Building Your Own BaaS With Apache Usergrid & Docker : Lessons Learned At Scale

Sungju Jin
sungju@apache.org

Speaker

Sungju Jin

- Apache Usergrid PPMC & Committer
- OSCON 2013 Speaker
- Previously
  - Korea Telecom
  - Korea Telecom Hitel
  - Samsung Electronics

Goal

- BaaS Architecture Overview
  - Requirements
  - Production
- Introduction to Apache Usergrid

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1. BaaS(Backend as a Service)

Cloud

- IaaS (Infrastructure As A Service)
- PaaS (Platform As A Service)
- MBaaS(Mobile Backend As A Service)

1. BaaS(Backend as a Service)

- Q : Why do you use BaaS in your system?
- A : Not to Repeat!

User	Group
Push	File
Auth	Data
Security	Social

User	Group
Push	File
Auth	Data
Security	Social

User	Group
Push	File
Auth	Data
Security	Social

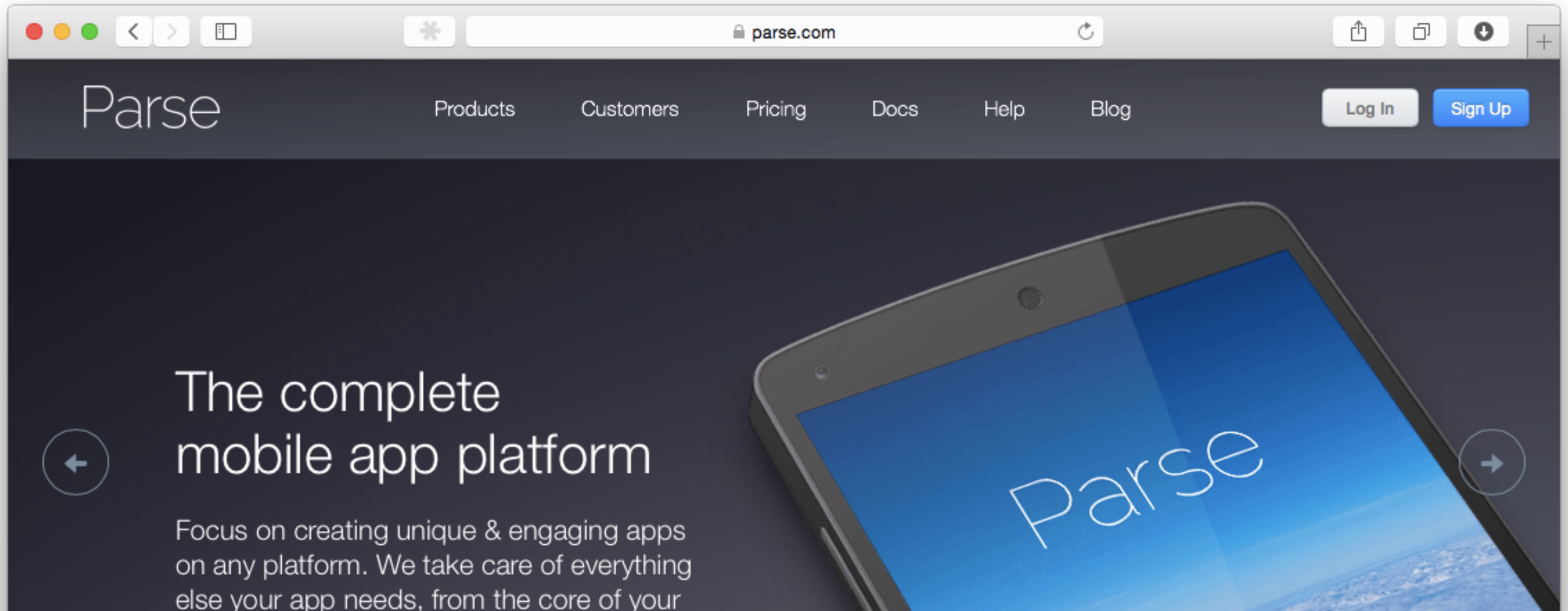
User	Group
Push	File
Auth	Data
Security	Social

1. BaaS(Backend as a Service)

- Q : What is difference between BaaS and server framework such as Spring framework?
- A : These two serve the same purpose, but with BaaS you can focus only on frontend.

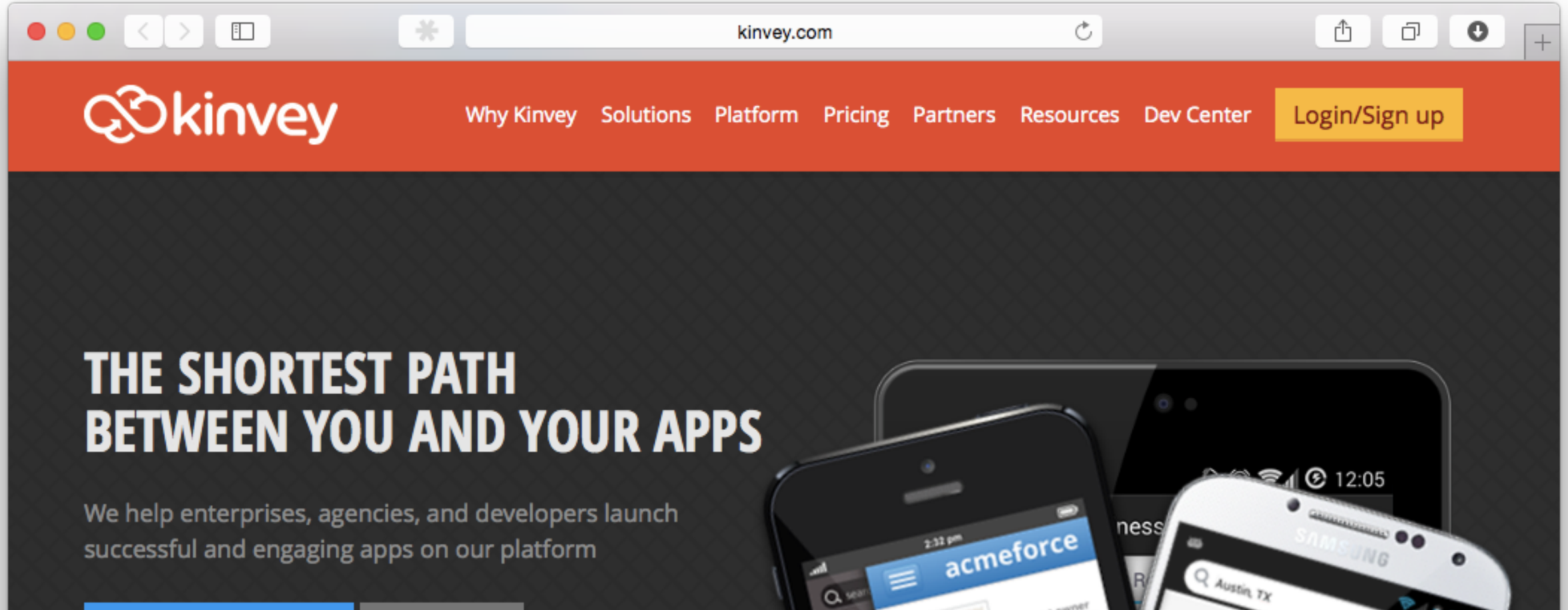
1. BaaS(Backend as a Service)

- Parse
  - Acquired by Facebook on April 25, 2013
  - Founded : June, 2011
  - Features : User, Push, Data, Cloud code, Hosting, Social



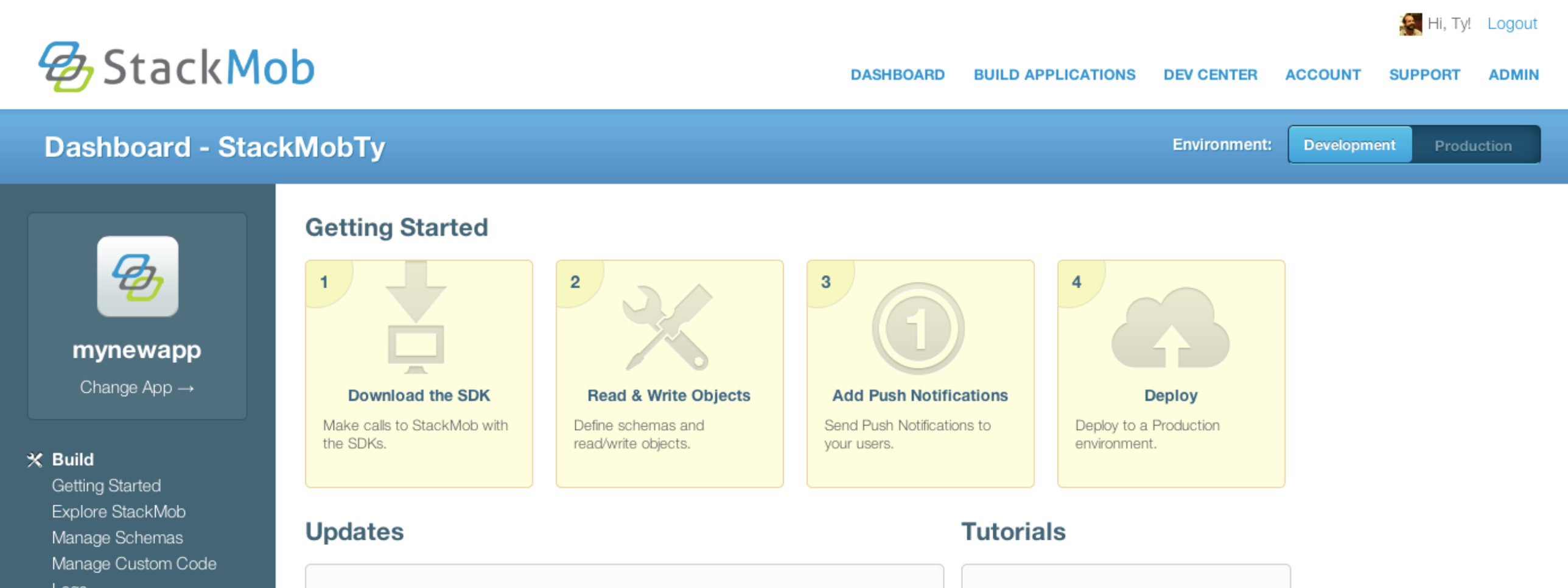
1. BaaS(Backend as a Service)

- Kinvey
  - $17.8 Million in 4 Rounds from 6 Investors
  - Founded : September, 2010
  - Features : User, Push, Data, Custom Apis



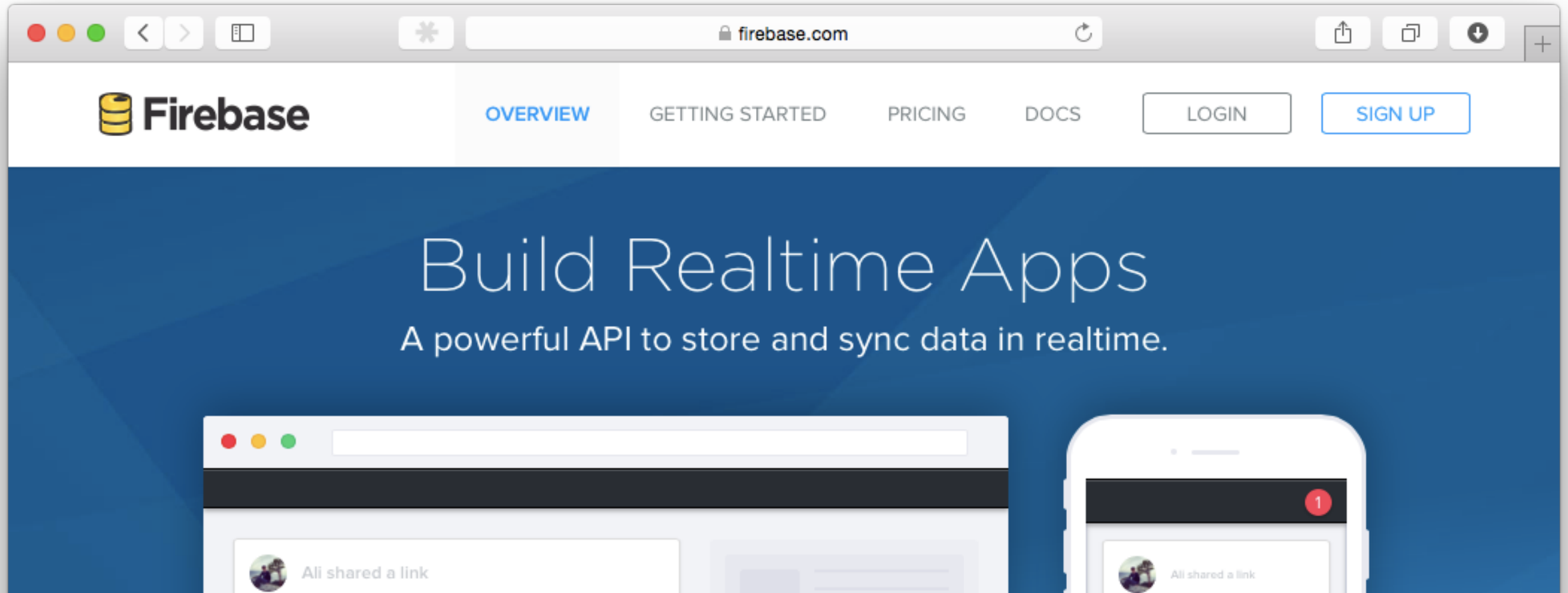
1. BaaS(Backend as a Service)

- Stackmob
  - Acquired by PayPal on December 17, 2013
  - Founded : January, 2010
  - Shutdown : May 11, 2014



