Blackboard Open Source Monitoring

By

Greg Lloyd

Submitted to
the Faculty of the School of Information Technology
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Technology

Greg Lloyd — Date: 4/20/15

Mark Stockman — Date: 4/20/15

University of Cincinnati
School of Information Technology
College of Education, Criminal Justice, and Human Services

May 2015

TABLE OF CONTENTS

ABSRACT

The University of Cincinnati (UC) provides online learning via the Blackboard Learn, learning management system (LMS). As time passed the product changed and so did the technologies that support it. The monitoring solution that monitors Blackboard had not grown with the LMS. This project will continue to use an open source system monitoring solution, Zabbix, while researching and adding new monitoring metrics. These metrics will give system administrators a better look into the Blackboard environment. To help tune and load test the LMS, JMeter was deployed. The ELK (Elasticsearch, Logstash, Kibana) stack was also implemented to help perform log analysis on the LMS. By combining all of these tools, Blackboard system administrators will be able to better diagnose and resolve performance-impacting problems.

STATEMENT OF NEED

Currently the Blackboard monitoring system is doing high level monitoring that would be completed on any monitored environment. With such high level monitoring the current solution is missing a lot of alerts that are crucial to the environment and stability of Blackboard. By extending the current monitoring solution, Zabbix, we can monitor Blackboard at a low level and alert system administrators of potential issues before the end user is impacted. For example, ActiveMQ, the Blackboard broker service, controls what app server handles certain backend processing tasks. When ActiveMQ fails over incorrectly those tasks do not take place. This can cause Blackboard to become extremely unstable and negatively impact the experience of the end-user.

To ensure that the monitoring solution is working properly a load testing solution will be needed. This will put load on specific components of Blackboard, which will then be reported in Zabbix. This need also has a secondary benefit; it can be used to help tune application servers based on specific load test.

To aid in troubleshooting performance issues that have been identified in Zabbix a log analysis tool will need to be deployed. This will allow system administrators to identify common problems across all nodes. Deploying a log analysis tool will also provide other metrics such as response time and bandwidth.

A final need of the project is for help desk technicians to have a peak into the environment to see if all application servers are reporting healthy. This will allow them to be able to verify claims of end user slowness or other system issues. This project has a primary emphasis on Networking and system administration.

SUPPORTING RESOURCES

To many system administrators monitoring an application is one of the first things done when an application is about to be deployed to production. If the system administrator doesn't have a clear view of what is going on in the environment it is like working with a black box. Nick Hardiman wrote an article on TechRepublic called *Bullet-proof your web service with the right monitoring setup*. In this article he states that all layers of hardware, networking, OS and applications must be checked. Later in his article he also states that "the performance of an application needs to be measured, and so does everything that can affect its performance" (Hardiman). This shows that a centralized monitoring solution is very important. It also shows that not only the application should be monitored but anything that could impact the performance of the application.

PROPOSED COURSE OF ACTION/SOLUTION

To achieve the goal of this project I utilized features in the current monitoring solution, Zabbix, that are currently not being used. I utilized the built-in SNMP monitoring tools to monitor the F5 BigIP load balancer that sits in front of the

Blackboard environment. I also used the JMX monitoring agent that will help monitor built-in functions within the Blackboard application. By adding those two technologies to the monitoring system I am now able to monitor parts of the Blackboard infrastructure that have never been monitored before.

To put load on the system I am using Apache's JMeter software. This tool allows you to create dynamic and custom use cases for load testing purposes. It also allows you to specify the amount of connections over time and how many iterations will be completed per test. This tool also graphs response times over time.

To analyze and gather system logs, the ELK stack has been deployed. The ELK stack consist of Elasticsearch, a JSON document store, Logstash, a log sending agent, and Kibana, a web interface that sits on top of Elasticsearch. This allows for custom dashboards and log analysis to be done in real-time.

To aid the help desk in troubleshooting a map was created in Zabbix. This will give them access to see when systems are reporting problems. They can then alert the appropriate technicians if the problem has not already been acknowledged.

USER PROFILES AND DESIGN PROTOCOLS

There are currently three user profiles that are initially impacted by this project. In the first use case, a help desk technician in the past had to email or IM the Blackboard System Administrators to get a status update on the Blackboard

system. With the implementation of this senior design project they will be able to login and view a monitoring map that shows the current status of Blackboard and its supporting infrastructure. Below is the map that was created to fulfill this.

