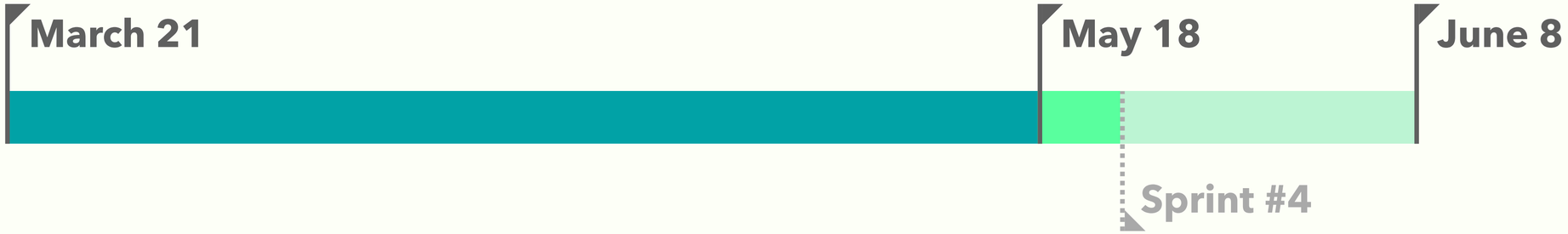
alphadoop

Doflamingo

An light-weight monitoring system for Apache Hadoop

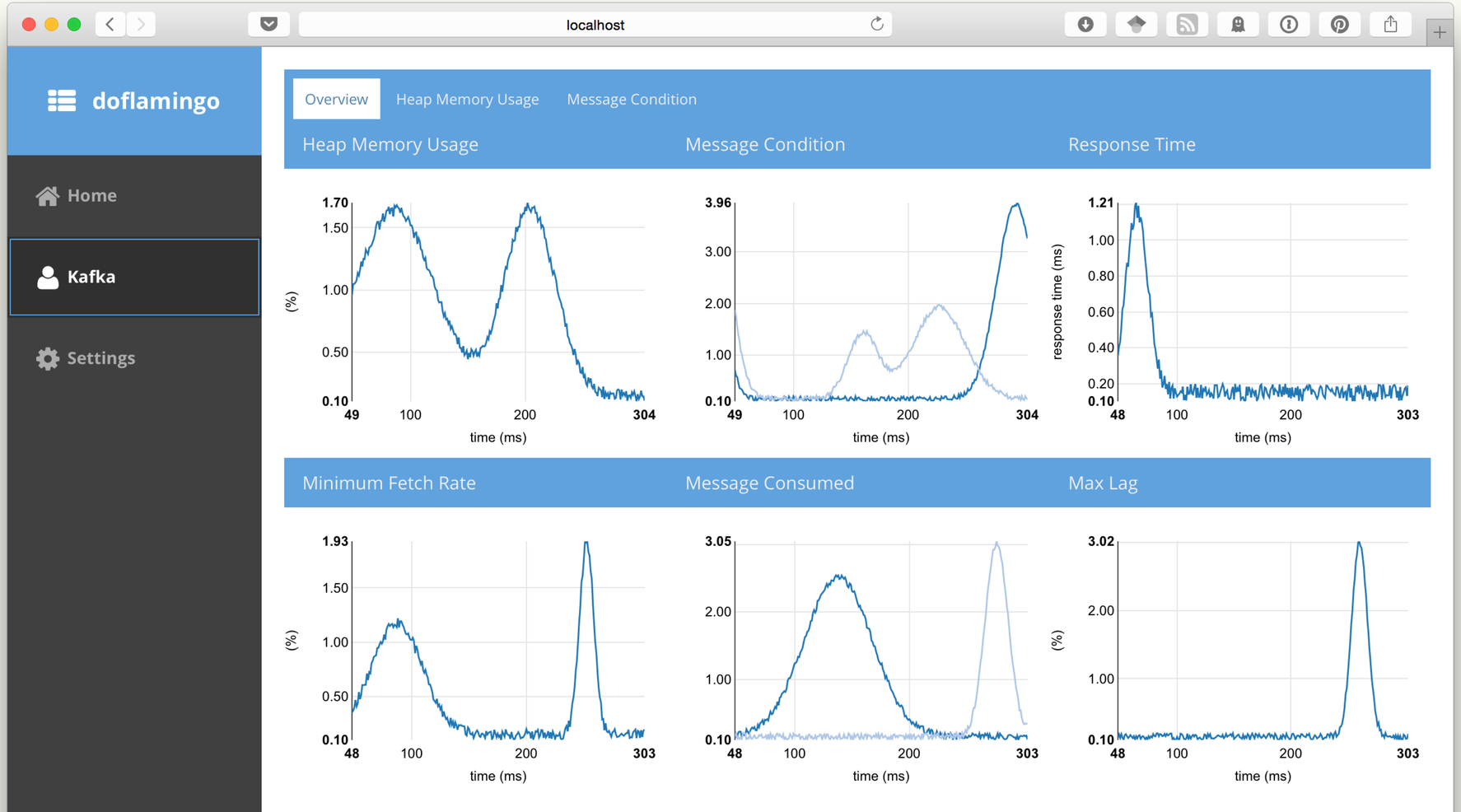
TITLE	**Kafka/ Zookeeper Monitoring Module built for Flamingo Ecosystem**
DURATION	**March 13, 2016 ~ June 8, 2016**
CLIENT	**EXEM**
PRESENTER	**ALPHADOOP**

TIMELINE



OBJECTIVES

_ LOOK OF PROGRAM



PART_01

SOFTWARE REQUIREMENTS SPECIFICATION

_ WHAT WE WILL DO

Collect Performance Metrics,
Visualize it, and
Integrate it with Flamingo.

_ WHAT WE WILL DO

Is all system working properly?

hadoop

kafka

Doflamingo

Of Course!

Check this out!

_ WHY WE NEED THIS PROJ

1. Hard to understand Hadoop

Distributed system — not intuitive

Unable to track fluctuant mass traffic

Eyes on only the upper level
— run and hope everything goes well

_ WHY WE NEED THIS PROJ

2. The Missing Link of Flamingo

Currently flamingo is able to monitor:

- Resources
- YARN application
- Map Reduce
- Nodes

_ HOW WE DO IT

Learn from other monitoring tools

Plenty of tools exists in the field – Learn from them and try to build up similar metrics

Build it into flamingo platform

There’s flamingo’s way of monitoring hadoop system. Add a new task into jobscheduler.

_ HOW WE DO IT

AGILE APPROACH

1 SPRINT = 2 WEEKS

TOTAL 5 SPRINTS along the semester

_ REQUIREMENTS

1. Built as a part of Flamingo system
2. Monitor and Report in Real-time
3. Utilize JVM ecosystem
4. Visualize the metrics, avoid numbers
5. Save metrics into Database
6. Special caution on log management

_ OBJECTIVES

O1: Set up an environment for Flamingo

O2: Define Kafka measurement metrics, visualization forms

O3: Implement API server which provides collected metrics

O4: Implement charts with Sencha

O5: Integrate with Flamingo Ecosystem

KAFKA MODULE **M1** →

O6: Define Zookeeper measurement metric, visualization

O7: Implement a Zookeeper monitoring module on Flamingo

ZOOKEEPER MODULE **M2** →

SPRINT 1

SPRINT 2

SPRINT 3

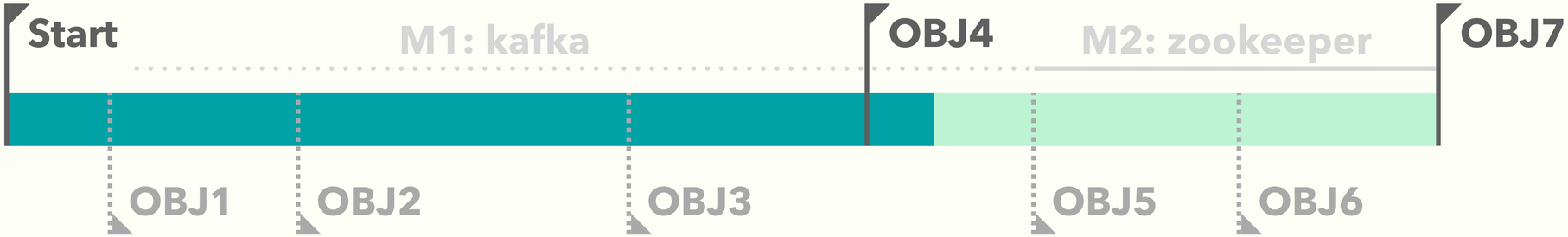
SPRINT 4

SPRINT 5

_ HOW FAR WE CAME

TIMELINE

OBJECTIVES



_ WHO WILL DO WHAT

TEAM _ ALPHADOOP

SEUNGHYO KANG *the hadoop master*

← Metric Analysis

RESTful Server →

JARYONG LEE *the spring master*

YOUNGJAE CHANG *the sencha master*

← Visualization

_ WE ARE RESPONSIBLE FOR:

1. Built as a open source software

Fork and request merge into flamingo

License/ Copyrights are same with flamingo

2. Bye-bye after spring semester

A/S are not supported after June 21, 2016

_ WE ONLY HAVE THESE:

LIMITED TIME: 10 WEEKS

No delay accepted – when semester ends, project should be ended

LIMITED DEVELOPERS: 3 PEOPLE

No one will help us
– no money to hire someone!

PART_02

BACKGROUND RESEARCH

_ SAY HELLO TO MONITORING

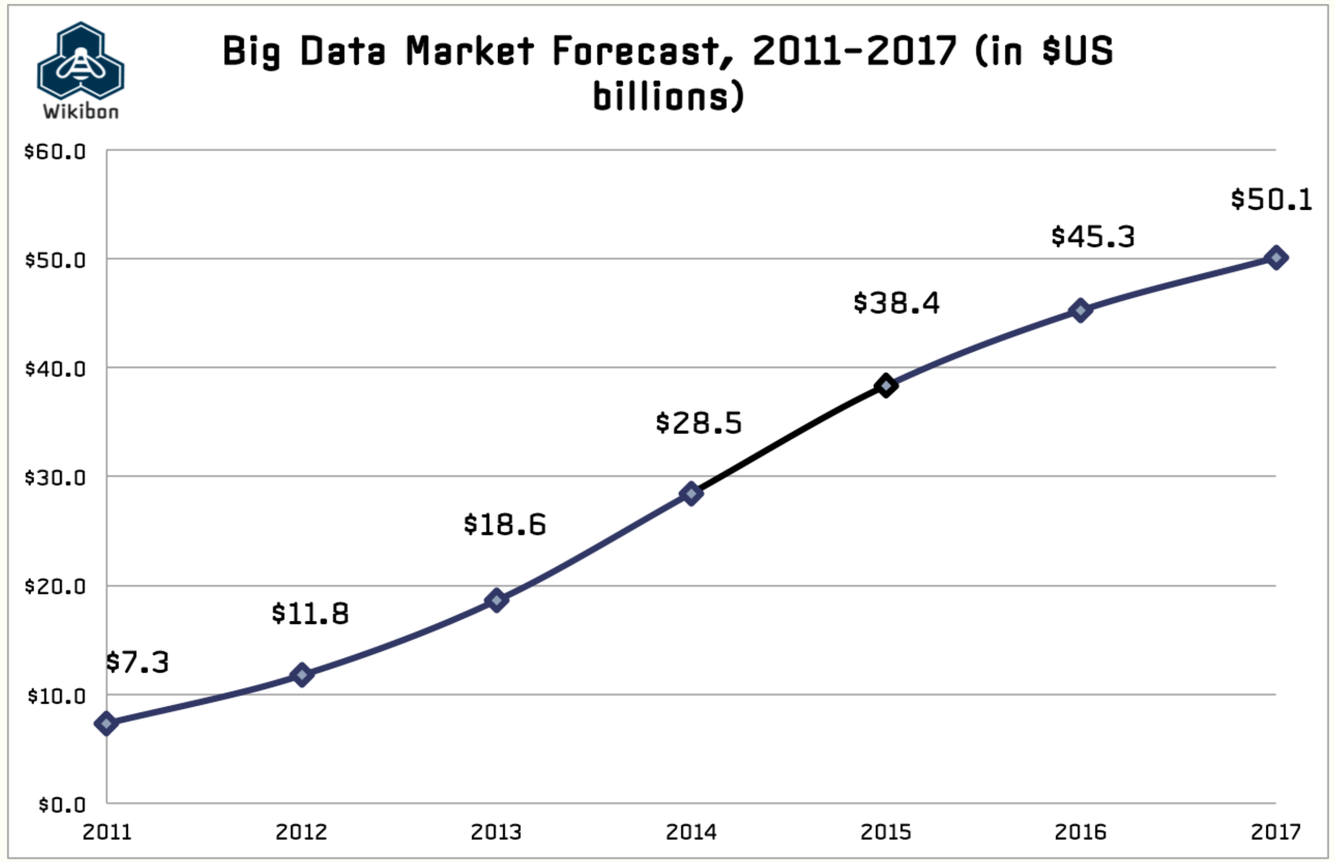
Seeing is believing

Software is intangible; so, where can we find it?

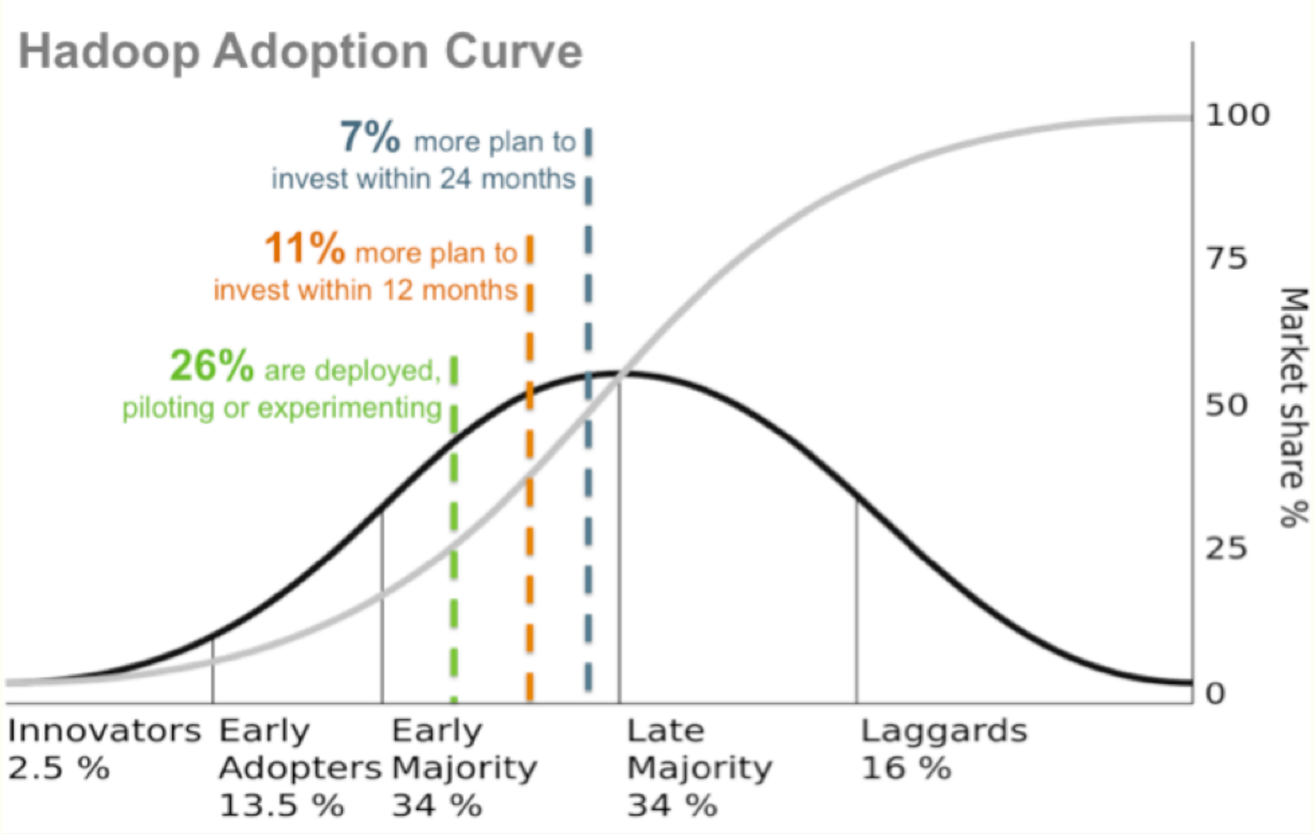
Bigdata: the buzz needs money

Hadoop is a money-eater:
10+ nodes, consulting, (expensive) engineers

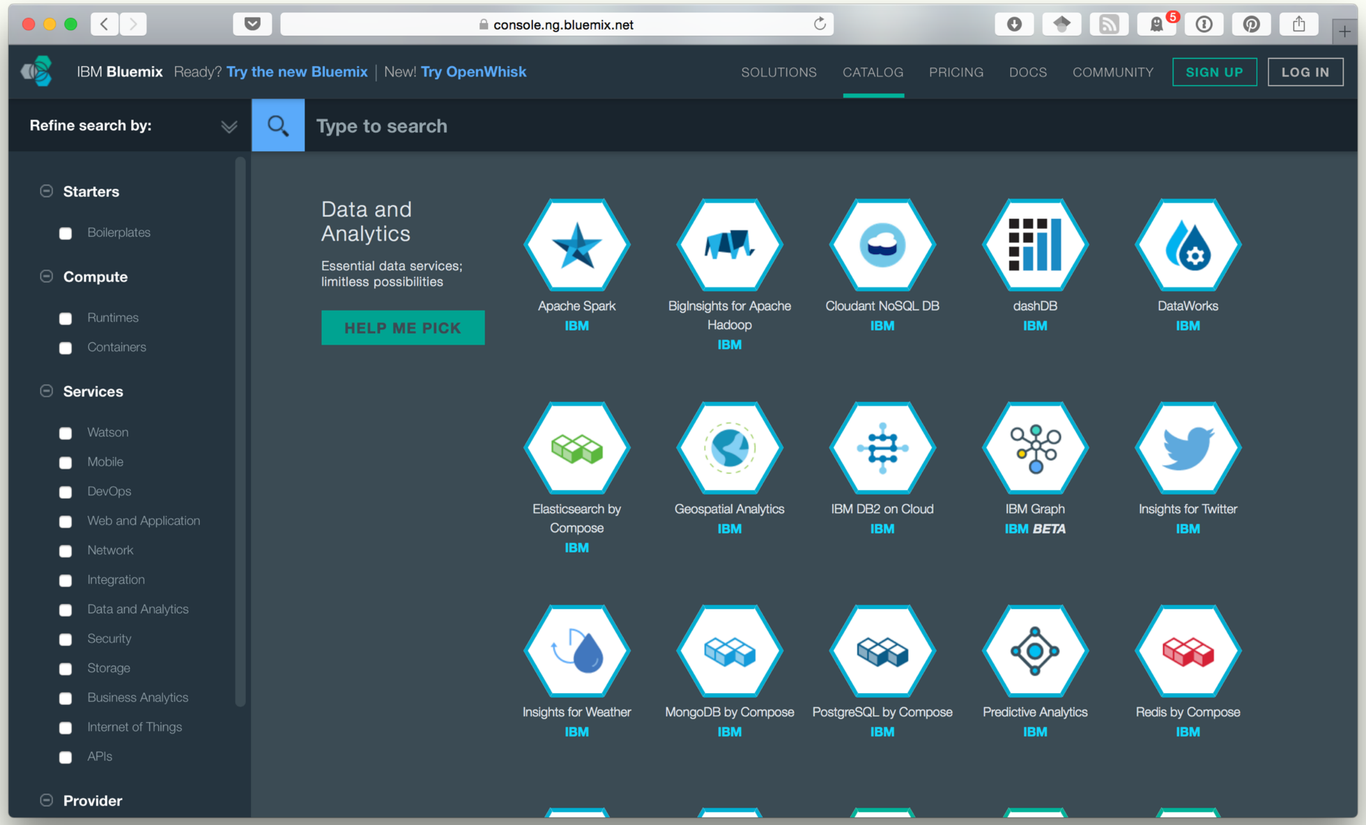
_ SAY HELLO TO MONITORING



_ SAY HELLO TO MONITORING



_ FUTURE OF FLAMINGO



Opensource

IBM

Bluemix

_ THE EFFECT OF OUR WORK

The ultimate control tower

Flamingo now monitors not only nodes,
but also modules that compose pipeline.

Opening up new possibility

The gathered metrics can be used for further
optimization or anomaly detection feature.

_ TECHNICAL DETAILS

[A] WHAT IS KAFKA?

A high-throughput distributed messaging system

BENEFITS

- Scalable
- High-throughput
- Distributable
- Low response time
- Save on data disk

USED IN

- LinkedIn
- Twitter
- Netflix
- Tumblr
- Foursquare

_ TECHNICAL DETAILS

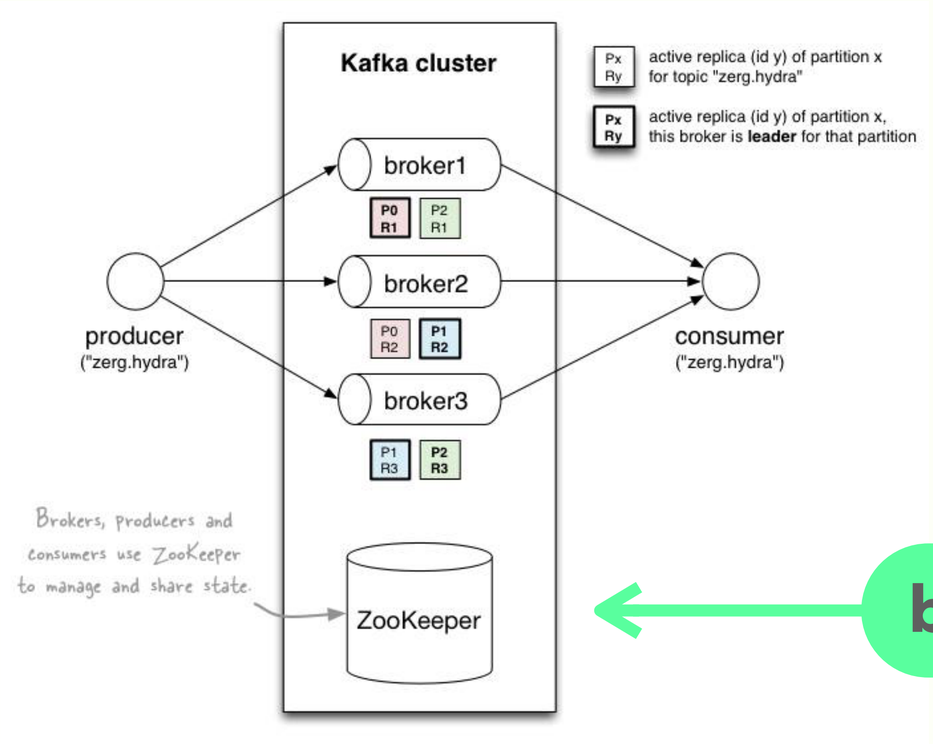
[A] WHAT IS KAFKA?

Kafka consists of producer, broker, and consumer, managed by **Zookeeper**

Producers send system messages to **brokers**
Brokers process them **distributively**
Consumers store the messages to their **disks**

_ TECHNICAL DETAILS

[A] WHAT IS KAFKA?



producer
generate message that fall into certain topics

consumer
subscribe specific topics & process the message

broker
stacks up logs based on topic

_ TECHNICAL DETAILS

[A] WHY KAFKA?

Store the messages in the **DISK**, not in the cache.

Consumers can rewind back to old data and re-consume them since they are in the disk for a certain period of time.

PULL model, not push model

consumer pull messages from broker without exceeding their limit; no drop occurs unlike producer-push model

_ TECHNICAL DETAILS

[B] WHAT IS ZOOKEEPER?

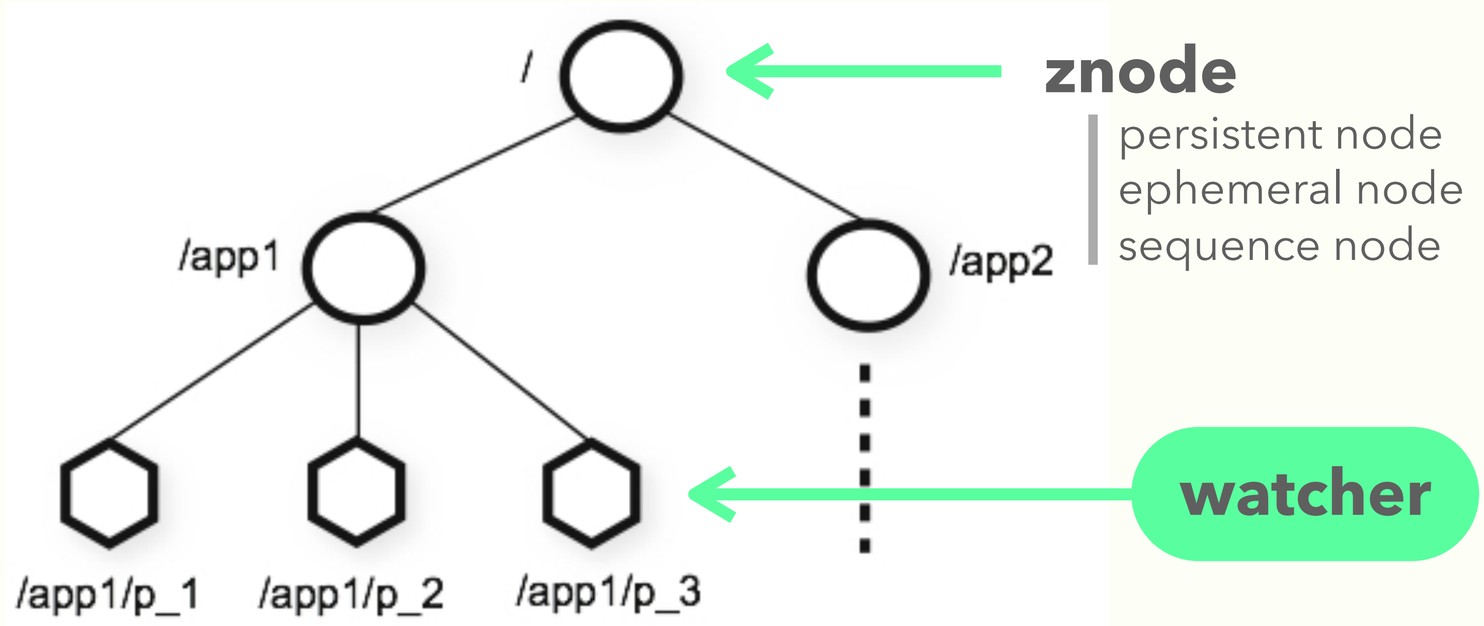
Handles various errors in distributed systems.

Four Features

Using name service to separate loads.
Using distributed lock to handle synchronization error
Error detection and recovery
Configuration management

_ TECHNICAL DETAILS

[B] WHAT IS ZOOKEEPER?



http://bcho.tistory.com/1016

PART_03

USER SCENARIO DESIGN

_ QUESTION

PHASE #1 **What is a monitoring?**

PHASE #2 **Why do we monitor?**

_ TWO NEEDS

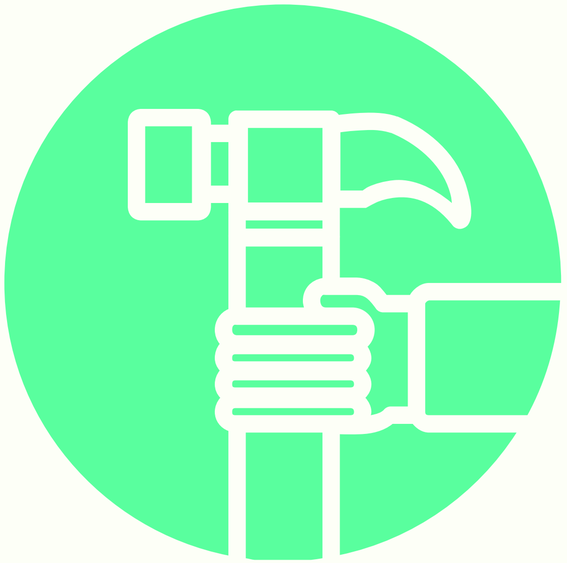
*To ensure the **normal operation** of the system*

*To find out the cause of **abnormal behavior***

_ TWO USERS

USER #1

Administrator



USER #2

Engineer

_ USERS

USER #1

Administrator

A. Hope everything stays normal
B. Determine whether to put more resources or not
C. Usually maintains a volume of system
D. Focus on real-time data

USER #2

Engineer

A. Fix the problem
B. Find out the cause of the problem by traveling the past data
C. Deeper understanding on whole system
D. Focus on specific events

_ DIFFERENT REQUIREMENTS

USER #1

Administrator

A. Visualize constantly changing statistics of sys.

B. At a glance view of metrics

C. Real-time update without user intervention

USER #2

Engineer

A. Visualize abrupt events

B. Can travel back to the past to find the cause of event

C. Detailed analysis on changing variables during specific timeframe

_ EXTERNAL INTERFACE

FUNC #1

Overview

A. Dashboard

B. ~~Configuration~~

FUNC #2

Timeline

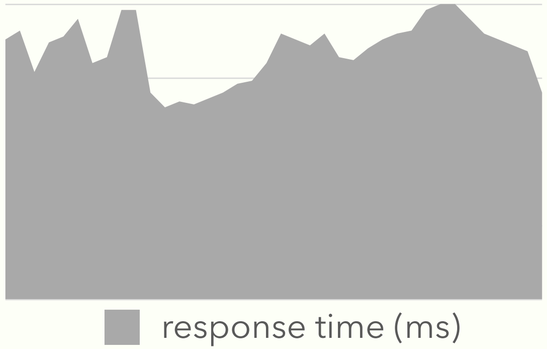
A. Event Timeline

B. Timemachine

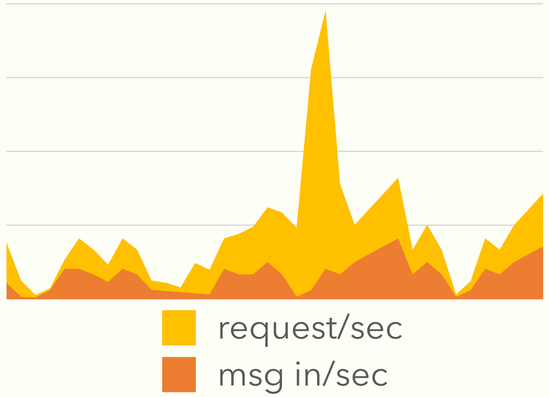
FUNC #1

Overview

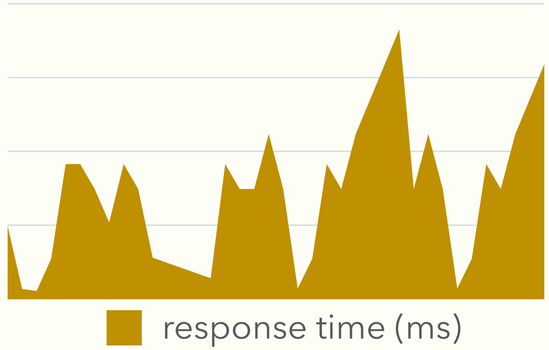
Heap memory usage



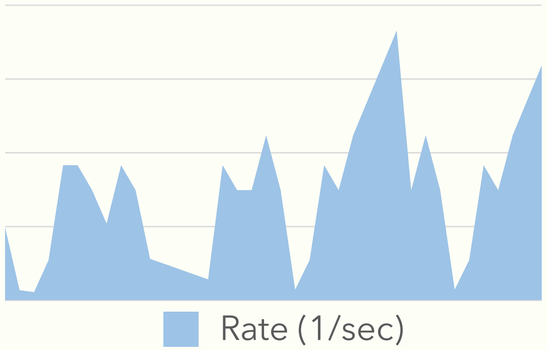
Message Condition



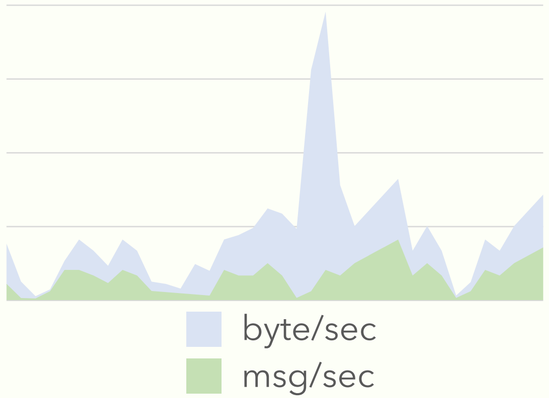
Response time



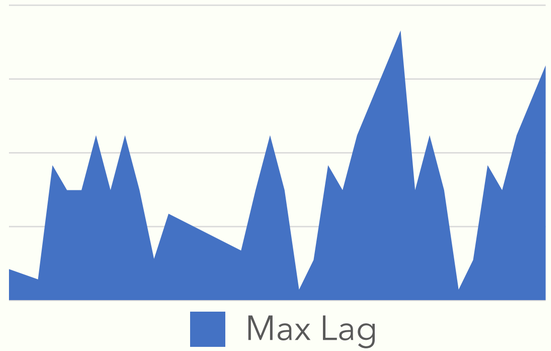
Minimum Fetch rate



Message Consumed

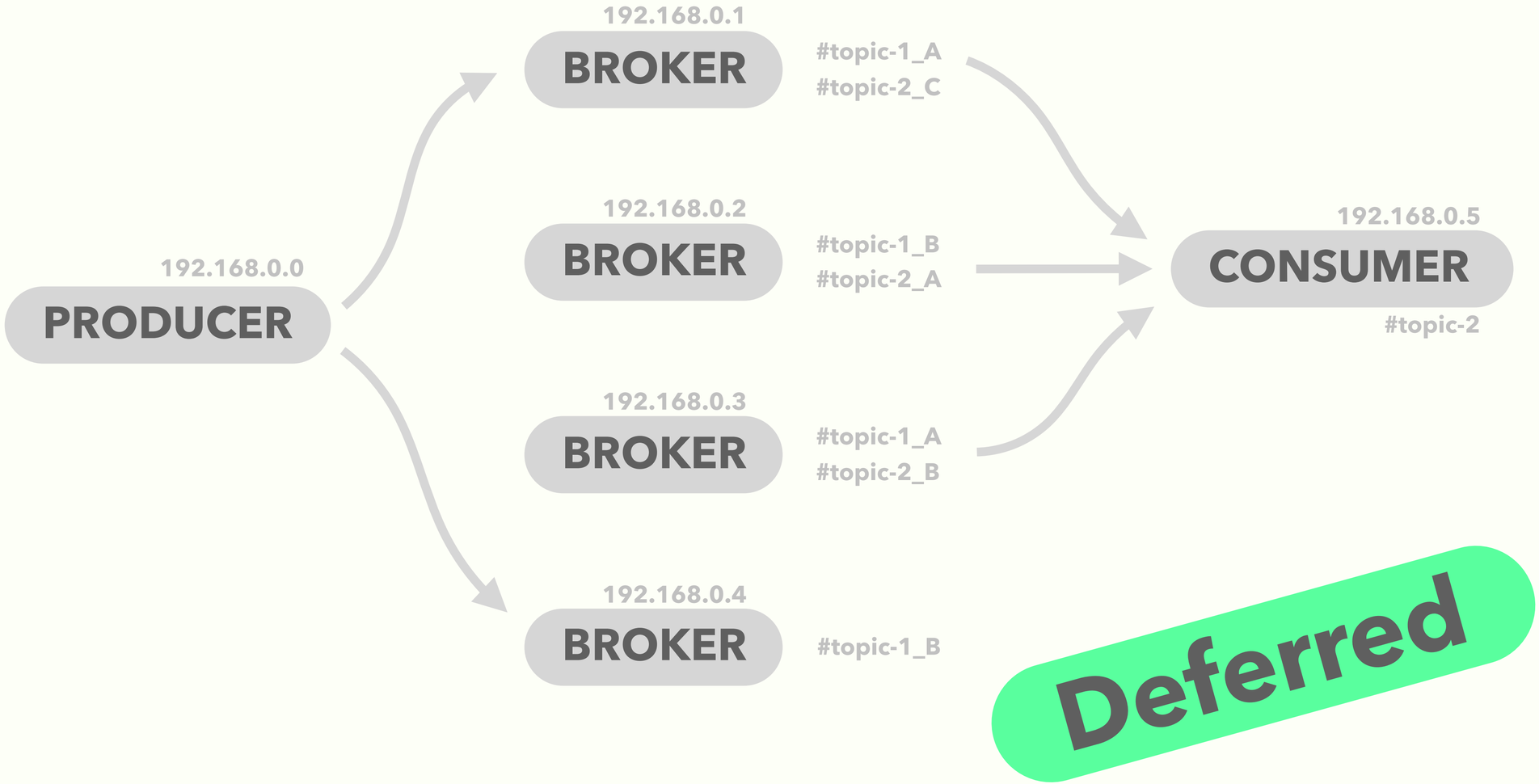


Max Lag



FUNC #1

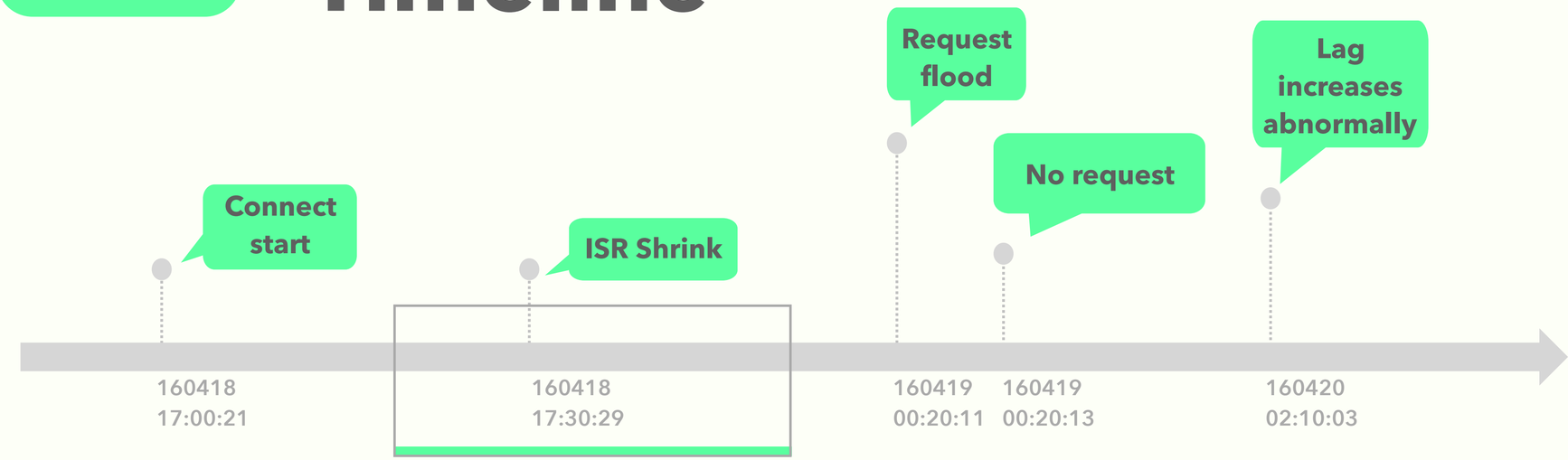
Overview



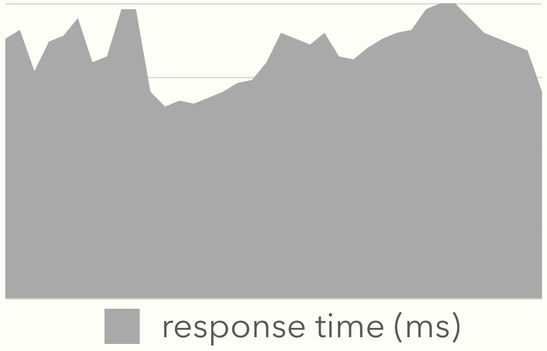
Deferred

FUNC #2

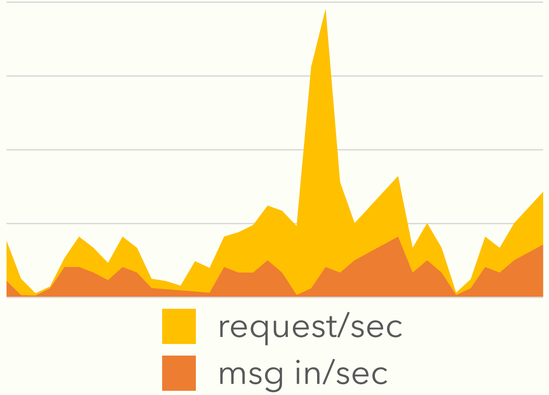
Timeline



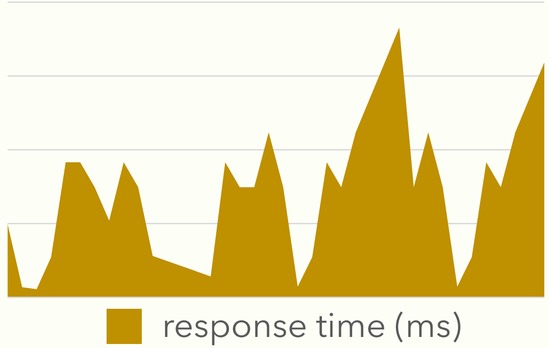
Heap memory usage



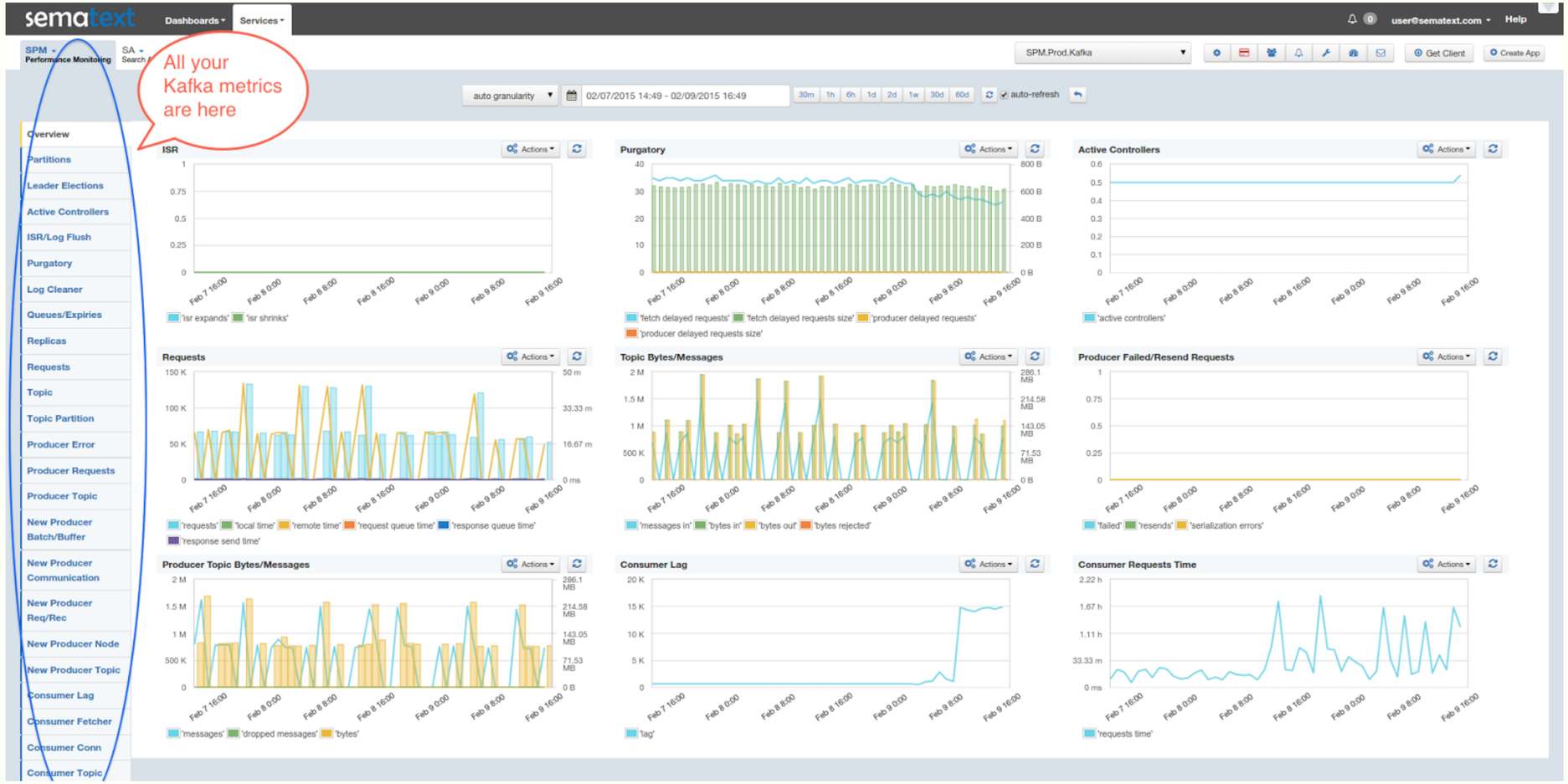
Message Condition



Response time



_ WHAT'S NEW?

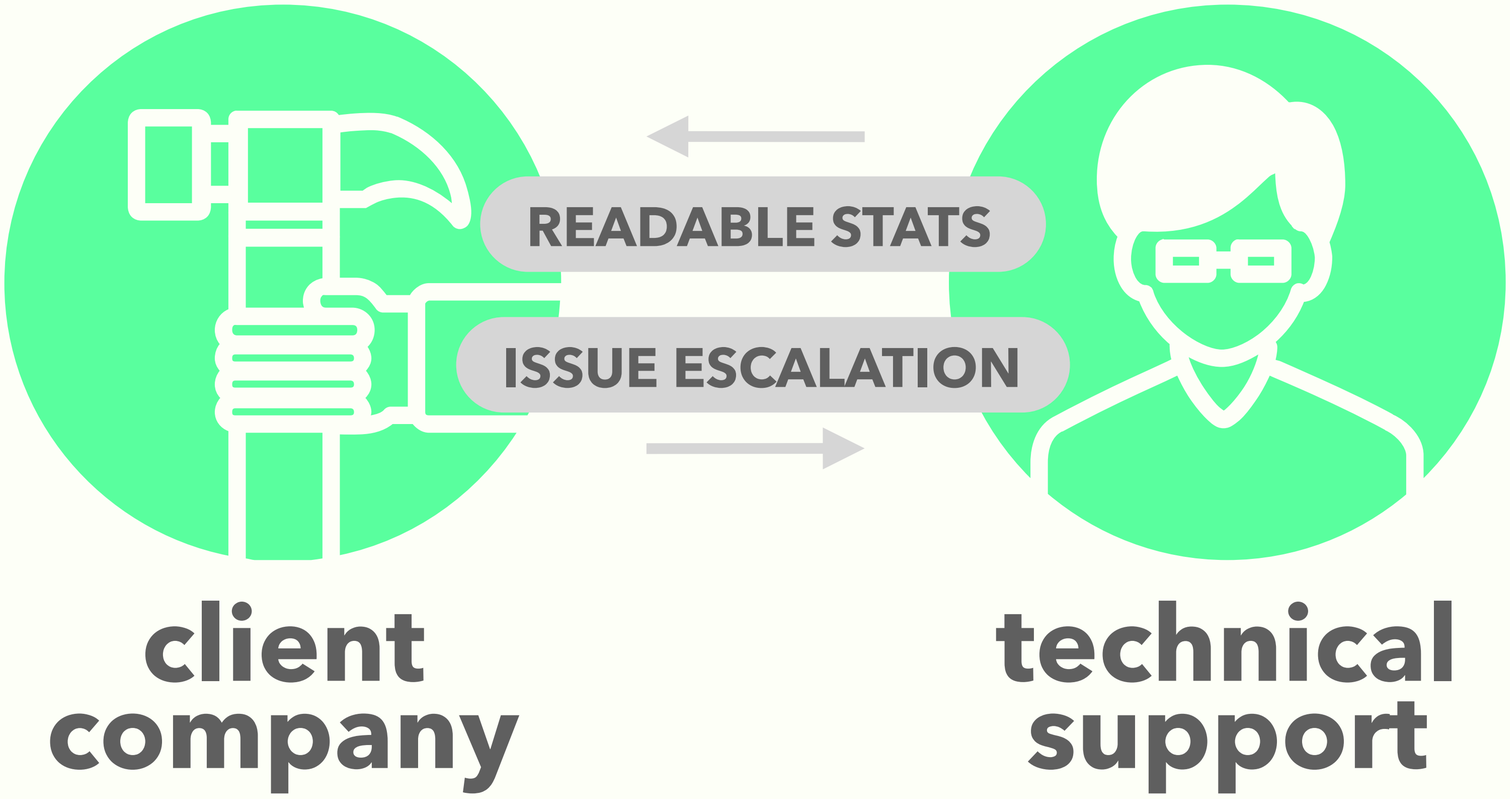


_ WHAT'S NEW?

Clear division of monitoring task

Further implication to BM

_ USER SCENARIO



PROGRESS_UPDATE

SPRINT #4
PROGRESS

_ SPRINT #4

US#1 : As a developer, I can easily plug-in MBean for visualization

- ~~Build General MBean Client Factory (Youngjae Chang)~~
- ~~Find appropriate D3 chart design for charts (Jaryong Lee)~~
- ~~Study websocket structure (Seunghyo Kang)~~
- ~~Design Database schema for saving metric history (Youngjae Chang)~~
- ~~Define API interface for data communication & update (Jaryong Lee)~~
- Design websocket communication structure (Seunghyo Kang)

US#2 : As a user, I can monitor Kafka Ecosystem

- Plug-In Kafka MBeans into Interfaces (Youngjae Chang)
- Place charts to fit designated Kafka monitoring module (Jaryong Lee)

Progress

Architecture

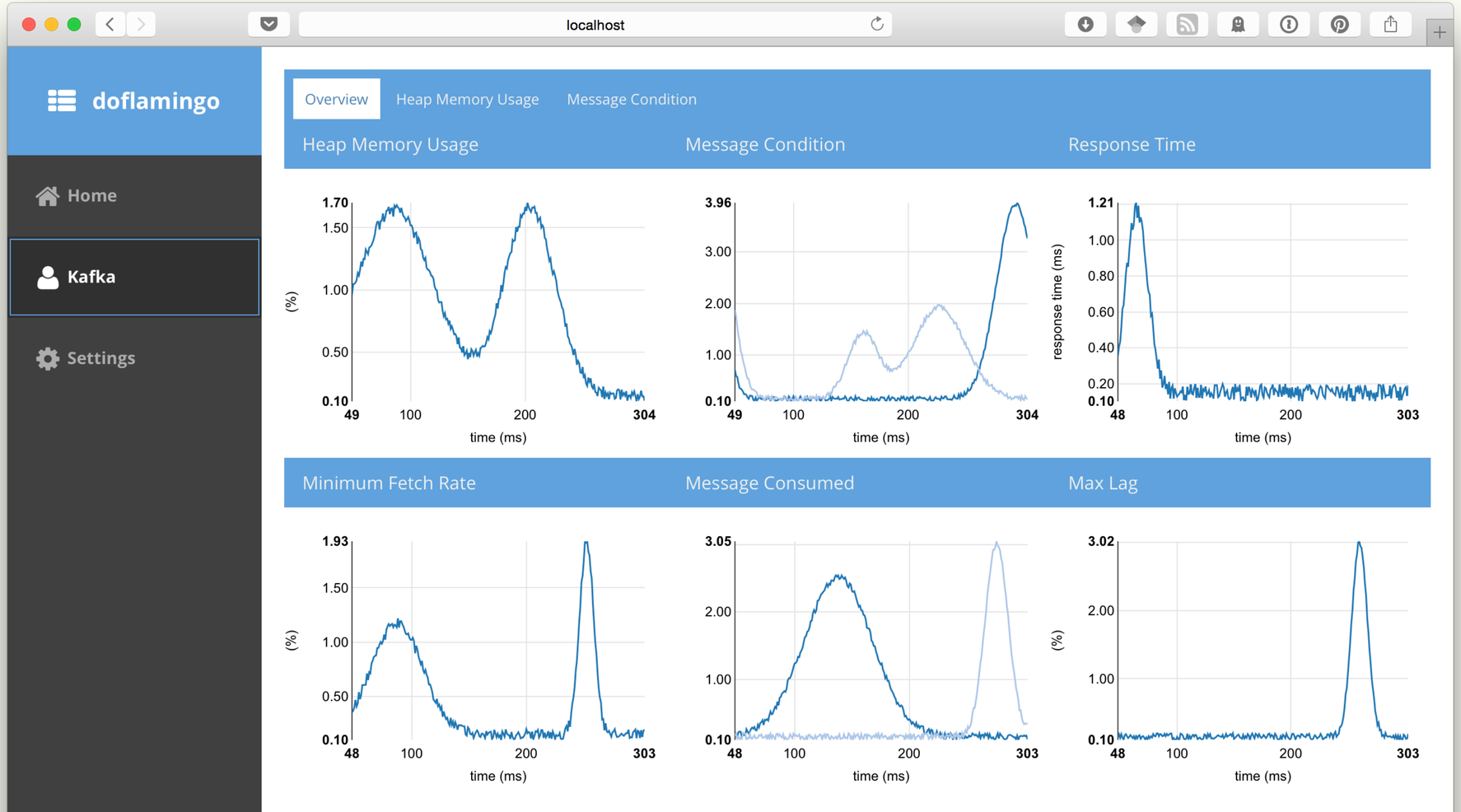
_ NEW ARCHITECTURE

Function	Flamingo	Our Stack
Collect	QuartzJob	JMXTrans
Store	MYSQL	Graphite (RRD Database)
Update	Ajax Query	Websocket
Draw	Sencha	D3.js

Progress

Architecture

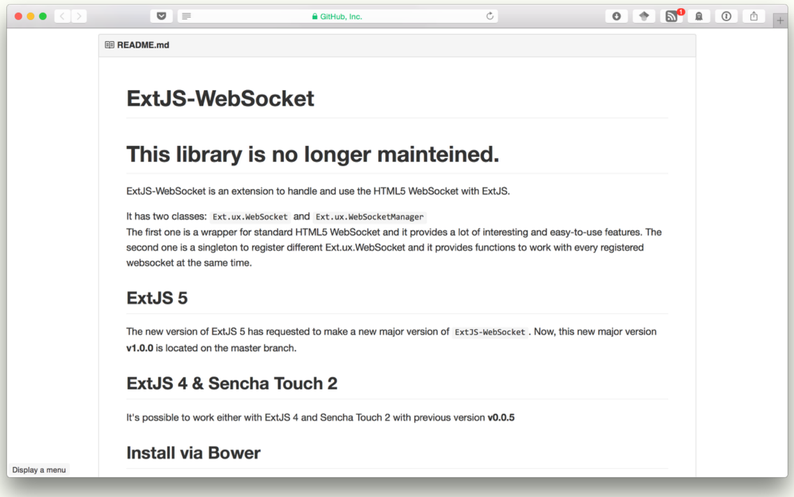
_ LOOK OF PROGRAM



Progress
Architecture
Obstacles

_ WEB SOCKET

Implanting client function to Sencha



Where should we put web socket function?

D3 charts are dynamically constructed.

How can we pass D3 instance as
callback function's arguments?

Sending only deltas v.s. whole data stream

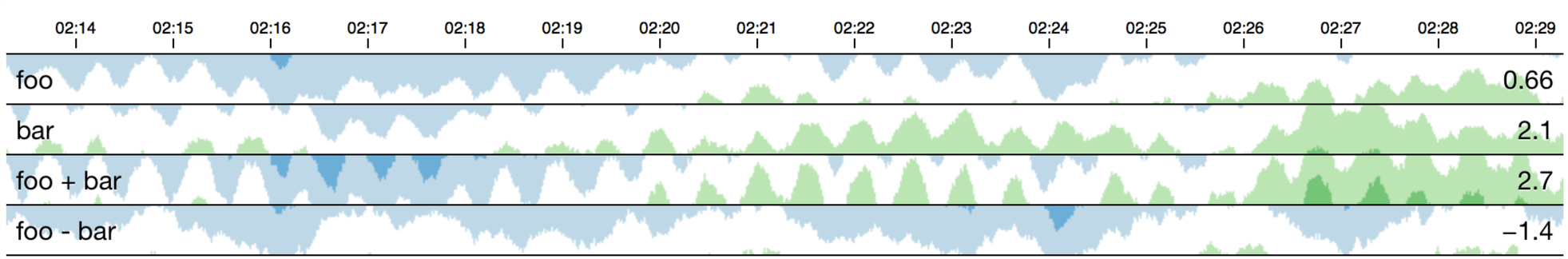
_ VISUALIZATION

Bubble timeline chart is too slow!

The cost of calculating changing shape is too high.

Surveying alternative options:

Cubism.js Square



_ VISUALIZATION

Timescale synchronization among timeline and multiple charts

Charts are drawn using nv-d3, we do not have transparent control over charts interactions

We have to implement custom interaction functions

_ METRIC MONITORING

Topic-specific metrics vs. Node-specific metrics

Kafka can be view as two features: topics and nodes

JMX is basically Node-specific though Kafka 0.9 also provides topic-specific lag information

Researching more on the topic-lag and its visualization

_ METRIC MONITORING

Specific Testing Scenario required

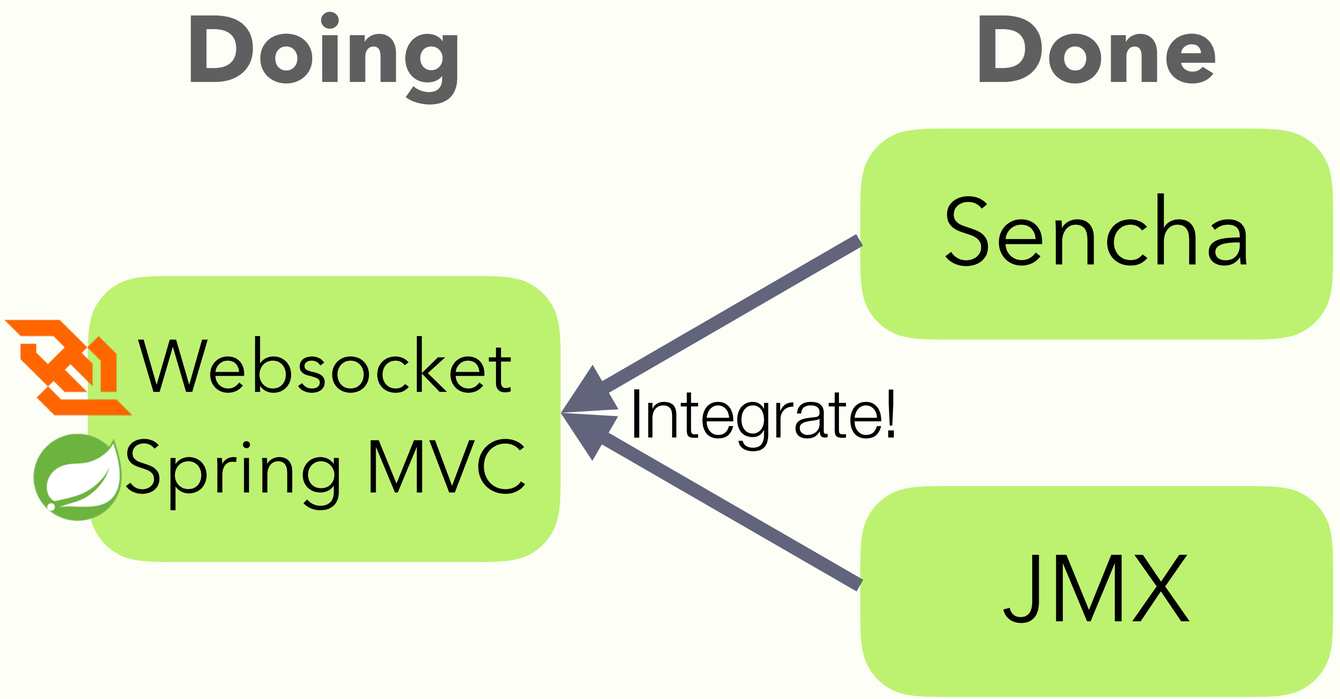
[1] Better design of metric visualization

[2] To demonstrate issue-finding functionality

REQUEST TO EXEM

Specified request will be sent via email

_ FUTURE PLAN



END