1. BaaS(Backend as a Service)

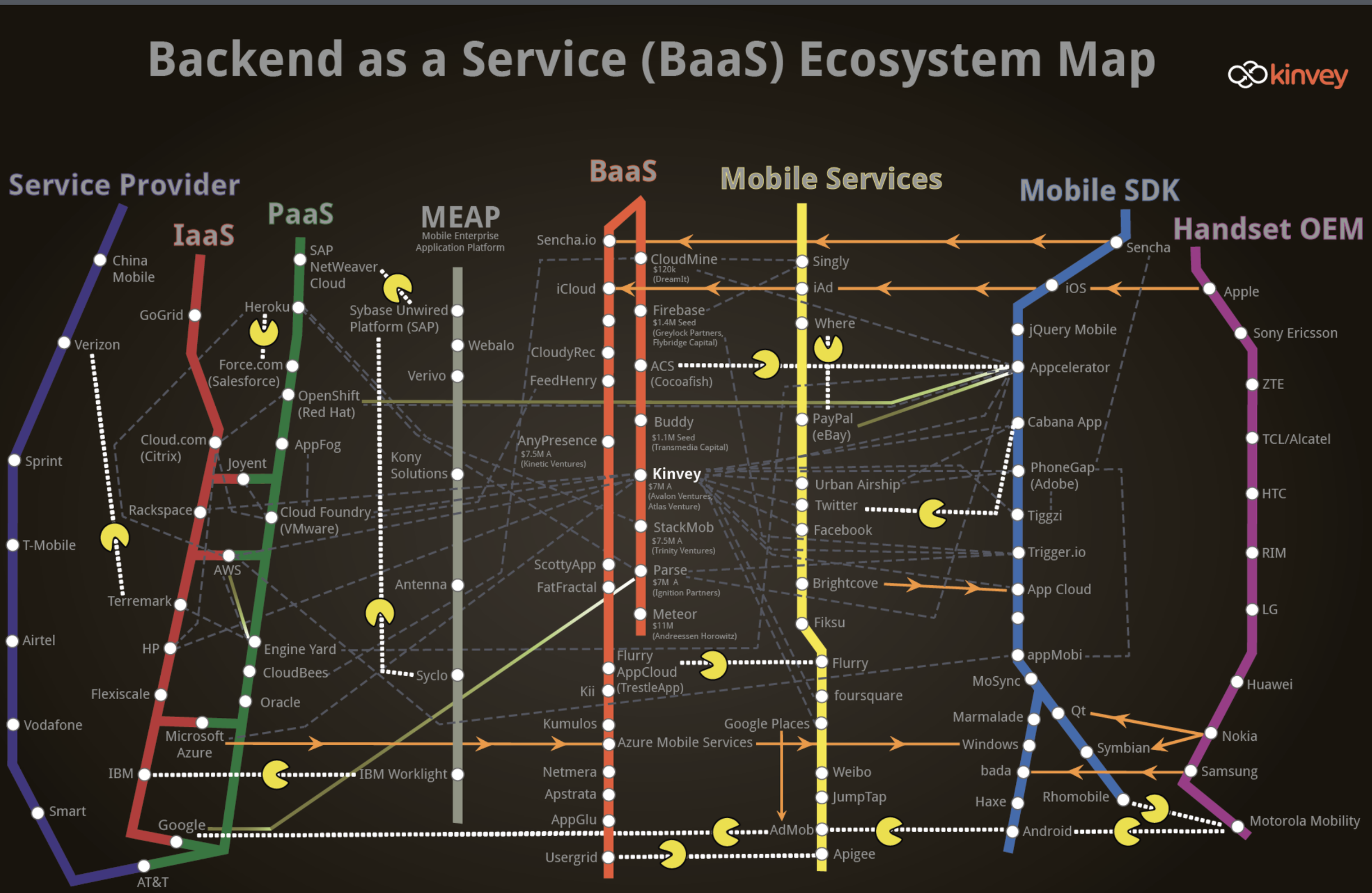
- Firebase
  - Acquired by Google on October 21, 2014
  - Founded : September, 2011
  - Features : Realtime Apps



1. BaaS(Backend as a Service)

- Since 2011, 2012 ~
- Private or Public
- Harmony of Full-stack Cloud Systems (IaaS, PaaS, BaaS)
  - Parse - Facebook
  - Paypal - Stackmob
  - Google - Firebase

1. BaaS(Backend as a Service)



http://www.kinvey.com/blog/65/mobile-cloud-backend-as-a-service-ecosystem-map-8211-all-roads-lead-to-baas

1. BaaS(Backend as a Service)

- **Why have own your BaaS?**
  - Lock-in
  - Always changes requirements
  - Data security
  - Integrate your legacy systems

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

2. Requirements for Building Your BaaS

- **Define your requirements!**
- Providers
  - Public or Private
  - ROI (Return On Investment)
  - Architecture ; Scalability, Availability, Consistently, Extensionality
- Users
  - Visualization (Dashboard, Data, Logs)
  - SLA
  - Lock-in
  - Features

2. Requirements for Building Your BaaS

Features

- Basic feature
- Data
- Users
- Security
- Devices
- Social
- File
- Email
- Notification
- Custom APIs
- Statistics
- Internal Systems

2. Requirements for Building Your BaaS

- **Basic feature**
  - API Versioning
  - Error code
  - API Burst limit
  - Blocking abused behaviors(Account / App / User / Group)
  - Timeout
  - Events (CRUD; User / Group / File / All of Collection)
  - Logging
  - Backups
  - Batch jobs

2. Requirements for Building Your BaaS

- **Data**
  - Data-structures; Key-Value / Tree / Graph
  - CRUD(Create/Read/Update/Delete)
  - Search
    - Searchable / Full-text search
    - Support query
    - Maximum size when parsing
- Support to Aggregation (COUNT, SUM, AVG, MAX, MIN)
- Sort; property number
- Schema / Schema-less
  - Primary, Unique
- Connector; MySQL / MariaDB / PostGres / Oracle
- Import / Export data

2. Requirements for Building Your BaaS

- **Users**
  - CRUD
  - Workflow
    - Validate email
    - Change password
  - Authentication
    - Token Policy (expire time, force to expire)
    - Two Factor Authentication
    - OAuth, LDAP, Facebook, Twitter, Github, KakaoTalk
- **Security**
  - Authorization
  - Scope; resources, users, groups
  - Action; permission (CRUD)

2. Requirements for Building Your BaaS

- **Devices**
  - CRUD
  - Mapping to user (1:1, 1:N, N:N)
  - Statistics
    - Connection frequency / User-agent / Countries
- **Social**
  - Feed
  - Follow / Unfollow / Follower
  - Event / Notification
- **File**
  - CRUD
  - Blob size
- **Email**
  - Template

2. Requirements for Building Your BaaS

- **Notification**
  - Web / Email / Mobile notification
  - Send; users, group, devices
  - Manage; gcm key, apns cert
  - Statistics; receiving rate, response rate
- **Custom APIs**
  - Business Logic
  - API Orchestration Layer
  - Transactions

2. Requirements for Building Your BaaS

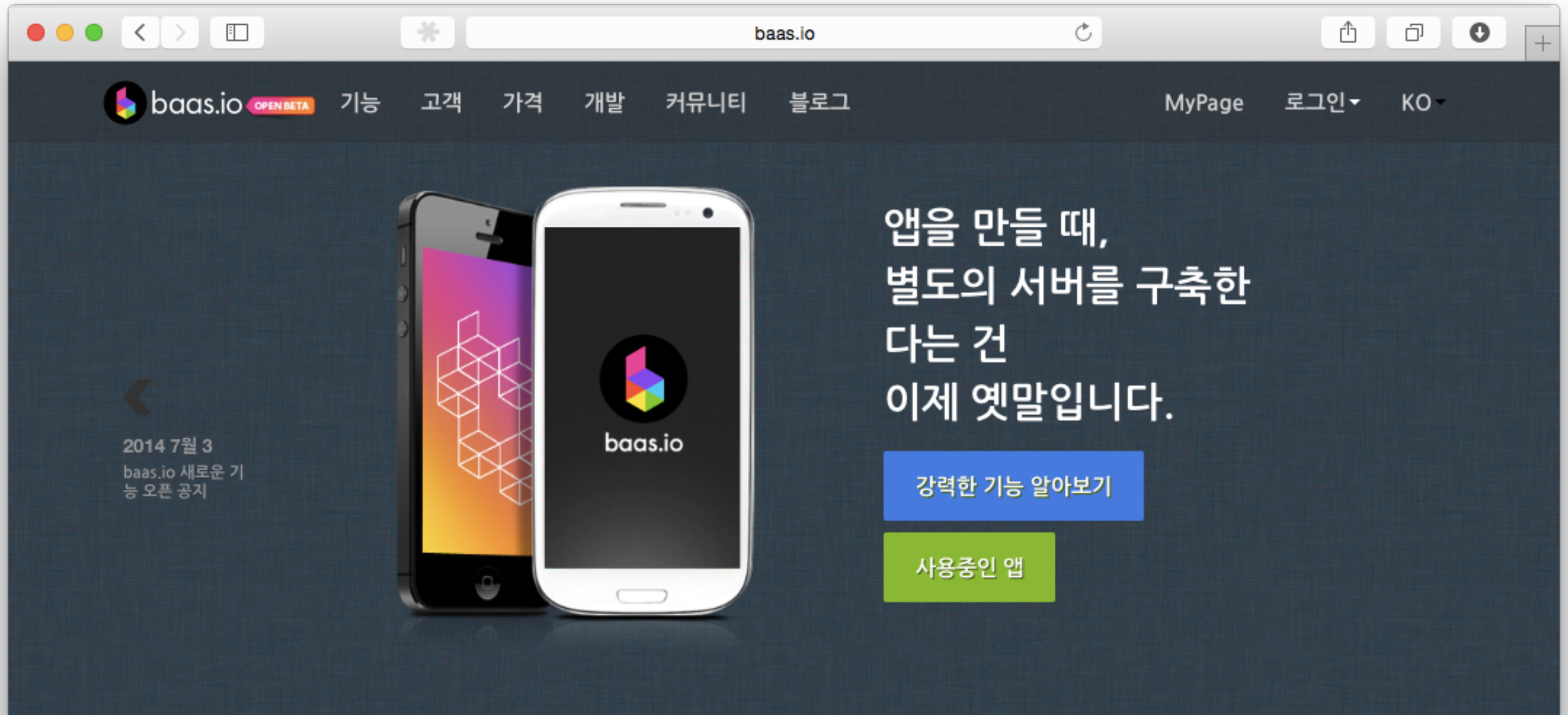
- **Statistics**
  - API Usages / Traffic / Push / File
- **Internal Systems**
  - Console Site
  - Marketing; Mailing
  - Monitoring; API, Systems
  - Deployment

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

3. BaaS in production (baas.io)

- Launched Nov 1, 2012
- Acquired by Korea Telecom on April 8, 2013
- Using Apache Usergrid(forked)



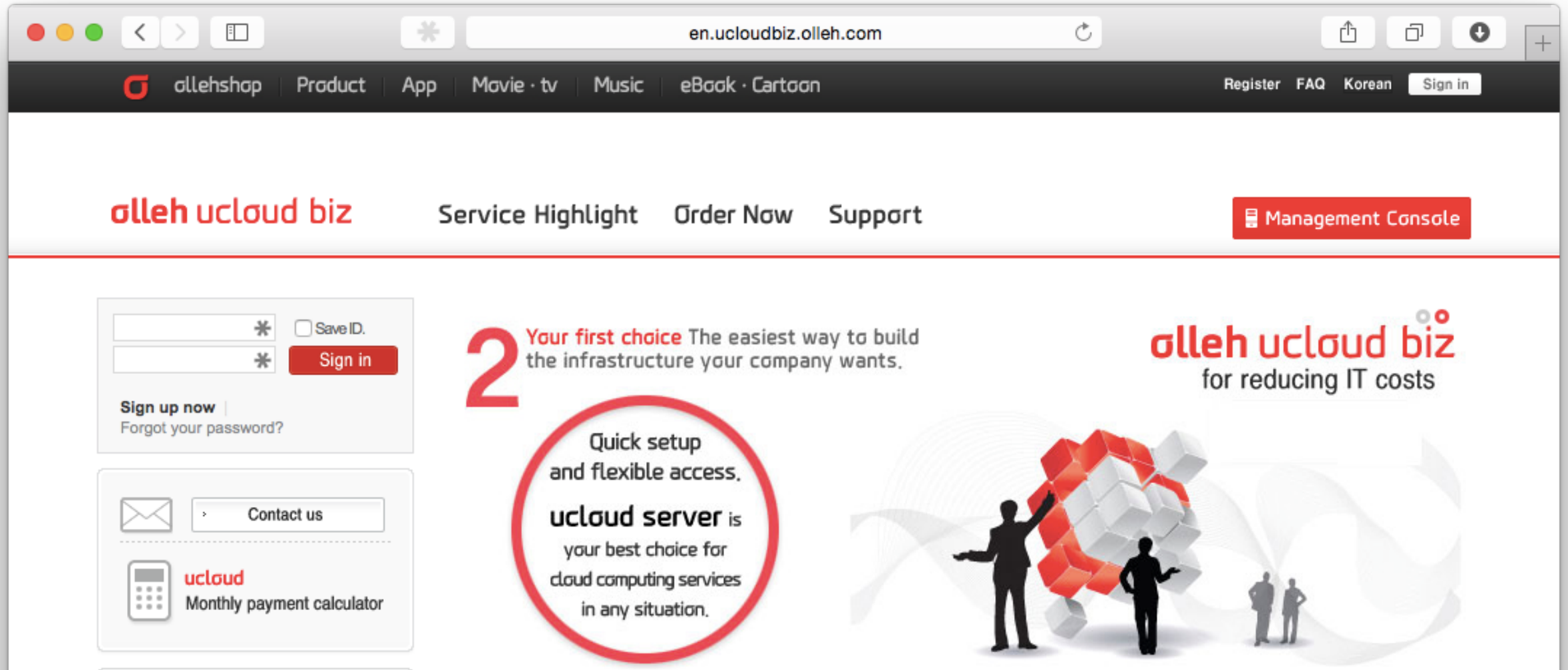
3. BaaS in production (baas.io)

- Features
  - Data
  - Social
  - User / Group / Devices
  - Push notification
  - File
  - Help Center
  - Custom APIs

3. BaaS in production (baas.io)

- Korea Telecom have own Private and Public IaaS(Infrastructure-as-a-Service) called UCloud Biz for Asian companies and developers.

- Korea Telecom think BaaS lead to increase their cloud business and usability.

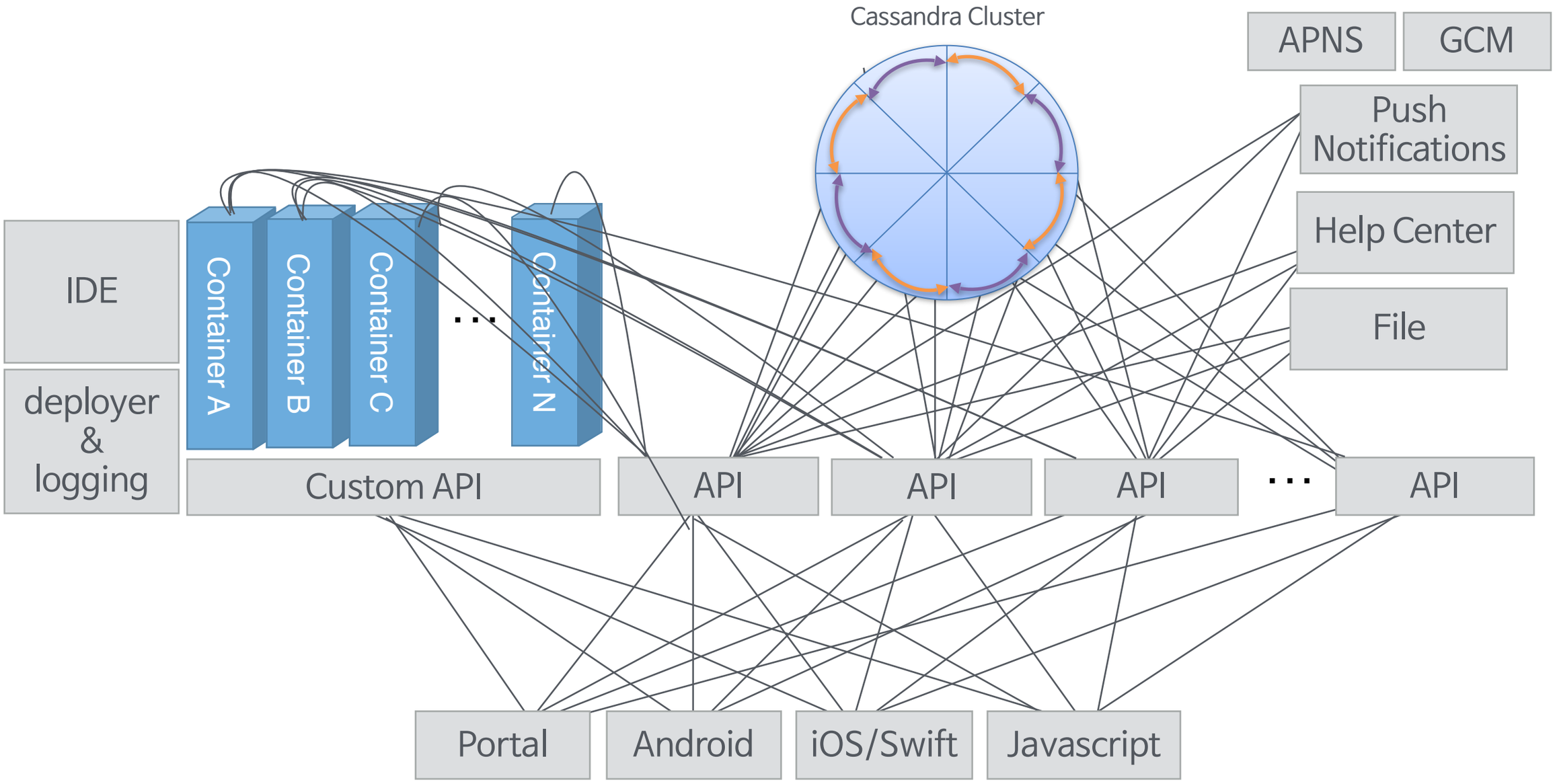


3. BaaS in production (baas.io)

- It serves hundreds of millions of API requests every month.
- It sends tens of millions of push notifications every month.
- It steady increase, growing at 10-20% each month.
- Billions of properties on cassandra clusters
- Over hundred of servers