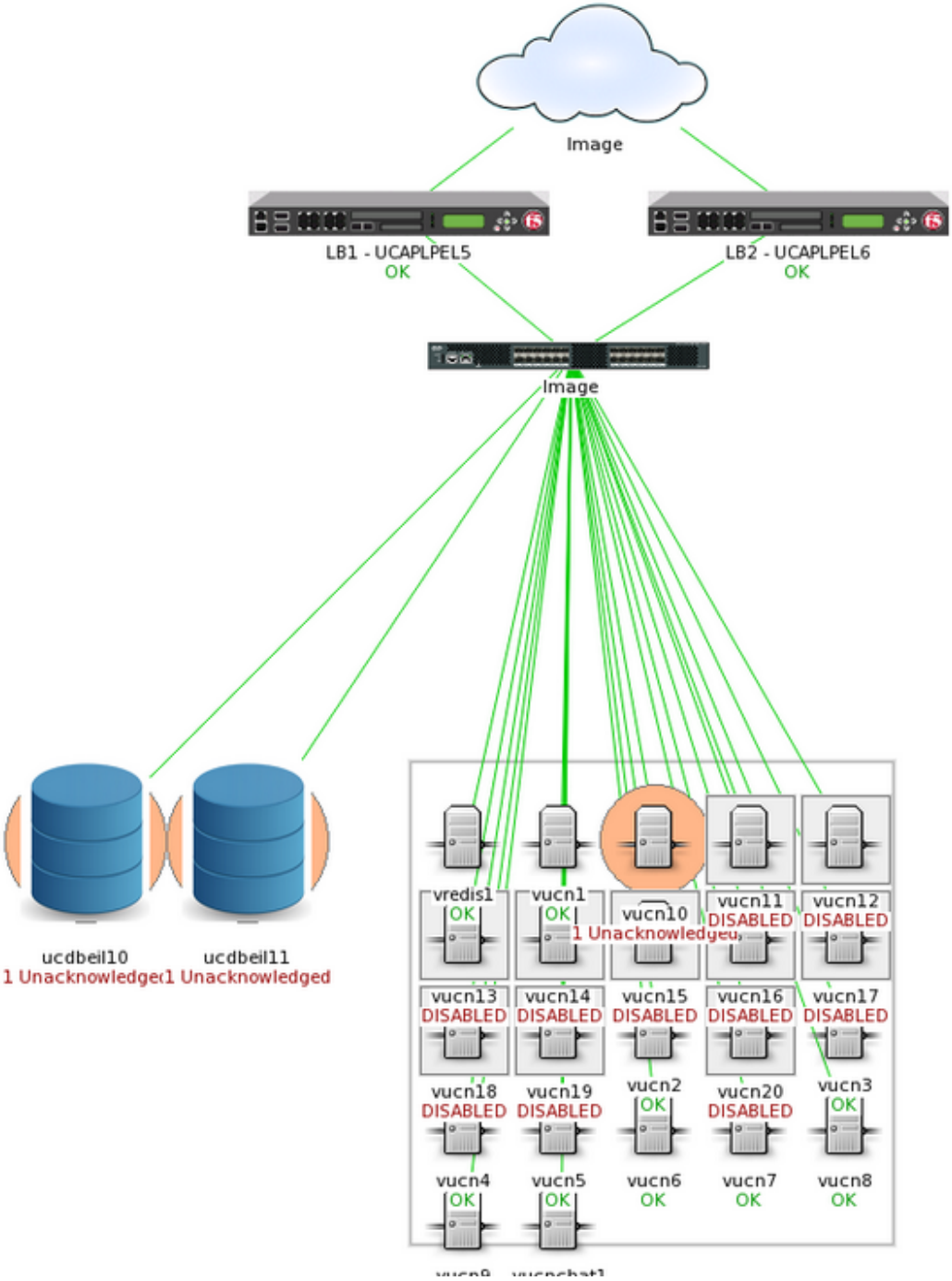

Figure 1. Zabbix Map

The second use case is for Junior Blackboard System Administrators. They will be able to create items in Zabbix and view dashboards, screens, maps, and edit existing settings. The right to create a trigger is reserved for a senior system administrator. This is due to the alerting mechanism that will page the on-call system administrator. If it is setup incorrectly it will alert a system administrator at the wrong time.

