



CLOUDFOUNDRY

brief introduction to CF-Abacus

what is it, its architecture, and its future



CF-Abacus team

<https://github.com/cloudfoundry-incubator/cf-abacus>

pic credit: <http://www.tecmaths.com>

v0.5.0
Feb. 3rd, 2016



agenda

- what is CF-Abacus? and what it's not
- examples of users of CF-Abacus
- architecture
 - overview - μ services
 - service provider APIs
- team and process
- status
- how you can contribute?
- references



what is CF-Abacus?

what are its main design points?

- pipeline of micro-services (μ services) processing data
- usage metering and aggregation functions are customizable
- usage submitted by service and runtime providers (anytime)
- usage processed by μ services pipeline for metering, rating...

what is it used for? usage reports useful for customer billing

what are some alternatives? none (comprehensive, OSS, for CF)



what CF-Abacus is not

what problems are we not solving

- billing or charging customers (need external billing service)
- making all service brokers usage common

what you should not use it for? bill directly to customers



where is CF-Abacus used today

IBM Bluemix

- Originally extracted from initial [Bluemix](#) public codebase
- Bluemix dedicated (slices of Softlayer)
- Bluemix local (installed on customer premises)

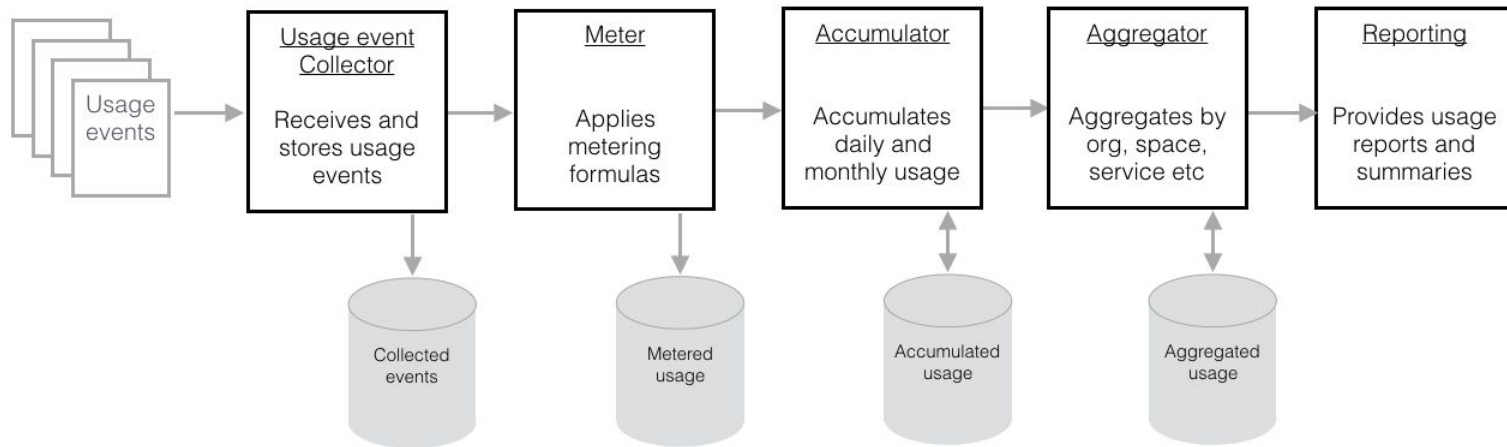
SAP-Hana - integration prototype moving to production in 2016