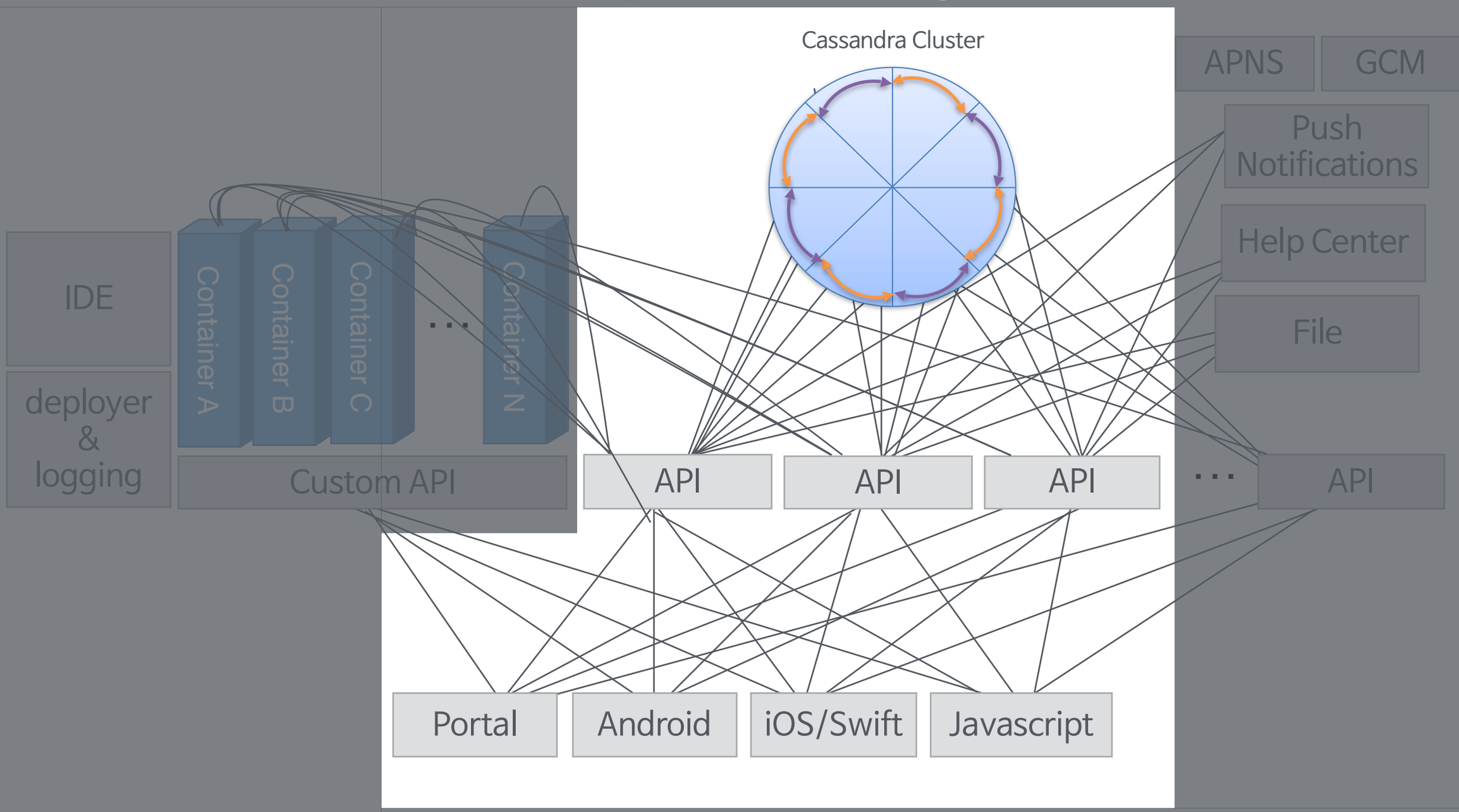
3. BaaS in production (baas.io)

- Architecture



3. BaaS in production (baas.io)

- Architecture



3. BaaS in production (baas.io)

- Thanks for Apache Usergrid team's efforts!
- Need to customize for our requirements
  - Integrate to legacy systems such as Push notification, Helpcenter, File
  - Extending APIs
  - Custom APIs (=Cloud code)
  - Improve performance
  - Isolation for data service even if perfectly
  - User Schema on Data service
  - Centralized logging
  - Billing & Analytics & Statistics
  - Monitoring
  - SDKs and error codes

3. BaaS in production (baas.io)

What to consider as provider?

- Who is your end user? private or public BaaS?
- How to support customized api?
- Schemaless vs Schema
- Transaction vs Performance
- How to monitor systems?
- How to make money?

3. BaaS in production (baas.io)

- **How to support customized api?**
  - We made custom API platform to support business logic using Usergrid & Docker & Jgit
- **Schemaless vs Schema**
  - We changed our system to support schema from schemaless system.
  - Many developers don't get out of SQL paradigm and not familiar with noSQL paradigm.
    They want to have their own schema for search.

3. BaaS in production (baas.io)

- **Transaction vs Performance**
  - We choose a performance.
  - Many developers don’t understand why the platform needs scale and why platform does not support transaction.
    - It make sense. but it is hard to implement requirements. It’ all about trade off between A and B.
- **How to monitor systems?**
  - Measure everything
  - Log all of data (API, File, Push, Network transactions)
  - Zabbix for system monitoring

3. BaaS in production (baas.io)

- **How to make money?**
  - If free tier has lots of features, they don't want to subscribe. However, if the free tier is not good, they don't use the platform.
  - Of course, there are lots of pricing model such as dedicated server.

3. BaaS in production (baas.io)

There are clues from your users or clients.

Take your time if you want to build own your BaaS

3. BaaS in production (baas.io)

Back to the technology!

- Reinventing the wheel!
- Open source is not silver bullet!
- Save your time.
  Apache Usergrid helps you!

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

Agenda

1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid
2) Basic Concepts
3) Restful APIs
4) Architecture
5) Data Processing
6) CRUD

4. Apache Usergrid

- Designed for multi-tenancy & operational predictability.
  Built on Java 7, Jersey & Apache Cassandra, with SDKs for iOS, Android, HTML5/JS, node.js, Ruby, Java, .NET, PHP — and so much more.

- http://usergrid.incubator.apache.org

- Creator, Ed Anuff (http://www.anuff.com)

- Since 2011.10.03 ~

apache usergrid_

The BaaS *not* made for Hipsters

Designed for multi-tenancy & operational predictability. Built on Java 7, Jersey & Apache Cassandra, with SDKs for iOS, Android, HTML5/JS, node.js, Ruby, Java, .NET, PHP – and so much more. Open source since 2011.

Currently undergoing incubation at the Apache Software Foundation

http://usergrid.incubator.apache.org/

Users

Sign up users, log in, reset passwords and more, in just one API call. You can put users in groups, assign roles or permissions, let users follow each other and access everything via OAuth 2.0, without writing a single line of server code.

Data

If you can express it in JSON, we can store it! Underneath everything is stored in a standard Cassandra instance, but we've added the ability to retrieve data via SQL, do graph queries, and even joins.

Files

Our asset storage can handle anything from text files to videos of several terrabytes, with automatic content-detection and full URL access control. In the back, everything goes Amazon S3 or other preferred cloud file store.

SDKs

Java-based

Trusted

4. Apache Usergrid

History

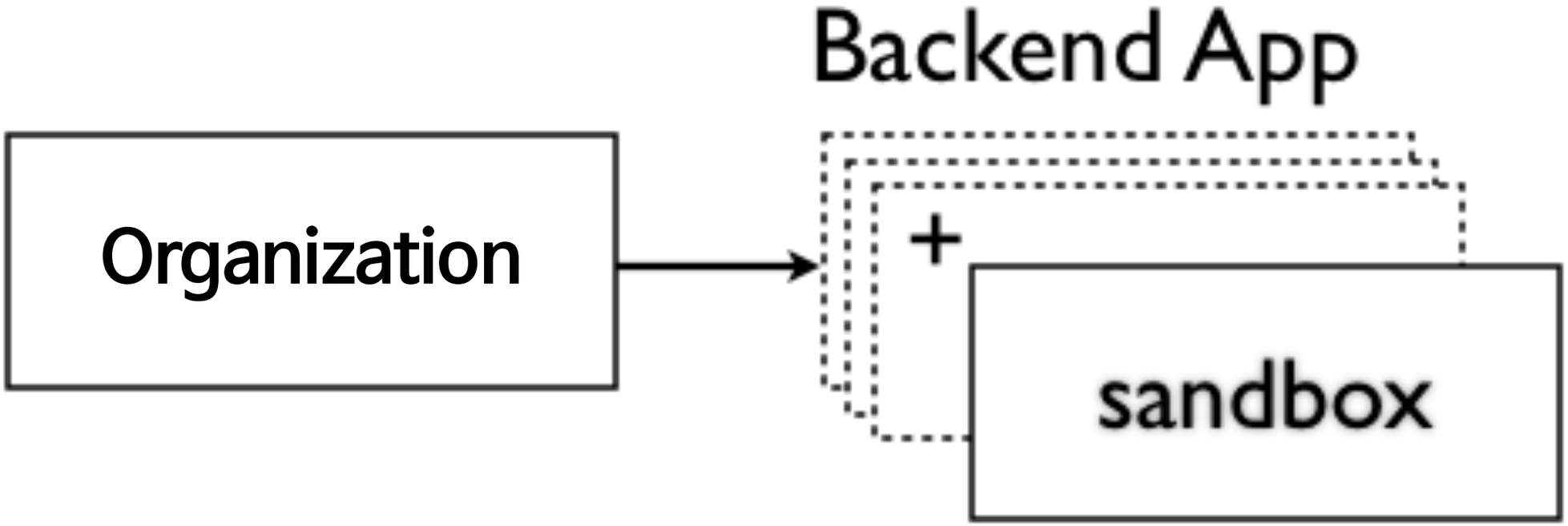
2011.10.03	Open sourced
2012.01.18 ~	Acquired by Apigee
2012.10 ~	Forked development by KTH
2013.10 ~	Joined Apache incubator project

Agenda

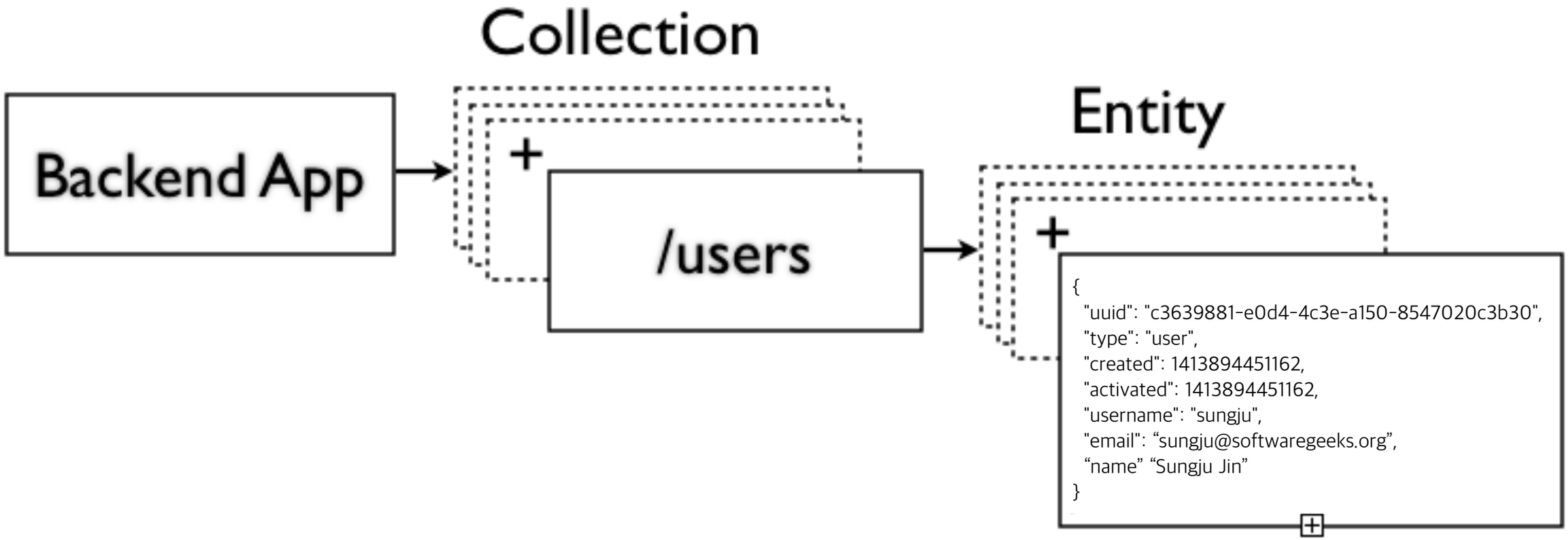
1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid
2) Basic Concepts
3) Restful APIs
4) Architecture
5) Data Processing
6) CRUD

4. Apache Usergrid > Basic Concepts



4. Apache Usergrid > Basic Concepts



4. Apache Usergrid > Basic Concepts

Resource	**POST** create	**GET** read	**PUT** update	**DELETE** delete
/items	create a new item	list items	bulk update items	delete all items
/items/ipad	error	show ipad	update ipad	delete ipad

4. Apache Usergrid > Basic Concepts

curl -X POST -i -H "Authorization: Bearer {auth_key}" -d '{"username":"bob","email":"bob@company.com"}' "https://api.domain.io/my-org-id/my-app-id/users"


```
{
    "action": "post",
    "application": "81c5c8b8-136a-11e2-8ed5-4061867ca222",
    "params": {},
    "path": "/users",
    "uri": "https://api.domain.io/my-org-id/my-app-id/users",
    "entities": [
        {
            "uuid": "37a71adc-136c-11e2-8ed5-4061867ca222",
            "type": "user",
            "created": 1349936628563,
            "modified": 1349936628563,
            "activated": true,
            "email": "bob@company.com",
            "picture": "https://www.gravatar.com/avatar/217195a2032ff3c42cf8711bd6334b0f",
            "username": "bob"
        }
    ],
    "timestamp": 1349936628483,
    "duration": 180,
    "organization": "my-org-id",
```


collection

application

organization

4. Apache Usergrid > Basic Concepts

Relationship

POST	https://api.domain.io/devices	Create device
POST	https://api.domain.io/users	Create user
POST	https://api.domain.io/users/{user}/devices	Create device to user relationship
POST	https://api.domain.io/devices/{device}/users	Create user to device relationship