The final use case is for the Senior System Administrator. They have full access to Zabbix and all of its supporting technologies. They are able to make changes on the fly and can implement a monitoring technique from start to finish.

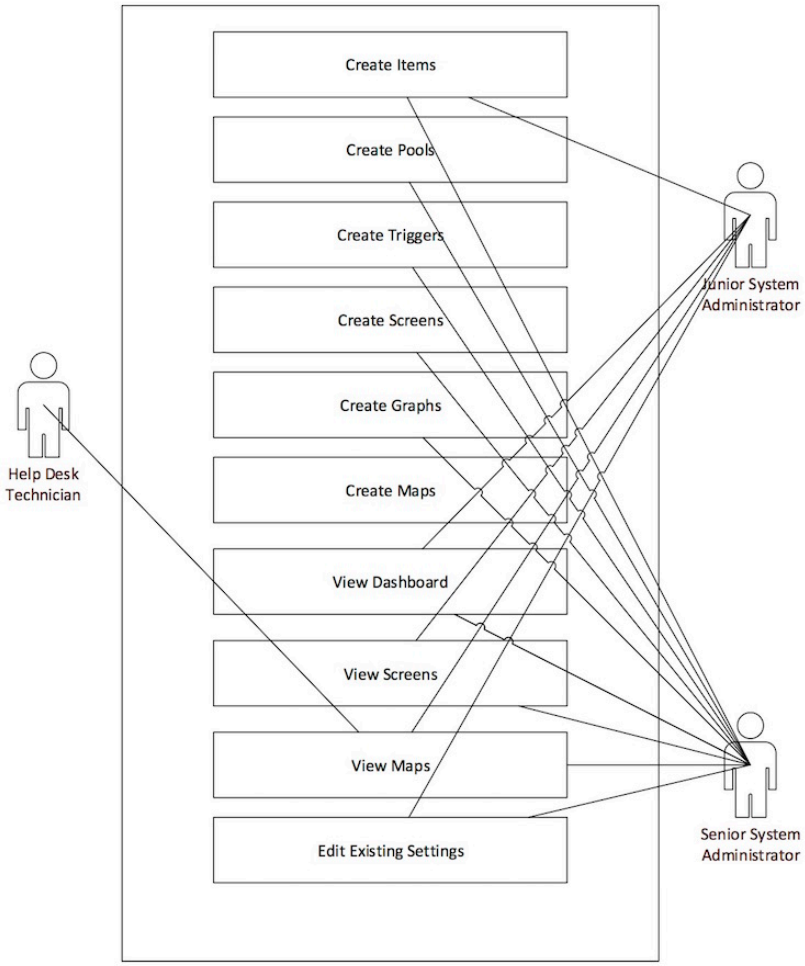

Figure 2. Use Case

TECHINCAL ELEMENTS

Zabbix 2.2 monitoring software is being used to monitor the environment. Zabbix requires a minimum of 128 MB of physical memory and 256MB of free disk space. The reason I am using Zabbix instead of other monitoring solutions is because the University of Cincinnati already has the infrastructure around Zabbix

implemented. Another reason to continue the use of Zabbix is its great UI and built-in functionality with SNMP and JMX. It also comes with a monitoring agent that can run on Linux or Windows based operating systems. Other tools were also investigated, such as Riverbed and Dynatrace. Both options were not selected, mainly due to cost. To deploy Riverbed it would have cost approximately $150,000 in licensing on top of hardware purchases and employee salaries. Other log analysis tools were also looked at such as Graylog. Graylog was not implemented due to its instability at the time of research.

TECHNICAL ELEMENTS IMPLEMENTATION

To be able to pull the SNMP OID's you have to know what you are looking for. To do this you need an SNMP browser. I decided to implement a Zabbix SNMP MIB browser that was written by the open source community (https://www.zabbix.com/wiki/howto/monitor/snmp/snmp_builder). This allowed me to place the load balancer MIB files on the Zabbix server and then search them for the correct OID. From there I was able to run snmpget on the public snmp interface on the F5 to pull the numeric OID and enter it as an item into Zabbix. After deploying SNMP monitoring I was able to successfully monitor Blackboard connections on both the public and private side of the load balancer. I was also able to find node availability, individual host connections, and pool connections.

