Python 探针用户指南

北京基调网络股份有限公司

简述

听云 python 探针（也称 Python Agent）是基于 Python 语言而研发的性能监测工具客户端，其主要目标为支持 WSGI 协议的 python web 框架。理论上基于只要是基于 WSGI 协议的 web 框架都能对其监测， 比如 django、tornado、flask 等。当然其中也包括一些该 web 框架支持的一些外围组件，例如 sql 数据库、nosql 数据库、第三方 http 调用等。其功能会随着版本的升级而得到更多的支持。

本文档所列明的支持的框架、数据库、软件环境等，是经过测试，且完全支持的。对于没有在本文档列出的框架以、组件、安装方式等将不被支持，但随着版本的更新会逐步完善。在使用听云 python 探针前请务必仔细阅读此文档，同时我们也极力保持文档与探针功能的同步更新。

探针在性能监测时，将会对页面加载、模板渲染、中间件调用、视图层、数据层、系统软件环境、硬件环境等数据进行跟踪和监测，更多关于监测的数据参考 http://report.tingyun.com/平台的数据监测。

欢迎使用听云，因为有你，我们将会做的更好！！

听云 python 团队

一、快速入门

1、环境要求

环境	要求	备注
操作系统	类 unix 系统，如 linux、FreebSD、MacOS X 等	windows 未经测试
Python	Cpython 2.6.x，2.7.x	
web 框架	Django、flask 等	更多参考支持框架
使用范围	兼容 WSGI 1.0 (PEP 333)	

2、快速安装

注意：快速安装需要用到 pip（推荐） 或者 setuptools 工具并需要连接互联网，如您的环境无法提供，请移驾源码安装。

- **安装探针：** pip install tingyun 或者 easy_install tingyun

- **生成本地配置文件:** tingyun-admin generate-config YourLicenseKey outputFile
  如：tingyun-admin generate-config 123-456-789-001 /tmp/tingyun.ini

- **设置探针配置文件环境变量。**
  如： export TING_YUN_CONFIG_FILE=/tmp/tingyun.ini

- **嵌入探针：**tingyun-admin run-program 您应用的启动命令 您应用启动参数
  如：tingyun-admin run-program ./uwsgi.sh start

说明：以上步骤将会设置探针的 log 输出为 /tmp/tingyun-agent.log。关于探针更详细的用法和配置，参考章节 探针工具和探针配置

3、部署步骤

3.1 安装探针

详情参考探针安装章节

3.2 配置探针

初次使用探针，建议对探针的默认配置进行修改，对探针配置文件提供了两个命令行工具 tingyun-admin generate-config 和 tingyun-admin check-config，可用这两个命令可实现生成配置文件和检查配置文件。

①如果通过一键启动方式需要对探针配置设置环境变量。
②如果通过手动嵌码，调用探针 api 的，可设置环境变量或者传入配置文件参数。

3.3 设置阻塞时间（可选）

阻塞时间是用于测量前端负载均衡接受到请求，到应用层接收到请求耗时。当并发访问量大的时候，对于衡量负载均衡 到 应用层时间非常有用。
详细设置方式参考阻塞时间章节

3.4 启动用户应用

启动详情参考探针启动章节，以及常见问题章节。

二、安装与配置

由于 python 极其灵活的安装方式， 建议使用虚拟环境，如 **virtualenv**, **pyvenv** 避免污染系统环境：

1、探针的安装

1.1 探针源码安装

安装包地址为 http://download.tingyun.com/agent/python/x.x.x/tingyun-agent-python-x.x.x.tar.gz，假设得到的安装包为 tingyun-agent-python-x.x.x.tar.gz。(x.x.x 为探针版本号)

- 假设得到的安装包位于/tmp/tingyun-agent-python-x.x.x.tar.gz，执行加压缩：
  tar –zxvf tingyun-agent-python-x.x.x.tar.gz –C /tmp

- 执行以下命令安装，即可将探针安装到当前的 python 环境。
  python /tmp/tingyun-agent-python-x.x.x/setup.py install

