

Indice

Premesse

L come logging

M come monitoraggio

Route

Store and aggregate


Visualize

Problemi

Conclusioni

Non solo distribuita, ma anche numerosa

- Un secondo effetto è che l'infrastruttura sarà **molto maggiore** di quella a cui è abituato l'IT tradizionale, e certi problemi di gestione si presentano solo su grande scala.
- O per meglio dire: sapere quello che accade nel **piccolo** non è un buon metro per capire cosa accade nel **grande**.



L'inizio

Tutto incomincia con . . . Patrick Debois che nel 2007 si trova a svolgere un lavoro “ibrido” di sviluppo e di sistemi, e non è contento di come sta procedendo.

Agile 2008 Andrew Shafer parla di “*Agile Infrastructure*”, o per meglio dire “monologa”.

23 giugno 2009 John Allspaw presenta il talk “*10+ deploys per day: Dev & Ops cooperation at Flickr*”.

30-31 ottobre 2009 Il primo *DevOps Days* a Ghent, in Belgio. Grande successo di pubblico e di critica.

*da una presentazione di Damon Edwards pubblicata su IT Revolutions.

Tutto incomincia con ... Patrick Debois che nel 2007 si trova a svolgere un lavoro “ibrido” di sviluppo e di sistemi, e non è contento di come sta procedendo.

Agile 2008 Andrew Shafer parla di “*Agile Infrastructure*”, o per meglio dire “monologa”.

23 giugno 2009 John Allspaw presenta il talk *"10+ deploys per day: Dev & Ops cooperation at Flickr"*.

*da una presentazione di Damon Edwards pubblicata su IT
Revolutions.

L'inizio

Tutto incomincia con . . . Patrick Debois che nel 2007 si trova a svolgere un lavoro “ibrido” di sviluppo e di sistemi, e non è contento di come sta procedendo.

Agile 2008 Andrew Shafer parla di “*Agile Infrastructure*”, o per meglio dire “monologa”.

23 giugno 2009 John Allspaw presenta il talk “*10+ deploys per day: Dev & Ops cooperation at Flickr*”.

30-31 ottobre 2009 Il primo *DevOps Days* a Ghent, in Belgio. Grande successo di pubblico e di critica.

*da una presentazione di Damon Edwards pubblicata su IT Revolutions.

L'inizio

#DEVOPS diventa un tema caldo in numerose conferenze: viene creato un formato, i DevOps Days, che in pochi anni si replicano per decine di volte in tutto il mondo.

Si enfatizza il tema di come funziona l'IT e di come dovrebbe invece funzionare. Il discorso verte sugli strumenti, su quali funzionano e su quali no, sulle *best practices* e sulle tecniche.

Il movimento oggi

NEL MARZO DEL 2011 anche Gartner si accorge del movimento e pubblica il suo oroscopo: “*The Rise of a New IT Operations Support Model*” che prevede che per il **2015** il movimento sarà passato, da una nicchia nell’ambito *cloud*, **all’adozione nel 20% delle imprese Global 2000.**

Si afferma a livello globale l’esistenza di un movimento *from practitioners, to practitioners.*

Il movimento oggi

NEL MARZO DEL 2011 anche Gartner si accorge del movimento e pubblica il suo oroscopo: “*The Rise of a New IT Operations Support Model*” che prevede che per il **2015** il movimento sarà passato, da una nicchia nell’ambito *cloud*, all’**adozione nel 20% delle imprese *Global 2000***.

Si afferma a livello globale l’esistenza di un movimento *from practitioners, to practitioners*.

Il movimento in Italia

ottobre 2012 La prima edizione dei DevOps Days italiana, a Roma, con quasi duecento partecipanti da tutto il mondo.

febbraio 2013 A Firenze si è tenuto il primo “Incontro DevOps Italia”, con 80+ persone presenti.

febbraio 2014 Bologna: il secondo “Incontro DevOps Italia”, con 120+ persone presenti.

aprile 2015 Bologna: il terzo “Incontro DevOps Italia”, con 200 persone presenti.

aprile 2016 Bologna: il quarto “Incontro DevOps Italia”, il primo aperto anche alla comunità internazionale.

People and process first. If you don't have culture, all automation attempts will be fruitless. (John Willis)

People and process first. If you don't have culture, all automation attempts will be fruitless. (John Willis)

Automate

2 AUTOMATIZZARE ogni azione. Se un'azione manuale può essere svolta da un programma, che lo si scriva. E lo si scriva secondo i crismi con cui si scrivono i programmi: il fatto che sia un programma per i sistemi (o per i *server*) non è un'offesa.

“Sistemista” non è un'offesa.

Automate

2 AUTOMATIZZARE ogni azione. Se un'azione manuale può essere svolta da un programma, che lo si scriva. E lo si scriva secondo i crismi con cui si scrivono i programmi: il fatto che sia un programma per i sistemi (o per i *server*) non è un'offesa.

“Sistemista” non è un'offesa.