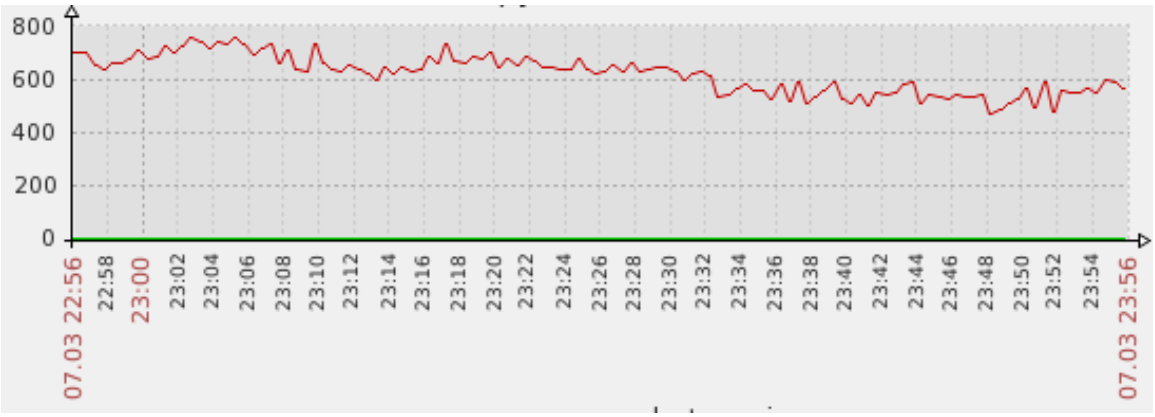

Figure 3. Zabbix Graph via SNMP values from F5 load balancer

In order to pull values from the Blackboard JVM (Java Virtual Machine) the Zabbix Java Gateway had to be installed on the Zabbix server. In order to find what JMX (Java Management Extensions) are available, I used JMXterm. This allowed me to browse the available domains and find which beans I would like to pull into Zabbix. After gaining an understanding of how to use JMX monitoring I was successfully able to pull which host is running as the ActiveMQ master. I was also able to use some openly developed Blackboard Templates that use some JMX functions. This allowed me to start implementing overall JVM statistics. Once this was completed I was able to create custom graphs, screens, and triggers to make looking at JMX data easier to understand.

In order to be able to pull the number of blocked threads within the jvm I had to create a custom BASH script. This script is located on a NFS mount that all servers in the environment connect to. From there I added an entry into cron on each app server to run the script. This script calls JSTACK a built in tool within the JDK that allows users to request thread dumps. The thread dumps then contain identifiers for what is blocked, waiting, or runnable. The script then counts the amount of blocked threads and puts it

into a text file. At that point Zabbix checks that text file and pulls the integer into Zabbix for evaluation. Below is graph that shows the recent blocking lock data.

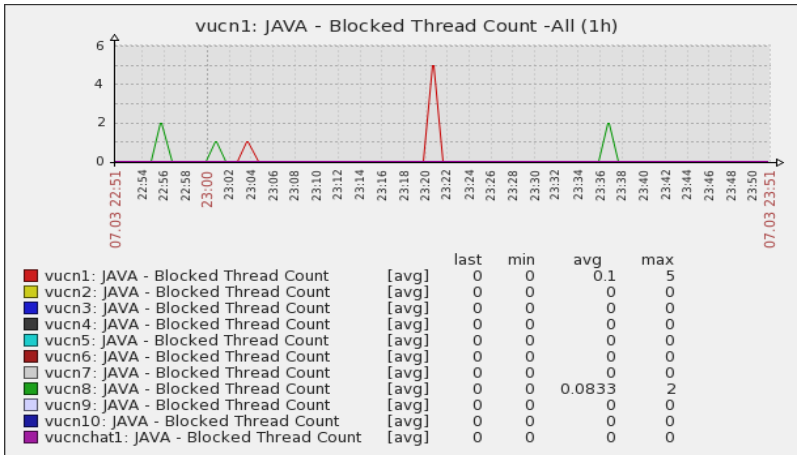

Figure 4. Blocked Thread Graph

To be able to send logs from the application servers to Elasticsearch I had to create grok patterns. These grok patterns allow for logstash to ship the logs in real time in a JSON format that is acceptable for Elasticsearch. To ensure that my grok patterns were correct I was able to use a real-time evaluator at https://grokdebug.herokuapp.com/. Grok patterns are located in Appendix A.

