在 Docker 中运行 Apache Phoenix 并启用远程调试

最近由于工作方面的原因需要解析 Apache Phoenix 底层的原始文件，也就是存在 HDFS 上的 HFile。但是由于 Phoenix 有自身的一套数据编码方式，但是由于本人对 Phoenix 这套根本就不熟悉，所以只能自己去看相关代码。但是 Apache Phoenix 是个大工程啊，不可能一个一个文件去找的，这会相当的慢。这时候我想到的是搭建一个 Phoenix 测试环境，然后调试。但是熟悉 Apache Phoenix 的同学肯定知道，如果想运行 Apache Phoenix，得部署 JDK、HDFS、Zookeeper、HBase、Phoenix！！！非常的繁琐。

为了简化测试环境的部署，一个可行的方案肯定是用 Docker。应该有小伙伴们也遇到这种问题，于是 Google 搜索了一遍确实发现了一些 Phoenix Docker 镜像。如 docker-phoenix、hbase-phoenix-docker 等。但是这些镜像已经很老了，基本都是几年前的，而我需要的是 Apache Phoenix 5.x，没办法，只能自己弄了。

于是基于 hbase-phoenix-docker 和 hbase-docker 两个 Docker 镜像，我弄了 Apache HBase 2.x 和 Apache Phoenix 5.x 的 Docker 镜像：

- https://github.com/397090770/hbase-phoenix-docker
- https://github.com/397090770/hbase-docker

注意：上面两个镜像都是用于测试的，在 Docker 中运行上面镜像会启动 HBase 伪分布式模式。如果需要线上使用，需要修改 HBase 的部署模式。

在 Docker 中运行 Apache Phoenix

有了上面的镜像之后，部署一个 Apache Phoenix 5.x 测试环境非常的容易，直接在命令行运行下面命令就可以将上面的 hbase-phoenix-docker 镜像拉到本地（这个和 git 的 pull 命令有点像）。

```
[iteblog@www.iteblog.com ~]$ docker pull iteblog/hbase-phoenix-docker:1.0
```

查看镜像是否下载完成

```
[iteblog@www.iteblog.com ~]$ docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
iteblog/hbase-phoenix-docker  1.0                 978d21a23ec3        21 hours ago        1.55GB
```

运行 Phoenix 镜像

```
[iteblog@www.iteblog.com ~]$ docker run -it -p 8765:8765 iteblog/hbase-phoenix-docker:1.0
```

注意，上面命令中的 -p 8765:8765 是将 Docker 镜像中的 8765 端口和本机的 8765 端口进行了映射，这样我们就可以直接在本地的终端的 8765 端口连接到 Phoenix QueryServer。
查看 Phoenix 镜像是否运行成功

```
[iteblog@www.iteblog.com ~]$  bin/sqlline-thin.py
Failed to find hbase executable on PATH, defaulting serialization to PROTOBUF.
Setting property: [incremental, false]
Setting property: [isolation, TRANSACTION_READ_COMMITTED]
issuing: !connect jdbc:phoenix:thin:url=http://localhost:8765;serialization=PROTOBUF none no
ne org.apache.phoenix.queryserver.client.Driver
Connecting to jdbc:phoenix:thin:url=http://localhost:8765;serialization=PROTOBUF
com.aliyun.phoenix.shaded.log4j:WARN No appenders could be found for logger (com.aliyun.
phoenix.shaded.org.apache.calcite.avatica.remote.HADriver).
com.aliyun.phoenix.shaded.log4j:WARN Please initialize the log4j system properly.
com.aliyun.phoenix.shaded.log4j:WARN See http://logging.apache.org/log4j/1.2/faq.html#noc
onfig for more info.
Connected to: Apache Phoenix (version unknown version)
Driver: Phoenix Remote JDBC Driver (version unknown version)
Autocommit status: true
Transaction isolation: TRANSACTION_READ_COMMITTED
Building list of tables and columns for tab-completion (set fastconnect to true to skip)...
133/133 (100%) Done
Done
sqlline version 1.2.0
0: jdbc:phoenix:thin:url=http://localhost:876> CREATE TABLE IF NOT EXISTS iteblog (
. . . . . .>   id BIGINT NOT NULL PRIMARY KEY,
. . . . . .>   blog VARCHAR(100)
. . . . . .> );
No rows affected (0.801 seconds)
0: jdbc:phoenix:thin:url=http://localhost:876> UPSERT INTO iteblog VALUES(1, 'https://www.ite
blog.com');
1 row affected (0.021 seconds)
0: jdbc:phoenix:thin:url=http://localhost:876> UPSERT INTO iteblog VALUES(2, 'http://books.ite
blog.com');
1 row affected (0.019 seconds)
0: jdbc:phoenix:thin:url=http://localhost:876> select * from iteblog;
+-----+--------------------------+
| ID  |          BLOG            |
+-----+--------------------------+
```

```
| 1   | https://www.iteblog.com   |
| 2   | http://books.iteblog.com  |
+-----+---------------------------+
2 rows selected (0.032 seconds)
0: jdbc:phoenix:thin:url=http://localhost:876>
```

