该故障的环境变量等信息；

[0043] 混沌结果自定义资源对象；混沌控制器运行故障演练后获取的相应混故障演练结果，混沌结果将故障演练结果写入到K8s；

[0044] 混沌控制器用于管理自定义资源的生命周期；所述混沌控制器为使用K8s Operator-SDK实现的控制器；

[0045] 混沌故障库用于存储混沌工程实验所需所有故障的实现过程；实际的故障注入是由故障库或故障执行器完成的，故障库包含了多个故障的实现，例如，针对云原生的如何杀死pod、如何引入Pod CPU占用、如何占用Pod内存或如何杀死Kubernetes节点，以及其他一些错误和降级。针对传统环境的如何对主机宕机，主机重启，主机CPU占用，主机内存占用，杀死主机上的进程等故障。针对不同的需要，还可以增加新的故障。

[0046] 混沌工程实验架构通过创建混沌引擎提交故障演练请求，获取相应的故障演练结果，并写入到K8s；混沌工程实验架构支持云原生环境的故障演练以及传统环境的故障演练。

[0047] 本实施例中，混沌工程实验架构采用如下方式扩展混沌故障库，

[0048] S1、初始化系统到应用集群中，采用k8s推荐的Operator扩展方式，创建相关的自定义资源以及自定义控制器；

[0049] S2、根据需求开发新的故障类型，并且创建新docker镜像，将docker镜像推送到

docker仓库;定义一个新的混沌实验,且该混沌实验包含了新故障的信息;

[0050] S3、根据需要通过Portal、Kubectl或者K8s Api Serve创建混沌引擎来提交故障演练请求,并获取故障演练结果。

[0051] 本实施例中,步骤S2包括如下具体内容,参见附图2;

[0052] S21、开发人员根据需求利用开发新的故障;不限定开发语言,支持传统的Python、Swift、JavaScript、C#、C、Ruby、PHP、Haskell、Java、C++或者Rust等,也支持脚本;

[0053] S22、开发人员编写Dockerfile,并且使用docker命令将新的故障打包成docker镜像,上传到镜像仓库;

[0054] S23、开发人员定义新的混沌实验,使该新的混沌实验包含该新的故障的信息,并将该新的混沌实验通过K8s命令或者K8s Api Server写入到K8s中;

[0055] S24、SRE工程师可以使用新的混沌实验实施故障演练。

[0056] 本实施例中,步骤S3具体包括如下内容,

[0057] S31、SRE工程师通过K8s Api Server或命令行的方式创建混沌引擎来提交新的故障演练请求,该故障演练请求包含了混沌实验以及混沌实验所需的环境变量信息;

[0058] S32、混沌控制器监测到故障演练请求后根据,该故障混沌实验的信息以及所需的环境变量信息从镜像仓库中拉取指定的镜像,创建故障演练Pod来实时具体的故障演练;

[0059] S33、判断该故障演练是针对云原生环境的故障演练还是针对传统环境的故障演练;

[0060] S34、若是针对云原生环境的故障演练,故障演练Pod通过调用K8s Api Server或者其他的一些方式对云原生环境实施故障演练,比如通过模拟杀服务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;获取故障演练结果后进入S36;

[0061] S35、若是针对传统环境的故障演练,故障演练Pod通过RPC远程调用的方式给指定的机器安装故障演练Agent,由Agent来是实施具体得演练,比如通过模拟传统环境的物理机宕机,重启,节点网络异常等来验证基础设施主从切换,主从同步是否正常。通过模拟服务的不可用,服务间的网络通讯异常等来验证服务层的高可用。通过模拟传统环境的调用延迟、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,微服务的强弱依赖是否正常,同时观察系统整体的QPS或RT是否受影响;获取故障演练结果后进入S36;

[0062] S36、故障演练Pod将故障演练结果通过混沌结果自定义资源对象写入到K8s中;