Agenda

1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid
2) Basic Concepts
3) Restful APIs
4) Architecture
5) Data Processing

4. Apache Usergrid > Restful APIs

Categories of API

Application API	• Token • Predefined Collections • Users, Groups, Roles, Activities, Devices, Events, Folders, Assets • Collections
Management API	• Token • Admin Users • Organizations

4. Apache Usergrid > Restful APIs

Collections (other than users, groups, and roles)

URI	Verb	Content Types	Action
/{org_id}/{app_id}/	GET	application/json	Retrieve all collections
/{org_id}/{app_id}/{collection}	POST	application/json	Create a new entity or collection
/{org_id}/{app_id}/{collection}/{uuid|name}	GET	application/json	Retrieve an entity
/{org_id}/{app_id}/{collection}/{uuid|name}	PUT	application/json	Update an entity
/{org_id}/{app_id}/{collection}/{uuid|name}	DELETE	application/json	Delete an entity
/{org_id}/{app_id}/{collection}?{query}	GET	application/json	Query a collection

4. Apache Usergrid > Restful APIs

Collections (other than users, groups, and roles)

URI	Verb	Content Types	Action
/{org_id}/{app_id}/{collection}/{entity_id}/{relationship}?{query}	GET	application/json	Query an entity's collections or connections
/{org_id}/{app_id}/{collection}/{first_entity_id}/{relationship}/{second_entity_id} or /{org_id}/{app_id}/{collection}/{first_entity_id}/{relationship}/{second_entity_type}/{second_entity_id}	POST	application/json	Add an entity to a collection or create a connection
/{org_id}/{app_id}/{collection}/{first_entity_id}/{relationship}/{second_entity_id} or /{org_id}/{app_id}/{collection}/{first_entity_id}/{relationship}/{second_entity_type}/{second_entity_id}	DELETE	application/json	Remove an entity from a collection or delete a connection

4. Apache Usergrid > Restful APIs

Access Token

URI	Verb	Content Types	Action
/management/token '{"grant_type":"client_credentials","client_id":"{client_id}","client_secret":"client_secret"}'	POST	application/json	Obtain an access token (access type = organization)
/management/token '{"grant_type":"password","username":"{username}":"password"="{password}"}'	POST	application/json	Obtain an access token (access type = admin user)
/management/{org_id}/{app_id}/token '{"grant_type":"client_credentials","client_id":"{client_id}","client_secret":"{client_secret}"}'	POST	application/json	Obtain an access token (access type = application)
/management/{org_id}/{app_id}/token '{"grant_type":"password","username":"{username}","password":"{password}"}'	POST	application/json	Obtain an access token (access type = application user)

4. Apache Usergrid > Restful APIs

Organizations

URI	Verb	Content Types	Action
/management/organizations|orgs	POST	application/json	Create an organization
/management/organizations|orgs/{org_name}|{uuid}	GET	application/json	Retrieve an organization
/management/organizations|orgs/{org_name}|{uuid}/activate?token={token}&confirm={confirm_email}	GET	application/json	Activate an organization
/management/organizations|orgs/{org_name}|{uuid}/reactivate	GET	application/json	Reactivate an organization
/management/organizations|orgs/{org_name}|{uuid}/credentials	POST	application/json	Generate organization client credentials

