REVIEW II

**Search API vs Streaming API**

Search goes back in time (up to about a week) to find tweets that have already been sent.  
Stream goes forward in time (starting from when you initiate the call) to capture new tweets in (more or less) real time as they are sent.

Search is easier to build well, and can work even if you have a less than robust infrastructure.  
Stream is easier to scale to volumes, but requires a very solid infrastructure to work.

Search is rate limited in how many calls you can make. 1000 is a trivial number of tweets to retrieve with search, and you can be done in a minute or two.

Stream is rate limited in how many tweets it will deliver (up to 1% of the total Twitter volume unless you pay). 1000 tweets is a trivial number for this as well, but there's no telling how long it will take for all 1000 to get posted to Twitter. You may have to leave a connection open for a long time to collect them all.

If you're just starting out, start with Search is my personal advice.

**Streaming API vs Rest API**

Strategy 1: Streaming API

* Open a single stream (POST statuses/filter) for all walls
* Each hashtag is added to the track parameter
* When new tweets arrive, they will be processed and sent to the corresponding wall
* ("one account, one application, one open connection" cf. <https://dev.twitter.com/discussions/14935>)

Problems with the Streaming API

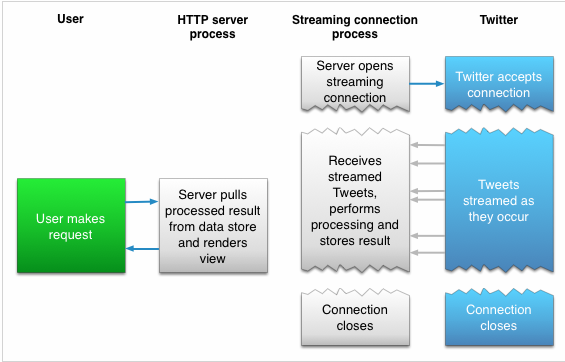
* Streaming api is limited to 400 keywords to track
* What to do if there are more than 400 keywords to track?
* Streaming api is limited to 1% of the tweets of the firehose
* It's very difficult to get above 1% of the firehose, but if you're tracking a term like "apple" it'd be pretty easy to exceed the 1%. (cf. [https://dev.twitter.com/discussions/63492](https://dev.twitter.com/discussions/6349))
* How can I handle such popular terms? Blacklist them?

Strategy 2: REST Search API\*\*

* Store user access tokens
* Poll the Search API (GET search/tweets) on behalf of the user, respecting the rate limits of 180 queries per 15 minute
* (cf. [https://dev.twitter.com/discussions/111411](https://dev.twitter.com/discussions/11141))

Problems with the REST Search API

* Polling
* Could get very expensive to poll the API for a lot of users.



There are three variations of the Twitter Streaming API:

1. [The Public Stream](https://dev.twitter.com/streaming/public). This allows your application to monitor public data on Twitter, such as public tweets, hashtag filters, et al.
2. [The User Stream](https://dev.twitter.com/streaming/userstreams). This allows you to track a user's tweet stream in real time. Part three of this series will focus on the user stream.
3. [Site Streams](https://dev.twitter.com/streaming/sitestreams). Site streams allow your application to monitor real-time Twitter feeds for a large number of users.

The job of your streaming implementation is to log the incoming events as quickly as possible and process them in the background using the REST API as necessary to harvest deeper data. Site streams require [prior approval from Twitter](https://dev.twitter.com/streaming/sitestreams#applyingforaccess), which are likely reserved for larger companies and developers.

Fortunately, there is a free, open-source library called [Phirehose](https://github.com/tutsplus/phirehose), which implements most of the streaming API requirements. This tutorial will describe how to integrate Phirehose into our open-source Birdcage application.

http://code.tutsplus.com/tutorials/building-with-the-twitter-api-using-real-time-streams--cms-22194