Others? various “kicking the tires”



architecture overview

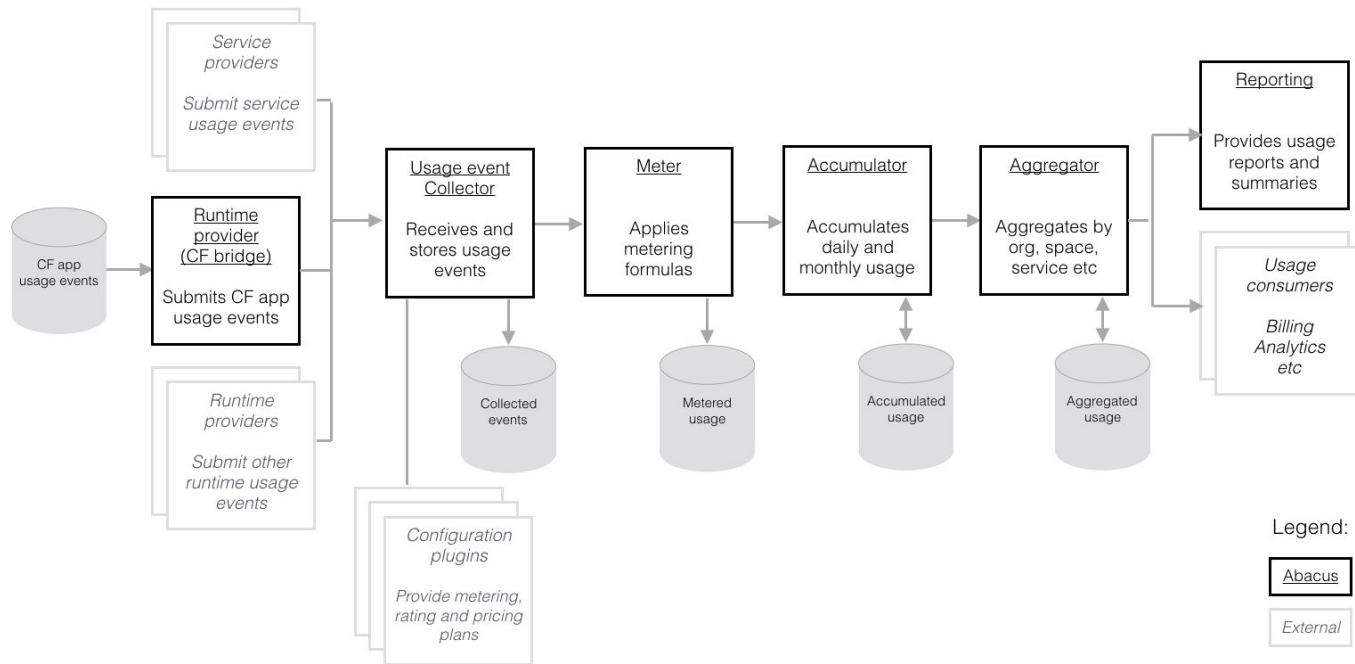
data processing pipeline architecture style





architecture overview (cont.)

cloud platform integration





demo

```
cd cf-abacus

# Point CF CLI to your local Cloud Foundry deployment and
# create a CF security group for the Abacus apps
bin/cfsetup

# Run cf push on the Abacus apps to deploy them to Cloud Foundry
npm run cfpush

# Check the state of the Abacus apps
cf apps

# You should see something like this
Getting apps in org <your organization> / space <your space>...
OK
```

name	requested state	instances	memory	disk	urls
abacus-usage-collector	started	1/1	512M	512M	abacus-usage-c
abacus-usage-meter	started	1/1	512M	512M	abacus-usage-m
abacus-usage-accumulator	started	1/1	512M	512M	abacus-usage-a
abacus-usage-aggregator	started	1/1	512M	512M	abacus-usage-a
abacus-usage-rate	started	1/1	512M	512M	abacus-usage-r
abacus-usage-reporting	started	1/1	512M	512M	abacus-usage-r
abacus-provisioning-stub	started	1/1	512M	512M	abacus-provisi
abacus-account-stub	started	1/1	512M	512M	abacus-account
abacus-dbserver	started	1/1	1G	512M	abacus-dbserve



architecture overview (*cont.*)

technology

- JavaScript using Node.js (node \geq v5.3 and npm \geq 3.3)
- development version is self-contained with PouchDB
- JSON for all data representation and output

deployment style

- deploy CF-Abacus μ services as CF apps into your CF env
- use a full CouchDB backend for production



service resource configuration

service brokers

1. onboarding to CF env
2. submit usage metering plans
3. create security token
4. submit usage