Agenda

1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid

2) Basic Concepts

3) Restful APIs

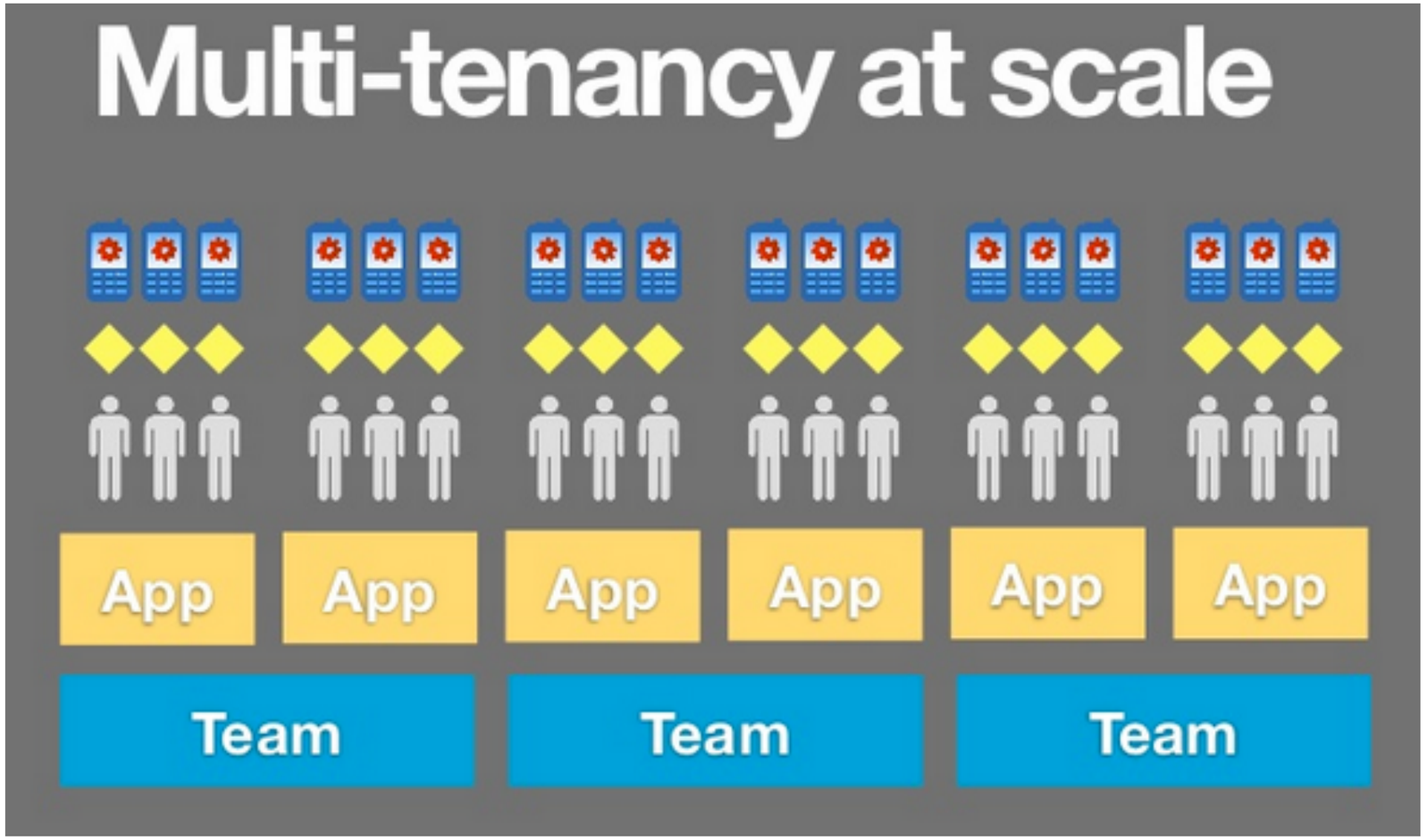
4) Architecture

5) Data Processing

4. Apache Usergrid > Architecture

Design Goal

- Designed for multi-tenancy on Cassandra
- Scalability



4. Apache Usergrid > Architecture

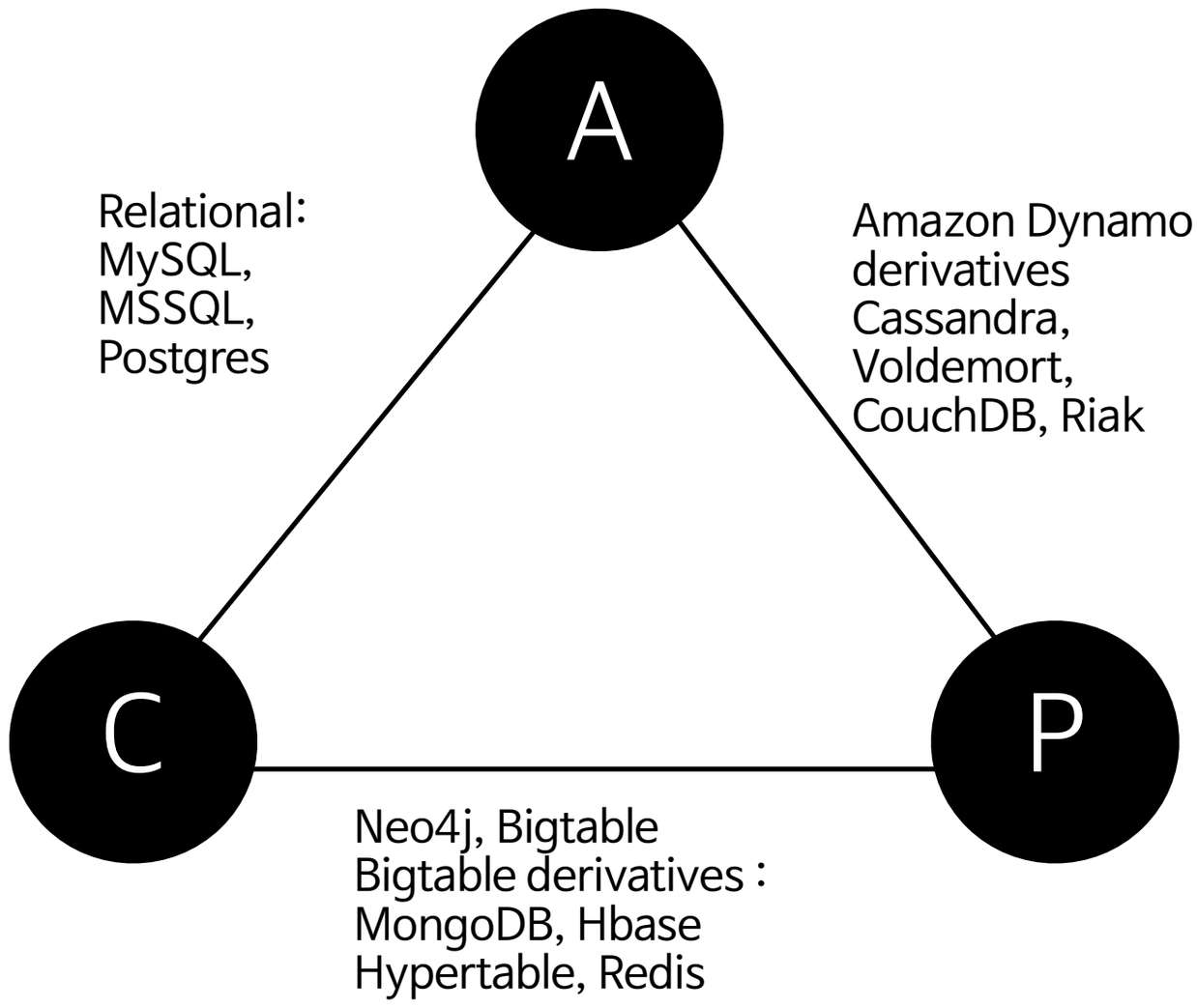
Design Goal > Background

- Brewer's theorem

  Consistency

  Availability

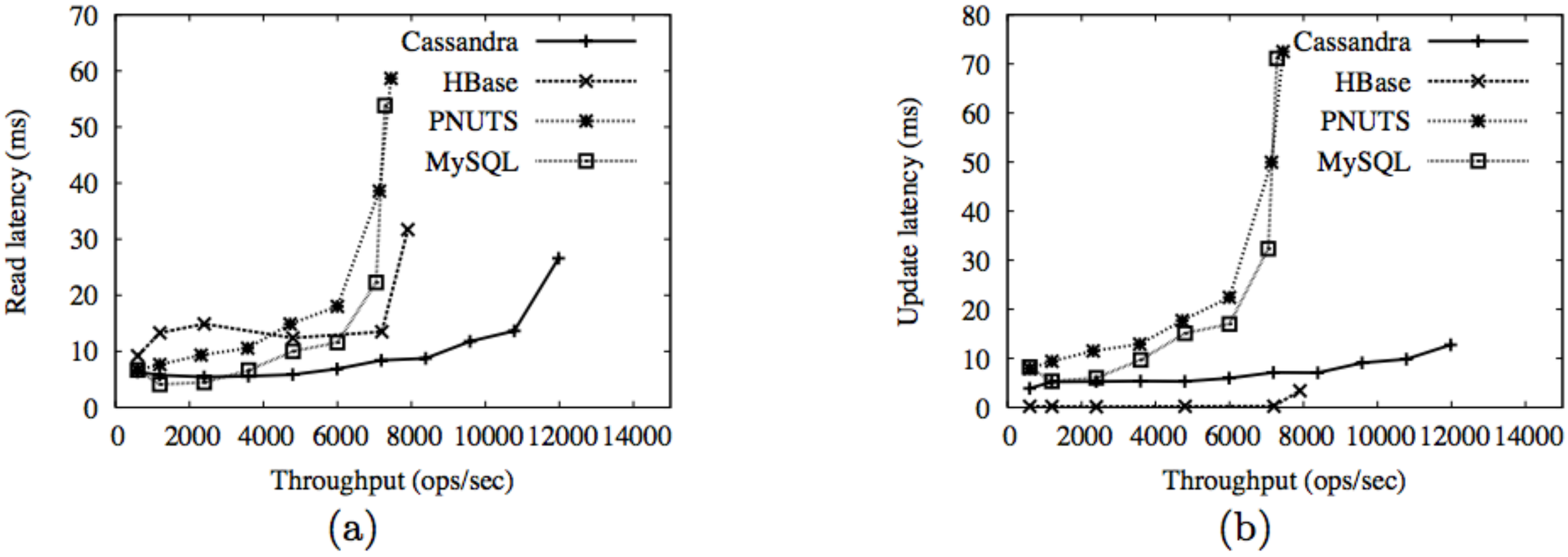
  Partition tolerance



4. Apache Usergrid > Architecture

Design background

- Benchmark

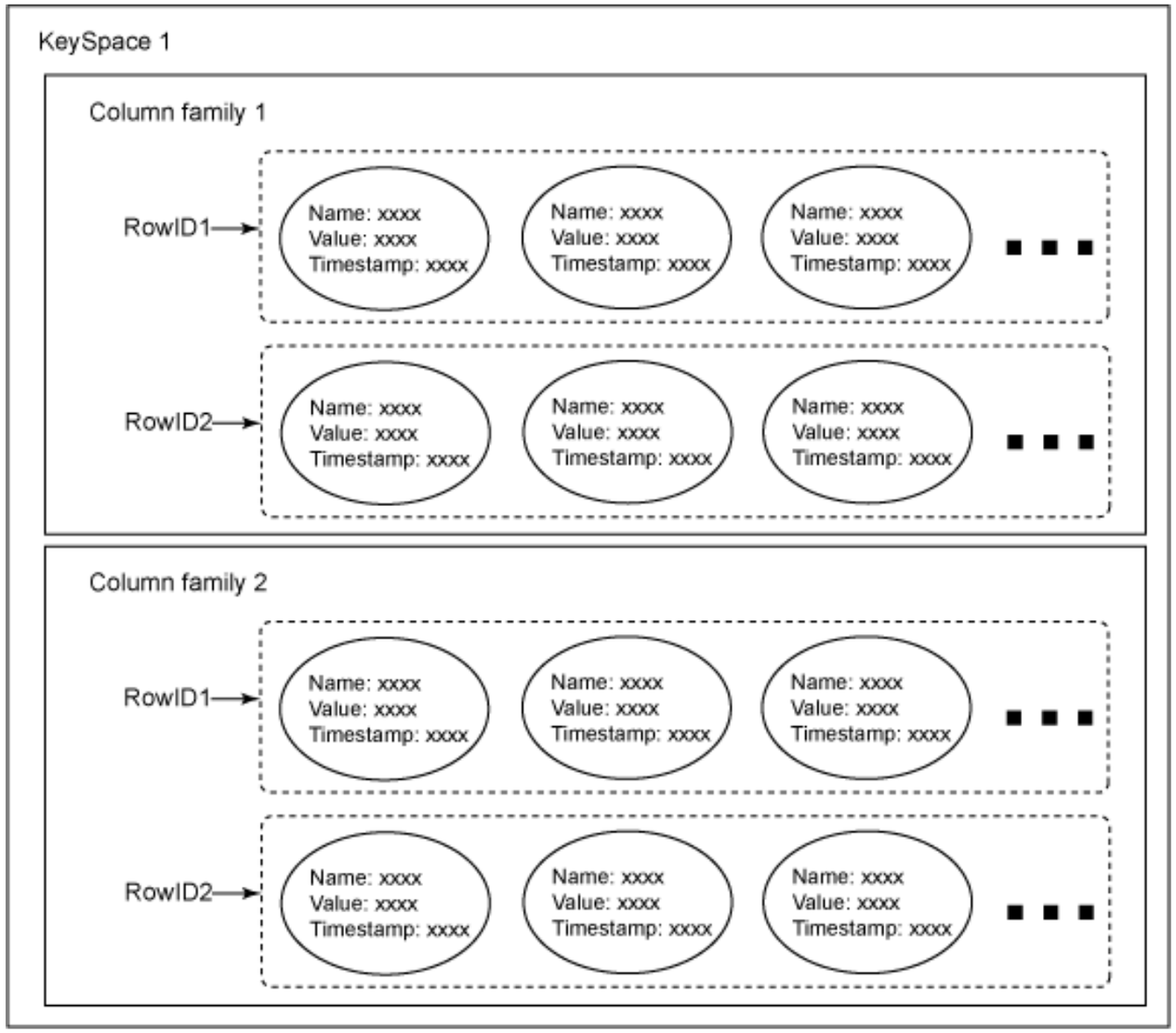


Workload A—update heavy: (a) read operations, (b) update operations. Throughput in this (and all figures) represents total operations per second, including reads and writes.

References : Benchmarking Cloud Serving Systems with YCSB

4. Apache Usergrid > Architecture

Design background



http://www.ibm.com/developerworks/library/os-apache-cassandra/

4. Apache Usergrid > Architecture

Design background

User	Cluster size	Node count	Usage	Now
~~Facebook~~	~~>200~~	~~?~~	~~Inbox search~~	Abandoned, Moved to HBase
Cisco WebEx	?	?	User feed, activity	
Netflix	200 TB	750 (50 cluster)		
Apigee	?	?	Data store	
Formspring	? (26 million account with 10 m responsed per day)	?	Social-graph data	
Urban airship, Rackspace, Open X, Twitter (preparing move to), ebay				

References :
http://planetcassandra.org/Company/ViewCompany
http://en.wikipedia.org/wiki/Apache_Cassandra

4. Apache Usergrid > Architecture

Design background

	Separate Database	Shared Database
Shared Schema	- Scalability + Isolation + Simple	+ Scalability - No Isolation - Complicated
Separate Schema	- Scalability + Isolation + Not Complicated	- Scalability - No Isolation - Complicated

4. Apache Usergrid > Architecture

Design background

	Separate Database	Shared Database
Shared Schema	Expensive	Interesting
Separate Schema	Very Expensive	Unwieldy

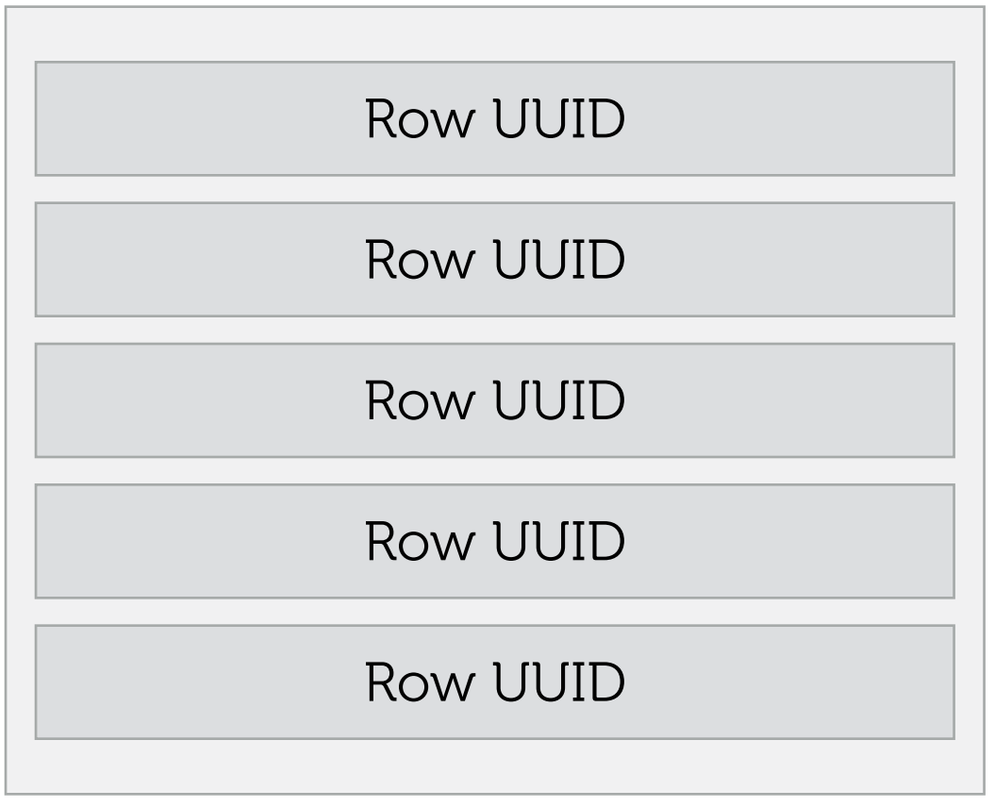
4. Apache Usergrid > Architecture

Design background

	Separate Database	Shared Database
Shared Schema	- Scalability + Isolation + Simple	+ Scalability - No Isolation - Complicated
Separate Schema	- Scalability + Isolation + Not Complicated	- Scalability - No Isolation - Complicated

4. Apache Usergrid > Architecture

Design background



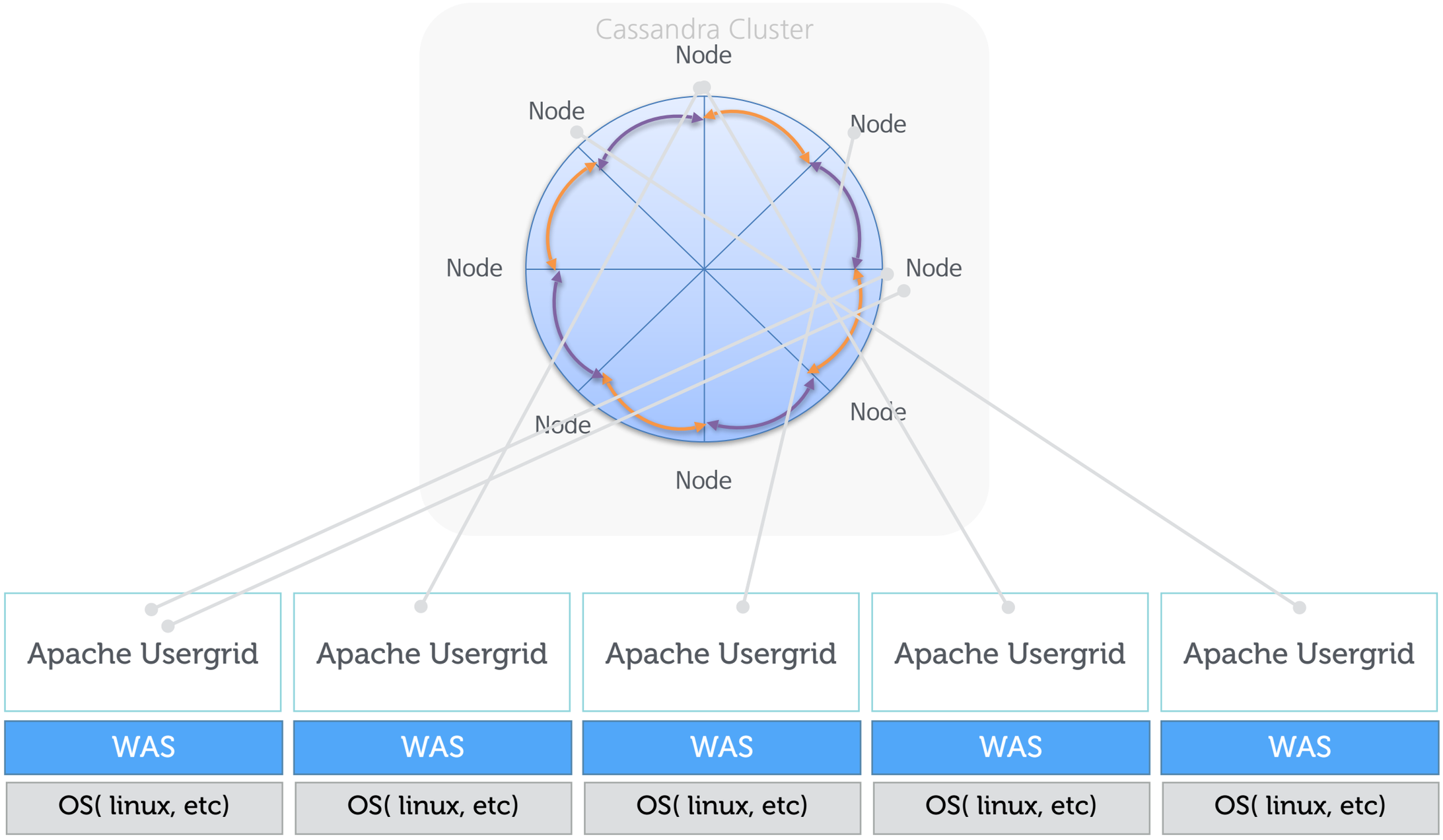
4. Apache Usergrid > Architecture

Design background

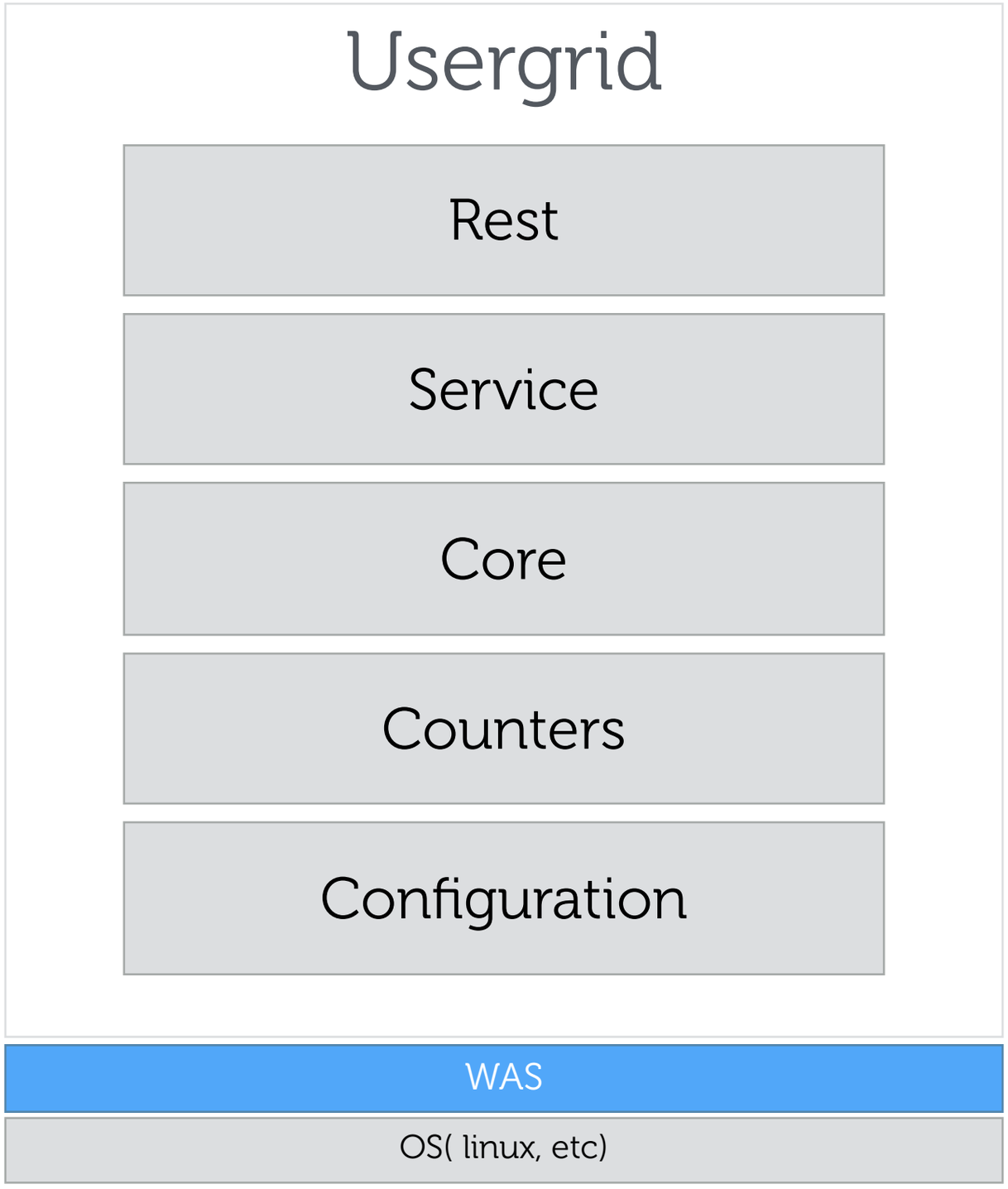
Multi-tenant Row Keys In Shared Keyspace

Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID
Tenant ID	Row UUID

4. Apache Usergrid > Architecture



4. Apache Usergrid > Architecture

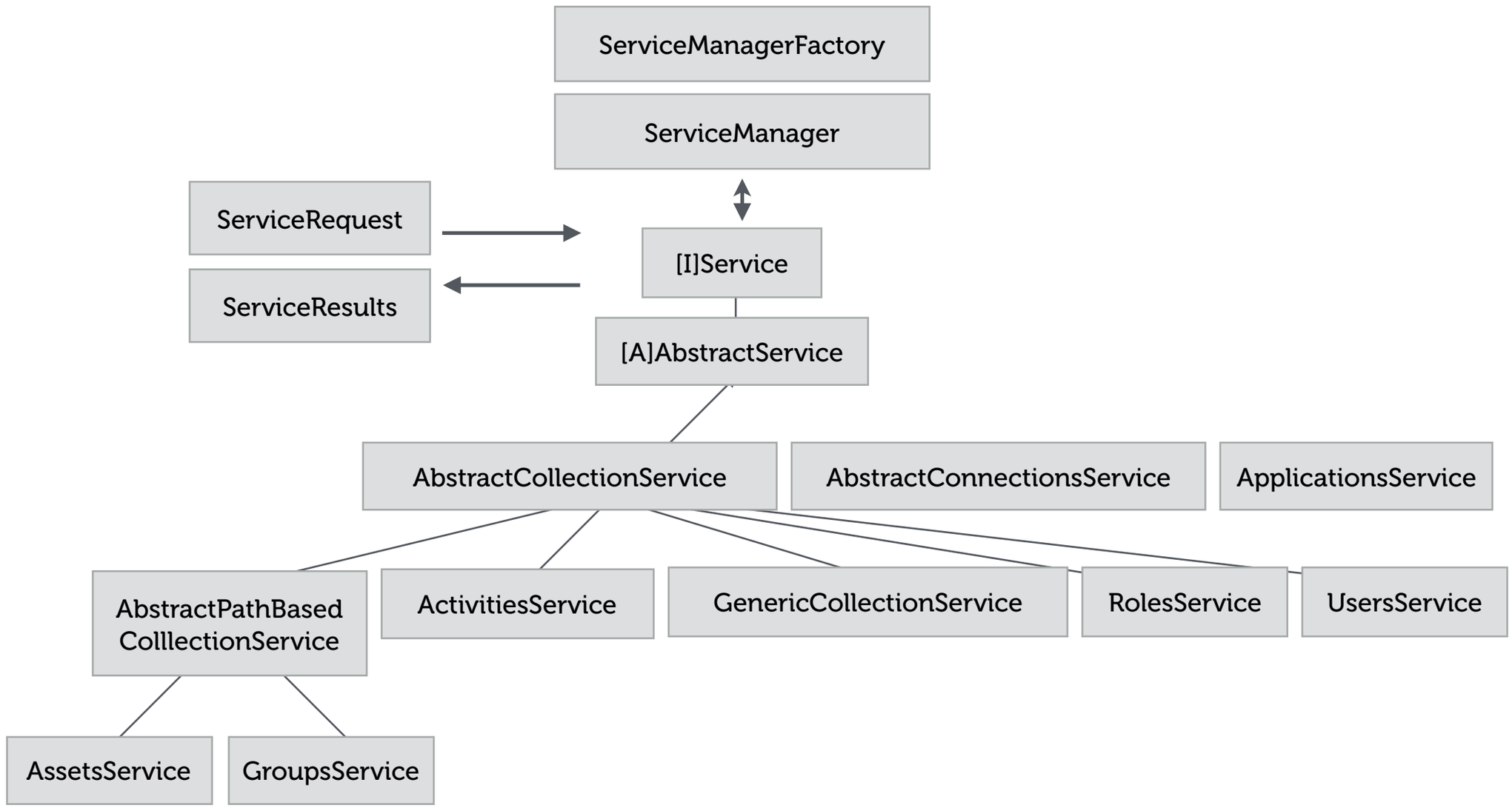


4. Apache Usergrid > Architecture

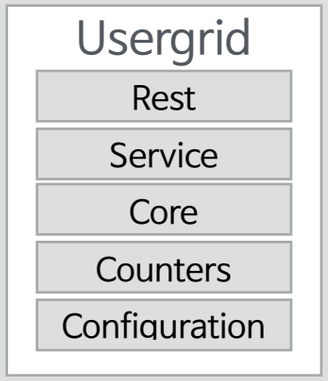
Usergrid

- Rest
- Service
- Core
- Counters
- Configuration

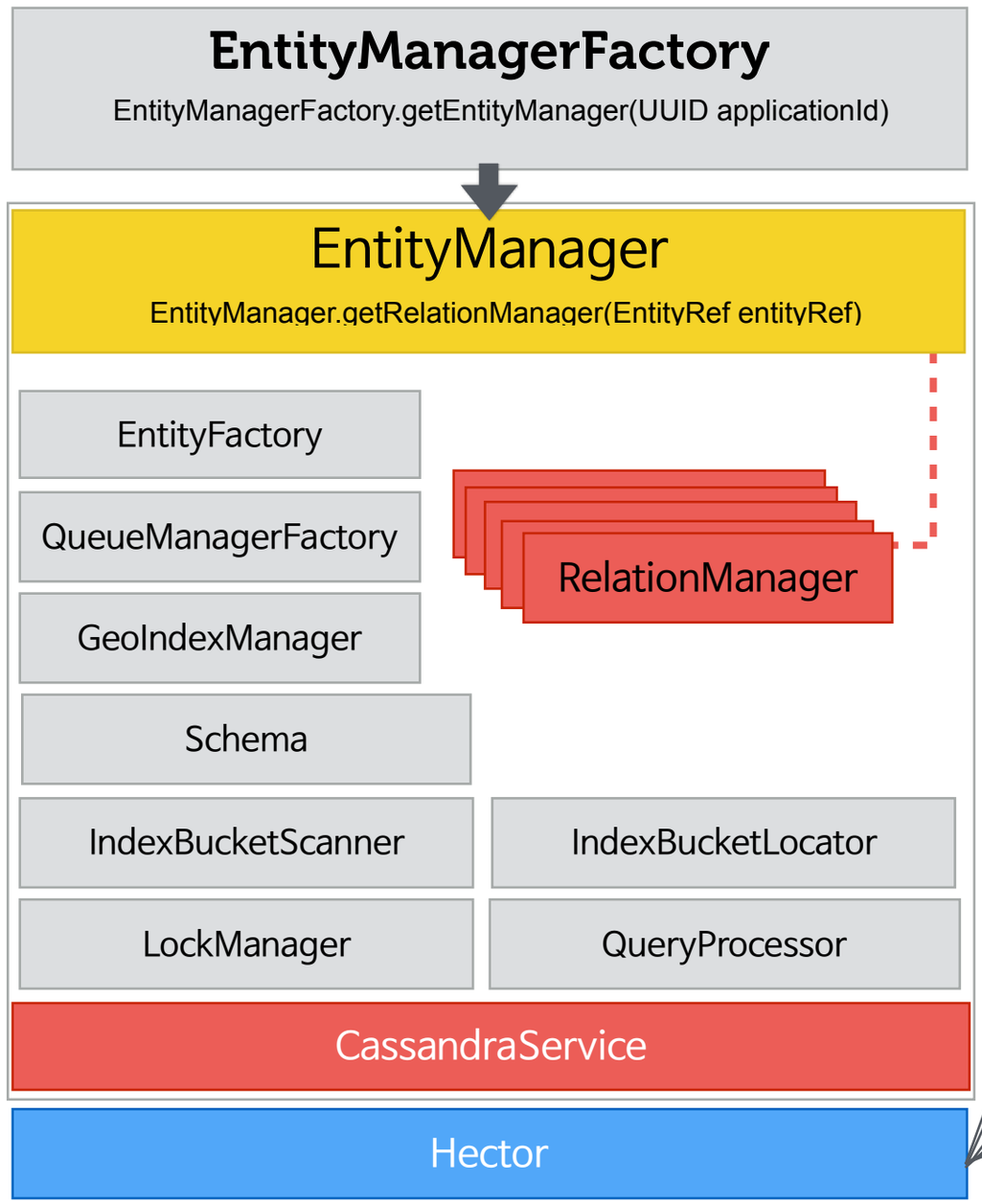
Service

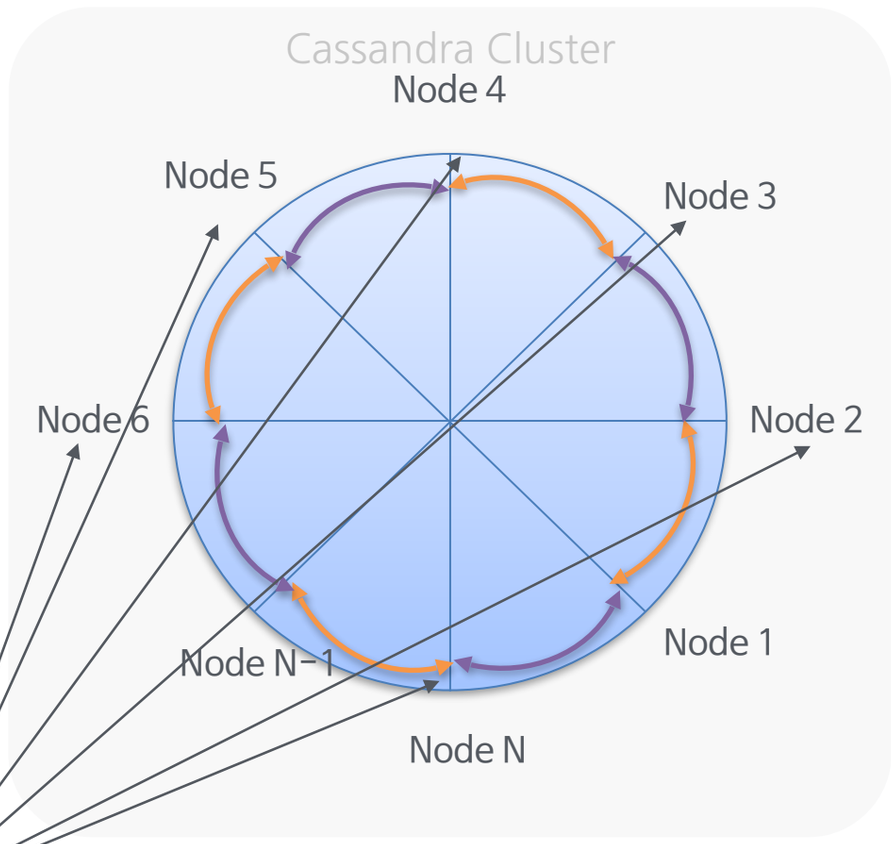


4. Apache Usergrid > Architecture



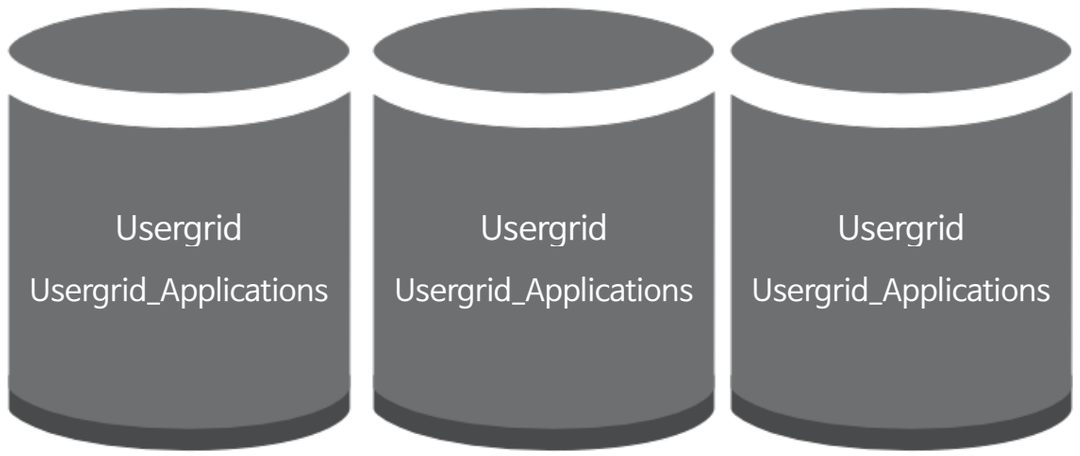
Core





4. Apache Usergrid > Architecture

Schema on Cassandra (Physical Layer)



KeySpaces

Usergrid

- Applications
- Properties
- Tokens

Usergrid_Applications

- Application_Aggregate_Counters
- Application_Roles
- Entity_Aliases
- Entity_Composite_Dictionaries
- Entity_Connections
- Entity_Counters
- Entity_Dictionaries
- Entity_Id_Sets
- Entity_Index
- Entity_Index_Entries
- Entity_Properties
- Consumer_Queue_Messages_Properties
- Entity_Metadata
- MQ_Consumers
- MQ_Counters
- MQ_Property_Index
- MQ_Property_Index_Entries
- Queue_Dictionaries
- Queue_Inbox
- Queue_Properties
- Queue_Subscribers
- Queue_Subscriptions

Agenda

1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid
2) Basic Concepts
3) Restful APIs
4) Architecture
5) Data Processing
6) CRUD

5. An Overview of Data Processing

- Metadata
  - Information of Organizations, Applications
  - Information of Collections
  - Information of Entity
- Real data
  - Name, Value

4. Apache Usergrid > Data Processing

- Collections
  - Organization > Application > Collections
  - Schema
    - Unique
    - Indexes (option, fulltextindexed)
      - Alias : Another lookup property
  - Information of added to the entity in collection
- Entity
  - Information of property
  - Property name
  - Property value

4. Apache Usergrid > Data Processing

Collections

- Predefined Collections
  - Users, Groups, Roles, Activities, Devices, Events, Folders, Assets

- Dynamic Collections
  - uuid (unique, required)
  - name (alias)
  - created (required)
  - modified (required)
  - … (indexed, fulltextIndexed)

4. Apache Usergrid > Data Processing

Collections

Predefined Collections

- Application
  - name (required, indexed, alias, unique)
  - title
  - accesstokenttl
  - description
  - collections
  - counters
  - activated
  - disabled
  - modified (required)
  - … (indexed, fulltextIndexed)
  - allowOpenRegistration, registrationRequiresEmailConfirmation, registrationRequiresAdminApproval, notifyAdminOfNewUsers
  - activities, assets, events, folders, groups, users, devices

4. Apache Usergrid > Data Processing

Collections

Predefined Collections

- Users
  - username (required, indexed, fulltextIndexed, alias, unique)
  - email (indexed, unique)
  - name (indexed, fulltextIndexed)
  - activated(indexed)
  - deactivated(indexed)
  - confirmed(indexed)
  - disabled(indexed)
  - picture(indexed)
  - created (required)
  - modified (required)
  - ... (indexed, fulltextIndexed)
  - connections, permissions, credentials, groups(linkedCollection), devices(linkedCollection), activities(linkedCollection), feed(linkedCollection), roles(linkedCollection)

4. Apache Usergrid > Data Processing

Collections

Predefined Collections

- Devices
  - name (required, indexed, alias, unique)
  - created (required)
  - modified (required)
  - … (indexed, fulltextIndexed)
  - users(linkedCollection)

4. Apache Usergrid > Data Processing

Collections

Predefined Collections

- Groups
  - path (required, indexed, alias, unique)
  - created (required)
  - modified (required)
  - … (indexed, fulltextIndexed)
  - users(linkedCollection), connections, credentials, permissions, activities(linkedCollection), feed(linkedCollection), roles(linkedCollection)

4. Apache Usergrid > Data Processing

Collections

Predefined Collections

- Role
  - name (required, indexed, alias, unique)
  - roleName(indexed)
  - title(indexed)
  - inactivity(indexed)
  - created (required)
  - modified (required)
  - ... (indexed, fulltextIndexed)
  - permissions
  - users(linkedCollection)
  - groups(linkedCollection)

Agenda

1. BaaS
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

1) Apache Usergrid
2) Basic Concepts
3) Restful APIs
4) Architecture
5) Data Processing
6) CRUD

4. Apache Usergrid > CRUD

Create an entity

```
Map<String, Object> properties = new
LinkedHashMap<String, Object>();
properties.put("username", "sungju");

Entity user = entityManager.create("user", properties);
```

4. Apache Usergrid > CRUD

Process of creating an entity

1. Precondition

    - All commands used batch_mutate in the Cassandra thrift API, so you should care to set thrift buffer size.