可以看到， Phoenix 镜像已经运行成功。

注意：

- iteblog/hbase-phoenix-docker 镜像直接基于 iteblog/hbase-docker 镜像开发的，所以我们 pull iteblog/hbase-phoenix-docker 镜像到本地， iteblog/hbase-docker 镜像也会一起拉到本地，而且 iteblog/hbase-phoenix-docker 镜像启动的过程会启动 HBase 伪分布式模式和 QueryServer，所以我们不需要单独再启动 iteblog/hbase-docker 镜像。
- 因为我们在启动 Phoenix 镜像的时候使用了 -p 8765:8765 参数，所以在启动 bin/sqlline-thin.py 的时候就不用指定 QueryServer 的连接地址了，因为默认就是 http://localhost:8765。

Apache Phoenix 远程调试

在上一小节我们已经下载了 Phoenix 的镜像，为了让我们能够调试 Phoenix，需要加上远程调试相关参数。可以通过 PHOENIX_QUERYSERVER_OPTS 环境变量将远程调试相关参数加载到 Phoenix 的 QueryServer 启动参数中。docker 在启动镜像时，可以通过 --env 参数指定环境变量，具体如下：

```
[iteblog@www.iteblog.com ~]$  docker run -it -p 8765:8765 -p 8888:8888 --env PHOENIX_QUERYSERVER_OPTS="-Xdebug -Xrunjdwp:transport=dt_socket,server=y,suspend=y,address=8888" iteblog/hbase-phoenix-docker:1.0
```

上面命令中的 -p 8765:8765 是为了在本地终端直接连接 Phoenix QueryServer；而 -p 8888:8888 是为了将 Phoenix Docker 里面的远程调试端口和本地的 8888 端口映射上，这样我们就可以到 Idea 中直接使用这个端口就行。

