ADP Capacity and Performance Analytics Service

Visualize with Grafana
GrafanaCon 2016

Carl De Pasquale, Ph. D.
Richard Flynn, Senior Director

11/28/2016

A more human resource.™

ADP Capacity and Performance Analytics Service

Principles of the Service

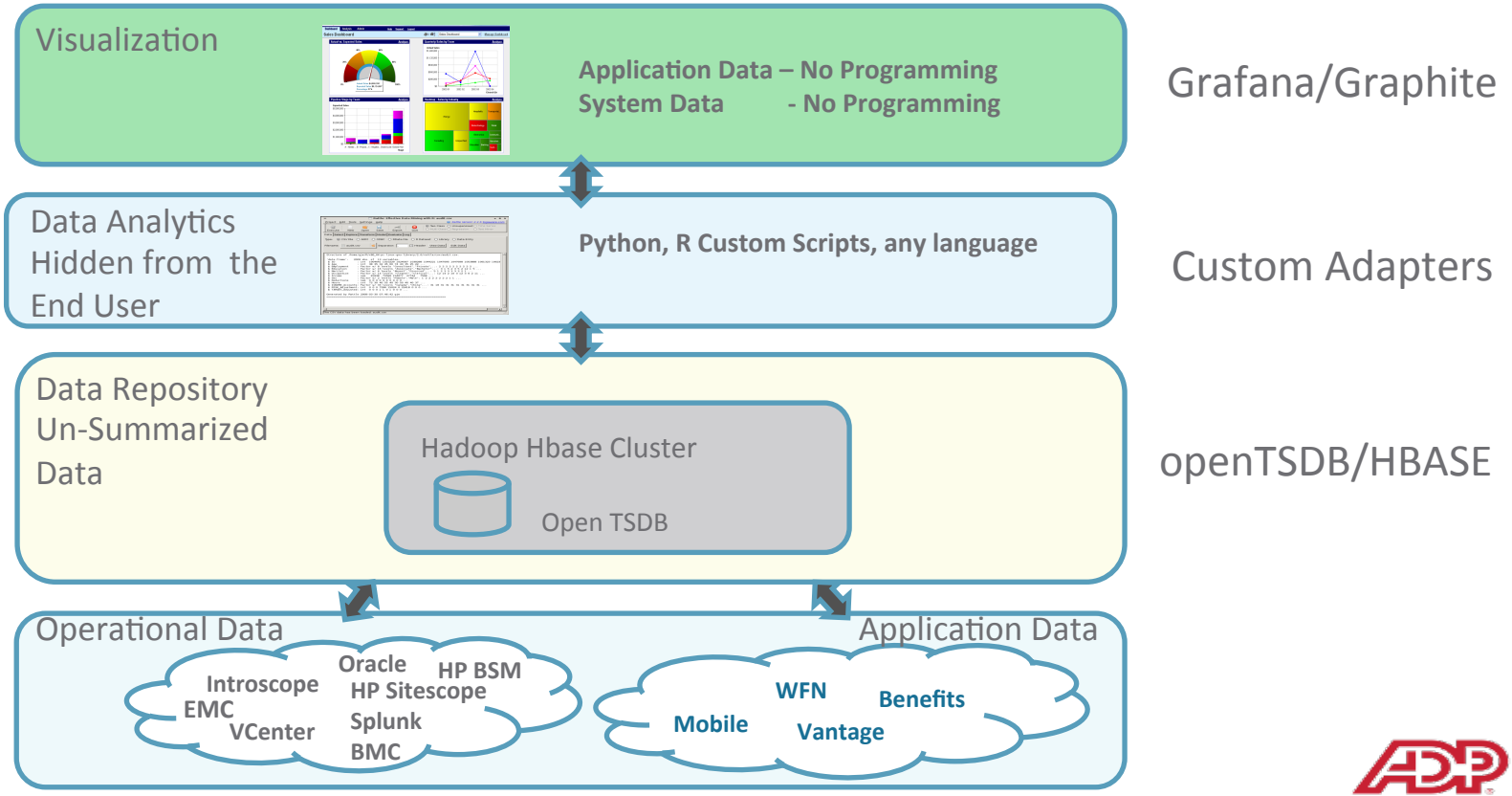
- Develop an approach to understand user behavior and its impact on the underlying infrastructure
- Provide easy to use statistical methods to evaluate data relationships
- Provide a single repository to retain non aggregated data
- Build a highly scalable and expandable clustered architecture

ADP Capacity and Performance Analytics Service

Principles of the Service (Continue)