提示： 安装完成后要配置探针才能正常使用，详情请参考探针配置章节，具体使用参考探针使用章节

1.2 在线安装

在线安装使用 python 官方代码源，需要用到 setuptools，或者 pip，更详细的用法请参考其官方文档。

- pip install tingyun 或者
- easy_install tingyun

提示： 安装完成后要配置探针才能正常使用，详情请参考探针配置章节，具体使用参考探针使用章节

2、探针升级

2.1 源码升级

源码方式升级探针，只需重新执行安装即可覆盖旧版本即可（详情参考<探针源码安装>），建议进入安装目录将安装的 tingyun 包给删除。

2.2 在线升级

注意： 在线升级时请注意对于安装时使用的工具，否则可能升级失败。并建议预先删除之前的老版本

- **pip 方式升级**
  pip install –U tingyun

- **easy_install 方式升级**
  easy_install –U tingyun

3、探针卸载

3.1 源码安装方式的卸载

如果以源码方式安装的探针，卸载探针时，只需将探针安装目录下的 tingyun 包（通常在 site-packages 下）以及可执行目录下的 tingyun-admin 删除即可。

3.2 在线安装方式卸载

- **pip 方式卸载**

  执行，pip uninstall tingyun

- **easy_install 方式卸载**

  执行，easy_install –m tingyun

注意，由于该命令只是将安装包信息从.pth 文件删除掉，安装包的脚本、包信息还在环境中，仅仅是导入时无法导入该安装包，若要完全移除安装包信息，需要手动删除安装包，详情参考源码方式卸载。

4、探针的配置

本小节主要讨论探针的本地配置，在安装包中提供了一份默认的配置文件 tingyun.ini（或者通过探针工具 generate-config 生成），该配置为标准 python ini 文件配置，详情参考 http://docs.python.org/library/configparser.html

注意：在使用时需要为该配置文件设置环境变量（TING_YUN_CONFIG_FILE），探针方能启动。 配置文件 tingyun.ini 中除了 license key 之外均有默认值，为了正常使用探针，license key 是必填项， 更多配置请参考下述说明。

Section	配置项（区分大小写）	备注
tingyun	license_key	字符型
	enabled	Boolean 型
	app_name	字符型
	log_file	字符型
	audit_mode	Boolean 型
	log_level	字符型
	ssl	Boolean 型
	auto_action_naming	Boolean 型
	action_tracer.log_sql	Boolean 型
tingyun:private	host	字符型
	port	正整数
tingyun:proxy	proxy_scheme	代理主机数据传输协议
	proxy_host	代理主机
	proxy_port	代理主机端口号
	proxy_user	代理账户
	proxy_pwd	代理账户所需密码

注：本地配置的修改需重启方能生效！

4.1 本地配置

采用标准的 ini 文件配置，标题书写格式为： [secionName] optionName

4.1.1 [tingyun] license_key

功能：账户认证标识，**必填项，使用探针前请务必填写该项**。
默认值： 无
说明：如缺少该配置项、或者配置项错误，探针能正常启动，log 输出会提示 license 无效，将不会采集并上报数据。

4.1.2 [tingyun] enabled

功能：客户端开启、禁用探针开关。
默认值：True，可选值 True, False, 以及 on, off，不区分大小写。
说明：如缺少该配置项、或者配置项错误，探针将使用值 True，开启探针功能。

4.1.3 [tingyun] app_name：

功能： 监控的应用名字
默认值：Python App
说明： 如缺少该配置项、空值、错误值等，将使用值 Python App 作为应用的名字上报数据。
探针支持多应用关连，该值可以设置为**英文分号**分割的多个应用名称。

4.1.4 [tingyun] log_file

功能： 指定探针 log 写入的文件名以及路径，**推荐使用绝对路径**。
默认值：/tmp/tingyun-agent.log，支持自定义系统文件路径、stdout、stderr
说明：若缺少该配置、空值、错误值等，探针可正常启动，log 将会输出到 stderr。

若指定了 stdout、或者 stderr，将会定向到系统标准输出。

由于 python 的 log 分割机制有缺陷，探针将会向一份日志文件中输出日志，请做好日志处理。

请确保您应用进程的用户对该目录和 log 文件有写入权限，否则将会输出到 stderr。

4.1.5 [tingyun] log_level

功能：指定探针 log 的日志级别
默认值：INFO，可选值 NOTSET, DEBUG，INFO, WARNING，WARN, FATAL ,ERROR，CRITICAL (不区分大小写)。
说明：若缺少该配置，或错误配置将默认使用 INFO 级别。