SECURITY

The Zabbix web frontend is secured over SSL. It also is connected to the UC LDAP server. In order to be able to login to the web frontend an account has to be created within the web UI and selected permissions must be applied. Once the account is created users can authenticate via LDAP. If the account does not exist it will not authenticate the user. As far as server side access to the zabbix system a sys admin must be logged in behind the F5 BigIP load balancer. To do that users mush ssh to an ssh server that is connected to LDAP with RSA two step authentication. This restricts system access to RSA users

that are in the allow policy. From there you can ssh to the ZAbbix server and authenticate again via LDAP to the Zabbix server. Since the data collected in the system is not PI or passwords the data does not need to be encrypted when being transferred from the app server/database/Bigip to the Zabbix server. Another security measure is the BigIP load balancer itself. This is because it acts as a firewall since it only has specific ports open to specific servers. The Kibana web interface is currently being deployed to shibboleth so that it will be protected via SAML authentication.

KNOWN ISSUES

Currently there is a known issue with Elasticsearch when using logstash that prohibits users from using a histogram with a type other than count (Saving configuration causes Facet). This is due to an index that was created via logstash that has 0 documents in it. This issue is still actively being investigated.

CONCLUSION

With the completion of this project, Blackboard administrators at UC can now monitor the Blackboard Learn environment at a much lower level than ever before. They can now view a Zabbix dashboard that shows them which node is processing a specific feed file. They can also look at a single screen to look over graphs of multiple system checks that are constantly monitoring the system. This will allow system administrators to catch problems early on and correct them much sooner than in the past. This will allow for a better end user experience and higher system uptime.

This project also has the potential to save UC approximately $55,000 if they decide to do another Blackboard performance audit. This is because I have been able to replicate what UC in the past has paid Blackboard Inc to load test and tune our Blackboard servers. Granted this project does not load test at the same level that Blackboard Inc does, it still load tests major system features with JMeter and then calculates load time via the Apache logs. This project also uses some of the same Zabbix templates that Blackboard Inc has developed (Tatsumi, N). In this project I actively used the JMX template and added to it support for ActiveMQ checks in our environment.

