```
                                "AWS Key Id is required");

    awsSecretKey = context.getString("awsSecretKey");
    Preconditions.checkArgument(!Strings.isNullOrEmpty(awsSecretKey),
                                "AWS Secret Key must be specified");

    bucket = context.getString("bucket");
    Preconditions.checkArgument(!Strings.isNullOrEmpty(bucket),
                                "Bucket name must be specified");

    endPoint = context.getString("endPoint");
    Preconditions.checkArgument(!Strings.isNullOrEmpty(endPoint),
                                "Endpoint cannot be null");

    batchSize = context.getInteger("batchSize", DEFAULT_BATCH_SIZE);
    objPrefix = context.getString("objectPrefix", DEFAULT_OBJECT_PREFIX);
    bufferSize = context.getInteger("bufferSize", DEFAULT_BUFFER_SIZE);
  }
}
```


该例子描述了写数据到 Amazon S3 bucket 的 Sink[s3]。该 Sink 是终端 Sink 的一个例子，用于写数据到存储系统。它从 Channel 读取数据且批量写出到 Amazon S3 的文本文件。

S3 Sink 继承自 AbstractSink 类实现了 Configurable 接口。当 Agent 启动时，框架通过调用 configure 方法配置 Sink。该 Sink 创建需要登录到 S3 服务的认证。Sink 连接的 bucket 名字和端点也从配置文件中读取。因为需要这些参数，这个方法使这些通过配置文件传递的参数值有效且不为空。可以选择性地传递例如 batchSize 参数和对象名前缀。如果没有传递它们，那就使用默认值。

然后 Flume 框架通过调用 start 方法启动 Sink。在这个方法中，Sink 建立需要的连接和 bucket 信息。

一旦 Sink 启动，Sink 运行器就循环调用 process 方法。在 process 方法中，Sink 通过调用 Channel 的 getTransaction 方法创建事务。使用 Transaction.begin 方法启动事务，然后 Sink 通过调用 take 方法尽可能多地以批量的大小从 Channel 中读取事件（或者直到 Channel 中没有更多可用的事件——此时 take 方法返回 null），并将它们写出到 S3。Sink 也会给每个事件预先添加上每个事件的长度，以确保我们可以从相同文件中读取多个事件。一旦数据写入成功，事务就会提交；否则事务回滚。最终，必须关闭事务（这应该总是在最后一个块中完成的）。如果没有可读的事件，Sink 运行器被要求在 process 方法中通过返回 Status.BACKOFF 来减慢速度；否则，该方法返回 Status.READY。如果抛出任何异常，Sink 运行器就捕获异常且自动后退。