CF env operator

- enable service usage

```
{
  "resource_id": "object-storage",
  "effective": 1420070400000,
  "measures": [
    {
      "name": "storage",
      "unit": "BYTE"
    },
    {
      "name": "api_calls",
      "units": "CALL"
    }
  ],
  "metrics": [
    {
      "name": "storage",
      "unit": "GIGABYTE",
      "meter": "(m) => m.storage / 1073741824",
      "accumulate": "(a, qty) => Math.max(a, qty)"
    },
    {
      "name": "thousand_api_calls",
      "unit": "THOUSAND_CALLS",
      "meter": "(m) => m.light_api_calls / 1000",
      "accumulate": "(a, qty) => a ? a + qty : qty",
      "aggregate": "(a, qty) => a ? a + qty : qty",
      "rate": "(p, qty) => p ? p * qty : 0",
      "summarize": "(t, qty) => qty",
      "charge": "(t, cost) => cost"
    }
  ]
}
```

POST|GET|PUT /v1/provisioning/resources/:resource_id/config



service provider APIs - resource usage

POST /v1/metering/collected/usage

```
{
  "usage": [
    {
      "start": 1396421450000,
      "end": 1396421451000,
      "organization_id": "us-south:54257f98-83f0-4eca-ae04-9ea35277a538",
      "space_id": "d98b5916-3c77-44b9-ac12-04456df23eae",
      "consumer_id": "app:d98b5916-3c77-44b9-ac12-045678edabae",
      "resource_id": "object-storage",
      "plan_id": "basic",
      "resource_instance_id": "d98b5916-3c77-44b9-ac12-04d61c7a4eae",
      "measured_usage": [
        {
          "measure": "storage",
          "quantity": 10
        },
        {
          "measure": "api_calls",
          "quantity": 10
        }
      ]
    }
  ]
}
```



service provider APIs - resource pricing

POST /v1/pricing/resources/:resource_id/config/:time

```
{
  "resource_id": "object-storage",
  "effective": 1420070400000,
  "plans": [
    {
      "plan_id": "basic",
      "metrics": [
        {
          "name": "storage",
          "prices": [
            {
              "country": "USA",
              "price": 1
            },
            {
              "country": "EUR",
              "price": 0.7523
            },
            {
              "country": "CAN",
              "price": 1.06
            }
          ]
        }
      ]
    },
    {
      "name": "thousand_light_api_calls",
      "prices": [
        {
          "country": "USA",
          "price": 0.03
```

```

        {
          "country": "EUR",
          "price": 0.0226
        },
        {
          "country": "CAN",
          "price": 0.0317
        }
      ]
    },
    {
      "name": "heavy_api_calls",
      "prices": [
        {
          "country": "USA",
          "price": 0.15
        },
        {
          "country": "EUR",
          "price": 0.1129
        },
        {
          "country": "CAN",
          "price": 0.1585
        }
      ]
    }
  ]
}
```