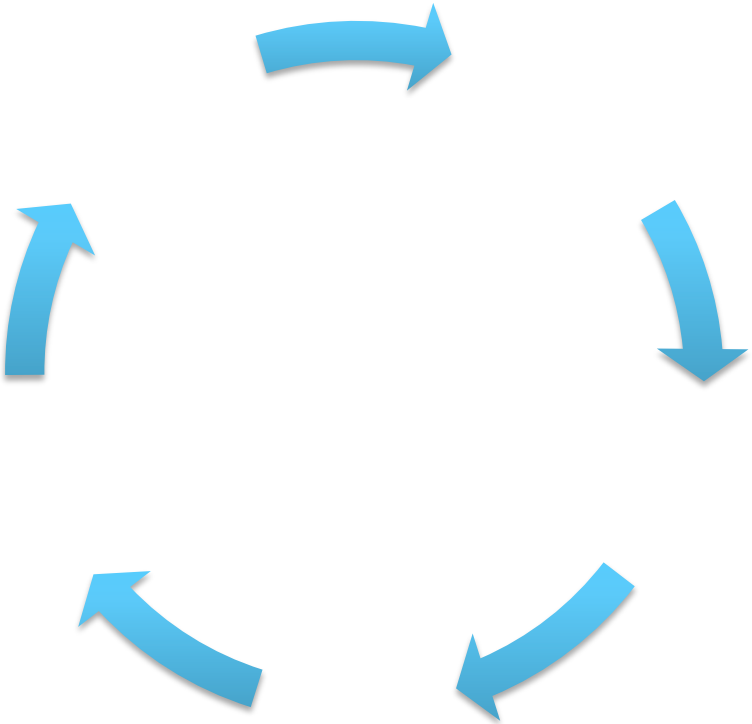
- Create a visualization layer that provides easy non-programmable access to data using prepackaged and user-driven dashboards
- Provide a service model available to all product and technology teams
- Use open source components

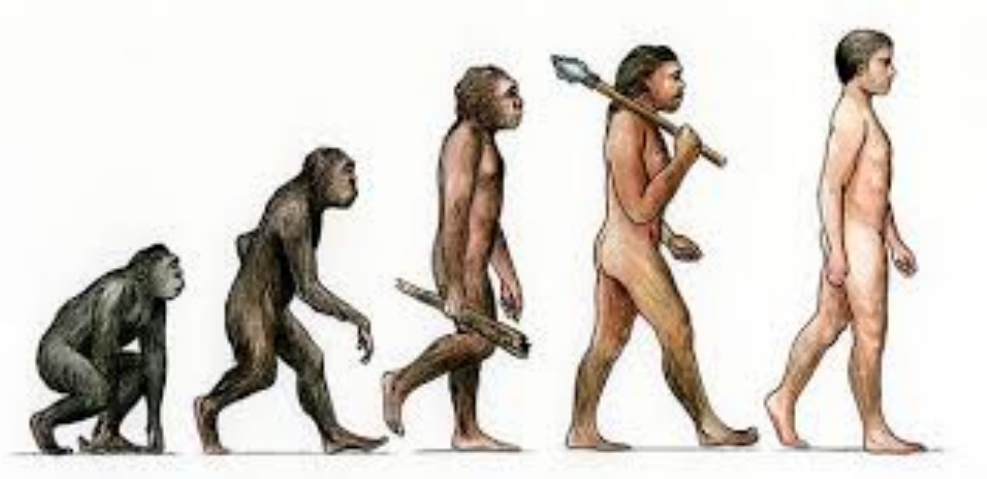
ADP Capacity and Performance Analytics Service



A more human resource.

Evolution





Evolution

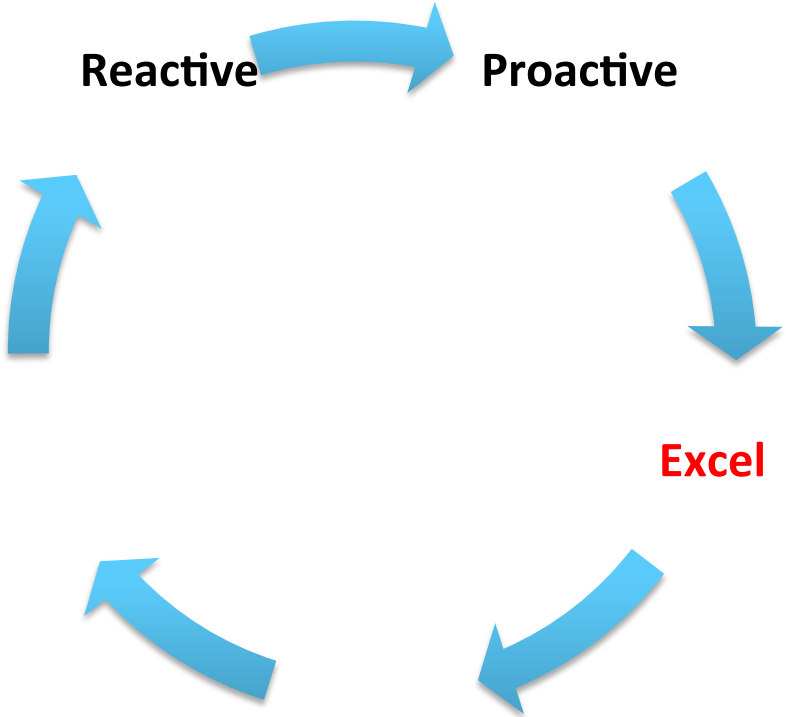
Reactive

Evolution

Reactive **Proactive**

A more human resource.