NETWORK DIAGRAM

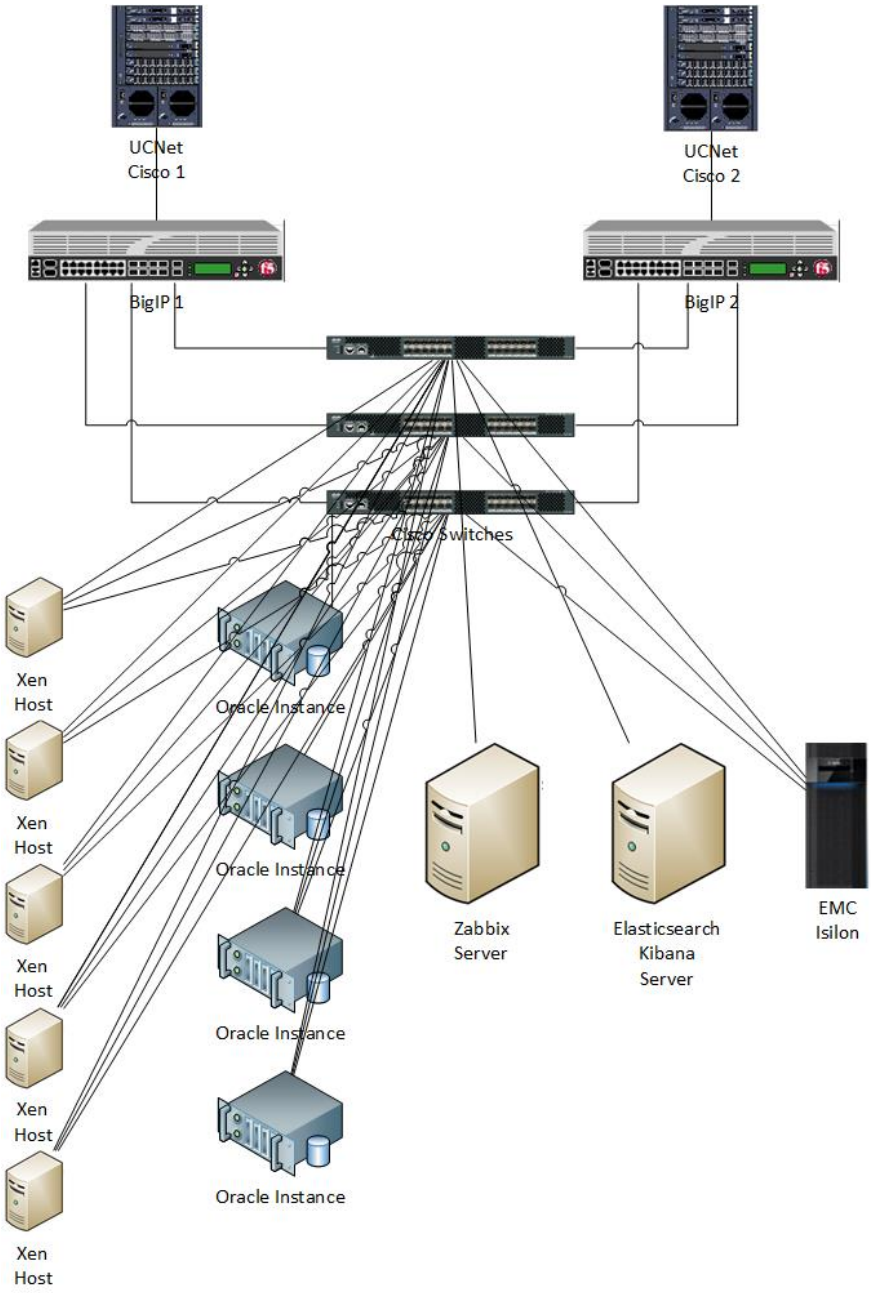

Figure 5. Physical Network Diagram

BUDGET

A Dell R710 cost approximately $1,500 but has already been purchased. Zabbix software is free open source software that has no cost. 250 man-hours are needed at $80 an hour, which comes out to $20,000. Since I am already a salaried employee and the R710 has been purchased in the past the net cost of the project is $0.

	Zabbix	AppInternals	DynaTrace
Software	$0	$150,000	$52,800
Server	$1,500	$3,000	$0
Implementation $80/Hr	$20,000	$5,000	$5,000
Total	$21,500	$158,000	$57,800

Table 1. Budget

TIMELINE

Task Name	Duration	Start	Finish	Predecessors
Initiate	**16 days**	**Mon 8/25/14**	**Mon 9/15/14**	
Find Project	10 days	Mon 8/25/14	Fri 9/5/14	
Authorize Project	6 days	Mon 9/8/14	Mon 9/15/14	
Planning	**20 days**	**Mon 9/15/14**	**Fri 10/10/14**	
Research	10 days	Mon 9/15/14	Fri 9/26/14	
Determine Architecture	3 days	Mon 9/22/14	Wed 9/24/14	
Determine Zabbix Items	7 days	Thu 9/18/14	Fri 9/26/14	
Determine Zabbix Triggers	2 days	Sat 9/27/14	Mon 9/29/14	
Determine Zabbix Graphs	2 days	Tue 9/30/14	Wed 10/1/14	
Determine Zabbix Maps	1 day	Tue 9/30/14	Tue 9/30/14	
Determine Zabbix Web	3 days	Wed 10/1/14	Fri 10/3/14	
Determine Zabbix Actions	1 day	Wed 10/1/14	Wed 10/1/14	
Determine Zabbix Screens	1 day	Wed 10/1/14	Wed 10/1/14	
Determine F5 SNMP Traps	4 days	Thu 10/2/14	Tue 10/7/14	
Determine if Piwik or Kibana are needed	3 days	Wed 10/8/14	Fri 10/10/14	
Executing	**105 days**	**Sat 10/11/14**	**Thu 3/5/15**	
Deploy Zabbix Items	15 days	Mon 11/3/14	Fri 11/21/14	24