[0063] S37、用户可通过K8s Api Server或者命令获取相应的故障演练结果。

[0064] 本实施例中,新的故障的信息包括该故障执行的docker镜像、故障的启动命令和参数、故障执行的环境变量以及执行该故障所需要的权限。

[0065] 实施例二

[0066] 本实施例中,通过一个具体故障库扩展的实例描述混沌工程实验架构如何实现故障扩展。

[0067] 故障为测试微服务接口的幂等性问题,接口的幂等是一个操作,不论接口被执行多少次,产生的效果和返回的结果都是一样的。传统的测试方法是通过压力测试来触发接口高并发调用的情况下是否存在幂等性问题,但这种测试方法并不能同时支持云原生上微

服务幂等性验证,不能覆盖比如高负荷下的高并发等场景下的幂等性验证。

[0068] 下面的故障是结合云原生下的混沌工程思想,加上传统的压力测试下的一种方案,同时支持云原生环境以及传统环境的接口幂等性测试。

[0069] 故障功能描述及实现:本故障主要提供两个功能,其一是针对指定的微服模拟在一定的时间内,包括网络延迟,网络损坏,网络重复,Cpu满载,内存负载等故障发生。其二是针对指定的微服务的接口,模拟多用户高并发调用,比如100个用户,1分钟内每个用户调用1000次,并且采集调用返回的指标,通过这些指标来验证接口的幂等性。

[0070] 本故障的实现方式有多种,针对第一个功能,可以进入微服务容器的命名空间,比如mnt、uts、ipc、pid、user、net、cgroup等命名空间,以及指定根目录和工作目录,执行比如宿主机上的tc命令触发容器进程的网络延迟,损坏等故障发生。dd命令触发容器进程的Cpu满载,内存负载等故障发生。针对第二个功能,可以参照一些压力测试工具来通过各种策略模拟多用户,高并发发送相同请求到微服务,并且采集请求返回的响应时间,成功率,失败率等数据保存到时序数据库中,比如Influxdb。

[0071] 故障扩展及实施:按照附图2的流程所示,将实现的故障程序build成镜像之后上传到镜像仓库,并且生成一个新的混沌实验写入到K8s,即完成整个故障的扩展。

[0072] 按照附图3所示,生成一个新的混沌引擎并且写入到K8s,即提交了一个新的故障演练请求,混沌引擎可以指定不同的环境变量来定义具体的故障场景,比如:压测的接口地址,接口参数,网络延迟时间,内存占用率等。可以通过获取混沌结果自定义对象来查看演练的结果,接口压测信息会保存到数据库,比如Influxdb,可以通过一些图形工具比如grafana来展示数据库中保存的幂等性测试结果。网络延迟,内存使用信息可以通过监控工具,比如promethus来查看。

[0073] 通过采用本发明公开的上述技术方案,得到了如下有益的效果:

[0074] 本发明提供了一种兼容云原生和传统环境的可扩展的混沌工程实验架构,本发明将故障演练和故障库分离管理,采用K8s自定义资源的方式来定义故障,通过将故障演练实施内容封装到Docker镜像中,通过Docker镜像仓库统一管理,升级Docker镜像,或者创建新的Docker镜像和K8s自定义资源即可升级或者扩展云原生混沌工程故障。扩展过程完全不影响故障演练平台的功能,从而大幅提升云原生混沌工程故障演练的可扩展性。针对不同的演练对象,采用K8s Api或者RPC的方式兼容云原生环境以及传统换镜架,大幅提升平台的兼容性。

[0075] 以上所述仅是本发明的优选实施方式,应当指出,对于本技术领域的普通技术人员来说,在不脱离本发明原理的前提下,还可以做出若干改进和润饰,这些改进和润饰也应视本发明的保护范围。