Evolution



Evolution

Reactive

Proactive
Manual
Performance
Data
Collection

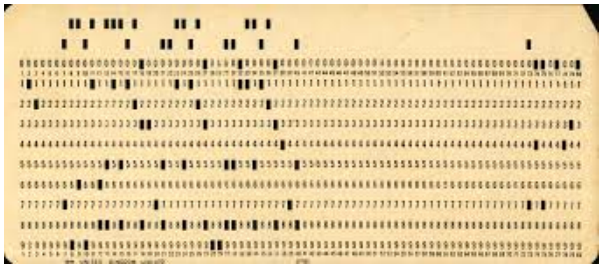
Excel

Proprietary software

A more human resource.

Evolution

Proactive
Manual
Performance
Data
Collection



Excel

Vendor
Product
Expensive
Feature
Limited

Open
software

Reactive

ADP Capacity and Performance Analytics Service

A flexible open software solution *that is easily*
Extensible *and* *provides*
Point & Click Analytics for Users *with a*
Well defined Data Model *that*
Easily, Inexpensively
(and Interminable) capable of
Retaining Un-Summarized data

A more human resource.™

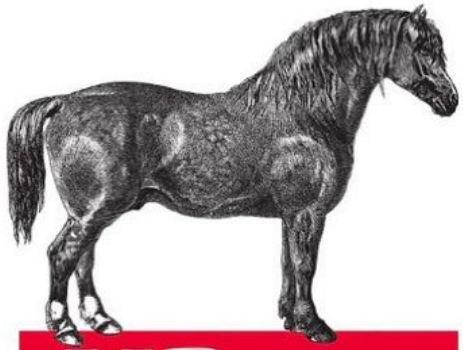
Grafana

graphite

Grafana

OPENTSDB

tCollector



A more human resource.

Data Visualization

Grafana

POD: P32A + P32B Server: dc2prezlm52mp4 + dc2prezlm56ws2 + dc2prezlm56ws3 + dc2prezlm56ws4 + dc2stezlmhdc2

Capacity Summary

Last Updated 04/25/2016

Average Client Growth per Month = 2500

Max Clients per Half Pod = 2800

BPT Details Here

Data Summary

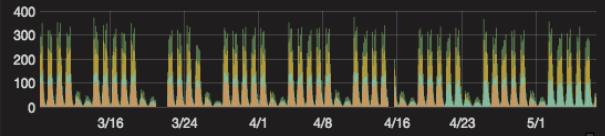
System metric and login data is currently in Splunk. Working on manually loading data into capacity warehouse until an autofeed is developed.

Splunk does retain enough data for capacity trending and correlation analysis.

Link to EZLM in Splunk

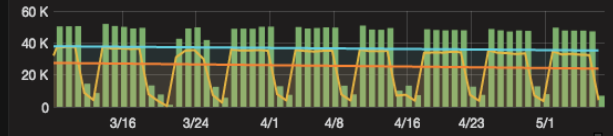
P32A + P32B

LOGINS and CLIENTS



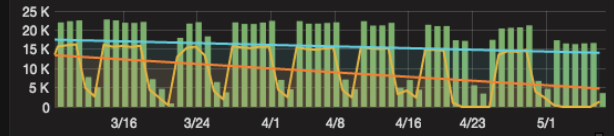
	min	max	avg	current
P32A.Total_Login	0	375	132	59
P32B.Total_Login	0	290	93	41
P32A.UniqueClients	0	147	57	32
P32B.UniqueClients	0	109	34	19

LOGINS FORECAST



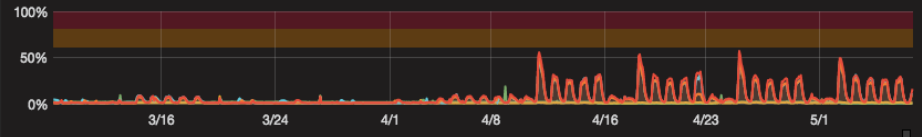
- P32A.Total_Login
- P32B.Total_Login
- linearRegression(summarize(Splunk.PROD.EZLM.Transactions.P32A.Total_Login, "1d", "su
- linearRegression(summarize(Splunk.PROD.EZLM.Transactions.P32B.Total_Login, "1d", "su

CLIENTS FORECAST



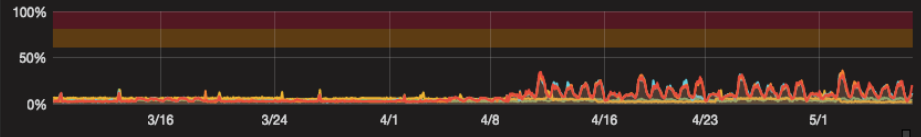
- P32A.UniqueClients
- P32B.UniqueClients
- linearRegression(summarize(Splunk.PROD.EZLM.Transactions.P32A.UniqueClients, "1d", "
- linearRegression(summarize(Splunk.PROD.EZLM.Transactions.P32B.UniqueClients, "1d", "

