**Docker稳定性测试计划**

**版本<0.2>**

**文档信息**

|  |
| --- |
| 标题:Docker稳定性测试计划 |
| 作者:丁文军 |
| 创建日期:2015/09 |
| 上次更新日期: |
| 版本:0.2 |
|  |
| 部门名称:测试部 |

**修订历史记录**

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2015.9.9 | 0.1 |  | 丁文军 |
| 2015.9.22 | 0.2 |  | 丁文军 |

目录

[1. 简介 3](#_Toc431650641)

[1.1. 背景 3](#_Toc431650642)

[1.2. 测试目的 3](#_Toc431650643)

[1.3. 测试工具 3](#_Toc431650644)

[1.4. 测试环境配置 3](#_Toc431650645)

[1.5. 参考资料 3](#_Toc431650646)

[2. 测试安排 3](#_Toc431650647)

[3. 测试方法和策略 4](#_Toc431650648)

[3.1. 容器接口测试 4](#_Toc431650649)

[3.2. 容器容错性测试 4](#_Toc431650650)

[3.3. 容器并发测试 4](#_Toc431650651)

[3.4. 容器压力测试 4](#_Toc431650652)

[4. 测试说明及案例 5](#_Toc431650653)

[4.1. 容器接口测试 5](#_Toc431650654)

[4.1.1. 功能测试 5](#_Toc431650655)

[4.1.2. 测试用例 5](#_Toc431650656)

[4.2. 容器并发测试 6](#_Toc431650657)

[4.2.1. 并发测试 6](#_Toc431650658)

[4.2.2. 测试用例 6](#_Toc431650659)

[4.3. 容器压力测试 6](#_Toc431650660)

[4.3.1. 仓库负载测试 6](#_Toc431650661)

[4.3.2. CPU压力测试 6](#_Toc431650662)

[4.3.3. 内存压力测试 7](#_Toc431650663)

[4.3.4. 磁盘IO性能测试 7](#_Toc431650664)

[4.3.5. 网络性能测试 7](#_Toc431650665)

[4.3.6. 测试用例 7](#_Toc431650666)

[5. 暂停标准和再启动需求 8](#_Toc431650667)

[6. 变更历史 8](#_Toc431650668)

# 简介

## 背景

docker作为民生金融通用云主要使用的云计算技术，可以把开发者从日常部署应用的繁杂工作中解脱出来，让开发者能专心写好程序；从系统工程师的角度来看也是一样，他们迫切需要从各种混乱的部署文档中解脱出来，让系统工程师专注在应用的水平扩展、稳定发布的解决方案上。

## 测试目的

本测试的测试对象是docker及可调dockerapi。主要对docker进行稳定性测试，同时对Docker集群devops管理平台可调dockerapi进行调用的可靠性测试，及调用的并发性能测试，得到docker在应用过程中的性能和功能边界，并初步分析测试结果是否符合期望。并将测试结果作为技术选型的重要标准。

## 测试工具

压力测试工具：nstress

网络测试工具：pipework,rsync

并发测试工具：locust

## 测试环境配置

民生内部服务器环境

## 参考资料

<<民生发布平台概要设计说明书2.0>>

# 测试安排

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **测试需求项** | **测试人员** | **时间安排** | **状态** | **描述** |
| 容器用例设计 | 丁文军/寇灵云 | 15/09/22-09/29 | 完成 |  |
| 容器并发测试 | 丁文军 | 15/09/22-09/25 | 完成 |  |
| 容器接口测试 | 寇灵云/丁文军 | 15/09/23-09/29 | 完成 |  |
| 容器容错测试 | 寇灵云/丁文军 | 15/09/23-09/29 | 完成 |  |
| 容器性能测试 | 丁文军 | 15/09/28-09/29 | 完成 |  |

# 测试方法和策略

该测试计划针对DOCKER1.8.1版本,进行选型测试,包括容器接口,压力,并发等测试. 采用黑盒测试方法.

## 容器接口测试

对docker主机调用api进行容器操作、镜像操作，对集群调用api进行容器操作和镜像操作。测试内容如下：

|  |  |  |
| --- | --- | --- |
| **序号** | **基本操作** | **可调api** |
| 1 | 获取容器列表 | /containers/json |
| 2 | 创建容器 | /containers/create |
| 3 | 启动容器 | /containers/start |
| 4 | 关闭容器 | /containers/stop |
| 5 | 删除容器 | /containers/delete |
| 6 | 容器日志 | /containers/logs |
| 7 | 容器监测 | /containers/stats |
| 8 | 获取镜像列表 | /images/json |
| 9 | 发布镜像 | /images/push |
| 10 | 下载镜像 | /images/pull |
| 11 | 删除镜像 | /images/delete |
| 12 | 基于swarm集群调用api |  |

## 容器容错性测试

对docker容器操作、镜像操作及集群操作调用的api进行负面(negative)测试。针对docker容器操作、镜像操作及集群操作api，通过传入错误的参数、多次重复相同操作等测试系统的容错性能。