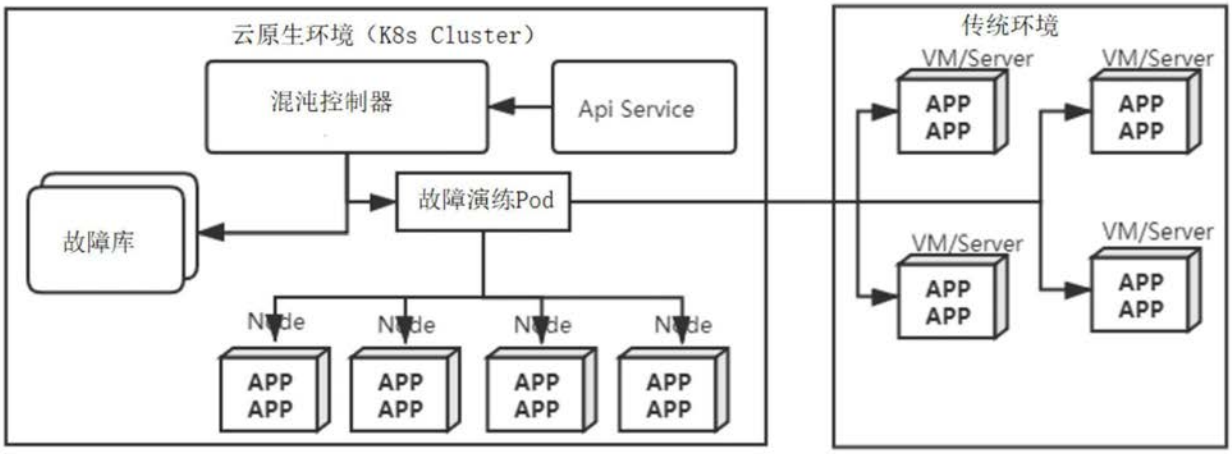

云原生环境（K8s Cluster）
混沌控制器
Api Service
故障库
故障演练Pod
Node
Node
Node
Node
APP
APP
传统环境
VM/Server
VM/Server
VM/Server
VM/Server

