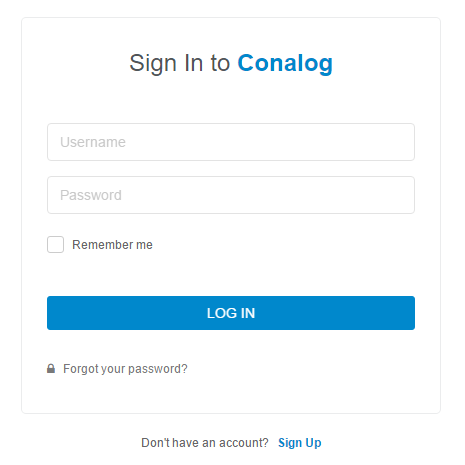
**Conalog User Manual**

*Version Beta1*

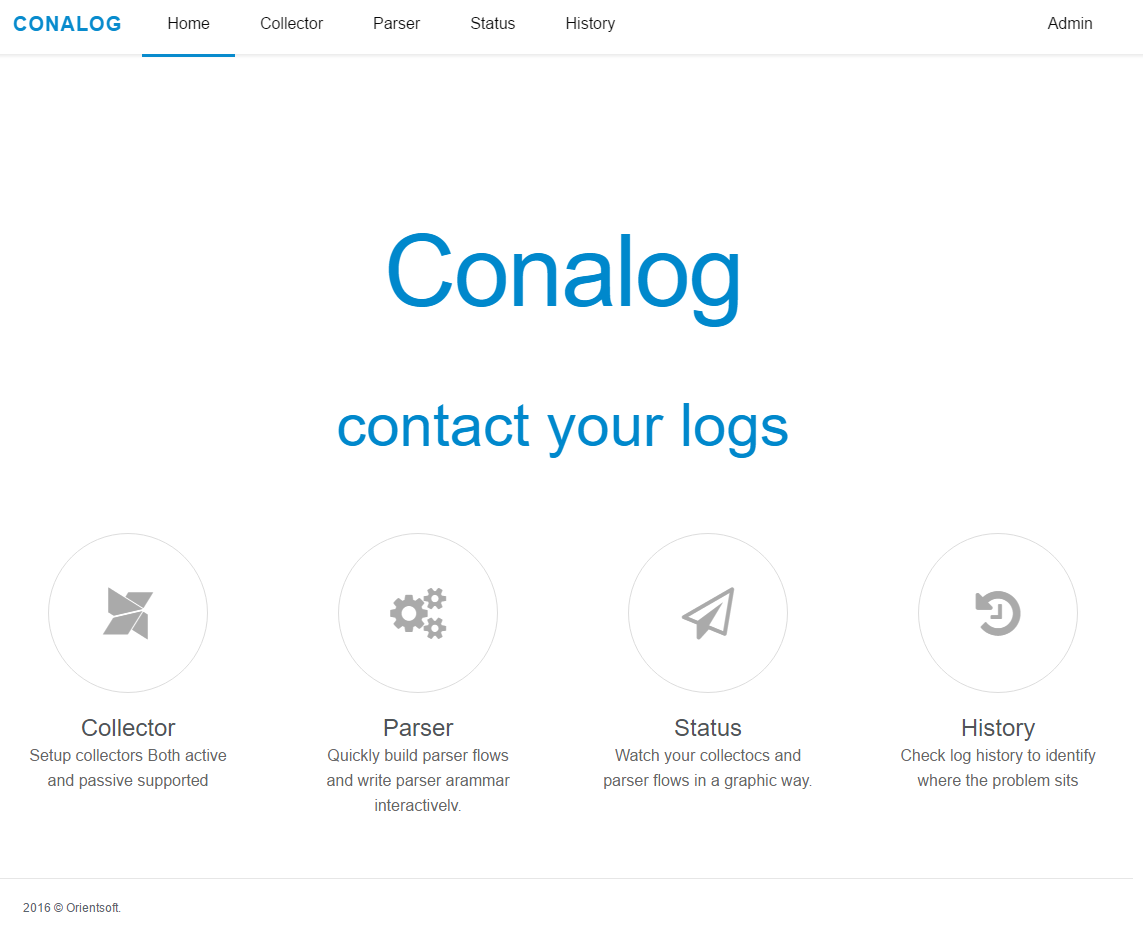
*2016.07.18*

**1. 用户验证**



默认用户名密码：admin/admininitpass。

**2. 首页**



从首页可以访问各个子功能模块，查看子功能模块的简介。

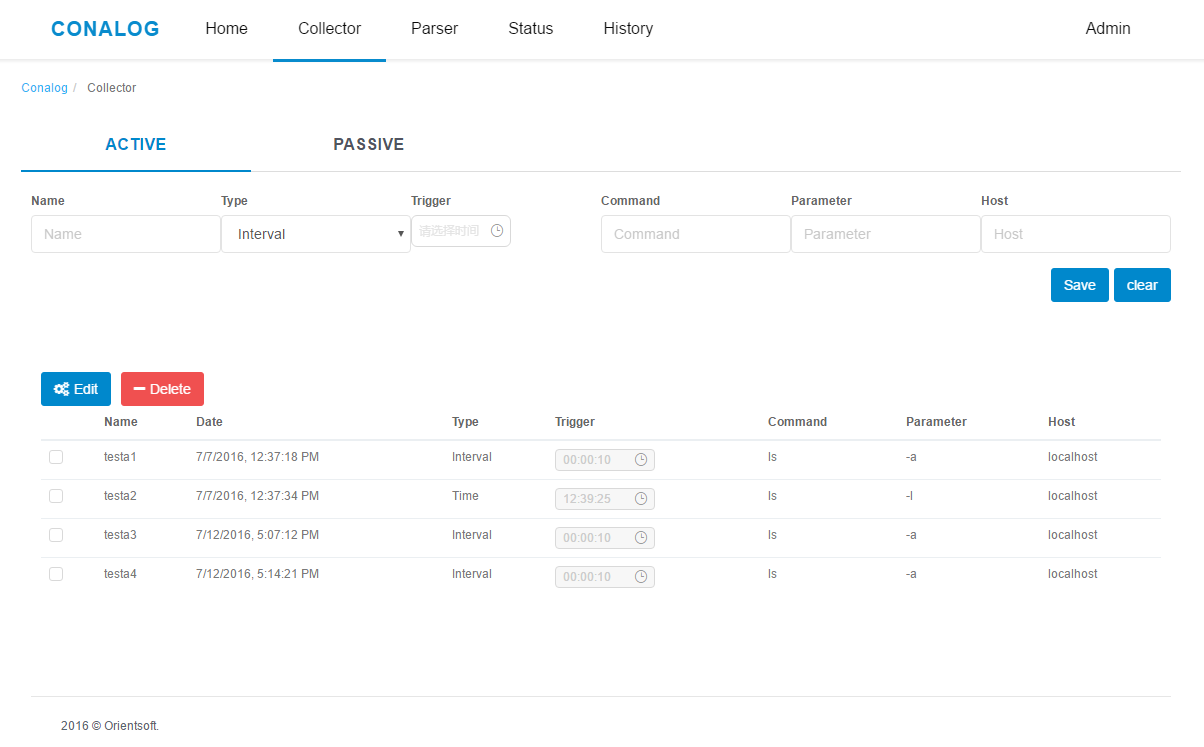
**3. Collector**

Collector分为两类，Active和Passive，分别提供了两个界面进行管理。

Collector将采集到的数据发送到对应的Redis通道上，ActiveCollector默认通道名为ac\_[Collector的名字]，PassiveCollector默认通道名为pc\_[Collector的名字]。对于错误信息，默认的通道名为：ac/pc\_err\_[Collector的名字]。所以同类型的Collector不能重名（若重名系统会提示错误，拒绝添加）。

Collector界面提供了详情表单和Collector列表。详情表单可以新建Collector，勾选Collector列表中的项，可以进行编辑、删除操作，编辑操作会加载**第一条**勾选的数据到详情表单中；删除操作在确认后，会一次性删除**所有**选定的项。