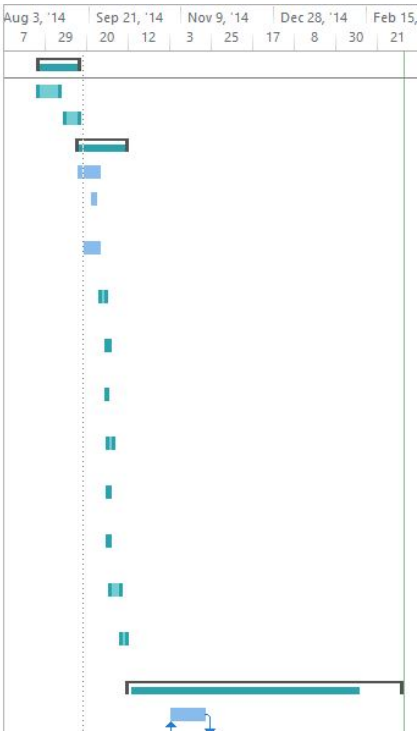

Figure 6. Gantt Chart 1

Deploy Zabbix Triggers	10 days	Mon 11/24/14	Fri 12/5/14	17
M1 - Upgraded Monitoring	0 days	Fri 12/5/14	Fri 12/5/14	
Deploy Zabbix Maps	10 days	Mon 12/8/14	Fri 12/19/14	18
Deploy Zabbix Web	5 days	Fri 12/19/14	Thu 12/25/14	
Deploy Zabbix Screens	3 days	Fri 12/26/14	Tue 12/30/14	21
Deploy Zabbix Graphs	5 days	Wed 12/31/1	Tue 1/6/15	22
F5 SNMP Traps	15 days	Mon 10/13/1	Fri 10/31/14	
Deploy Zabbix Actions	5 days	Wed 1/7/15	Tue 1/13/15	23
M2 - Zabbix Improvements done	0 days	Tue 1/13/15	Tue 1/13/15	
Deploy Kibana or Piwik If necessary	5 days	Wed 1/14/15	Tue 1/20/15	25
M3 - Log analysis tool implemented	0 days	Tue 1/20/15	Tue 1/20/15	
Load Generator	15 days	Wed 1/21/15	Tue 2/10/15	27
M4 - Jmeter implemented	0 days	Wed 2/11/15	Wed 2/11/15	
Monitoring & Controlling	**15 days**	**Mon 2/9/15**	**Fri 2/27/15**	
Determine environment impact	5 days	Mon 2/9/15	Fri 2/13/15	
Verifying Restart Script	3 days	Mon 2/16/15	Wed 2/18/15	32

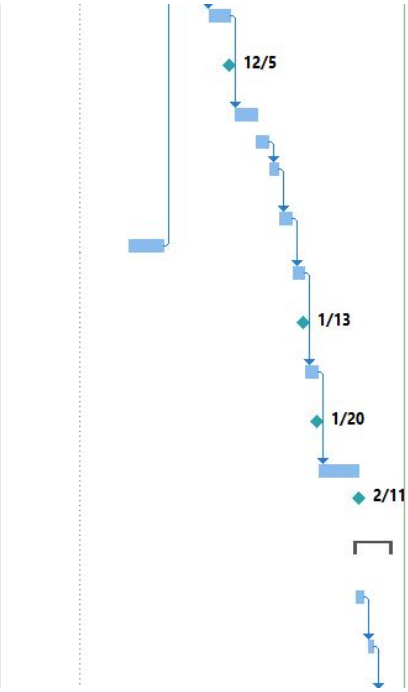

Figure 7. Gantt Chart 2

Resolving false positives	7 days	Thu 2/19/15	Fri 2/27/15	33
Closing	**20 days**	**Fri 2/27/15**	**Thu 3/26/15**	
implement any items missed or extra items	15 days	Fri 2/27/15	Thu 3/19/15	
Verify all systems functioning	5 days	Thu 3/19/15	Wed 3/25/15	
M5 - Project Finished	0 days	Fri 3/27/15	Fri 3/27/15	

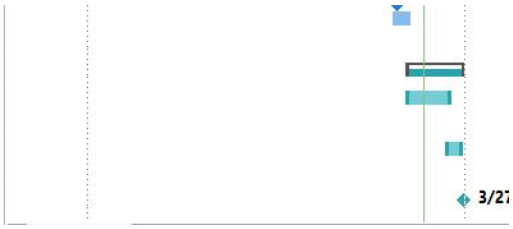

Figure 8. Gantt Chart 3

References

Hardiman, Nick. "Bullet-proof Your Web Service with the Right Monitoring Setup." TechRepublic. N.p., 28 Aug. 2012. Web. 1 Nov. 2014.