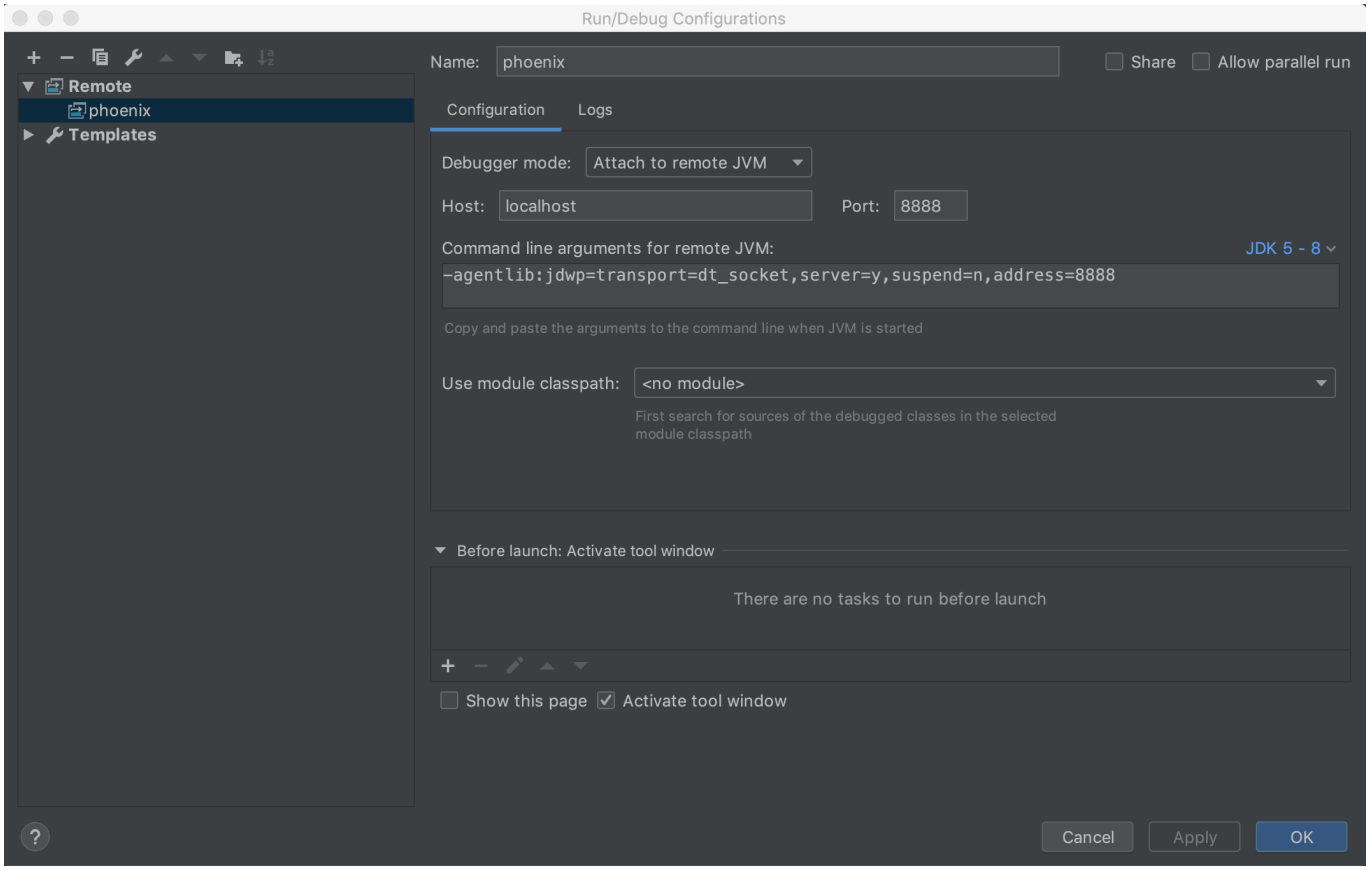
上面命令启动过程中可以看到输出的日志里面有下面一句日志

```
phoenix-queryserver stdout | Listening for transport dt_socket at address: 8888
```