图1

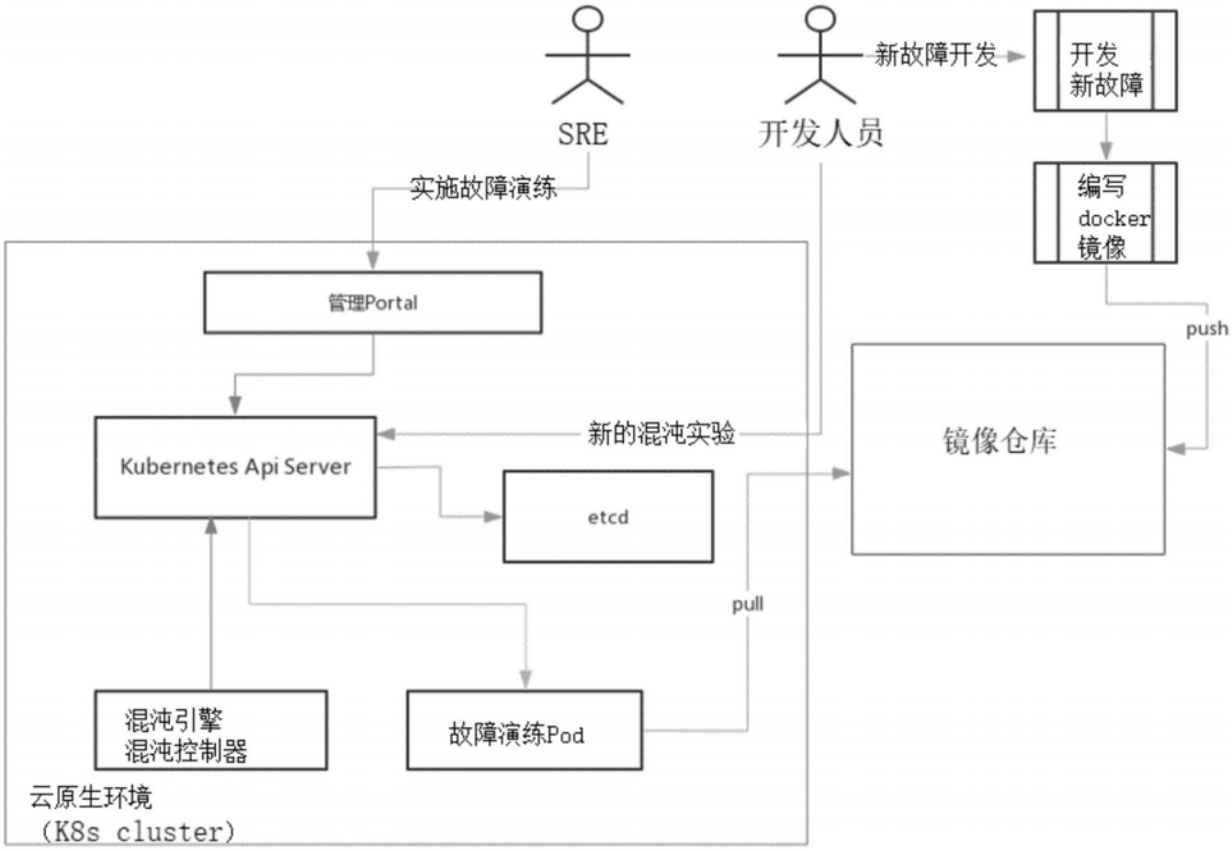

SRE
开发人员
新故障开发
开发
新故障
编写
docker
镜像
实施故障演练
管理Portal
push
新的混沌实验
Kubernetes Api Server
etcd
镜像仓库
pull
混沌引擎
混沌控制器
故障演练Pod
云原生环境
(K8s cluster)

图2

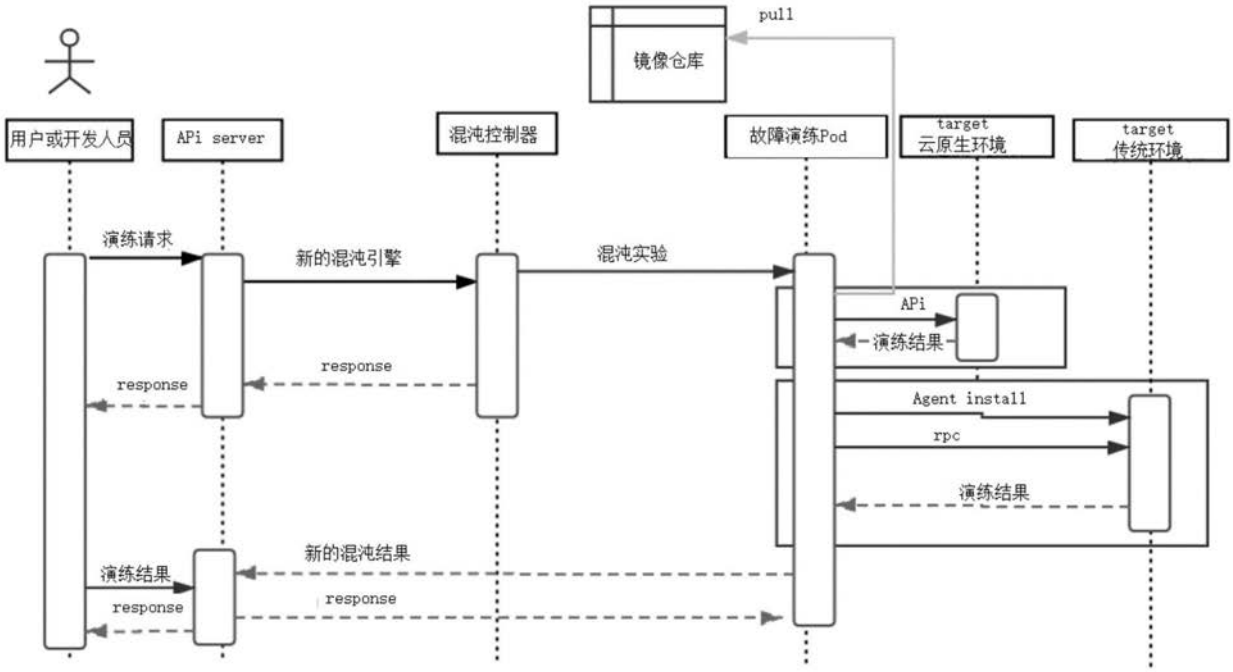

pull
镜像仓库
用户或开发人员
APi server
混沌控制器
故障演练Pod
target
云原生环境
target
传统环境
演练请求
新的混沌引擎
混沌实验
APi
演练结果
response
response
Agent install
rpc
演练结果
新的混沌结果
演练结果
response
response

图3