## 容器并发测试

对docker主机调用docker-remote-api进行容器操作、镜像操作的并发测试，对集群调用docker-remote-aip进行容器操作、镜像操作的并发测试。并发量测试范围为：10,20,50,100。

1. 单一容器和镜像并发请求测试。
2. 容器混合操作并发请求测试，权重比例

|  |  |  |
| --- | --- | --- |
| **序号** | **基本操作** | **权重** |
| 1 | 获取容器列表 | 20 |
| 2 | 创建容器 | 10 |
| 3 | 启动容器 | 10 |
| 4 | 关闭容器 | 5 |
| 5 | 删除容器 | 1 |

## 容器压力测试

1. 多台docker主机上传和下载镜像对仓库负载的性能测试。
2. 容器中CPU、内存、disk IO和带宽过高是否会影响Docker，导致同一物理机上所有容器失效
3. 物理主机CPU、内存、disk IO和带宽过高是否会影响Docker，导致同一物理机上所有容器失效

# 测试说明及案例

本测试计划,只是列出简单测试用例的测试点和测试优先级,具体的测试用例,需要查看细节的测试用例文档.

## 容器接口测试

### 功能测试

通过使用curl发送docker-remote-api，验证docker-remote-api 可调功能测试。

### 测试用例

以下每个测试点在详细测试用例中需要被细化为针对docker主机场景的测试和针对swarm集群场景的测试。

|  |  |  |  |
| --- | --- | --- | --- |
| **功能模块** | **测试功能内容** | **测试点** | **优先级** |
| 容器管理 | 列出容器信息 | 调用/containers/json列出容器 | P1 |
| 容器基本操作 | 调用/containers/create创建容器 | P1 |
| 用不存在的镜像创建容器 | P2 |
| 使用错误的参数创建容器 | P2 |
| 调用/containers/id/stop关闭容器 | P1 |
| 关闭不存在的容器 | P2 |
| 关闭已处于关闭状态的容器 | P2 |
| 调用/containers/id/start启动容器 | P1 |
| 启动已经启动的容器 | P2 |
| 启动不存在的容器 | P2 |
| 调用DELETE /containers/id删除容器 | P1 |
| 删除处于运行状态的容器 | P1 |
| 删除不存在的容器 | P2 |
| 容器日志 | 调用/containers/id/logs获取容器日志信息 | P1 |
| 获取已关闭容器的日志信息 | P2 |
| 在获取容器日志过程中关闭容器 | P2 |
| 在获取容器日志过程中删除容器 | P2 |
| 容器监控 | 容器监控 | 调用/containers/id/stats监控容器资源 | P1 |
| 监控容器过程中启动一个资源消耗高的应用 | P1 |
| 监控已关闭容器资源消耗 | P1 |
| 在监控容器过程中删除容器 | P2 |
| 在监控容器过程中关闭容器 | P2 |
| 镜像操作 | 获取镜像信息 | 调用/images/json列出镜像信息 | P1 |
| 镜像基本操作 | 调用/images/create创建镜像 | P1 |
| 调用DELETE /images/id删除镜像 | P1 |
| 调用/images/push上传镜像到仓库 | P1 |
| 容器运行过程中不可以删除容器依赖的镜像 | P1 |
| 容器关闭状态下不可以删除容器依赖的镜像 | P1 |
| 在使用镜像生成容器过程中删除镜像 | P2 |
| 仓库操作 | 仓库基本操作 | 搜索仓库中的镜像 | P1 |
| 搜索仓库中不存在的镜像 | P2 |
| 删除仓库中的镜像 | P2 |
| 删除仓库中的不存在的镜像 | P2 |

## 容器并发测试

### 并发测试

1. 利用locust发送restful请求给docker主机，对容器操作的单个操作进行并发测试，关注请求的响应时间，每秒处理请求个数和出错概率。
2. 利用locust发送restful请求给docker主机，对容器的创建，删除，关闭，启动等按照一定权重进行并发测试，关注请求的响应时间，每秒处理请求个数和出错概率。
3. 利用locust发送restful请求给swarm集群，对容器操作的单个操作进行并发测试，关注请求的响应时间，每秒处理请求个数和出错概率。
4. 利用locust发送restful请求给swarm集群，对容器的创建，删除，关闭，启动等按照一定权重进行并发测试，关注请求的响应时间，每秒处理请求个数和出错概率。

### 测试用例

以下每个测试点在详细测试案例中需要被细化为针对docker主机场景的测试和针对swarm集群场景的测试。

|  |  |  |  |
| --- | --- | --- | --- |
| 并发测试 | 单操作并发测试 | 每秒并发10/20/50/100个用户创建容器 | P1 |
| 每秒并发10/20/50/100个用户开启容器 | P1 |
| 每秒并发10/20/50/100个用户关闭容器 | P1 |
| 每秒并发10/20/50/100个用户获取容器列表 | P1 |
| 每秒并发10/20/50/100个用户获取容器日志 | P1 |
| 每秒并发10/20/50/100个用户获取镜像列表 | P1 |
| 多操作并发测试 | 按照权重进行容器的并发创建/删除/关闭/启动/获取列表操作和镜像的创建/发布/获取列表操作 | P1 |