%CPU USED



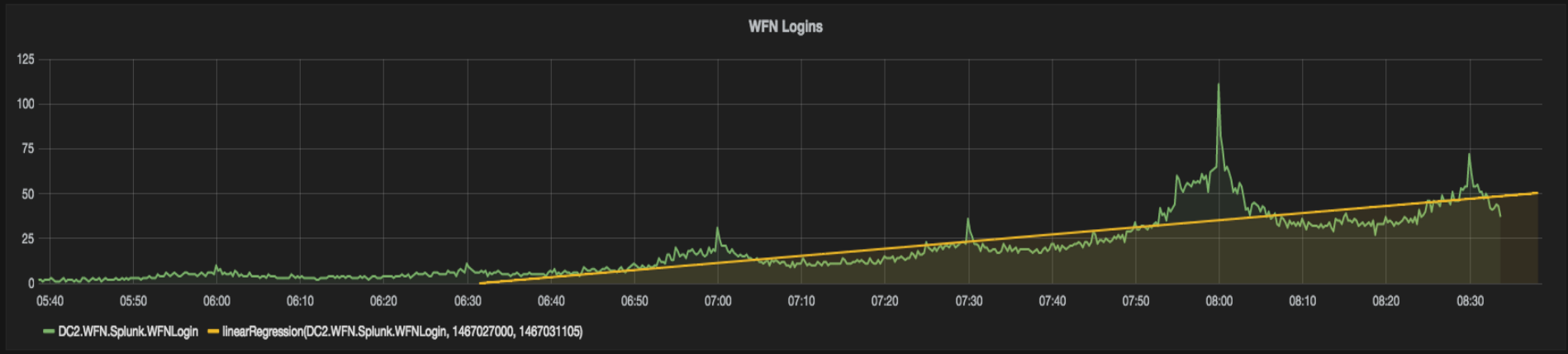
	min	max	avg	current
dc2stezlmhdc2	0%	19%	2%	0%
dc2prezlm52mp4	1%	8%	1%	1%
dc2prezlm56ws3	0%	49%	7%	10%
dc2prezlm56ws4	0%	47%	6%	8%

%MEMORY USED



	min	max	avg	current
dc2prezlm52mp4	2%	15%	4%	4%
dc2stezlmhdc2	1%	15%	5%	2%
dc2prezlm56ws4	2%	32%	8%	11%
dc2prezlm56ws3	2%	36%	8%	8%

WFN Logins | Back to dashboard | Zoom Out | Last 3 hours | Refresh every 5m



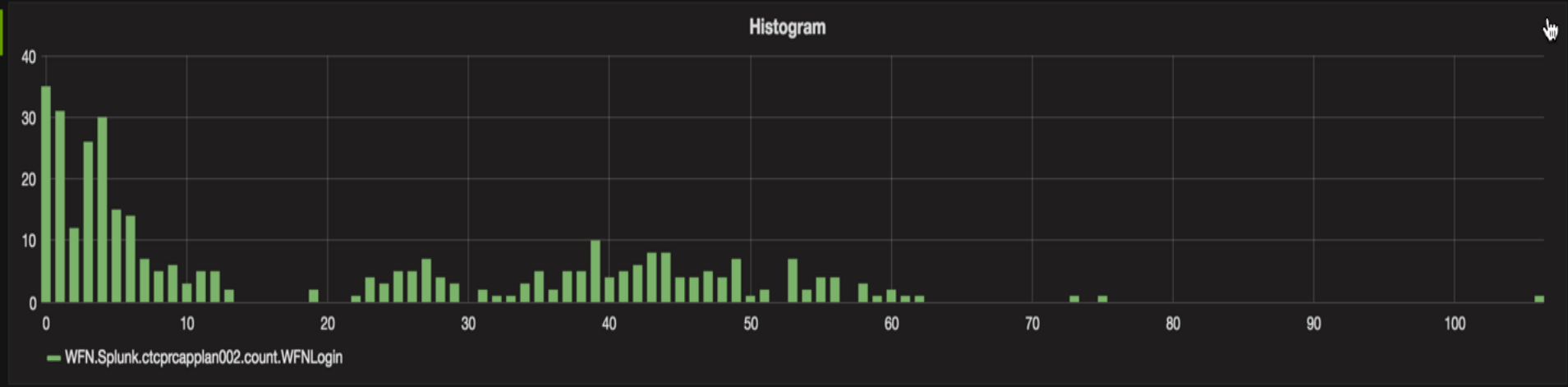
Graph | General | Metrics | Axes & Grid | Display Styles | Time range | Back to dashboard

