# Gearpump–TAP Integration Design Document

Terminology

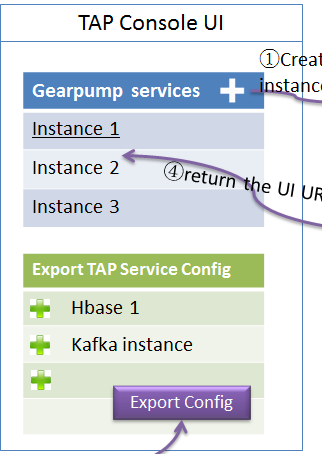
|  |  |  |
| --- | --- | --- |
| TAP | Means | Trusted Analytics Platform |
|  |  |  |
|  |  |  |

## Overview



## Module Requirements Specification

### TAP Console UI



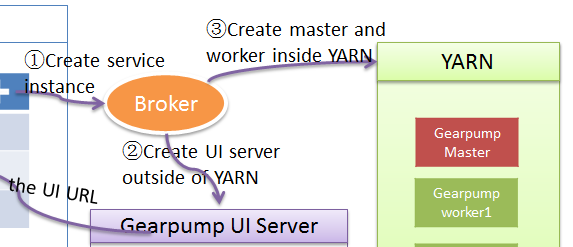
**Create new ICON for Gearpump:**

1. A new Gearpump ICON in market place, user are allowed to create a gearpump service instance by clicking the ICON.
2. When user create new service instance, it should call the broker to create the gearpump cluster. The broker will return the UI server address of this gearpump cluster.
3. TAP console will show the UI server address in TAP console UI.

**Create new UI page in TAP console to export service configuration:**

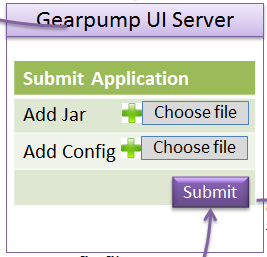
1. A new page named like “services” should be created in TAP console UI.
2. In this page, user can pick the service instance he is interested of.
3. User can export the configuration for this TAP service instances.
4. The exported configuration contains all necessary information to access the TAP services, like HBase, or Kafka.
5. The schema of this configuration file is defined at <https://github.com/gearpump/gearpump_tap_integration/blob/master/src/main/java/io/gearpump/tap/TapConfig.java>
6. There should provide a Util to convert this configuration file to something HBaseSink or KafkaSink can understand.

### Gearpump Service Broker



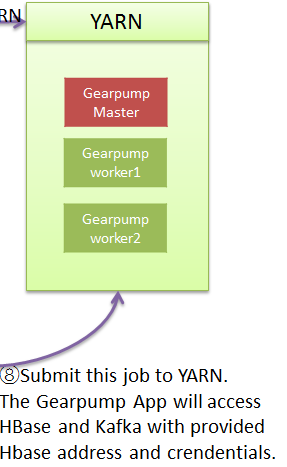
1. Upon receiving “create service instance” request from TAP console UI, the broker should be be invoked to create the Gearpump cluster.
2. It shall use the YarnClient in Gearpump to start the Gearpump cluster(master and worker) on YARN. And the YARN client should return some URL of master address.
3. It should use the master address in above step to create the Gearpump UI server **outside of YARN.**
4. It should report the UI server back to TAP, and show it as a link in TAP UI console.
5. User can access the Gearpump UI server by following the link in TAP UI Console.

### Gearpump UI Server



1. The Gearpump UI server should allow user to submit a jar along with configuration file.
2. The configuration contains all information to access TAP service like HBase service, or Kafka service.
3. The jar will parse the above configuration, and transform it to object which KafkaSink and HBaseSink can understand.
4. Gearpump UI server should be able to authentication itself by simple authentication.
5. It is bonus point if Gearpump UI can by authenticated by OAUTH.

### Gearpump YARN Module



1. Gearpump master and worker daemons shall be able to be started in YARN environment, which is protected by Kerberos.
2. For the application run inside YARN, it should be able to access HBase, which is protected by Kerberos.
3. If there are some secret file needed in this process, the platform need to make sure the secret is well protected, without the risk of information leakage.

### User Written Gearpump DAG application:

1. To use the TAP service like HBase, or Kafka. User are needed to modify their implementation of Gearpump application.
2. It is expected that the Main class will:
   1. Parse the TAP service configuration file, which is exported from the TAP console UI that contains address and credential information.
   2. The TAP configuration file is submitted as normal HOCON file. There is **NO** additional MANIFEST file needed.
   3. After the parsing, the MainClass should translate the TAP configuration to the format Which Gearpump DAG Sink can understand. For example, for Gearpump HBaseSink, it will require a org.apache.commons.Configuration, so the MainClass should translate the TAP Config to org.apache.commons.Configuration.
   4. After the configuration conversion, the MainClass can compose the DAG, and submit that to the Gearpump UI server.
3. During this process, make sure the secret keytab files are well protected.

## Work Item breakdown

See other meeting notes sent by Kam.