4.1.6 [tingyun] ssl

功能： 指定使用 http 或 https 传输协议
默认值：True，可选值 True, False, 以及 on, off，不区分大小写。
说明： 若缺少该配置、配置错误等，将使用值 True，使用 https 协议传输数据。

4.1.7 [tingyun] audit_mode

功能：是否将提交上报的数据将会输出到 log 中以备审计，False 关闭审计，反之开启。（该 log 以 INFO 级别输出）
默认值：False，可选值 True, False, 以及 on, off，不区分大小写。
说明：若缺少该配置、配置错误等，将使用值 False，关闭审计模式。该部分 log 将以“Agent capture”开头， info 级别输出。

4.1.8 [tingyun] auto_action_naming

功能：设置是否开启自动事务命名，如开启自动命名，uri 名字将会做为 action 的名字。

默认值： True，可选值 True, False, 以及 on, off，不区分大小写。

说明：若缺少该配置、配置错误等，将使用值 True，开启自动命名。

4.1.9 [tingyun] action_tracer.log_sql

功能：事务跟踪时 SQL 语句的记录只写到本地日志文件中，不提交到数据采集服务上。（输出级别为 INFO 级别 log）

默认值： False，可选值 True, False, 以及 on, off，不区分大小写。

说明：若缺少该配置、配置错误等，将使用值 False，该部分 log 将以“Log sql is opened”开头。

4.1.20 [tingyun:private] host

敬告：需要私有化探针时才需配置此选项，常规用户无需理会该选项！

功能：私有化时，用于配置内网的服务器重定向地址。

提示：如果配置文件没有 section [tingyun:private]，请手动添加该 section，然后再配置 host 等选项。

4.1.21 [tingyun:private] port

敬告：需要私有化探针时才需配置此选项，常规用户无需理会该选项！

功能：私有化时，用于配置内网的服务器用于重定向地址开放的端口号。

4.1.22 [tingyun] urls_merge

功能：在关闭自动事务命名时，合并 uri 作为事务名称。
默认值： True，可选值 True, False, 以及 on, off，不区分大小写。

说明：
该参数用于开启 url 自动合并，仅当配置自动事务命名关闭，该选项开启时 url 自动合并才生效，其命名规则如下：
- uri 中除字符'/'外，出现的连续数字的将会被字符'*'替换。
- uri 中字符'/'中间部分的值为数字的将会被字符'*'替换。

4.1.23 [tingyun] verify_certification

功能：是否验证探针服务器网站证书。
默认值：False，当启用 ssl 加密传输数据时，不验证探针服务器的网站的证书信息。可选值 True, False, 以及 on, off，不区分大小写。

说明：
如果 python 包 certifi 的版本大于 2015.04.28 时，该证书使用 sha256 加密认证网站证书信息，但由于探针服务器证书颁发机构的根证书使用的是 sha1 加密，所以使用该版本之后的版本将会导致认证服务器证书失败的，以至于上传探针监测的数据失败，所以暂时的解决方案是，关闭服务器证书的认证。

4.1.24 [tingyun:exclude] plugins

功能：配置不需要监测的 python 包
默认值：空
说明：plugin 已英文半角逗号分隔，其支持的**包名**称如表 4-1-24 所示,使用示例:
[tingyun:exclude]
plugins=mysql,memcahced

表 4-1-24

数据库包名	框架包名	对外调用包名	模板以及其他
mysql	django	urllib	jinja2
pymysql	flask	urllib2	mako
oursql	web2py	urllib3	gevent
oracle	webpy	thrift	
postgresql	tornado4	requests	
psycopg2	nova	httplib2	
psycopg2ct			
psycopg2cffi			
pyodbc			
mongodb			
redis			
memcached			

4.1.25 [tingyun:proxy] proxy_host

功能：配置 http/https 代理服务器主机地址,用户探针链接外网上报性能数据。
默认值： 无
说明：该配置可与配置项 proxy_port, proxy_user, proxy_pwd 配合使用， 该选项支持格式：schme://user:password@host/path