**3.1 ActiveCollector**



ActiveCollector指的是，每次调用都会在**有限时间内返回数据并退出**的Collector。比如，从数据库定期采集数据的脚本Collector，调用之后连接数据库读取数据，完成之后立即返回退出。（如果在ActiveCollector配置使用长时间执行的脚本、tail -f命令等，则Collector状态会一直卡在Pending，这种情况应该使用PassiveCollector进行管理。）

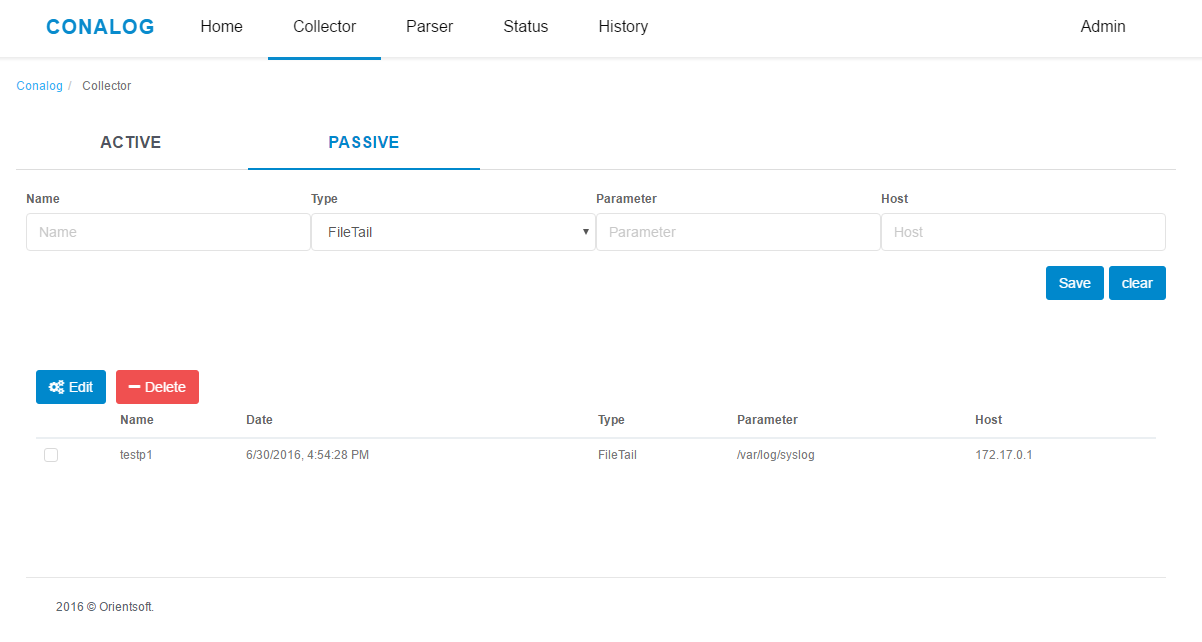
ActiveCollector应该读取命令行参数，并将数据输出到STDOUT和STDERR上。STDOUT上的数据会发送到正常数据的Redis通道；STDERR上的信息会被认为是错误信息，自动发送到错误信息Redis通道。不管是正常消息还是错误信息，Conalog都会**按行原文**发送到Redis通道，留待后续Parser等进行处理。

可以设置ActiveCollector的Type为周期性的调用（Interval），也可以设置为每天定时调用（Time）。

Parameter参数会被作为命令行参数，在Collector启动时加载。

（\*2016.07.18注，目前ActiveCollector 不支持远程执行，所以Host参数请设置为localhost。Issue详情跟踪：<https://github.com/Orientsoft/conalog/issues/19>。）