2. Pre-processing

3. Handling metadata

4. Loop through all the properties

4. Apache Usergrid > CRUD

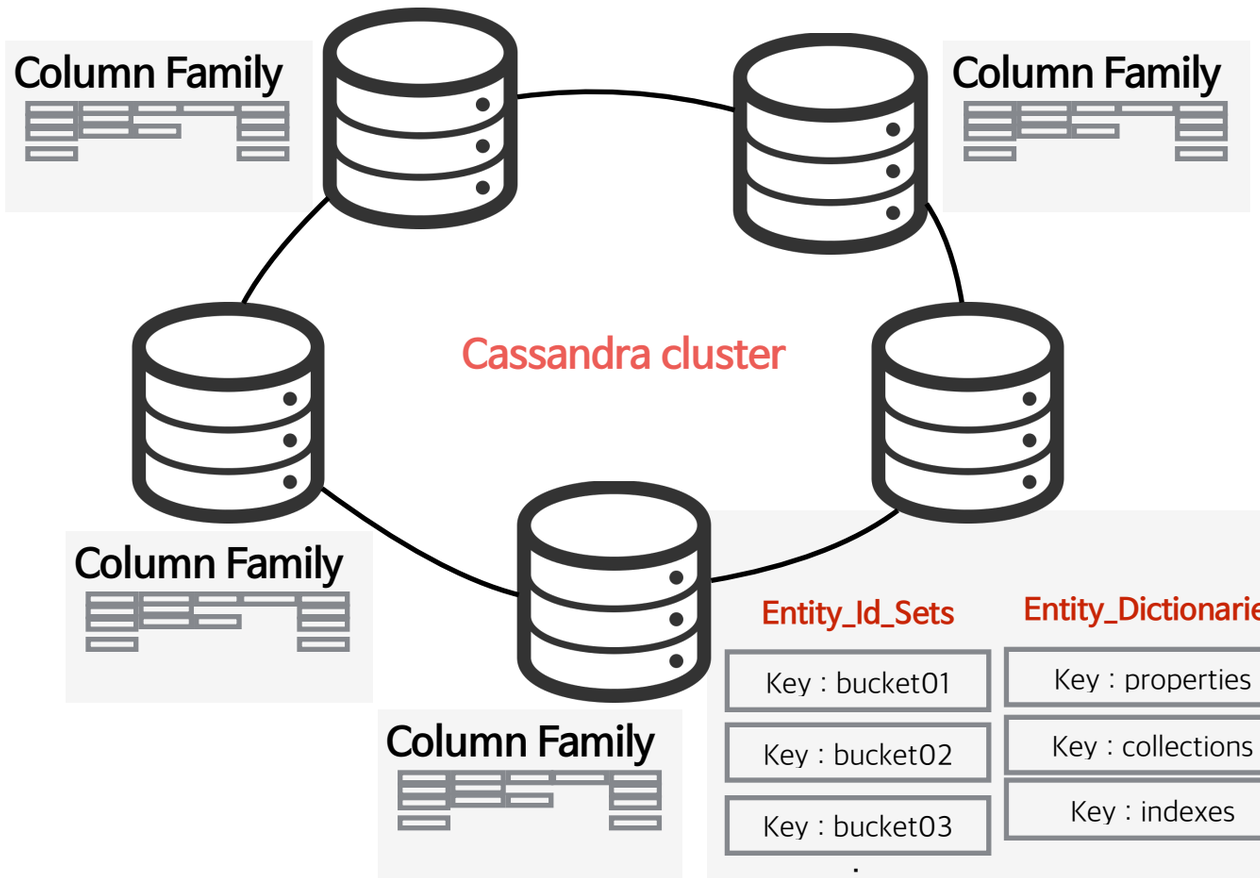
Process of creating an entity (con't)

Indexing

1) Background
    1) http://anuff.com/2010/07/secondary-indexes-in-cassandra/
    2) http://anuff.com/2011/07/cassandra-summit-sf-2011-presentation/
    3) http://anuff.com/2011/02/indexing-in-cassandra/
2) Index Type - COLLECTION, CONNECTION, GEO
3) Get tokens based on lucene 3.0 (token separator, space)
    1) fulltextIndex = false, (Key="name", Value="Sungju Jin")
    2) fulltextIndex = true, (Key="name", Value="Sungju Jin"), (Key="name.keyword", Values="sungju, jin")
4) Delete all matching index in Entity_Index from the entries retrieved from Entity_Index_Entries in the previous step
5) Delete all columns in Entity_Index_Entries that were previously retrieved
6) Insert new entry in Entity_Index
7) Insert new entry in Entity_Index_Entries

4. Apache Usergrid > CRUD

Usergrid on Cassandra (Physical Layer)



Column Family

Entity_Id_Sets	Entity_Dictionaries	Entity_Unique	Entity_Properties	Entity_Index	Entity_Index_Entries
Key : bucket01	Key : properties	Key	Key(uuid)	Key : bucket01	Key
Key : bucket02	Key : collections	Entity_Aliases		Key : bucket02	
Key : bucket03	Key : indexes	Key		Key : bucket03	
⋮				⋮	
Key : bucket20				Key : bucket20	

Entity_Composite_Dictionaries

Key : container_entities

4. Apache Usergrid > CRUD

Update an entity

```
UUID uuid = UUID.fromString("0177c265-a596-11e3-b4cd-406c8f07e929");
Entity entity = EntityFactory.newEntity(uuid,"users");
entity.setProperty("username", "sungju1");
```

```
Entity user = entityManager.update(entity);
```

4. Apache Usergrid > CRUD

Delete an entity

```
UUID uuid = UUID.fromString("0177c265-a596-11e3-b4cd-406c8f07e929");
Entity entity = EntityFactory.newEntity(uuid,"users");
Entity user = entityManager.delete(entity);
```

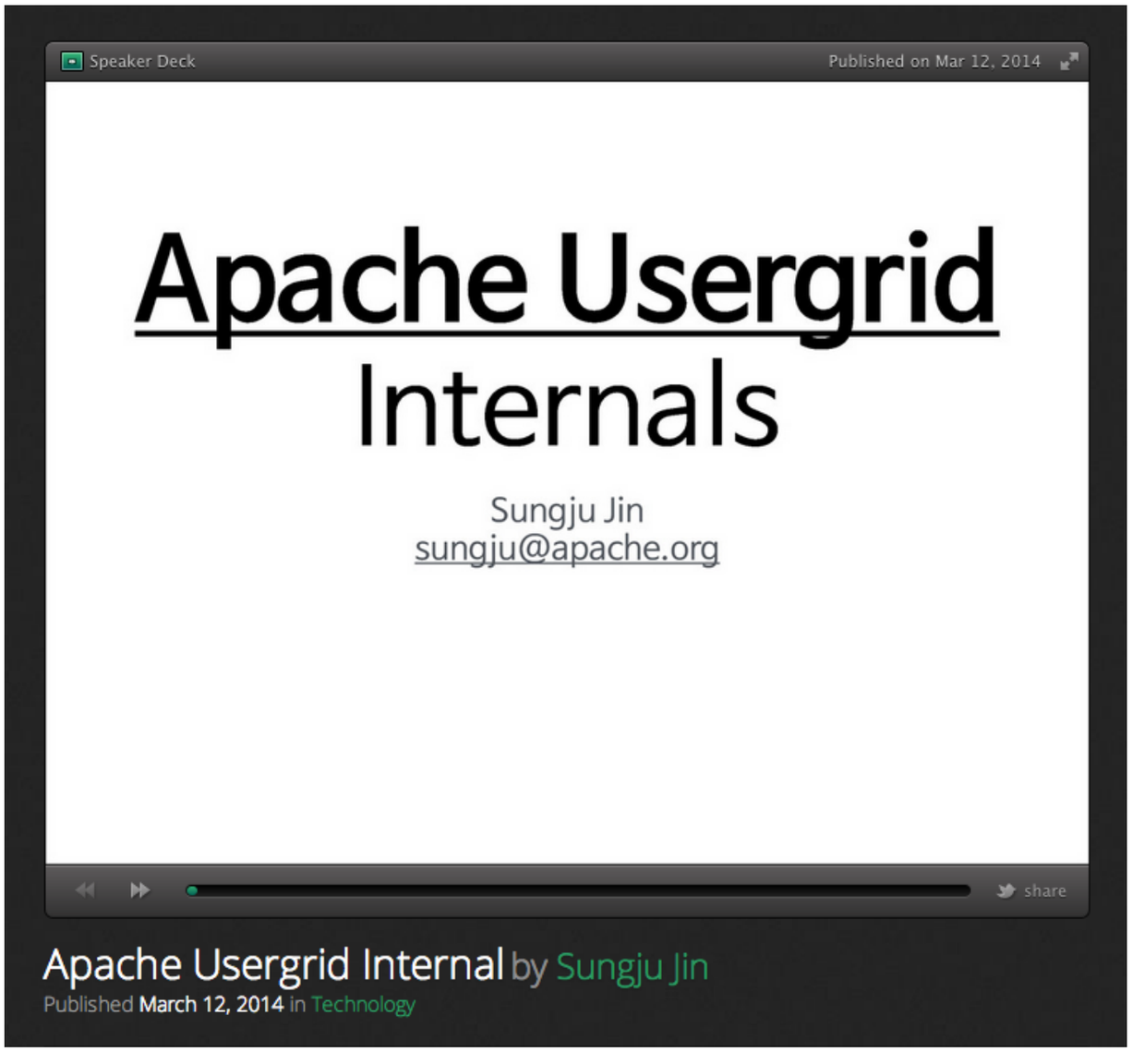
4. Apache Usergrid > CRUD

Read an entity

1. Get an entity by UUID
2. Get an entity by Alias
3. Retrieve entities
4. Retrieve entities by Query

4. Apache Usergrid > CRUD

- If you're interest in Usergrid 1.0 internal, this presentation will help you!



https://speakerdeck.com/sungjuly/apache-usergrid-internal

4. Apache Usergrid 2.0

Currently developing Usergrid 2.0

- Re-implement persistent layer
  - To improve performance
    - MVVC(Multiversion concurrency)
    - Replace C* driver to Astyanax from Hector
    - Store index data using ElasticSearch
    - Reactive system using RxJava

4. Apache Usergrid

Lost of benefit if you adopt Usergrid

- Restful APIs are really great!
- Great abstraction
- Scalability

Remember Usergrid philosophy

- If scalability is more important to your system than consistency or isolation, Apache Usergrid will help you!
- If not, you should re-implement persistent layer based on your requirement.

Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

5. Apache Usergrid & Docker

Custom APIs(=Cloud Code)

- Business Logic
- API Orchestration Layer
- Transactions

5. Apache Usergrid & Docker

- Write code. and then deploy. Have your own api!

- Call http://host/org/app/custom/yourapi

yourapi.js

```
var sample = function (request, response) {
    if (request == null) {
        response.error(error);    //HTTP Status Code : 400
    } else {
        var results = {
            title: "hello world"
        }
        response.finish(results);    //HTTP Status Code : 200
    }
};

runnable.function = sample;
```

5. Apache Usergrid & Docker

- If you’re private provider, isolation and security are not required.

- If you’re public provider, you have to consider isolation and security for custom APIs.

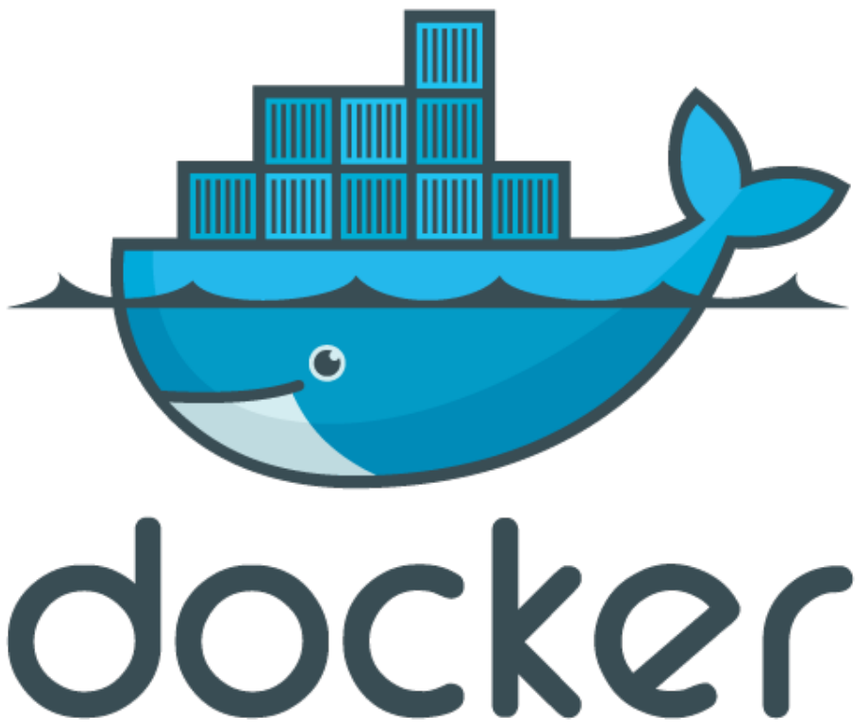
  - Limitation resource (cpu, memory, disk)

  - Security (don’t allow way such as file system)

5. Apache Usergrid & Docker

- Docker - An open platform for distributed applications for developers and sysadmins.