4.1.26 [tingyun:proxy] proxy_port

功能：配置代理服务器主机端口号
默认值：无

4.1.25 [tingyun:proxy] proxy_user

功能：配置代理服务器用户名
默认值：无

4.1.26 [tingyun:proxy] proxy_pwd

功能：配置代理服务器用户密码，该选项和 proxy_user 对应。
默认值：无

4.1.26 [tingyun:proxy] proxy_scheme

功能：配置代理服务器数据传输协议。
默认值：http

4.1.27 [tingyun] tornado_wsgi_adapter_mode

功能：启用 tornado wsgi 应用模式
默认值：False， 支持 false， true， on， off，不区分大小写。
说明：
当使用第三方容器部署（如 gevent，uwsgi）tornado 应用（tornado wsgi 应用）时，需要开启该选项，探针才能正常工作。

4.2 环境变量

4.2.1 TING_YUN_CONFIG_FILE

该环境变量用于设置配置文件的路径，探针在启动的时候需要加载该文件，如 export TING_YUN_CONFIG_FILE='/tmp/tingyun.ini'。文件名跟路径随意配置，**唯一要确保的就是应用程序运行用户需要有权限去读取该文件**。

配置该环境变量后，可以使用探针工具 tingyun-admin check-config 来检查该设置是否正确。

注意：使用该命令是需要应用程序运行的用户操作，否则测试结果可能不准确。

4.2.2 DISABLE_TING_YUN_AGENT

该变量如果出现在 HTTP 请求的 HEADER 中，并且值为"真"，则该次请求的性能值将被忽略。

4.3 请求阻塞时间

阻塞时间用于提取客户端发起请求到应用处理请求之间阻塞的时间， 即 **请求阻塞时间 = 请求到达前端服务器 到 应用服务器收到请求的时间**

只需在 web Server 中配置相应的头即可，根据不同的应用服务器配置方案如下：

Web Server	App Server	Web Server 配置项
Nginx	uwsgi	uwsgi_param HTTP_X_QUEUE_START "s=$msec";
	tornado	proxy_set_header X-QUEUE-START "s=$msec";
	gunicorn	proxy_set_header X-QUEUE-START "s=$msec";
	gevent	proxy_set_header X-QUEUE-START "s=$msec";
	FASTCGI	fastcgi_param HTTP_X_QUEUE_START "s=$msec";
	SCGI	scgi_param HTTP_X_QUEUE_START "s=$msec";
Apache	mod_wsgi	RequestHeader set X-QUEUE-START "%t"

三、使用与维护

为了响应更多的需求，我们将会根据客户的反应、需求、以及探针自身的因素，进行不定期的升级。为了更好的用户使用体验，升级后我们会第一时间通知您，届时请升级到最新版。同时也欢迎您向我们提供宝贵的意见与建议！

反馈信息邮箱：python@tingyun.com

1、局限性、限制

※ 由类似 greenlet 等第三方协程、线程调用模块。其监测性能数据可能会有偏差.

