|  |  |
| --- | --- |
| 产品名称Product name | 密级Confidentiality level |
| 概要设计文档 | *内部* |
| 产品版本Product version | Total pages 共页 |
|  |

***Service\_guardian概要设计文档***

|  |  |  |  |
| --- | --- | --- | --- |
| Prepared by  拟制 | *杨寅* | Date  日期 | 2016-03-28 |
| Reviewed by  审核 |  | Date  日期 |  |
| Approved by  批准 |  | Date  日期 |  |



ZHICloud Technologies Co., Ltd.

致云科技有限公司

All rights reserved

版权所有 侵权必究

（REP01T01 V0.1 / for internal use only）

（REP01T01 V0.1 / 仅供内部使用）

# 文档介绍

## 1.1目的

明确说明各功能的实现方式，确定软件的全部需求和软件组成模块，确定各模块的功能和用户接口，以此作为编码设计的依据和基础。

## 1.2范围

仅限公司内部开发设计人员阅读。

## 1.3术语

# 总体设计

## 2.1 模块功能

监控服务节点进程运行状态，重启异常退出服务节点。

## 2.2 工作原理

1、通过配置文件获取被监控服务节点名称；

2、读取服务节点运行PID文件获得服务节点进程PID及执行命令。

3、轮询判断/proc/{pid}/cmdline的执行命令是否与服务节点执行命令一致；不一致视为异常重启节点，反之正常，进行下一轮轮询。

## 2.3 三方库使用

Boost库。

# 软件架构

## 3.1 模块结构描述

## 3.1.1 类图



## 3.1.2 监控管理模块（monitor\_manager.cpp）

1、读取监控配置文件，获得被监控节点名称。

2、初始化节点监控实例。

3、创建轮询服务节点。

4、每隔2秒钟轮询所有被监控节点状态。

5、执行节点重启接口。

## 3.1.3 监控节点模块（monitor\_node.cpp）

1、生成被监控节点pid文件名称。

2、获取被监控节点pid。

3、提供进程异常检测接口。

4、提供异常节点重启命令接口。

## 3.2 功能流程图



# 内部实现

## 4.1 数据结构定义

|  |  |  |
| --- | --- | --- |
| 监控管理类 |  |  |
| 日志实例 | logger | 系统日志输出接口 |
| 监控配置文件实例 | config\_file | 用户添加、移除被监控服务节点;  /var/zhicloud/config/service\_guardian/data/monitor.ini |
| 监控节点容器 | monitor\_node | 存储已添加的被监控服务节点。 |

|  |  |  |
| --- | --- | --- |
| 监控节点容器 |  |  |
| 服务节点名称 | Monitor\_node->first | 被监控节点的名称 |
| 服务节点监控实例 | Monitor\_node->second | 被监控节点实例完成对节点异常检测及执行重启动作 |

|  |  |  |
| --- | --- | --- |
| 节点监控类 |  |  |
| 服务节点进程启动命令文件 | Cmdline\_file | /proc/{pid}/cmdline\_file |
| 服务节点进程运行pid文件 | Pid\_file | /var/zhicloud/running/{节点名称}.pid |

## 4.2主要功能类定义

## 4.2.1 监控管理类

本类主要完成对所有被监控节点的注册，状态监测及重启等工作，主要涉及成员函数及数据成员如下。

class MonitorManager

{

public:

typedef std::unique\_lock<std::recursive\_mutex> lock\_type;

MonitorManager(const string& config\_root, const string& logger\_name);

virtual ~MonitorManager();

bool initializer();

void start();

void stop();

void \_execute();

void \_mainProcess();

protected:

private:

logger\_type logger;

const string config\_file;

map<string, MonitorNode> monitor\_node;

mutable std::recursive\_mutex monitor\_mutex;

std::thread \_mainThread; //监控各个节点运行状态异常节点重启的线程

bool \_shutdownMainThread;

Event \_mainThreadingEvent;

};

## 4.2.1 节点监控类

主要完成被监控节点的基本信息初始化，生成pid文件名和被监控节点名称，监测并抛出被监控节点的异常信息，提供异常检测及节点重启接口。

class MonitorNode

{

public:

MonitorNode();

MonitorNode(const MonitorNode& other);

virtual ~MonitorNode();

MonitorNode& operator=(const MonitorNode& other);

bool initializer(const string& name);

void ganeratePidFileName(string& pid\_file);

void ganerateLocalNodeName(string& name);

void getPid(int& pid, string& cmd);

void getProcExeLinkPath();

bool checkProc();

void restartNode();

void saveLog();

private:

void getNewerLog(const string& dir, const string& filer\_name, path& newer\_file);

void moveFile(const path& src, const path& dst);

protected:

private:

const static string config\_root;

const static string log\_path;

const static string data\_path;

const static string pid\_path;

const static string exe\_path;

string node\_conf;

string node\_name;

string restart\_command;

string pid\_file;

string cmdline;

string cmdline\_file;

string local\_node\_name;

uint64\_t restart\_count;

};

# 接口定义

## 5.1监控管理类

|  |  |  |
| --- | --- | --- |
| **接口名称** | **输入输出** | **功能介绍** |
| **initializer** | **N/A** | **根据配置文件注册被监控服务节点，分配节点监控实例。** |
| **start** | **N/A** | **启动监控主线程** |
| **stop** | **N/A** | **停止监控主线程** |
| **\_mainProcess** | **N/A** | **监控主线程轮询检查各节点进程的状态，并重启异常节点。** |

## 5.2节点监控类

|  |  |  |
| --- | --- | --- |
| **接口名称** | **输入输出** | **功能介绍** |
| **initializer** | **N/A** | **根据配置文件注册被监控服务节点，分配节点监控实例。** |
| **start** | **N/A** | **启动监控主线程** |
| **stop** | **N/A** | **停止监控主线程** |
| **\_mainProcess** | **N/A** | **监控主线程轮询检查各节点进程的状态，并重启异常节点。** |

# 参数配置

## 6.1 参数配置

监控配置文件部署路径:/var/zhicloud/config/service\_guardian/data/monitor.ini

监控配置文件实例:

[DEFAULT]

node\_count = 6

node\_0 = http\_gateway

node\_1 = control\_server

node\_2 = data\_server

node\_3 = storage\_server

node\_4 = intelligent\_server

node\_5 = node\_client

## 6.2 启停方式

应用程序部署路径:/home/zhicloud/service\_guardian

启动方式: ./service\_guardian start

停止方式: ./service\_guardian\_stop

重启方式: ./service\_guardian restart