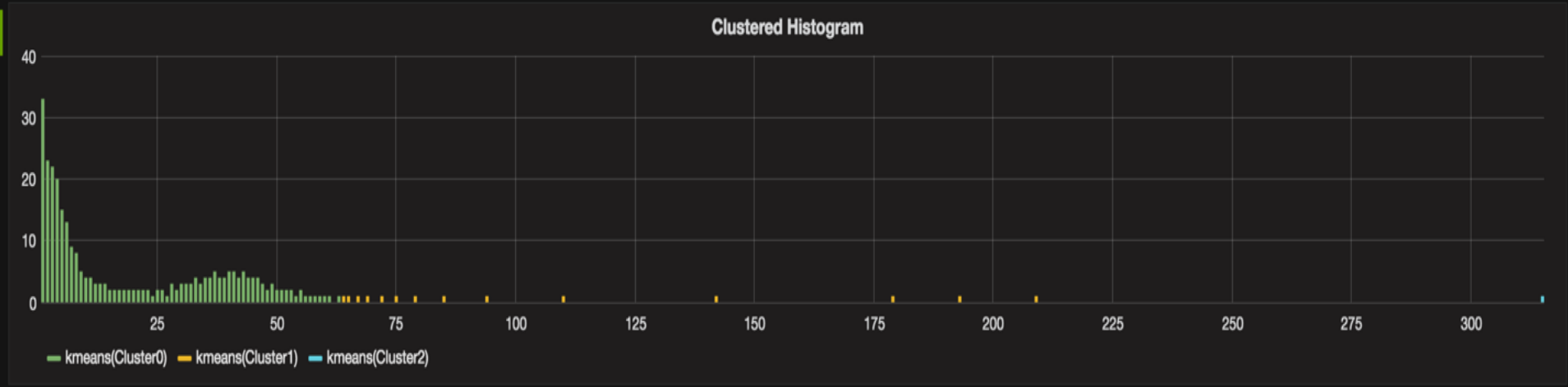
A	DC2	WFN	Splunk	WFNLogin	+		
B	linearRegression(DC2.WFN.Splunk.WFNLogin, '07:30 20160627')						

+ Query

Cache timeout	60	Max data points	auto		
shorter legend names	series as parameters	stacking	templating	max data points	

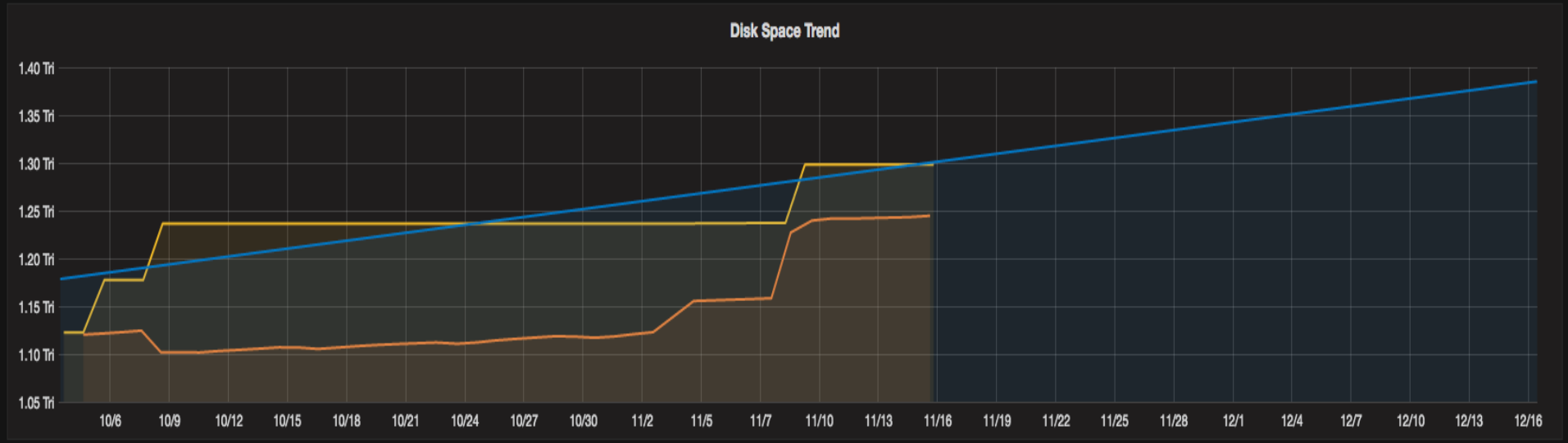
Graphite


Histogram
40
30
20
10
0
0
10
20
30
40
50
60
70
80
90
100
WFN.Splunk.ctcprcapplan002.count.WFNLogin


Clustered Histogram
40
30
20
10
0
25
50
75
100
125
150
175
200
225
250
275
300
kmeans(Cluster0)
kmeans(Cluster1)
kmeans(Cluster2)