※ 不支持未在本文档内注明支持的其他框架、模块。

※ 随着探针升级更多的框架、组件将得到支持，敬请期待。

2、探针使用

首先，在探针安装完之后，需要对探针进行配置，源代码中包含的配置文件需要手动添加 licensekey，或者使用 generate-config 生成配置文件。

然后，对配置文件设置环境变量 TING_YUN_CONFIG_FILE，最后启动探针即可，具体使用方式如下。

2.1 探针启动

2.1.1 一键嵌入探针

在设置好配置文件之后，通过 linux 命令行可一键嵌入 python 探针，使用方式如下：

tingyun-admin run-program 您应用启动命令 您应用启动参数

例如：

※ tingyun-admin run-program python manage.py runserver

※ tingyun-admin run-program uwsgi_shell.sh start

2.2.2 手动探针嵌码

手动探针嵌码需要在您的应用的关键地方调用探针的 API， 关于 API 请参阅<探针 API>章节。手动嵌码大致有以下逻辑：

初始化探针 >> 包装 WSGI 程序入口 >> 包装函数/方法

注意事项：

初始化探针， 需在所有执行用户代码之前操作，否则可能不能检测到某些 python 模块。

WSGI 程序入口，需要在 WSGI 入口处调用，如非标准 WSGI 程序暂不支持手动嵌码。

本文档注明支持的 web 框架以及组件请勿手动嵌码，否则可能造成数据冗余、探针异常等问题。

2.2 探针停止

探针会随着您应用程序的停止而停止，所以大致有以下两种正常的停止方式：

※ 直接使用您应用程序的停止方式停止探针，如 uwsgi_shell.sh stop

※ 使用探针命令方式停止探针，如 tingyun-admin run-program uwsgi_shell.sh stop

2.3 探针工具

由于探针使用了配置文件，以及环境变量方式配置探针，该工具用于检测探针配置的合法性，其使用方式如下：

探针命令行命令	探针命令参数 1	探针命令参数 2	探针命令参数 3
tingyun-admin	check-config	[config_file_path]	
	generate-config	[license_key]	output_file

注意：以上命令不会对探针的 license 的合法性、有效性做检查

check-config 示例：

1、如果您对探针的配置文件，设置了环境变量，使用默认参数即可对配置文件进行检测和校验，其命令如下：

tingyun-admin check-config

2、对指定的配置文件进行校验（假设探针配置文件放置在/tmp/目录下）。

tingyun-admin check-config /tmp/tingyun.ini

3、注意事项，运行该命令的用户需要和您应用程序进程所属用户一致，才能有效的检查配置是否合理。

generate-config 示例：

1、提供 license ，此时 license 会自动写入配置文件中，output_file 为必须选项。

tingyun-admin generate-config 123-456-789-10 /tmp/tingyun.ini

2、稍后修改 license，output_file 为必须选项.

tingyun-admin generate-config /tmp/tingyun.ini

若要修改更详细的配置，需手动修改，请参考<本地配置>

3 私有化

警告：普通用户无需关心此选项，如需私有化请联系我们的私有化团队

Python 探针支持私有化部署，只需在配置文件中增加以下配置即可：

```
[tingyun:private]
host=YourprivateServer
port=YourprivatePort
```

详情参考配置选项安装与配置的 4.2.0、4.2.1

4 探针 API

探针支持的用户 API 如下所示，所有的 API 在代码处均有示例代码，建议参阅源代码，后期会逐步开放更多 API 的支持，敬请期待。

所在模块以及函数	功能
tingyun.api.init_tingyun_agent	初始化探针
tingyun.api.wsgi_app_decorator	打包 WSGI 程序入口
tingyun.api.function_trace_decorator	函数/方法性能追踪

4.1 init_tingyun_agent

init_tingyun_agent(config_file=None):
功能：初始化探针程序。
参数：
config_file: 探针的配置文件路径，建议使用绝对路径。如果未提供该参数，探针会寻找环境变量`TING_YUN_CONFIG_FILE`指向的配置文件。若两个地方都没有提供，探针将不会工作。
返回值：无
使用方式：直接调用
注意事项：
必须在所有用户代码之前调用。

4.2 wsgi_app_decorator

wsgi_app_decorator(framework='xx', version='xx')
功能：打包 WSGI 应用入口
参数：
framework：探针所在框架，默认为`xx', 若无框架类别可不填或自定义。
version: 该框架（或自定义框架）使用的版本号，默认值为 `xx`。
返回值: 函数
使用方式:
装饰器调用,如@wsgi_app_decorator(framework='customer', version='0.1')

4.3 function_trace_decorator

function_trace_decorator(name=None, group=None)
功能：函数性能追踪
参数：
name：被追踪函数的名字，如未提供，则使用当前被追踪函数名字作为该性能追踪的名称。
group: 被追踪函数的分组， 未提供则分组为`Function`
返回值: 函数
使用方式:
装饰器调用,如@function_trace_decorator(name='home_page', group='views')