service provider APIs - usage reporting

GET /v1/metering/organizations/:organization_id/aggregated/usage/:time

```
{
  "start": 1435622400000,
  "end": 1435708799999,
  "processed": 1435708800000,
  "organization_id": "us-south:a3d7fe4d-3cb1-4cc3-a831-ffe98e20cf27",
  "charge": 46.09,
  "id": "k-a3d7fe4d-3cb1-4cc3-a831-ffe98e20cf27-t-0001435622400000",
  "spaces": [
    {
      "space_id": "aaee239-f3f8-483c-9dd0-de5d41c38b6a",
      "charge": 46.09,
      "consumers": [
        {
          "consumer_id": "app:d98b5916-3c77-44b9-ac12-045678edabae",
          "charge": 46.09,
          "resources": [
            {
              "resource_id": "object-storage",
              "charge": 46.09,
              "aggregated_usage": [
                {
                  "metric": "storage",
                  "quantity": 1,
                  "summary": 1,
                  "charge": 1
                },
                {
                  "metric": "thousand_light_api_calls",
                  "quantity": 3,
                  "summary": 3,
                  "charge": 0.09
                },
                {
                  "metric": "heavy_api_calls",
                  "quantity": 300,
                  "summary": 300,
                  "charge": 45
                }
              ]
            }
          ]
        }
      ]
    }
  ]
}
```

```
,
  "plans": [
    {
      "plan_id": "basic",
      "charge": 46.09,
      "aggregated_usage": [
        {
          "metric": "storage",
          "quantity": 1,
          "summary": 1,
          "cost": 1,
          "charge": 1
        },
        {
          "metric": "thousand_light_api_calls",
          "quantity": 3,
          "summary": 3,
          "cost": 0.09,
          "charge": 0.09
        },
        {
          "metric": "heavy_api_calls",
          "quantity": 300,
          "summary": 300,
          "cost": 45,
          "charge": 45
        }
      ]
    }
  ],
  "resources": [
    {
      "resource_id": "object-storage",
      "charge": 46.09,
      "aggregated_usage": [

```



project process and current team

incubation project - created to explore and test (optional to core)

distributed commit process

IBM engineers (@jsdelfino, @betafood, @kruely)

SAP engineers (@hsiliev & @georgethebeatle)

Independent (@sasrin)

[Own tracker](#) for all work items and [Github issues](#)

team



CLOUDFOUNDRY



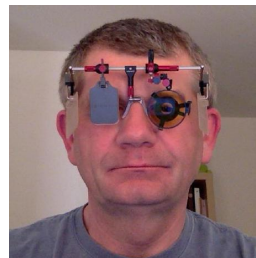
Jean-Sebastien Delfino
IBM
Committer



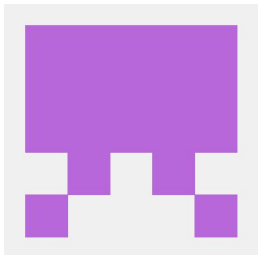
Hristo Iliev
SAP
Committer



**Saravanakumar
Srinivasan 'Assk'**
Committer



Piotr Przybylski
IBM - Bluemix



Benjamin Cheng
IBM
Committer



Georgi Sabev
SAP
Committer



Kevin Yudhiswara
IBM
Committer



**Rajkiran
Balasubramanian**
IBM - Bluemix



status and future

recent updates

- flexible metering and rating plan configuration API
- dynamic usage reporting using GraphQL
- usage accumulation in time windows

near-term

- Mongo-DB support
- async queuing for multi-datacenter deployments
- usage processing failure management



status (*cont.*)

near-term (*cont.*)

- concourse build pipeline

longer-term

- built-in default UIs (on-boarding and usage reporting)
- CF-Abacus-as-a-Service (via service broker)
- consuming and providing usage notifications
- dynamic rolling and slack time windows



how can you contribute

integrator

- “kick the tires” - try deploying CF-Abacus into your env
- create UI for onboarding and usage report presentation
- integrate with your CF service brokers that report usage

service developer

- support submitting usage to CF-Abacus
- implement the [usage submission API](#)



how can you contribute (*cont.*)

developer

- “kick the tires” - try deploying CF-Abacus into your env
- create any new issues you find on Github
- write code, tests, and submit pull requests

tester and documentation

- test with other brokers
- documentation needs improvements



references

<https://github.com/cloudfoundry-incubator/cf-abacus>

[CF-Abacus Tracker](#) project

[IRC](#) channel, [Slack](#) and [Gitter](#), and [CF mailing list](#)

project [README](#) and [FAQs](#)

[APIs doc](#) for integration overview

thank you



CLOUDFOUNDRY

감사합니다 Natick
Grazie Danke Ευχαριστίες Dalu
Thank You Köszönöm
Спасибо Dank Gracias
谢谢 Merci Seé
Obbrigado
ありがとう

credit: <http://knowyourmeme.com/photos/522333-language>