Contents

1 Sentry Authentication

1.1 LDAP Installation

1.2 Hive Configuration

2 Hive Debug

3 Hive Hook

3.1 Hooks lifecycle

There are several types of hooks depending on at which stage you want to inject your custom code:

- Driver run hooks (Pre/Post)
- Semantic analyzer hooks (Pre/Post)
- Execution hooks (Pre/Failure/Post)
- Client statistics publisher

If you run a script the processing flow looks like as follows:

1. **Driver.run()** takes the command
2. **HiveDriverRunHook.preDriverRun()** `HiveConf.ConfVars.HIVE_DRIVER_RUN_HOOKS`

3. **Driver.compile()** starts processing the command: creates the abstract syntax tree
4. **AbstractSemanticAnalyzerHook.preAnalyze()** `AbstractSemanticAnalyzerHook.preAnaly`
5. Semantic analysis
6. **AbstractSemanticAnalyzerHook.postAnalyze()** `HiveConf.ConfVars.SEMANTIC_ANALYZER_`
7. Create and validate the query plan (physical plan)
8. **Driver.execute()** : ready to run the jobs
9. **ExecuteWithHookContext.run()** `HiveConf.ConfVars.PREEXECHOOKS`
10. **ExecDriver.execute()** runs all the jobs
11. For each job at every `HiveConf.ConfVars.HIVECOUNTERSPULLINTERVAL` interval:*ClientStatsPublisher.run()* is called to publish statistics `HiveConf.ConfVars.CLIENTSTA` If a task fails: **ExecuteWithHookContext.run()** `HiveConf.ConfVars.ONFAILUREHOOKS`
12. Finish all the tasks
13. **ExecuteWithHookContext.run()** `HiveConf.ConfVars.POSTEXECHOOKS`
14. Before returning the result **HiveDriverRunHook.postDriverRun()** `HiveConf.ConfVars.HIVE_DRIVER_RUN_HOOKS`
15. Return the result.

3.2 Usage in DDW

3.2.1 ExecuteWithHookContext

1. HiveConfmapreduce.job.queuenameCDH
2. HiveHistoryHive SQL

```
 /**
 * 1 ) pool,(mem/cpu),job
 * 2 )
 * Created by xkwu on 2017/1/17.
 */
public class CustomExecuteWithHookContext implements ExecuteWithHookContext {
    private static final Log LOG = LogFactory.getLog(CustomExecuteWithHookContext.class
```

```
    private static TailerTracker tailerTracker = new TailerTracker();

    static {
        tailerTracker.start();
    }
    @Override
    public void run(HookContext hookContext) throws Exception {
        try {
        UserGroupInformation ugi = hookContext.getUgi();
        hookContext.getConf().set("mapreduce.job.queuename", ugi.getUserName());//1)qu

            QueryPlan queryPlan = hookContext.getQueryPlan();
            int jobs = Utilities.getMRTasks(queryPlan.getRootTasks()).size();
            if (jobs > 0) {
                String histFileName = SessionState.get().getHiveHistory().getHistFileN
                tailerTracker.track(hookContext.getUserName(), histFileName);//2)HiveH
            }
        } catch (Exception e) {
            LOG.error(e.getMessage(), e);
        }
    }
}
```

3.2.2 AbstractSemanticAnalyzerHook

1. drop databases,create databases
2. table

```
/**
 *
 * ,.
 *
 *
 * Created by xkwu on 2017/1/18.
 */
public class CustomSemanticAnalyzerHook extends AbstractSemanticAnalyzerHook {
    private static final Log LOG = LogFactory.getLog(CustomSemanticAnalyzerHook.class)
    private Pattern pattern = Pattern.compile("(STORED\\s+?AS\\s+?)(parquet|rcfile|orc

    @Override
```

```
    public ASTNode preAnalyze(HiveSemanticAnalyzerHookContext context,
                              ASTNode ast) throws SemanticException {
        switch (ast.getToken().getType()) {
            case HiveParser.TOK_CREATETABLE:
                Matcher m = pattern.matcher(context.getCommand());
                if (!m.find()) {
                    LOG.info(context.getUserName() + ": " + context.getCommand());
                    new EmailThread("/hive/notifyAboutCreateTable", "cmd=" + context.g
                }
                break;
            case HiveParser.TOK_CREATEDATABASE:
            case HiveParser.TOK_DROPDATABASE:
                String adminAccount = ConfigUtils.prop.getProperty("ADMIN_ACCOUNT");
                if (StringUtils.isEmpty(context.getUserName()) || !context.getUserName
                    throw new SemanticException("Only the admin accounts of ddw bigdat
                }
                break;
            default:
                break;
        }
        return ast;
    }
}
```