5 探针的维护

如果配置了 log（建议配置 log），探针的所有信息将会根据设定的日志级别输出到 log 中，如果发现有任何异常，可根据 log 输出，获取更多的支持和帮助。

若需获取更多的帮助，请联系我们的技术支持。

四、模块支持

听云 python 探针是根据 WSGI 协议而为 web 框架定制的性能监测客户端，理论上基于只要是基于 WSGI 协议的 web 框架都能对其监测。当然其中也包括一些该 web 框架支持的一些 python 的第三方包，例如 sql 数据库、nosql 数据库、第三方 http 调用等。其功能会随着版本的升级而得到更多的支持，目前其支持的组件以及框架结构如下所述。

目前所有的组件，包括数据库、第三方调用等，暂时没有提供独立的 api，都依赖于 web 框架而存在。

虽然所有基于下述框架下所有 wsgi 应用都可以支持， 下述的所有支持，均经过验证和测试的。如果您发现有下述框架下的 wsgi 应用有不支持的情况，请向联系我们。

1 框架

1.1 支持的 web 框架

支持框架	建议版本	备注
django	1.4.x - 1.7.x	
flask	0.6 以上	
Webpy	0.3.x	
Web2py	2.8.1-2.12.x	
Tornado	v3.x.x-v4.x.x	不支持 browser 探针自动嵌码
bottle	0.10.x-0.12.x	

1.2 支持组件以及部署方式

框架性能监测组件	部署方式
请求中间件	Uwsgi（必须开启--enable-threads 和 --single-interpreter 选项）
视图中间件	gunicorn
模板响应中间件	mod_wsgi
响应中间件	AJP
异常中间件	FASTCGIT
视图函数、模板渲染等	SCGI
异常捕获	独立的 WSGI 容器
其他	CGI

2 数据库支持

Python 探针目前支持以下数据库性能监测，若您站点使用其他类型的数据库 api，其数据库的性能将不会被采集。随着探针的升级，更多的模块将会支持。

数据库	数据库模块	备注
mysql	mysql-python	(1.2.3-1.2.5)
	pymysql	(0.6.x-0.7.x)
oracle	cx_Oracle	(5.1.x-5.2.x)
PostgreSQL	psycopg2	(2.3.x-2.6.x)
	psycopg2ct	psycopg2ct(0.2.x-2.x)
	psycopg2cffi	(2.5.x-2.7.x)
ODBC	pyodbc	(2.1.x-3.0.x)
Memcached	python-memcached	(1.4.x-1.5.x)
MongoDB	pymongo	(2.x-3.x)
Redis	redis	

3 外部调用支持

仅支持以下列表中的组件的性能采集，若您站点使用其他类型的外部调用模块，其性能数据库将不会被采集。随着探针的升级，更多的模块将会支持。

模块	备注
urllib	
urllib2	
urllib3	对外调用时，支持跨应用追踪
requests	对外调用时，支持跨应用追踪
thrift	
httplib2	对外调用时，支持跨应用追踪

4 python 与系统平台

python	平台
Cpython2.6.x 2.7.x	类 unix 系统，如 linux、FreebSD、MacOS X 等

5 其他部分

模块	备注
Jinja2	2.3 以上版本
mako	v0.7.x-v1.0.x
django-piston	0.2.x

五、部署详解与常见问题

1、部署详解

下列步骤为正常情况下探针部署运行的步骤，这里假设探针已经安装成功，安装步骤参考 探针安装。

下列步骤有些是可选的，均已（可选）标注。以下步骤仅适用于一键嵌入方式部署探针。

1.1 生成配置文件（可选）

探针安装包中有一份自带的 tingyun.ini 配置文件，源码安装在修改后配置后，可以直接使用该文件，也可以通过命令生成配置文件。

使用命令：tingyun-admin generate-config 可生成配置文件，详情参考探针工具，使用在线安装时该命令尤其有用。

1.2 为探针配置环境变量

对探针设置配置文件有两种方案，为了最大的灵活性，推荐使用第一种，两种方式如下：

方式一：设置环境变量，详情参考 环境变量 TING_YUN_CONFIG_FILE。

方式二：不推荐，设置绝对配置文件路径，'/opt/tingyun.ini'，确保该文件存在，并且运行程序用有权限读取。如果设置了探针配置文件环境变量，该配置文件不生效。