## 容器压力测试

### 仓库负载测试

1. Docker主机push镜像至仓库，利用CPU和网络监测脚本，监测仓库主机网络IO和系统的负载情况。
2. 多台Docker主机从同仓库服务器pull镜像，利用CPU和网络监测脚本，监测仓库主机网络IO和系统的负载情况，并监测镜像获取机制。

### CPU压力测试

1. 生成未绑定CPU的容器，在容器中使用nstress工具耗损CPU资源，监测并记录docker服务和其他容器运行状态
2. 生成绑定特定CPU的容器，在容器中使用nstress工具耗损CPU资源，监测并记录docker服务和其他绑定不同CPU的容器运行状态
3. 开启cpu-1 数量个容器，对容器绑定特定CPU，在容器中发布一个httpd应用，利用nstress耗损CPU资源，利用locust监测httpd访问请求的平均响应时间及每秒处理事务数。

### 内存压力测试

1. 生成一个未限制内存使用的容器，在容器中使用nstress工具耗损内存资源，监测并记录docker服务运行状态和内存耗损情况。
2. 生成一个限制内存使用的容器，在容器中使用nstress工具耗损内存资源，监测并记录docker服务运行情况，内存耗损情况和内存突破最大使用限制时容器的运行情况。
3. 开启cpu-1 数量个容器，每个容器限制为(主机内存-2)/(cpu-1)的内存容量，在容器中发布一个httpd应用，利用nstress耗损内存资源，利用locust监测httpd访问请求的平均响应时间及每秒处理事务数。

### 磁盘IO性能测试

1. 创建一个容器，对容器不进行磁盘IO限制，在容器中利用IOZONE进行读写性能测试。在docker主机中利用IOZONE进行读写性能测试。对比两次测试结果。
2. 创建cpu-1 数量个容器，每个容器限制为(主机内存-2)/(cpu-1)的内存容量，容器绑定不同的CPU，对每个容器进行磁盘IO限制，在容器中利用IOZONE进行读写性能测试。关注每个容器磁盘IO读写性能，及测试过程中docker主机的运行情况。

### 网络性能测试

1. 创建cpu-1 数量个容器，每个容器限制为(主机内存-2)/(cpu-1)的内存容量，在容器中并行利用rsync进行大文件上传和下载测试，监测容器网卡接收/发送，丢包率和网络延时情况。
2. 创建cpu-1 数量个容器，每个容器限制为(主机内存-2)/(cpu-1)的内存容量，在容器中并行利用rsync进行小文件上传和下载测试，监测容器网卡接收/发送，丢包率和网络延时情况。

### 测试用例

以下每个测试点在详细测试案例中细化测试脚本及测试前准备工作。

|  |  |
| --- | --- |
| **测试功能内容** | **测试点** |
| 仓库压力测试 | 发布镜像并查看仓库服务器负载 |
| 并发5个docker主机下载镜像并查看仓库服务器负载 |
| CPU压力测试 | 容器不绑定CPU情况下进行CPU压力测试 |
| 容器绑定CPU情况下进行CPU压力测试 |
| 开启cpu-1个容器并消耗CPU情况下容器中应用访问性能测试 |
| 内存压力测试 | 不限定容器使用内存情况下进行内存压力测试 |
| 限定内存使用内存情况下进行内存压力测试 |
| 开启cpu-1个容器并消耗内存情况下容器中应用访问性能测试 |
| 磁盘IO限制测试 | 对比docker主机和容器中的IO读写性能测试 |
| 开启cpu-1个容器并发写情况下对比容器性能 |
| 带宽限制测试 | 开启cpu-1个容器进行大文件传输网络性能测试 |
| 开启cpu-1个容器进行小文件传输网络性能测试 |

# 暂停标准和再启动需求

1.若开发暂停，则相应测试也暂停。

2.项目再启动时，测试进度重新安排或顺延。

3.如有新的项目需求，则在原测试计划下做相应的调整。

4.若项目中止，则对已完成的测试工作做测试活动总结。

# 变更历史

正常情况下测试应依测试计划按期进行，但可能出现的一些不可预见的变更，影响测试的进度，需有所记录。

1. 出于某些原因，开发组暂停开发，测试相应暂停，测试进度从再启动日起顺延。
2. 出于某些原因，开发组不能按时提交待测试的模块，测试进度顺延。
3. 出于某些原因（未修改完、出差、放假、停电等），开发组不能按时修改Bug，测试进度顺延。

原则上，实际测试时间不大于计划测试时间。

|  |  |  |  |
| --- | --- | --- | --- |
| **变更ID** | **变更描述** | **变更影响的范围** | **结果** |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |