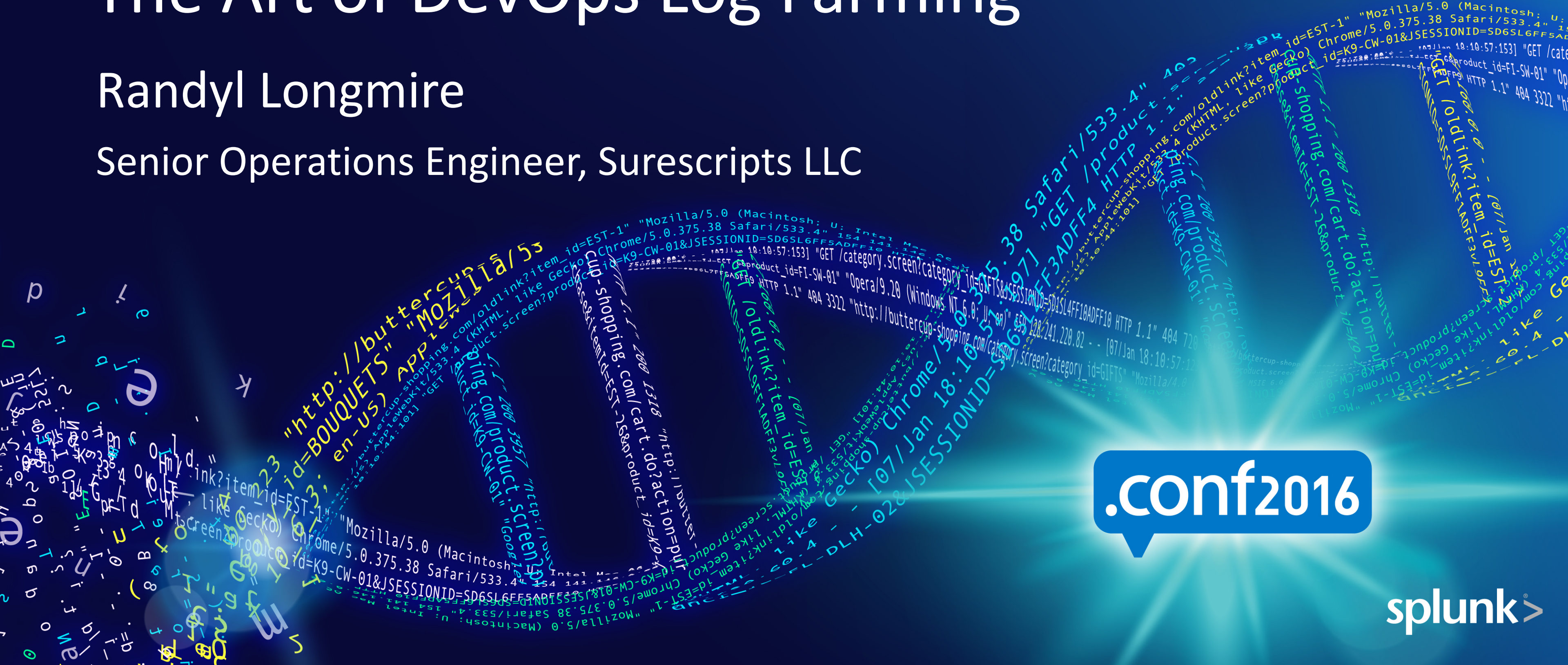
Finding Straw in a Hay Field

The Art of DevOps Log Farming

Randyl Longmire

Senior Operations Engineer, Surescripts LLC



.conf2016

splunk>

Disclaimer

Agenda

What are we doing here?

- Introductions
- Where in the DevOps cycle this session is focused
- Turning the ‘hay field’ of log entries into a valuable resource using Splunk software
- Queries, transactions, alerts, and automation
- Summary
- What’s next?

Who is Surescripts?

Surescripts is How Healthcare Gets Connected.

A nationwide health information network securely connecting doctors' offices, hospitals, pharmacists, and health plans through an integrated and technology neutral platform.

- We partner with more than 700 EHR applications used by over 900,000 healthcare professionals and more than 1,000 hospitals, impacting more than 270 million insured lives.

- We process more than 6 billion transactions each year, including nearly 700 million medication histories, more than 1 billion e-prescriptions and nearly 10 million clinical messages.

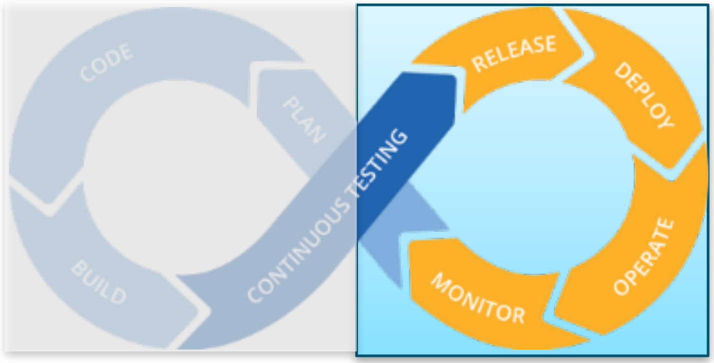
Who is Randyl Longmire?

The ‘Problem Resolver’

- Senior Operations Engineer (10 yrs.) at Surescripts
- Born and raised in the Northwest United States
- 14 years in Healthcare Technology
- 20+ years in Computer Support, Systems and Operations

The Scope of this Session

Which elements of the DevOps cycle are we focusing on?



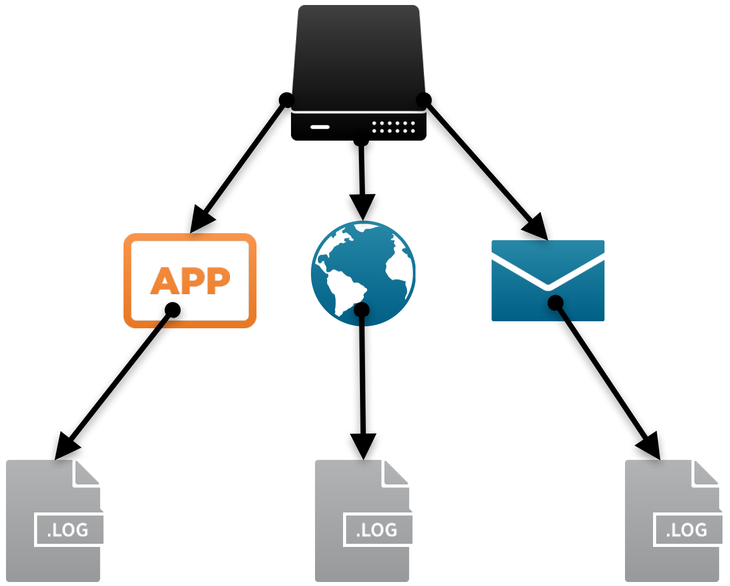
- Server and software deployments
- Monitoring of server and application health
- Troubleshooting and problem resolution

Scenario 1: Error Troubleshooting

Troubleshooting the old way

- We are alerted to an HTTP 500 error on a single site
- Using a text editor or log parser, we manually search for anything that looks like an error around the time that it was reported
- We then manually correlate this error with logs from related systems around the same timestamp

The anatomy of the haystack



Modern methods of finding a needle in the haystack

Typical method of finding a needle in the haystack

From a haystack to a hay field

Surescripts Hosted Web Apps

- 1800 servers (haystacks)
- 68 million log entries daily

Finding Straw in a Hay Field With Splunk Software

Query scope can range from very focused to very general

Query to find the string "TimeoutException" in a single log type on a single server:

```
Index=webApps host=webServer1 sourcetype=appLogs TimeoutException
```

Query to find the string "TimeoutException" in any log on any server within the 'webApps' index:

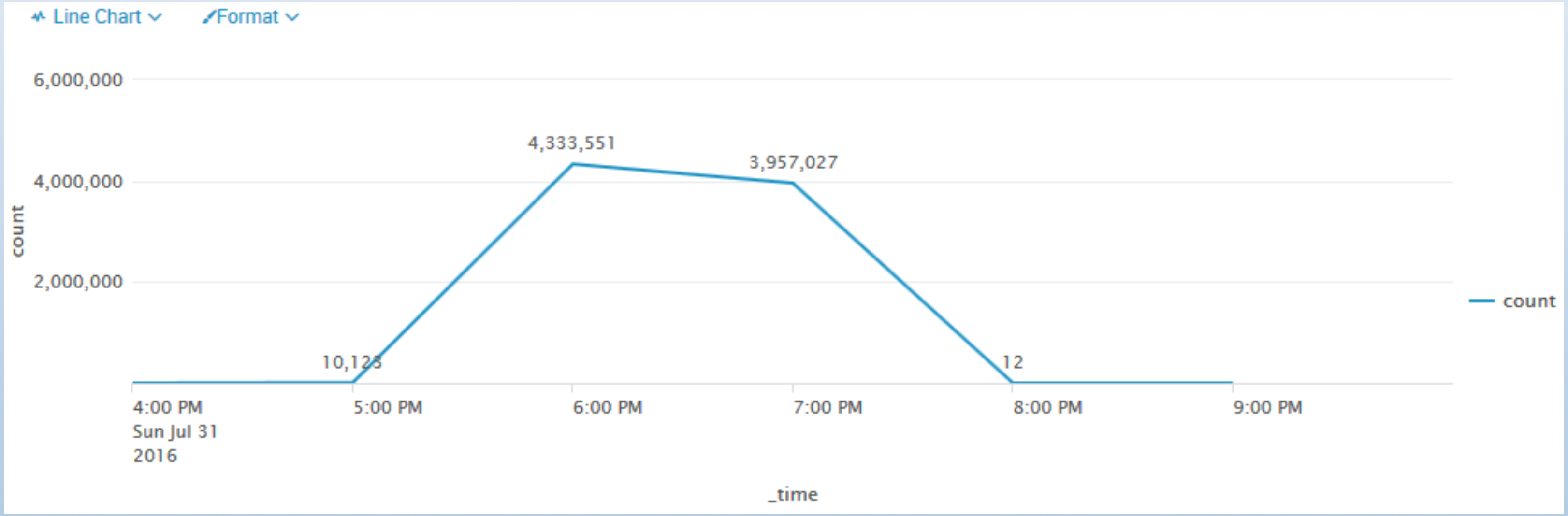
```
Index=webApps TimeoutException
```

Using Timechart to Graph the History

Using the same query, we can now look into the past

Query to find the string “TimeoutException” and visualize the frequency through time:

Index=webApps sourcetype=appLogs TimeoutException | timechart count span=1h



Scenario 2: Alerts

Using Alerts for proactive monitoring

- Now that we have the power of Splunk queries available, we can use them to create proactive alerts.
- When the same Timeout Exception error occurs, we can now be alerted to it immediately as well as trigger other actions.

Using Alerts for Error Detection

Step 1: Define the Query

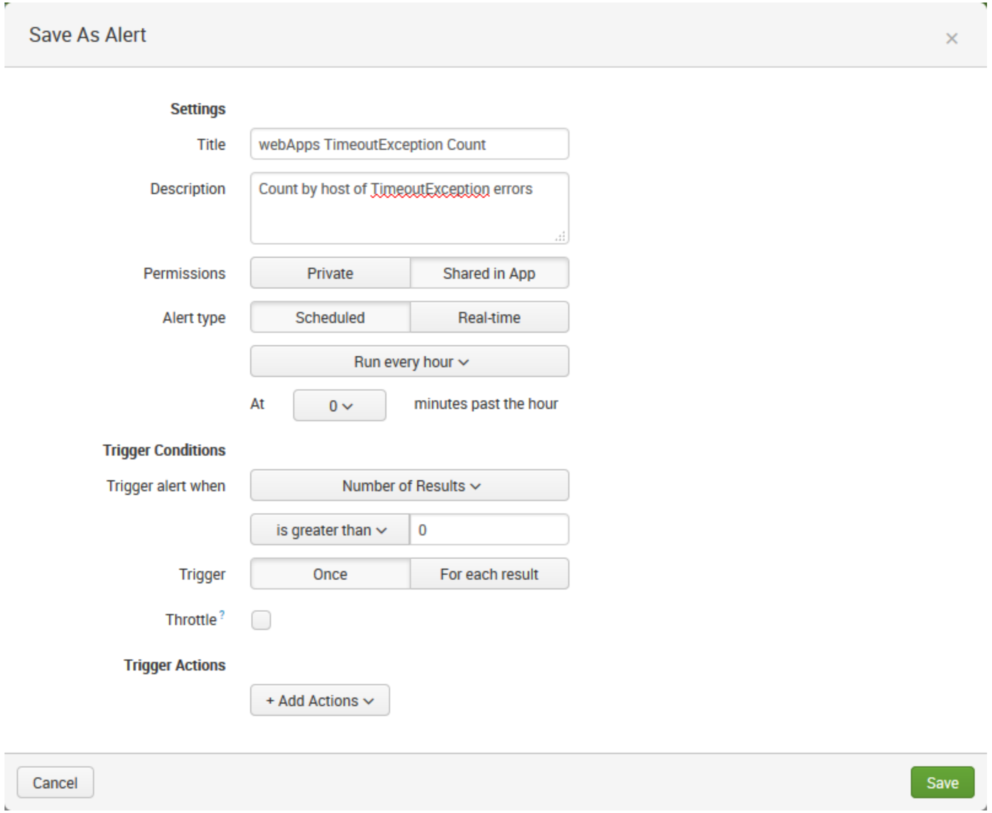
```
Index=webApps TimeoutException | stats count by host
```

Using Alerts for Error Detection

Step 2: Set the Alert Type and Trigger Condition

Scheduled or Real-Time

Trigger based on result counts



Save As Alert

Settings

Title	webApps TimeoutException Count
Description	Count by host of TimeoutException errors
Permissions	Private / Shared in App
Alert type	Scheduled / Real-time
	Run every hour
	At 0 minutes past the hour

Trigger Conditions

Trigger alert when	Number of Results
	is greater than 0
Trigger	Once / For each result
Throttle	☐

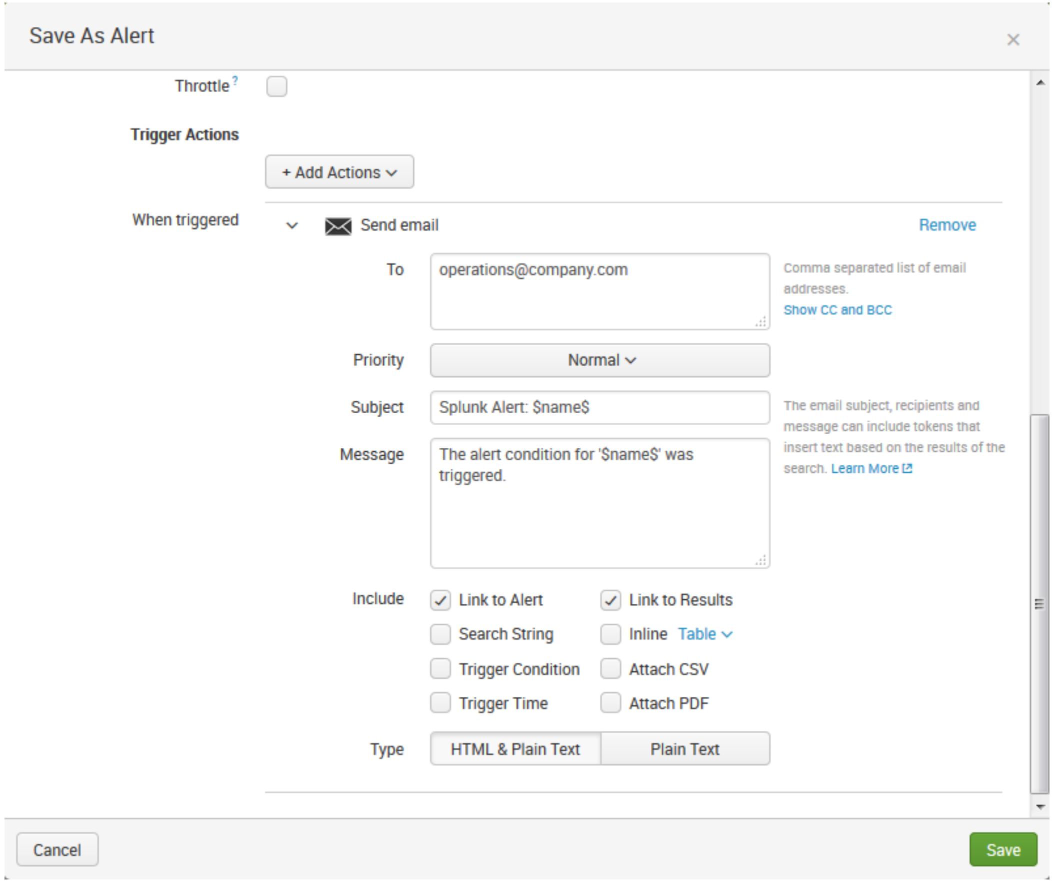
Trigger Actions

splunk> .conf2016

Using Alerts for Error Detection

Step 3: Specify Trigger Actions

- Log events
- Send an Email
- Run a script
- Open a ServiceNow incident
- POST to a webhook URL
- etc



Save As Alert

Throttle: ☐

Trigger Actions: + Add Actions

When triggered: Send email — Remove

To: operations@company.com

Comma separated list of email addresses. Show CC and BCC

Priority: Normal

Subject: Splunk Alert: $name$

Message: The alert condition for '$name$' was triggered.

The email subject, recipients and message can include tokens that insert text based on the results of the search. Learn More

Include: ✓ Link to Alert, ✓ Link to Results, ☐ Search String, ☐ Inline Table, ☐ Trigger Condition, ☐ Attach CSV, ☐ Trigger Time, ☐ Attach PDF

Type: HTML & Plain Text | Plain Text

Scenario 3 –Automation

Using PowerShell with the Splunk REST API

- The Problem:
  - Partner application has version dependencies with our application
  - When one side upgrades, the connectivity is broken until the other side upgrades
- Solution before Splunk
  - Support ticket is opened requesting an upgrade on or after a certain date
  - Connectivity would be broken for anywhere from hours to days
- Solution with Splunk
  - PowerShell script runs query against Splunk API every 30 minutes
  - When an unsupported version error is detected in the logs from any of the 1800 servers, the upgrade for that server is queued automatically

Using the REST API with PowerShell

Step 0: Declare the search query and parameters

```
#region Variables
     # Splunk Server Address
         [string]$SplunkServer = "https://splunk.example.com:8089"
     # Splunk API Username
         [string]$SplunkAPIUser = "splunkAPI"
     # Limit the number of results
         [int]$resultLimit = 1000
     # Splunk Search String
         [string]$SearchString = "search index=""webapps"" source=""*webApp.log""
     ""unsupported ver*"" | stats count as ErrorCount by host | head $resultLimit"
     # Value for time frame from now to search
         [string]$incrementValue = "-5m" # -5m,-5h,-5d, etc
     # Seconds to wait for the job to complete
         [int]$timeLimit = 60
     # Generate hashtable of body contents used to perform the search
         $RestBody = @{
                  search=$SearchString
                  output_mode="json"
                  earliest_time="$incrementValue"}
#endregion Variables
```

Using the REST API with PowerShell

Step 1: Get an API Session Key

```
# Get a session Key from Splunk API
#region GetSessionKey
     $object = @{
     "username" = $SplunkAPIUser
     "password" = $(GetSecret $SplunkAPIUser)
     }
     try {
         $token = Invoke-RestMethod -Uri "$SplunkServer/services/auth/login/" -Body
     $object -Method Post
     }
     catch {
         log "Error getting Splunk login token. $($_.Exception)"
         exit
     }
     $header = @{
     "Authorization" = "Splunk $($token.response.sessionKey)"
     }
#endregion GetSessionKey
```

Using the REST API with PowerShell

Step 2: Submit the search job

```
# Submit the search job
#region SubmitJob
    try {
        $JobID = (Invoke-RestMethod -Method Post -Uri
    "$SplunkServer/services/search/jobs/" -Headers $header -Body
    $restBody -ErrorAction Stop).sid
    }
    catch {
        log "Error submitting Query to Splunk API. Please check your
    search parameters and try again."
        log "Error Detail: $_"
        exit
    }
#endRegion SubmitJob
```

Using the REST API with PowerShell

Step 3: Wait for the job to complete

```
$JobStatus = Invoke-RestMethod -Method Post -Uri "$SplunkServer/services/search/jobs/$jobID"
-Headers $header -ErrorAction Stop
# Wait for job to complete
While (((($JobStatus.entry.content.dict.key | where {$_.name -eq "dispatchState"})."#text")
-ne "DONE") -and (!($timeOut))){
    If (((New-TimeSpan $startTime (Get-Date))).totalSeconds -gt $timeLimit) {
              log "Timeout exceeded!"
              # Delete the job and exit
              try {
                        $JobDelete = Invoke-RestMethod -Method DELETE -Uri
"$SplunkServer/services/search/jobs/$jobID" -Headers $header -ErrorAction Stop
                        exit
              }
              catch {
                        exit
              }
    }
    sleep -Seconds 1
    $JobStatus = Invoke-RestMethod -Method Post -Uri
"$SplunkServer/services/search/jobs/$jobID" -Headers $header -ErrorAction Stop
}
```

Using the REST API with PowerShell

Step 4: Get the results

```
# Get search results from the job
######################
    try{
                $JobResults = Invoke-RestMethod -Method Get -Uri
"$SplunkServer/services/search/jobs/$jobID/results?output_mode=json&count=0" -Headers $header -
ErrorAction Stop
                log "$($JobResults.results.Count) Search Results received"
    }
    catch {
                log "Could not obtain search results from Splunk API. $_"
    }

# Delete the job
    try {
                $JobDelete = Invoke-RestMethod -Method DELETE -Uri
"$SplunkServer/services/search/jobs/$jobID" -Headers $header -ErrorAction Stop
                log "Job Deleted successfully"
    }
    catch {
                log "Could not delete Splunk job ($JobID). $_"
    }
}
```

Using the REST API with PowerShell

Step 5: Process the results

```
# Process the search results
# We could also limit this list by only returning results with an
ErrorCount over a specified number
[array]$results = $JobResults.results.host

ForEach($hostname in $results) {
    # Do some automation based on the results, in our case queue the
server for an upgrade.
}
```

splunk> .conf2016

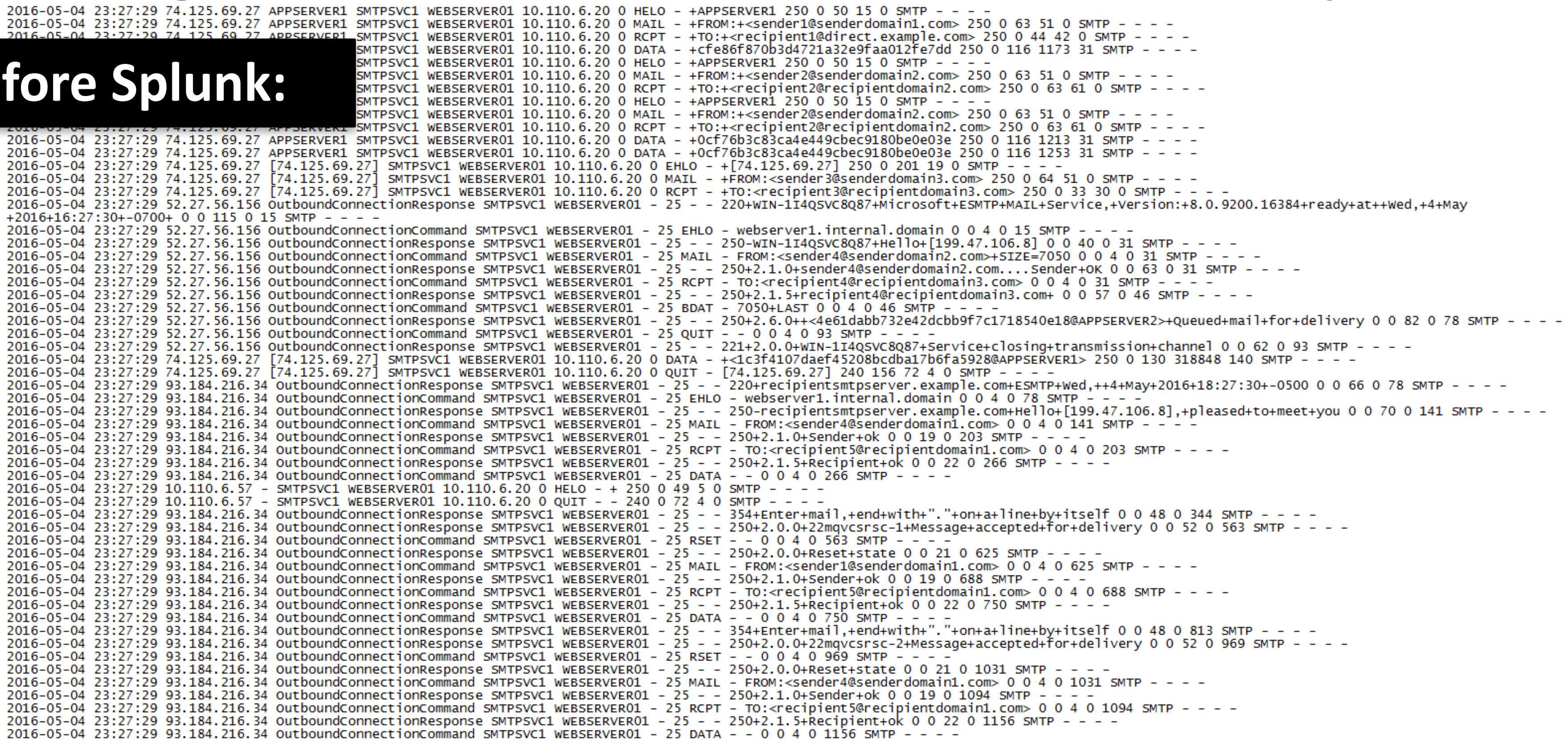
Bonus Scenario – Log Readability

Using transactions to create readable SMTP logs

- The Problem:
  - Customer reports an SMTP message was sent but never delivered
- Solution before Splunk
  - Support ticket is opened reporting SMTP details for missing message
  - Manually searching and parsing SMTP logs for hours, if they still exist
- Solution with Splunk
  - Use a ‘transaction’ query to display all communication threads from sender’s IP

Reading SMTP communication before Splunk

Before Splunk:

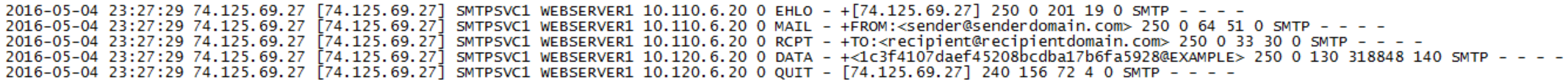


```
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 HELO - +APPSERVER1 250 0 50 15 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 MAIL - +FROM:+<sender1@senderdomain1.com> 250 0 63 51 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 RCPT - +TO:+<recipient1@direct.example.com> 250 0 44 42 0 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 DATA - +cfe86f870b3d4721a32e9faa012fe7dd 250 0 116 1173 31 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 HELO - +APPSERVER1 250 0 50 15 0 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 MAIL - +FROM:+<sender2@senderdomain2.com> 250 0 63 51 0 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 RCPT - +TO:+<recipient2@recipientdomain2.com> 250 0 63 61 0 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 HELO - +APPSERVER1 250 0 50 15 0 SMTP - - - -
                                            SMTPSVC1 WEBSERVER01 10.110.6.20 0 MAIL - +FROM:+<sender2@senderdomain2.com> 250 0 63 51 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 RCPT - +TO:+<recipient2@recipientdomain2.com> 250 0 63 61 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 DATA - +0cf76b3c83ca4e449cbec9180be0e03e 250 0 116 1213 31 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 APPSERVER1 SMTPSVC1 WEBSERVER01 10.110.6.20 0 DATA - +0cf76b3c83ca4e449cbec9180be0e03e 250 0 116 1253 31 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER01 10.110.6.20 0 EHLO - +[74.125.69.27] 250 0 201 19 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER01 10.110.6.20 0 MAIL - +FROM:<sender3@senderdomain3.com> 250 0 64 51 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER01 10.110.6.20 0 RCPT - +TO:<recipient3@recipientdomain3.com> 250 0 33 30 0 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 220+WIN-1I4QSVC8Q87+Microsoft+ESMTP+MAIL+Service,+Version:+8.0.9200.16384+ready+at++Wed,+4+May
+2016+16:27:30+-0700+ 0 0 115 0 15 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 EHLO - webserver1.internal.domain 0 0 4 0 15 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250-WIN-1I4QSVC8Q87+Hello+[199.47.106.8] 0 0 40 0 31 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 MAIL - FROM:<sender4@senderdomain2.com>+SIZE=7050 0 0 4 0 31 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.0+sender4@senderdomain2.com....Sender+OK 0 0 63 0 31 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RCPT - TO:<recipient4@recipientdomain3.com> 0 0 4 0 31 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.5+recipient4@recipientdomain3.com+ 0 0 57 0 46 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 BDAT - 7050+LAST 0 0 4 0 46 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.6.0++<4e61dabb732e42dcbb9f7c1718540e18@APPSERVER2>+Queued+mail+for+delivery 0 0 82 0 78 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 QUIT - - 0 0 4 0 93 SMTP - - - -
2016-05-04 23:27:29 52.27.56.156 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 221+2.0.0+WIN-1I4QSVC8Q87+Service+closing+transmission+channel 0 0 62 0 93 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER01 10.110.6.20 0 DATA - +<1c3f4107daef45208bcdba17b6fa5928@APPSERVER1> 250 0 130 318848 140 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER01 10.110.6.20 0 QUIT - [74.125.69.27] 240 156 72 4 0 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 220+recipientsmtpserver.example.com+ESMTP+Wed,++4+May+2016+18:27:30+-0500 0 0 66 0 78 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 EHLO - webserver1.internal.domain 0 0 4 0 78 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250-recipientsmtpserver.example.com+Hello+[199.47.106.8],+pleased+to+meet+you 0 0 70 0 141 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 MAIL - FROM:<sender4@senderdomain1.com> 0 0 4 0 141 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.0+Sender+ok 0 0 19 0 203 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RCPT - TO:<recipient5@recipientdomain1.com> 0 0 4 0 203 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.5+Recipient+ok 0 0 22 0 266 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 DATA - - 0 0 4 0 266 SMTP - - - -
2016-05-04 23:27:29 10.110.6.57 - SMTPSVC1 WEBSERVER01 10.110.6.20 0 HELO - + 250 0 49 5 0 SMTP - - - -
2016-05-04 23:27:29 10.110.6.57 - SMTPSVC1 WEBSERVER01 10.110.6.20 0 QUIT - - 240 0 72 4 0 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 354+Enter+mail,+end+with+"."+on+a+line+by+itself 0 0 48 0 344 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.0.0+22mqvcsrsc-1+Message+accepted+for+delivery 0 0 52 0 563 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RSET - - 0 0 4 0 563 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.0.0+Reset+state 0 0 21 0 625 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 MAIL - FROM:<sender1@senderdomain1.com> 0 0 4 0 625 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.0+Sender+ok 0 0 19 0 688 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RCPT - TO:<recipient5@recipientdomain1.com> 0 0 4 0 688 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.5+Recipient+ok 0 0 22 0 750 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 DATA - - 0 0 4 0 750 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 354+Enter+mail,+end+with+"."+on+a+line+by+itself 0 0 48 0 813 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.0.0+22mqvcsrsc-2+Message+accepted+for+delivery 0 0 52 0 969 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RSET - - 0 0 4 0 969 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.0.0+Reset+state 0 0 21 0 1031 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 MAIL - FROM:<sender4@senderdomain1.com> 0 0 4 0 1031 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.0+Sender+ok 0 0 19 0 1094 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 RCPT - TO:<recipient5@recipientdomain1.com> 0 0 4 0 1094 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionResponse SMTPSVC1 WEBSERVER01 - 25 - - 250+2.1.5+Recipient+ok 0 0 22 0 1156 SMTP - - - -
2016-05-04 23:27:29 93.184.216.34 OutboundConnectionCommand SMTPSVC1 WEBSERVER01 - 25 DATA - - 0 0 4 0 1156 SMTP - - - -
```

