Pivotal Cloud Foundry

Charge Back strategies

3/2017

Pivotal

Typical PCF Charge Back strategies

1. Fixed Plans
2. Metered services (Pay-Per-Use)
3. Combination of Fixed + Metered

1. Fixed Plans

- Pay weekly/monthly/yearly per allocation model
- Controlled by PCF Quota Plans for Orgs and Spaces. Example:

Plan	Total RAM	RAM per App	Routes	Service Instances
trial	2G	1G	2	1
small	10G	2G	50	10
large	100G	10G	1000	100

- Easier to to administer, automate and grow
- Consumers not forced to self police consumption costs
- Platform team required to monitor a bit
  (Why do you want a 128 gig quota? Can we help?)

2. Metered Services (Pay-Per-Use)

- Based on consumption of RAM, CPU, Services, etc
- Pay **after** use
- Must be able to charge customer frequently via a bill or report
- More difficult to implement from a process and automation standpoint
- Forces customers to self-police consumption in order to lower their costs

3. Combination of Fixed + Monitored

- Offer fixed plans and quotas for Application Instances
- Charge for instances of select Services and/or Buildpacks
- Fixed plan for AI's simplifies model, though complexity of monitoring automation of pay-per-use instances remains

Usage Data Collection: Fixed plans

- Get list of all Orgs

  `cf curl /v2/organizations`

- For each Org, get its corresponding quota details and pricing

  `cf curl /v2/quota_definitions/<quota_guid>`

- Build report of Orgs/Spaces and Quotas Charge Back

Usage Data Collection: Metered services 1/2

- For each PCF Org/Space, for the week/month/year, get:
  - Application Instances usage
    - RAM allocated
    - Disk allocated
    - Duration of instances (e.g. in seconds)
  - Services Instances usage
    - Service Plan ID and info
    - Duration of instances (e.g. in seconds)

Usage Data Collection: Metered services 2/2

- Define pricing structure for App usage, e.g.
  - X *dollars* for *1 Gb RAM / hour*
  - Y *dollars* for *1 Gb of Disk / hour*
- Define pricing structure for each Service type and plan, e.g.
  - RabitMQ `default` plan: W *dolllars / hour*
  - RabitMQ `large` plan: Z *dolllars / hour*
- Calculate and produce report/bill for each Org/Space, e.g.

$$OrgA_{usage} = (X \cdot \sum_{GbRAM/hr}) + (Y \cdot \sum_{GbDisk/hr}) + (\sum_{AllServices/hr} ...)$$

APIs and Tools for Usage Data Collection

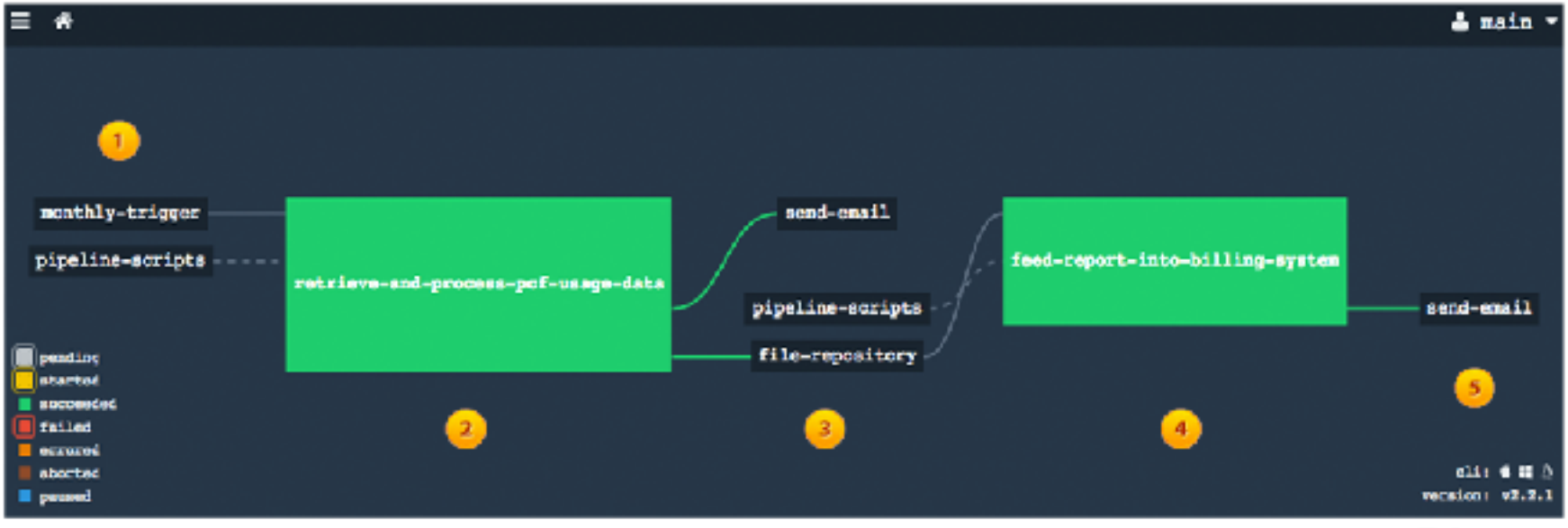
- Cloud Foundry API - App Usage Events
- App usage firehose nozzle
- Abacus
- ***PCF Accounting Report API***

PCF Accounting Report API

- Documentation
- Collects applications and service usage information for each Org and persists it for 90 days (vs. 30 days from CF API events)
- Very usefull to collect granular usage data for *Metered Services*: API endpoints: `/app_usages` and `/service_usages`
- Used by PCF Apps Manager - Accounting report
- Can be used to create custom Org and Space usage reports

Example: Usage Report producer

- PCF Usage Report producer
- Concourse CI pipeline that collects usage data for all Orgs
- Uses the PCF Accounting Report API
- Consolidates a single JSON report containing App and Services usage information about all Orgs and Spaces
- JSON report can be fed into billing systems or dashboards



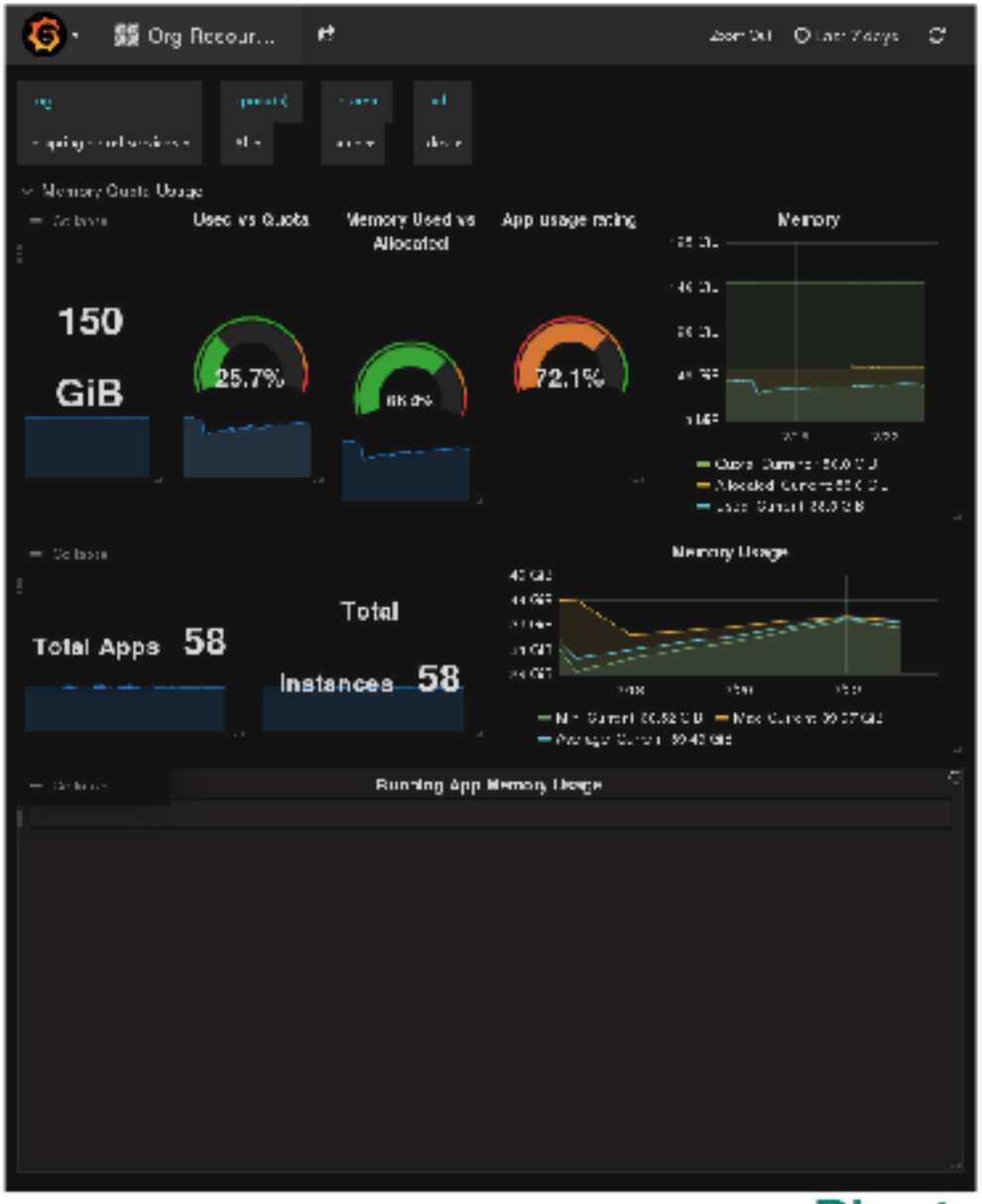
Example: Usage Report producer

- Sample JSON output report

```
{
  "start_date": "YYYY-MM-DD",   // report start date
  "end_date": "YYYY-MM-DD",     // report end date
  "total_app_instance_count": integer,
  "total_app_memory_used_in_mb": integer,
  "total_disk_quota_in_mb": integer, ...
  "organizations": [    // array of organization objects
    {  // organization object
      "name": "string", ...
      "total_app_instance_count": integer,
      "total_app_memory_used_in_mb": integer,
      "total_disk_quota_in_mb": integer,
      "spaces": [  // array of all spaces of this org
        { // space object
          "name": "string",  ...
          "app_usages": [ // array of applications
            { // app usage object
              "guid": "string",...
```

Example: Org Usage Grafana Dashboard

- Sample dashboard
- App usage data fed into a TSDB
- Filters per org, space, interval and environment



Questions?

Thanks!