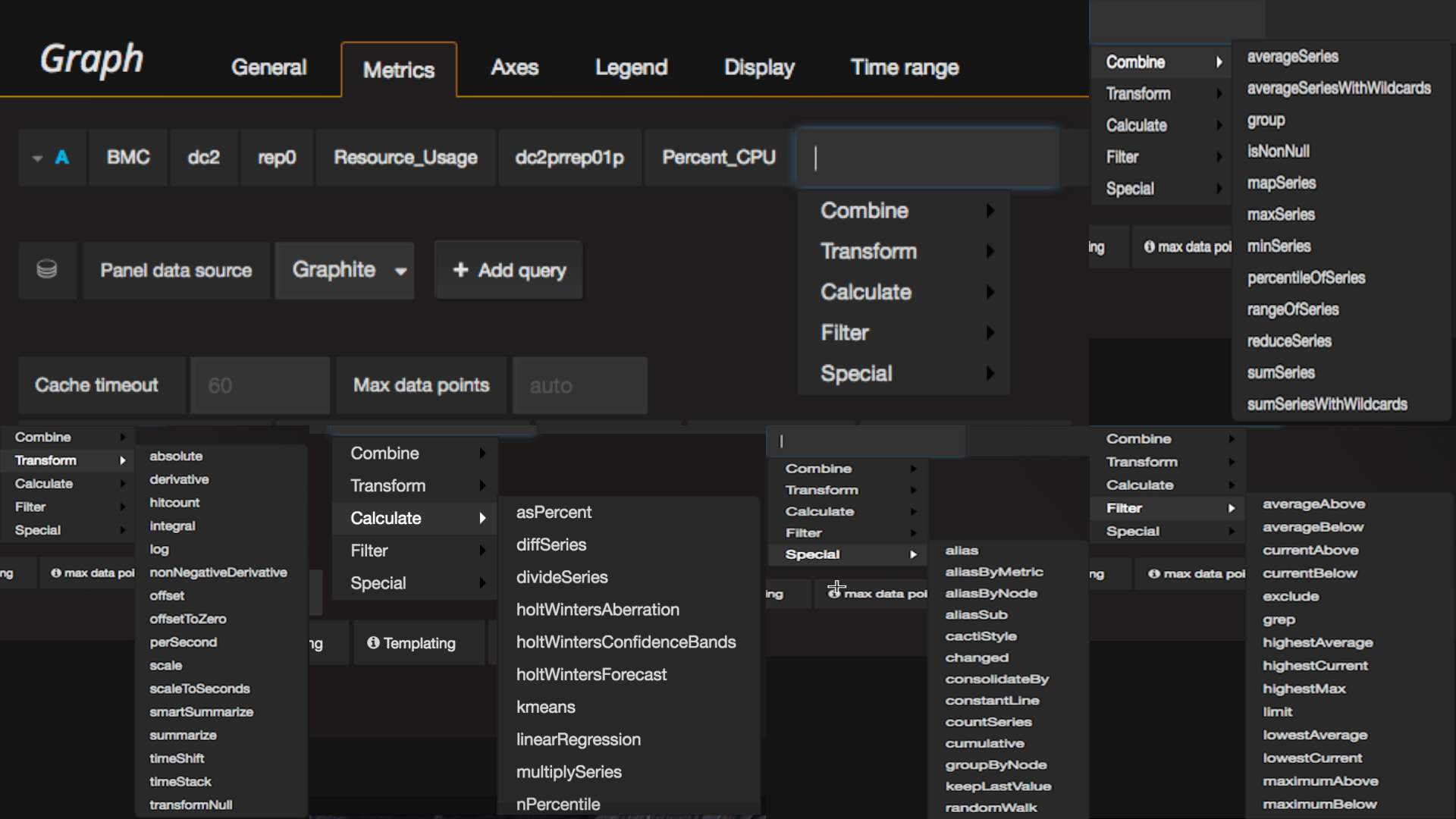
New dashboard

Back to dashboard | Zoom Out | Oct 3, 2016 09:03:19 to Dec 17, 2016 10:03:19 | Refresh every 5m



Graph | General | Metrics | Axes | Legend | Display | Time range

A	OEM	PROD	database	rac_database	wfc53p	DATA1_TablespaceAllocatedSpaceMB	+	
B	OEM	PROD	database	rac_database	wfc53p	DATA1_TablespaceUsedSpaceMB	+	
C	OEM	PROD	database	rac_database	wfc53p	DATA1_TablespaceAllocatedSpaceMB	linearRegression(00:00 20161001)	+


Graph
General
Metrics
Axes
Legend
Display
Time range
A
BMC
dc2
rep0
Resource_Usage
dc2prrep01p
Percent_CPU
Panel data source
Graphite
+ Add query
Cache timeout
60
Max data points
auto
Templating
Combine
Transform
Calculate
Filter
Special
absolute
derivative
hitcount
integral
log
nonNegativeDerivative
offset
offsetToZero
perSecond
scale
scaleToSeconds
smartSummarize
summarize
timeShift
timeStack
transformNull
asPercent
diffSeries
divideSeries
holtWintersAberration
holtWintersConfidenceBands
holtWintersForecast
kmeans
linearRegression
multiplySeries
nPercentile
alias
aliasByMetric
aliasByNode
aliasSub
cactiStyle
changed
consolidateBy
constantLine
countSeries
cumulative
groupByNode
keepLastValue
randomWalk
averageSeries
averageSeriesWithWildcards
group
isNonNull
mapSeries
maxSeries
minSeries
percentileOfSeries
rangeOfSeries
reduceSeries
sumSeries
sumSeriesWithWildcards
averageAbove
averageBelow
currentAbove
currentBelow
exclude
grep
highestAverage
highestCurrent
highestMax
limit
lowestAverage
lowestCurrent
maximumAbove
maximumBelow

OpenTSDB

Data Model

OpenTSDB

ADP
A more human resource.

OPENTSDB

MetricName TimeStamp MetricValue [TagName=TagValue]*

CPU 1465830900 10.0 Host=XYZZY

Example tags

- **source=**
- **host=**
- **processingCenter=**
- **application=**
- **other=**

OPENTSDB

A more human resource.

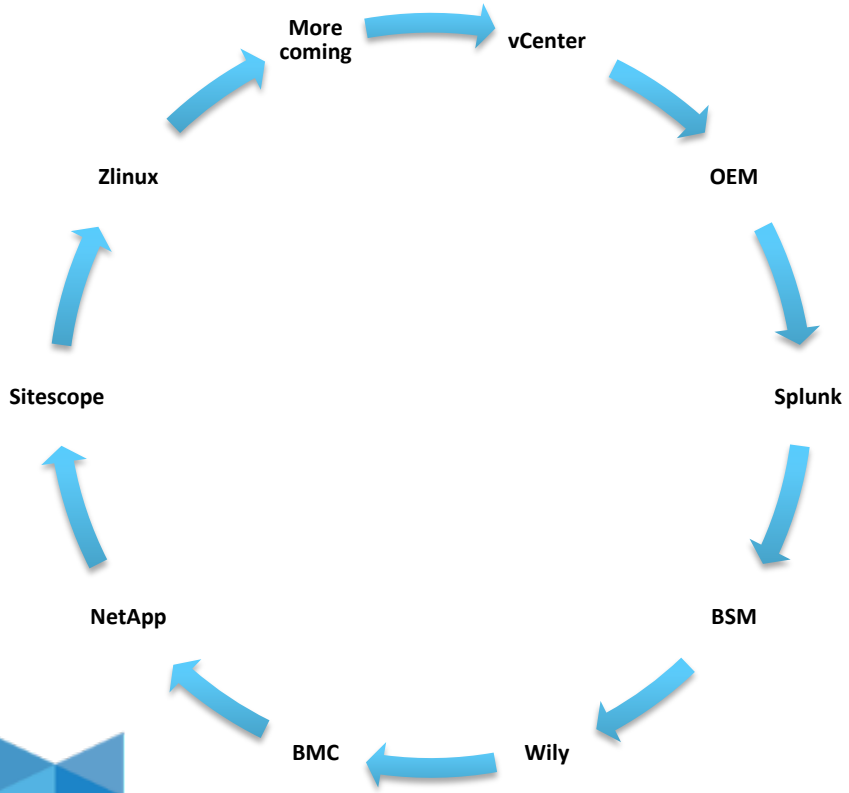
OPENTSDB

Data Sources & Adapter Scheduling

OPENTSDB

A more human resource.

Data Sources



58 adapters, more coming

- tCollector - schedules adapters
  - Adapters stored in directories with numerical names indicating
    - Time sweep
      - Adapters stored in directory ‘300” run every 5 minutes

- Adapters may be written in any language;
  - Adapter written in Python, R, GoLang
  - Adapter spools results to standard out

- tCollector loads data into OpenTSDB/HBASE

ADP Capacity and Performance Analytics Service

- Four HBASE Tables
  - TSDB – 6, 258, 246, 167 raw non-aggregated data points
  - 200 GB storage compressed

OpenTSDB

tCollector

Sample Adapter Code

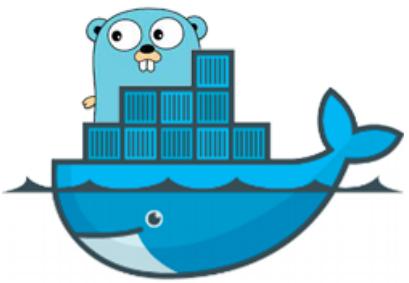
ADP
A more human resource.

Splunk

```
baseurl = 'https://' + ip + ':443
searchQuery = 'search index=…timechart span=5m limit=50 count as TotalLogins

########Establish Session
serverContent = httplib2.Http(disable_ssl_certificate_validation=True).request(baseurl +
'/services/auth/login',
   'POST', headers={}, body=urllib.urlencode({'username':userName,
'password':password}))[1]
sessionKey = minidom.parseString(serverContent).getElementsByTagName('sessionKey')[0].childNodes[0].nodeValue

#Submit Query get query ID
returnedSid =  httplib2.Http(disable_ssl_certificate_validation=True).request(baseurl + '/
servicesNS/-/ezlm_main/search/jobs','POST',
   headers={'Authorization': 'Splunk %s' % sessionKey},body=urllib.urlencode({'search':
searchQuery}))[1]
xmldoc = minidom.parseString(returnedSid)
for element in xmldoc.getElementsByTagName('sid'):
  sid = element.firstChild.nodeValue
```

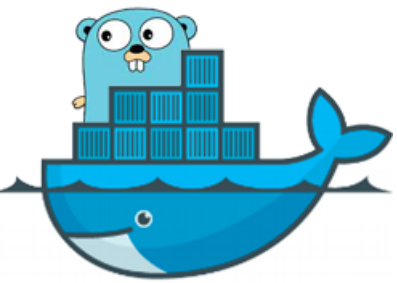
Splunk

```
#########wait for query to finish
statusQuery = '/services/search/jobs/%s/' % sid
while notDone:
   searchstatus = httplib2.Http(disable_ssl_certificate_validation=True).request(baseurl + statusQuery,
    'GET', headers={'Authorization': 'Splunk %s' % sessionKey},
       body=urllib.urlencode({'username':userName, 'password':password}))[1]
   doneStatus = re.compile('isDone">(0|1)')
   doneStatus = doneStatus.search(searchstatus).groups()[0]
   if (doneStatus == '1'):
      notDone = False

#Retrieve results
resultsQuery = '/services/search/jobs/%s/results?output_mode=csv' % sid
searchResults = httplib2.Http(disable_ssl_certificate_validation=True).request(baseurl +
resultsQuery,
    'GET', headers={'Authorization': 'Splunk %s' % sessionKey},
body=urllib.urlencode({'username':userName, 'password':password}))[1]

############Format results and write to stdout By processing searchResults
```



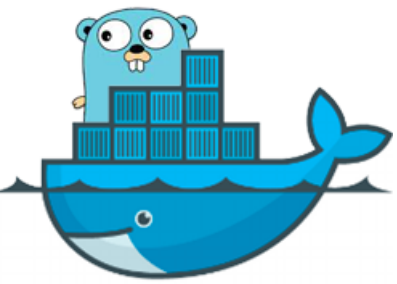
Docker

```
package main
import (
        "fmt"
        "log"
        "time"
        "github.com/fsouza/go-dockerclient"
)
func containerCpuPercent(x float64, y float64) *big.Float {
  numerator,  denominator, percent100 :=
                              big.NewFloat(x), big.NewFloat(y), big.NewFloat(100)
  z    := new(big.Float).Quo(numerator, denominator)
  z     = new(big.Float).Mul(z,percent100)
  return z
}
func main() {
        endpoint := "unix:///var/run/docker.sock"
        client, err := docker.NewClient(endpoint)
        if err  != nil {
                log.Fatal ("NewClientConnect Failed\n" )
        }
        statsOption   := docker.StatsOptions{}
        done          := make(chan bool)
        errC          := make(chan error, 1)
```



Docker

```
for ;; {
        containers, _ := client.ListContainers (docker.ListContainersOptions {All: false})
        for _, container :=  range containers {
                stats                   := make(chan *docker.Stats)
                statsOption             = docker.StatsOptions{}
                statsOption.ID          = container.ID
                statsOption.Stream  = true
                statsOption.Stats       = stats
                statsOption.Done        = done
                go func() {
                        errC <- client.Stats(statsOption)
                }()
                done <- true
                if prevCpuStats[name] != 0 {
                  numerator   := float64(currCpuStats[name]-prevCpuStats[name])
                  denominator := float64(prevCpuStats[name])
                  cpuPercent := containerCpuPercent (numerator, denominator )
                  fmt.Printf ( "cpuUsage %v %v  container=%s\n", tyme, cpuPercent, container.Names )
                }
                prevCpuStats[name] = currCpuStats[name]
        } //container loop
        time.Sleep ( 5 * time.Second )
        fmt.Println()
} // infinite loop }//end main
```



Docker

```
cpuUsage 1478970269 0.0036428  container=[/sysproc]
cpuUsage 1478970269 0  container=[/ucp-controller]
cpuUsage 1478970270 0  container=[/ucp-swarm-ca-proxy]
cpuUsage 1478970272 0  container=[/ucp-swarm-ca /ucp-swarm-ca-proxy/cfssl]
cpuUsage 1478970273 0  container=[/ucp-ca-proxy]
cpuUsage 1478970274 0  container=[/ucp-ca /ucp-ca-proxy/cfssl]
cpuUsage 1478970274 0.000212623  container=[/ucp-swarm-manager]
cpuUsage 1478970276 8.28166e-05  container=[/ucp-swarm-join]
cpuUsage 1478970279 0  container=[/ucp-proxy]
cpuUsage 1478970279 0.000965825  container=[/ucp-kv]
```

```
read.opentsdb <- function(server.url, start.date, end.date=NA, agg="avg",
                                    opentsdb.metric="MetricName", opentsdb.tags="TagName)
        require (RCurl)
        qStr <- paste(server.url, "/q?start=", start.date, "&end=", end.date, "&m=",
                                                        agg, ":", opentsdb.metric, opentsdb.tags, "&ascii",sep="")
        tsdb <- getURL (qStr)
        tsdb <-gsub(' ',',',tsdb)
        tsdb<-unlist(strsplit(tsdb,"\n"))
        i <- 0
        df <- data.frame(stringsAsFactors=FALSE)
        while(i<=length(tsdb))
        {
                t <- unlist(strsplit(tsdb[i],","))
                metric <- t[1]
                time   <- as.numeric(t[2])
                value  <- as.numeric(t[3])
                tag1   <- t[4]
                tag2   <- t[5]
                tag3   <- t[6]
                tag4   <- t[7]
                tag5   <- t[8]
                i <- i + 1
                df1 <- data.frame (metric,time,value,tag1,tag2,tag3,tag4,tag5,stringsAsFactors=FALSE)
                df <- rbind (df,df1)
        }
        return (df)
}
```