1.3 检查配置文件（可选）

使用命令 tingyun-admin check-config 可检查配置文件的有效性。

1.4 用探针启动用户应用

使用 tingyun-admin run-program 启动用户应用程序。执行该操作后，正常情况下探针 log（探针配置文件中配置）会有以下类似 log 信息输出。

```
... tingyun.embattle.inspection 154 INFO - Detect hooker tingyun.armoury ...
    ..........
... tingyun.logistics.transportation.engine 134 INFO - create session with license.
... Successful register application to agent server with name: ....
    ..........
... tingyun.battlefield.dispatcher 166 INFO - Spend * to harvest all applications
```

下面对异常情况一一列举说明：

一、 探针 log 中没有任何 log 信息输出，且一段时间后报表无数据（有延时）

描述：探针 log 中没有任何 log 输出，除了”agent check log file config input message”,该消息为使用 tingyun-admin check-config 命令生成。

可能性一：探针没有启动，探针安装的 python 包位置与应用使用的 python 包的位置不一致导致。

如，探针安装在了系统默认的 python 包目录下，而用户应用程序使用了虚拟环境。

解决方案：将探针装在用户应用使用的虚拟环境中。

可能性二：探针为启动，配置文件读取权限问题，或者指向配置文件不存在导致。

如，探针配置文件只有 root 能读取，但实际应用程序运行用户为 test，则读取失败，此时用户应用的 log 中将会有一条错误信息输出。

解决方案：设置正确的配置文件，可使用 tingyun-admin check-config 检查

二、 探针 log 中没有任何 log 信息输出，且一段时间后报表有数据（有延时）

可能性一：探针配置文件中配置的探针 log 文件对于客户应用程序运行用户没有权限写入。

如，探针 log 的权限为 root 可写，其他用户可读，但应用程序运行用户为 test，则无法写入 log 信息。

解决方案：对探针 log 设置正确的权限。

三、探针 log 仅有部分 log 输出，且一段时间后报表有数据（有延时）

探针 log 中仅有类似于”... tingyun.embattle.inspection 154 INFO - Detect hooker tingyun.armoury ... ”这样的 log

可能性一：用户应用启动后，对 python 的 logging 模块做了配置，使用了 disable_existing_loggers=True 参数，禁止掉了第三方 log 的输出。

或者，python logging 模块优先初始化了探针 log，而后监控的应用程序又调用了 logging.config.fileConfig()函数。

解决方案：设置 disable_existing_loggers=False

2、探针其他常见问题

1、更新了本地配置，为什么没有生效？

如果更新本地配置文件，探针系统不能自动识别，目前解决方案为：重启应用（探针也被重启）。

2、怎样确认探针配置文件、配置选项是否正确、合理

我们提供了命令行工具 tingyun-admin check-config 检查配置文件，详情参考<探针工具>

3、私有化时，探针部署成功，log 中一堆 license 无效的错误

部署私有化时，没有对 host、port 参数，设置 section，导致数据往公网服务器上报所致。

解决方案，对私有化设置 section 段。

4、报表显示采集的请求响应时间信息误差偏大

这种情况很可能是因为探针没有采集到应用阻塞时间，整个请求响应时间 阻塞时间 + 应用层处理时间。

解决方案，在 web 服务器设置上请求头，以便探针处理阻塞时间。详情参考请求阻塞时间

附加说明

该章节对于一些探针的技术点、限制、特点等进行一些详细说明。

1、关于 uwsgi 的说明

--enable-threads，uWSGI 默认情况下，没有开启多线程的支持，由于探针基于多线程模式，所以必须开启此模式，否则探针无法正常启动。如果 uwsgi 使用了选项—threads，将自动开启选项--enable-threads。

--single-interpreter， 默认情况下 uWSGI 为了隔离应用环境，以便运行多个应用，在启动的时候会启动子进程来运行某个应用，而不是在主进程中执行。 为了更好的适应探针环境请开启这个选项， 开启后不会有其他任何的副作用，它也是安全而有效的方式。

2、关于 tornado 的说明

Tornado 即是 web 容器，又是 web 框架，建议使用 tornado 自身容器 IOLoop 方式部署， 探针默认此规则。

当使用第三方容器，如 gevent、uwsgi 部署时，需要在配置文件配置 tornado wsgi 模式探针才能正常工作，详情参考配置[tingyun] tornado_wsgi_adapter_mode