igvita.com

A goal is a dream with a deadline.

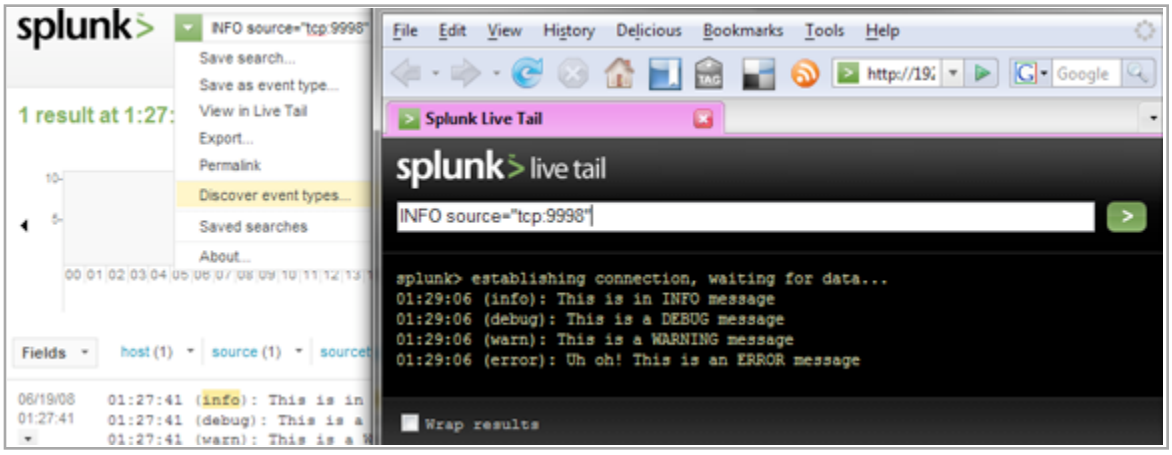
-
- Blog
- Projects
- About
- Archives
-

Splunk Your Distributed Logs in EC2

Managing log files is like herding cats, except its worse: a typical LAMP / Rails stack will easily generate a dozen logs in different locations. The fun part is, of course, the debugging and server administration tasks which involve the forensic task of tracking down each of these files and then cross-referencing timestamps to figure out what really happened. We've all dealt with this problem at one point or another.

Hence, when I recently stumbled on Splunk, I knew I had to try it - it, looked, awesome. The setup was a breeze and I was up and running in no time - no more fgrep / egrep for me! If you want to give it a try, take a look at Michael Wilde's video (Splunkin' with Amazon EC2), which he posted just a few days ago. Michael provides a great walkthrough, but I think he overlooked an awesome feature of Splunk, which is especially useful for the EC2 environment, and one we're planning to make heavy use of at AideRSS: distributed logging over TCP!

Managing Distributed Logs from the Cloud



Managing logs is hard as it is, and now imagine you have several (dozen) servers in EC2, the process becomes a chore, the debugging is hard, and frustration abounds. To address this problem, I've setup

Splunk to listen on a TCP port for any network traffic - if all others servers log to this host, then you will have a centralized, indexed log repository for all of your services. A simple Ruby-based *NetworkLogger* class later, and we're in business:

> networklogger.rb

```ruby
# igvita.com (Ilya Grigorik)
require 'rubygems'
require 'socket'

class NetworkLogger
  def initialize(host = 'localhost', port = '8888')
    @socket = TCPSocket.new(host, port)
  end

  def log(level, msg)
    @socket.write "#{Time.new.strftime("%H:%M:%S")} (#{level}): #{msg}\\n"
  end

  def method_missing(method, *args)
    if method.to_s =~ /(info|debug|warn|error)/i
      return self.log(method.to_s, args[0])
    end
  end
end

logger = NetworkLogger.new('localhost', 9998)

# Send log messages to remote Splunk server
logger.info "This is in INFO message"
logger.debug "This is a DEBUG message"
logger.warn "This is a WARNING message"
logger.error "Uh oh! This is an ERROR message"
```

networklogger.rb (NetworkLogger source)

Downloads: 686 File Size: 0.7 KB

It's a simple example, but it highlights the possibilities - you can stream any set of logs to a centralized server for easy debugging, live monitoring, and alerts and notifications. Best of all, it's a free solution as long as you stay below 500mb of indexed log data, per day.

Update: Michael Wilde posted posted a video reply with a demo of what he considers the coolest feature of Splunk to be: transactions. I didn't know about, and have to agree, thanks Michael!