**3.2 PassiveCollector**



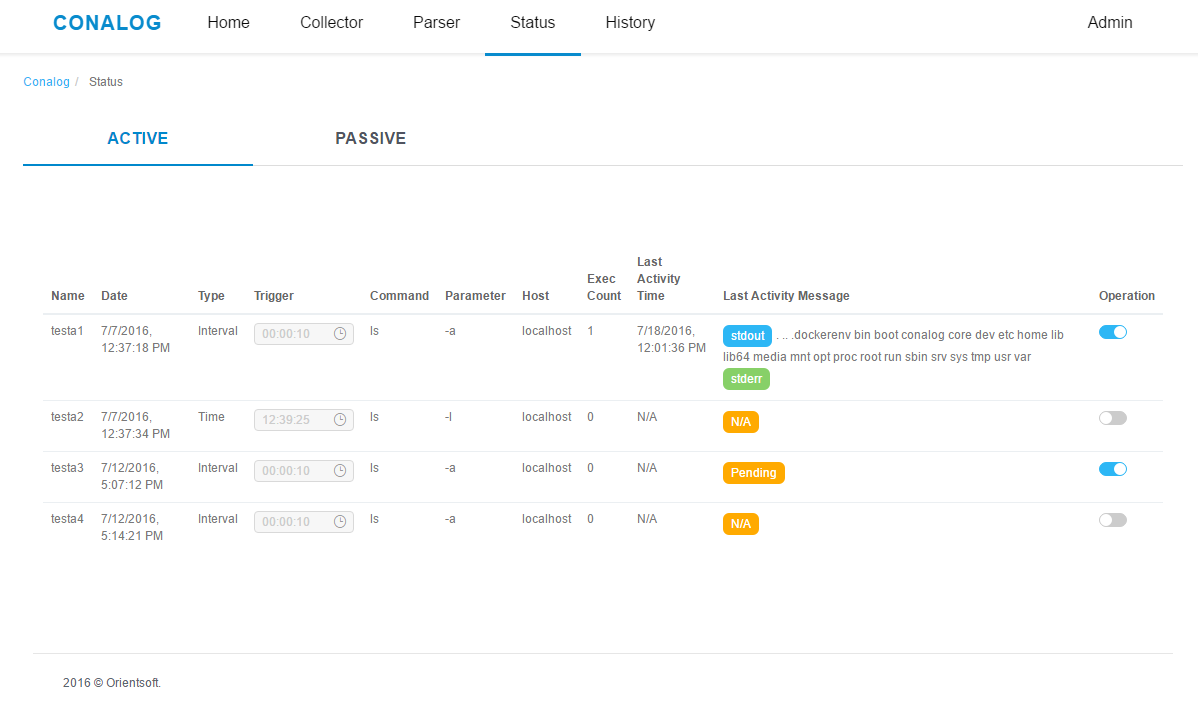
PassiveCollector指的是，每次调用不会退出，而是**以行的形式长期输出数据**的Collector。比如，tail -f命令。（如果在PassiveCollector配置使用短期执行脚本，则PassiveCollector的状态会反复重置。）

PassiveCollector目前支持FileTail模式，请配置Parameter参数为需要监听的文件名。

由于很多需要监听的文件位于客户机上，PassiveCollector提供了远程连接能力。配置Host参数为客户机地址，则PassiveCollector会通过SSH连接到远程机器上，并远程调用tail -f命令。如果是Docker启动的Conalog并且希望监听宿主机的文件，需要设置Host参数为宿主机在容器内的网桥地址（一般为容器IP网段的xxx.xxx.xxx.1）。

（\*2016.07.18注，为了保证远程连接的安全性，需要在后台配置文件中列出远程SSH用户名/密码，未列出的主机无法连接。Beta1阶段暂时需要手工在前台填入设好的Host，之后的测试版会自动加载后台配置好的Host列表。另外，也会加入对用户名/密码加密的功能，相关Issue详情：<https://github.com/Orientsoft/conalog/issues/13>、<https://github.com/Orientsoft/conalog/issues/6>。）