- Docker has also isolation platform such as LXC

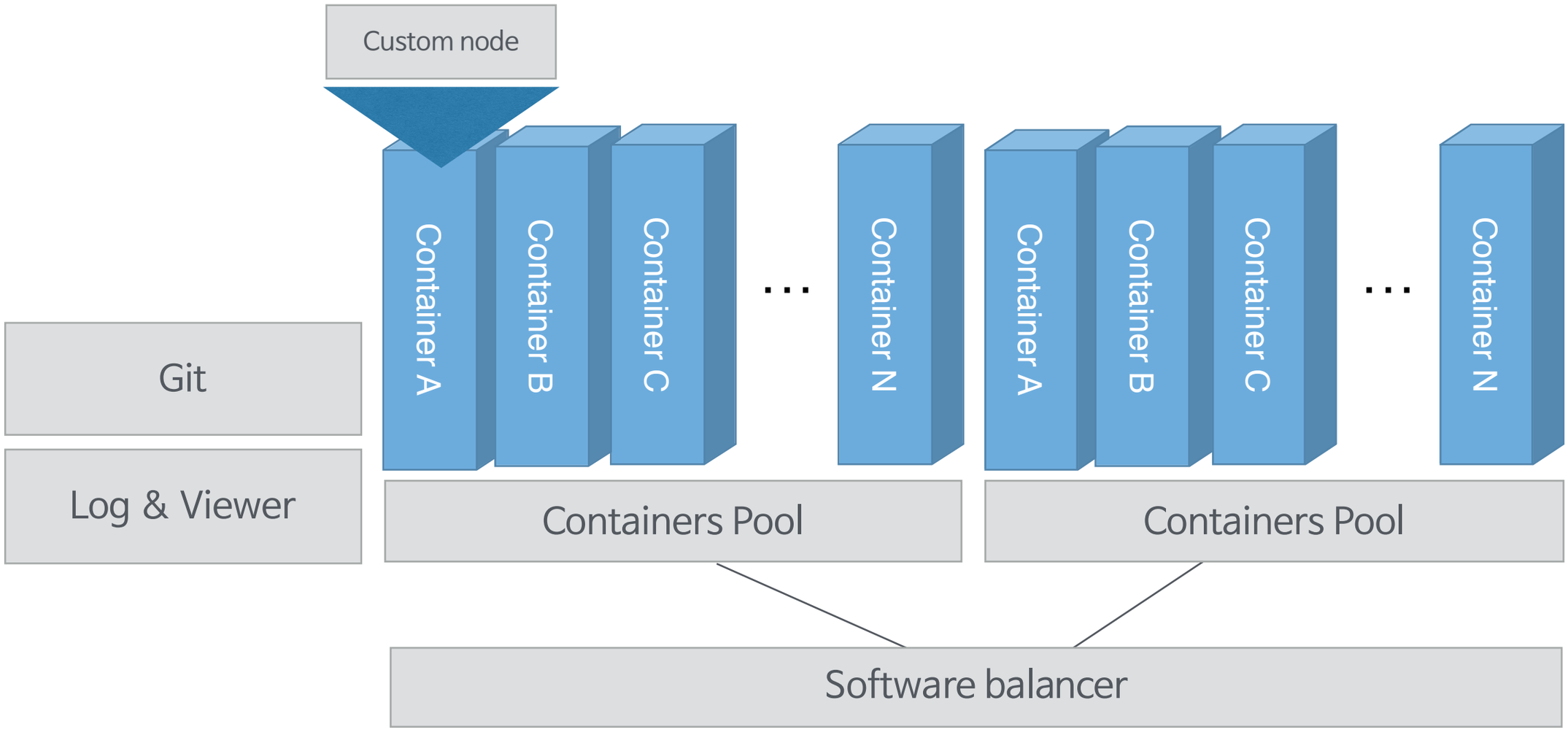


5. Apache Usergrid & Docker

- Implement custom APIs using open source
  - Isolation environment : Docker
  - Authentication : Apache Usergrid
  - Security : Customized node.js
  - Managing Code version : JGit
  - Logging : fluentd, flume
  - Log viewer : web socket

5. Apache Usergrid & Docker

- Architecture



Agenda

1. BaaS(Backend-as-a-Service)
2. Requirements for Building Your BaaS
3. BaaS in Production
4. Apache Usergrid
5. Apache Usergrid & Docker
6. Lessions Learned

Define Requirements!

https://www.flickr.com/photos/coenversluis/5441924762

Private or Public BaaS

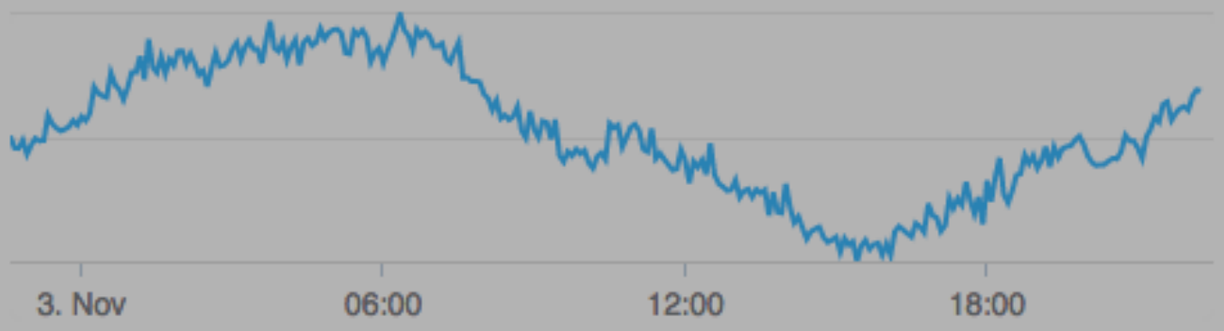
https://www.flickr.com/photos/christinawelsh/5690769079

Measure Everything

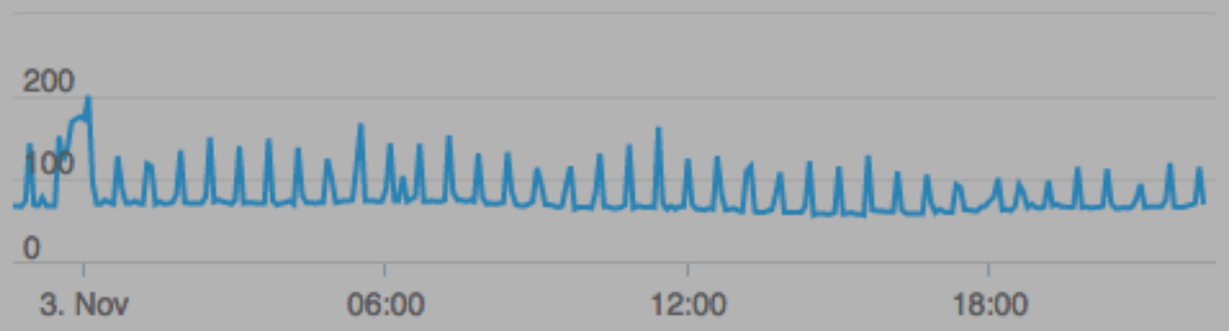
https://www.flickr.com/photos/laurenmanning/5658962691

Publish system status

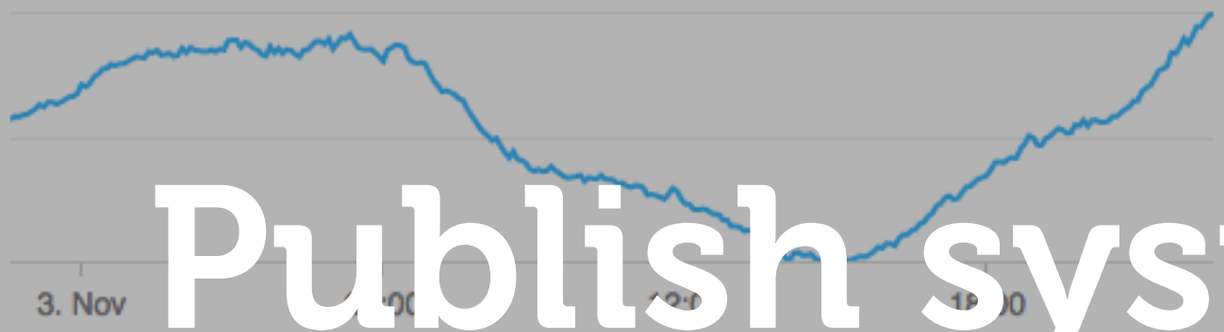
Website Requests



Website Response Time 81ms

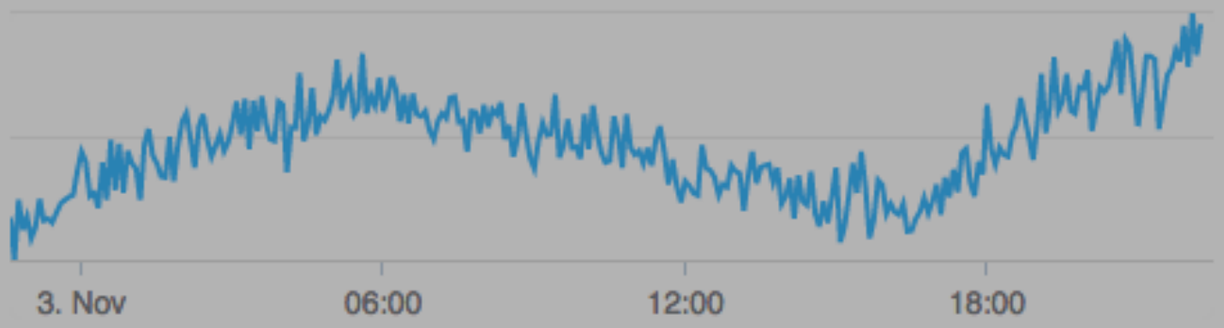


Player Requests

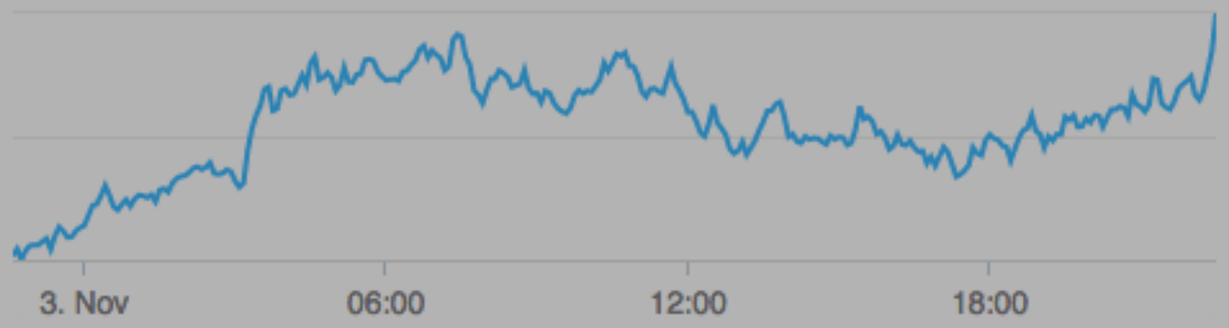


Player Response Time 25ms

Uploads



Active Encoder Jobs



Searches

Errors 0.06/sec

Define your policy (burst limit, unused app)

https://www.flickr.com/photos/amchu/3860550438

6. Lessions Learned

- Define Requirements!
- Private vs Public BaaS
- Measure Everything
- Publish system status
- Define policy (burst limit, unused app)

Goal

- BaaS Architecture Overview
  - Requirements
  - Production
- Introduction to Apache Usergrid

Thanks

Q & A

Thanks

Do you want to contribute?

http://usergrid.incubator.apache.org/docs/contribute-code/

How to Contribute Code

contribute to this article on github

Usergrid is an open source project developed by folks like you. Anybody can contribute, but we do have some rules in place and we do like to review and discuss changes in public on our mailing lists and on our Github account.

We use Github as our code review and collaboration system and we use the Apache Git repo as our official repository of record, that's the code that we release when we make an official release.

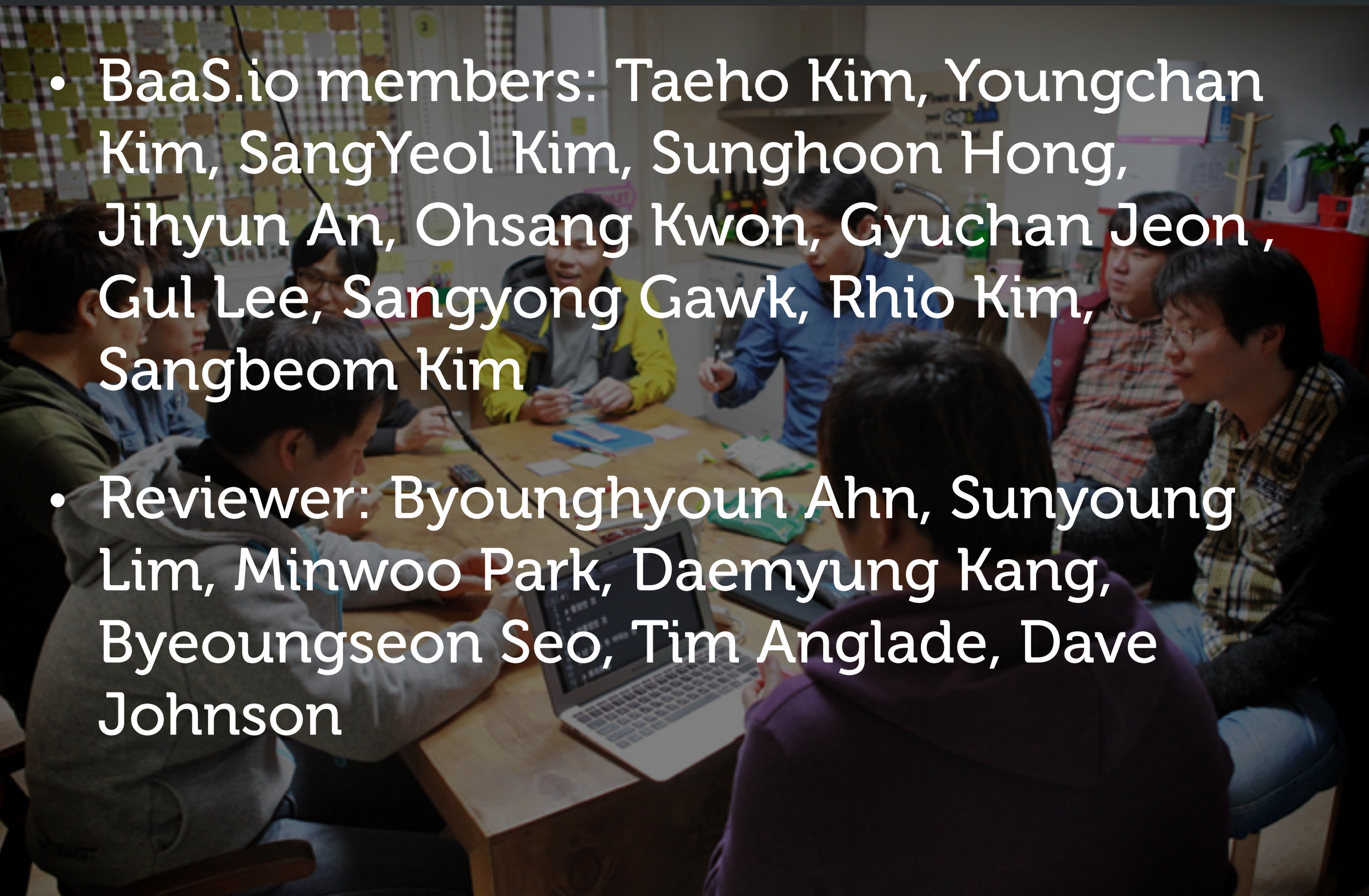
We have two slightly different procedures for small fixes and for more significant changes.

For small fixes

We welcome small "drive-by" contributions from impatient developers! If you just have a small fix that you want to contribute then these are the steps you should follow to get your changes into Usergrid.

- **Fork usergrid/usergrid**. Fork this repo: https://github.com/usergrid/usergrid. You will work in your fork. Clone it to a directory on your computer and...
- **Make your changes** in your fork and don't forget to create tests for those changes.

Special Thanks



- BaaS.io members: Taeho Kim, Youngchan Kim, SangYeol Kim, Sunghoon Hong, Jihyun An, Ohsang Kwon, Gyuchan Jeon , Gul Lee, Sangyong Gawk, Rhio Kim, Sangbeom Kim
- Reviewer: Byounghyoun Ahn, Sunyoung Lim, Minwoo Park, Daemyung Kang, Byeoungseon Seo, Tim Anglade, Dave Johnson

References

- Open Source Mobile Backend on Cassandra
  http://www.slideshare.net/edanuff/open-source-mobile-backend-on-cassandra
- Cassandra Indexing Techniques
  http://anuff.com/2011/07/cassandra-summit-sf-2011-presentation/
- Secondary indexes in Cassandra
  http://anuff.com/2010/07/secondary-indexes-in-cassandra/
- Indexing in Cassandra
  http://anuff.com/2011/02/indexing-in-cassandra/
- Apache Cassandra
  http://www.slideshare.net/starlight60/apache-cassandra-32301526