这说明我们的远程调试参数配置成功。我们需要到 Idea 中连接这个端口。在 Idea 中依次进行下面操作

Run -> Edit Configurations... -> 弹出对话框 -> 点击左上角的 + 号 -> 弹出的菜单选择 Remote

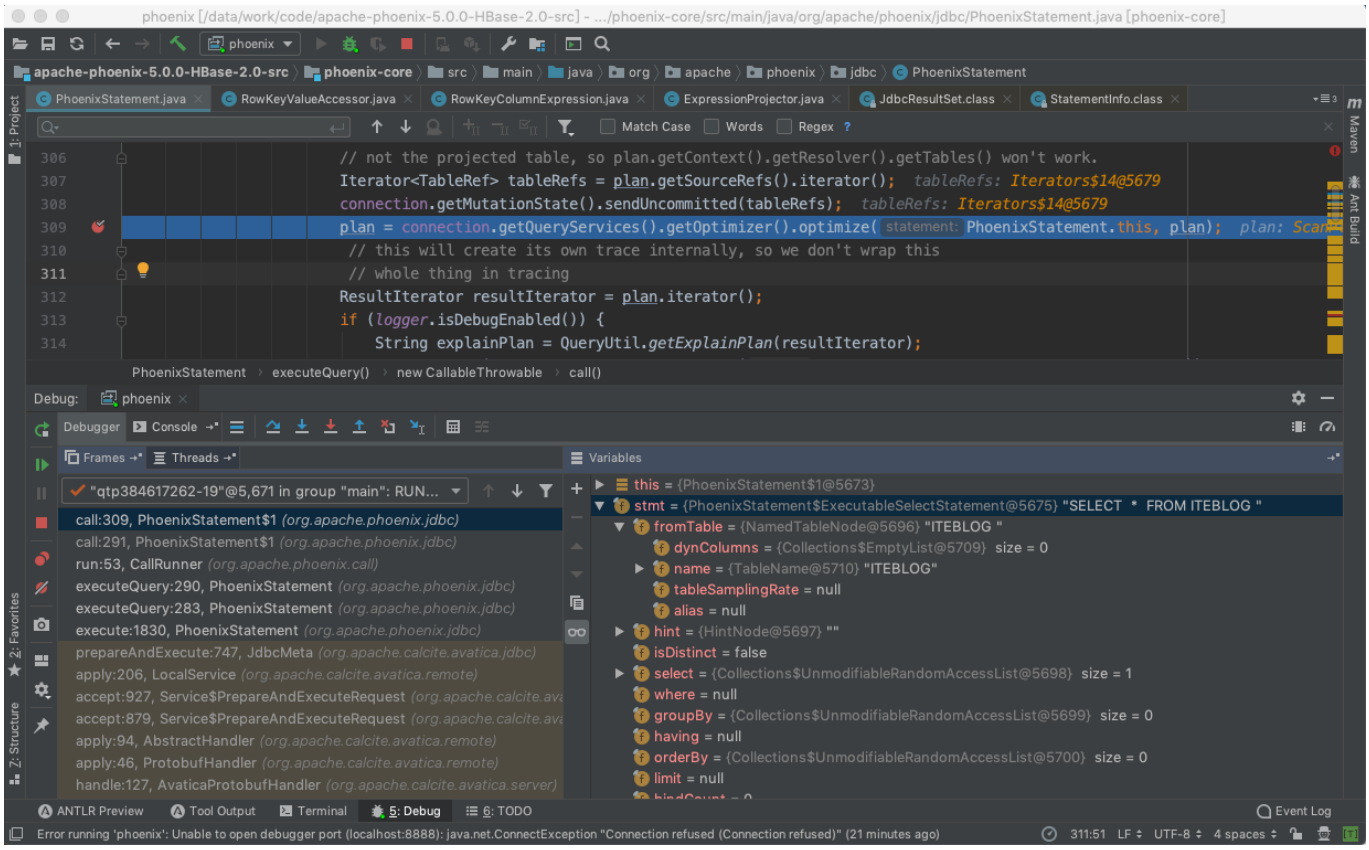
然后在新的界面填写如下的信息，点击 OK ：



最后我们到 Idea 编辑界面点击 Debug 按钮（也就是绿色小虫子的按钮）。这时候 Idea 就和 Docker 里面的测试环境连接上了。我们到终端启动 bin/sqlline-thin.py，并创建好相关表，然后在 PhoenixStatement.java 类中添加相关断点，最后执行下面的查询语句

```
[iteblog@www.iteblog.com ~]$  bin/sqlline-thin.py
0: jdbc:phoenix:thin:url=http://localhost:876> select * from iteblog;
```

这时候我们可以看到 Idea 出现了以下的界面了。



如果想及时了解Spark、Hadoop或者HBase相关的文章，欢迎关注微信公众号：iteblog_hadoop

到这里，我们就可以到任何我们感兴趣的地方打断点，然后进行相关调试了。