Converting SMTP logs into readable transactions

After Splunk:

index=webApps sourcetype=iis c_ip="74.125.69.27" | transaction c_ip startswith=EHLO endswith=QUIT maxspan=4s



```
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER1 10.110.6.20 0 EHLO - +[74.125.69.27] 250 0 201 19 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER1 10.110.6.20 0 MAIL - +FROM:<sender@senderdomain.com> 250 0 64 51 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER1 10.110.6.20 0 RCPT - +TO:<recipient@recipientdomain.com> 250 0 33 30 0 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER1 10.120.6.20 0 DATA - +<1c3f4107daef45208bcdba17b6fa5928@EXAMPLE> 250 0 130 318848 140 SMTP - - - -
2016-05-04 23:27:29 74.125.69.27 [74.125.69.27] SMTPSVC1 WEBSERVER1 10.120.6.20 0 QUIT - [74.125.69.27] 240 156 72 4 0 SMTP - - - -
```

Summary

- Before Splunk
  - Log files were more of a management task than a useful tool
  - The manual process of log parsing was tedious and time-consuming

- With Splunk
  - Log files are an empowering resource across all aspects of DevOps
  - Queries can target a broad scope or laser focus for error identification and troubleshooting
  - Alerts provide pro-active monitoring and automation
  - Timecharts enable graphing for dashboards and historical data
  - The API opens the power of Splunk to countless other applications

What's Next?

- Be Creative
  - Splunk's applications expand with your imagination
- Be Collaborative
  - Use the Splunk community tools
- Be Adventurous
  - Discover new commands and methods and ways they can be applied
- Be Inspired
  - Adapt and transform existing solutions into new and exciting tools

THANK YOU