Rounin, D. (2013, June 23). Parse Apache2 Error logs with Grok for Logstash. Retrieved February 13, 2015, from http://stackoverflow.com/questions/17331593/parse-apache2-error-logs-with-grok-for-logstash

Saving configuration causes Facet: [0]: (key) field [@timestamp] not found in histogram panel · Issue #834 · elasticsearch/kibana. (2014, January 15). Retrieved March 8, 2015, from https://github.com/elasticsearch/kibana/issues/834

Tatsumi, N. (2014, July 25). Blackboard/zabbix-ext. Retrieved March 8, 2015, from https://github.com/blackboard/zabbix-ext

Appendix A – GROK Patterns

```
BBAPACHELOG %{IPORHOST:clientip} %{USER-:ident} %{USER-:auth}
\[%{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME} %{INT:ignore}\]
"%{NOTSPACE:method} %{NOTSPACE:request} HTTP/%{NUMBER:httpversion}"
(%{INT:httpcode})? (%{NUMBER:bytes})?( %{QS:agent})?(
"JSESSIONID=%{NOTSPACE:jsessionid};)?( _ga=%{NOTSPACE:lbcookie};)?(
%{NOTSPACE:sitecookie};)?( JSESSIONID=%{NOTSPACE:jsessionid2};)?(
session_id=%{NOTSPACE:session_id};)?(
s_session_id=%{NOTSPACE:s_session_id}")?

BBTOMCATACCESS %{IPORHOST:clientip} (-
%{NOTSPACE:connector}|(%{IPORHOST:clientip} %{NOTSPACE:connector}))
(%{USER:user_pk1}|{unset id}) %{TOMCATACCESSTIME:datetime}
"%{NOTSPACE:method} %{NOTSPACE:request} HTTP/%{NUMBER:httpversion}"
(%{INT:httpcode})? ((-)|(%{INT:bytes})?) (%{QS:agent} %{NOTSPACE:sessionid})
%{NOTSPACE:responsetime} %{NOTSPACE:2int} ((-)|(%{INT:bytes2})?)

TOMCATACCESSTIME \[%{MONTHDAY}/%{MONTH}/%{YEAR}:%{TIME}
%{INT:ignore}\]

APACHE_ERROR_TIME %{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}

APACHE_ERROR_LOG \[%{APACHE_ERROR_TIME:timestamp}\]
\[%{LOGLEVEL:loglevel}\] (?:\[client %{IPORHOST:clientip}\]
){0,1}%{GREEDYDATA:errormsg} (Rounin, D)

BBTOMCATERROR %{LOGLEVEL:loglevl}   \| jvm %{INT:jvm_id}    \|
%{YEAR:year}/%{MONTHNUM:month}/%{MONTHDAY:day} %{TIME:time} \|
%{GREEDYDATA:message}
```

Appendix B – Blocked Thread Script

```
#!/bin/bash

threaddump=/home/bbuser/threaddump.txt
bb_logs=/usr/local/blackboard/logs
threaddumparchives=$bb_logs/tomcat/thread_dump
datetime=$(date +"%Y-%m-%d-%T")


# Find Bb pid
bb_pid=`ps aux | grep bbuser | grep tomcat | grep 768m | awk '{print $2}'`
echo $bb_pid

# Use jstack to create a thread dump
/usr/local/jdk/bin/jstack $bb_pid > $threaddump

# Count blocked threads
BlockedCount=`cat $threaddump | grep BLOCKED | wc -l`


if [ "$BlockedCount" -gt 0 ];
 then
     /bin/touch /tmp/blockedthreads;
     echo $BlockedCount > /tmp/blockedthreads;
     date; echo "$BlockedCount BLOCKED THREADS.....";
     # Archive off the thread dump when more than 4 blocked threads
     if [ "$BlockedCount" -gt 4 ];
      then
          # Need to verify directory exist
          if [ ! -d "$threaddumparchives" ];
           then
               echo "Creating " $threaddumparchives;
               /bin/mkdir $threaddumparchives;
          fi
          # Now moving thread dump to archive location
          echo "Moving the thread dump to " $threaddumparchives;
          /bin/cp $threaddump $threaddumparchives/$datetime.txt;
     fi
 else
     /bin/touch /tmp/blockedthreads;
     echo $BlockedCount > /tmp/blockedthreads;
     date; echo "NO BLOCKED THREADS.....";
fi
```