**4. Status**



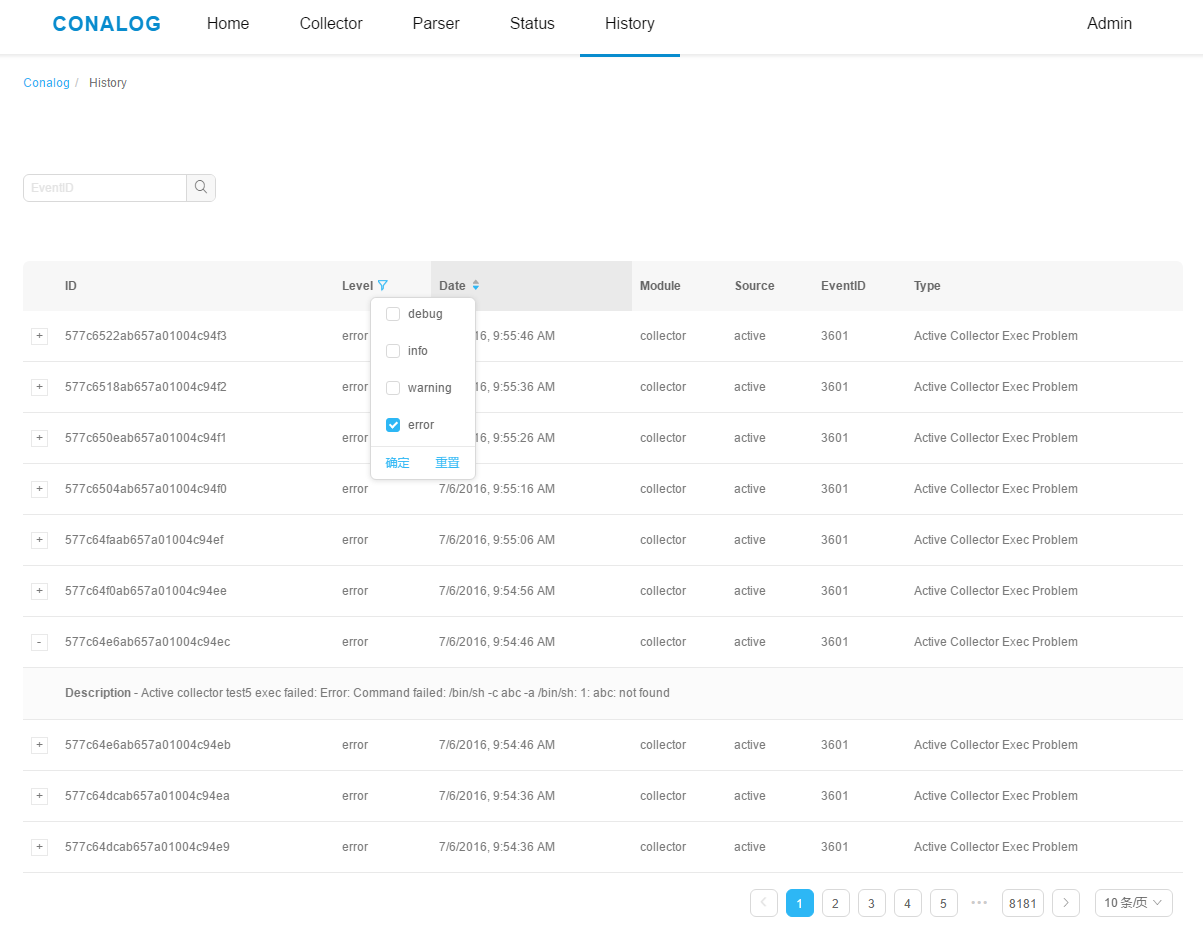
Status可以实时展现所有Collector的状态，也可以启停Collector。

可以在Status界面快速检阅Collector最后执行的状态，其中Last Activity Time说明了最后一条数据的时间，而Last Activity Message给出了最后一条数据的内容。对于ActiveCollector来说，Exec Count就是脚本或命令被调用的次数；对于PassiveCollector来说，是采集到的数据的行数。

可以通过开关对Collector进行启停操作，停止后最后执行状态清零，启动后重新开始计算。

Conalog的Collector执行状态集中保存在外部状态缓存上（目前是Redis），当Conalog系统重启之后，会自动启动所有原来正在执行的Collector，无需人工介入。

**5. History**



History模块展示了Conalog系统本身的日志。这些日志直接保存在MongoDB中，便于检索分析。

用户可以给日志等级加过滤器，也可以根据日期排序，对于某个单独的日志类型，可以直接通过EventID进行搜索。用户还可以展开单条日志查看该日志的详细信息。