Automate

2 AUTOMATIZZARE ogni azione. Se un'azione manuale può essere svolta da un programma, che lo si scriva. E lo si scriva secondo i crismi con cui si scrivono i programmi: il fatto che sia un programma per i sistemi (o per i *server*) non è un'offesa.

“Sistemista” non è un'offesa.

Corollario: *Infrastructure as code*

Se solo il codice definisce i componenti dell'infrastruttura significa che questa **non può** essere determinata da:

1. configurazioni manuali,
2. cose che si *clicciano* di qua e di là,
3. persone (*a.k.a.* consulenti) che arrivano e fanno cose.

Corollario: *Infrastructure as code*

Se solo il codice definisce i componenti dell'infrastruttura significa che questa **non può** essere determinata da:

1. configurazioni manuali,
2. cose che si *clicciano* di qua e di là,
3. persone (*a.k.a.* consulenti) che arrivano e fanno cose.

Corollario: *Infrastructure as code*

Se solo il codice definisce i componenti dell'infrastruttura significa che questa **non può** essere determinata da:

1. configurazioni manuali,
2. cose che si *cliccano* di qua e di là,
3. persone (*a.k.a.* consulenti) che arrivano e fanno cose.

Measure everything

3 MISURARE ogni componente dell'infrastruttura. Il concetto di *monitoring* non è affatto nuovo, l'innovazione è nell'avere degli strumenti che permettano di controllare **tutte le parti**.

Nell'approccio tradizionale si controlla **solo la parte sistemistica** mentre la parte applicativa ha — nella migliore delle ipotesi — al più una soluzione *ad hoc*.

Measure everything

3 MISURARE ogni componente dell'infrastruttura. Il concetto di *monitoring* non è affatto nuovo, l'innovazione è nell'avere degli strumenti che permettano di controllare **tutte le parti**. Nell'approccio tradizionale si controlla **solo la parte sistemistica** mentre la parte applicativa ha — nella migliore delle ipotesi — al più una soluzione *ad hoc*.

4 CONDIVIDERE un progetto comune, un obiettivo, delle pratiche, delle tecniche, degli strumenti, fra gruppi eterogenei, e che hanno obiettivi differenti (complementari).

Sharing is the loopback in the CAMS cycle. Creating a culture where people share ideas and problems is critical. (John Willis)

Forse non si è capito ma ...

... dire che solo il codice definisce l'infrastruttura, e che ogni azione deve essere automatizzata ... ovvero trasformata in *software* ... implica che chiunque adotti queste pratiche, e indipendentemente dal nome con cui si fa chiamare, è un ...

PROGRAMMATORE !

Forse non si è capito ma ...

... dire che solo il codice definisce l'infrastruttura, e che ogni azione deve essere automatizzata ... ovvero trasformata in *software* ... implica che chiunque adotti queste pratiche, e indipendentemente dal nome con cui si fa chiamare, è un ...

PROGRAMMATORE !

Forse non si è capito ma ...

... dire che solo il codice definisce l'infrastruttura, e che ogni azione deve essere automatizzata ... ovvero trasformata in *software* ... implica che chiunque adotti queste pratiche, e indipendentemente dal nome con cui si fa chiamare, è un ...

PROGRAMMATORE !

Forse non si è capito ma . . .

. . . dire che solo il codice definisce l'infrastruttura, e che ogni azione deve essere automatizzata . . . ovvero trasformata in *software* . . . implica che chiunque adotti queste pratiche, e indipendentemente dal nome con cui si fa chiamare, è un . . .

PROGRAMMATORE !

Il *log management* è un problema risolto

- ▶ Ok non proprio.
- ▶ Magari non è verissimo in ogni contesto, ma è un problema per cui esistono **soluzioni note** da anni (*decenni*).
- ▶ Giusto per fare un confronto:
 - ▶ `/var/log` esiste su ogni server Linux **da sempre**,
 - ▶ un posto **standard** dove tenere le metriche **non è mai stato previsto** (e tuttora non c'è).

Alternative per il log management

VISTA L'ABBONDANZA DI ALTERNATIVE, tutte di qualità, è molto probabile che le vostre necessità di log management siano soddisfatte da uno dei seguenti progetti:

- ▶ ELK, eventualmente (R)ELK, con Redis o Rabbitmq per modellare i canali con memoria - <https://www.elastic.co>
- ▶ Heka - <https://github.com/mozilla-services/heka/>
- ▶ Graylog2 - <https://www.graylog.org/>

Alternative per il log management

VISTA L'ABBONDANZA DI ALTERNATIVE, tutte di qualità, è molto probabile che le vostre necessità di log management siano soddisfatte da uno dei seguenti progetti:

- ▶ ELK, eventualmente (R)ELK, con Redis o Rabbitmq per modellare i canali con memoria -

<https://www.elastic.co>

- ▶ Heka -

<https://github.com/mozilla-services/heka/>

- ▶ Graylog2 - <https://www.graylog.org/>

Alternative per il log management

VISTA L'ABBONDANZA DI ALTERNATIVE, tutte di qualità, è molto probabile che le vostre necessità di log management siano soddisfatte da uno dei seguenti progetti:

- ▶ ELK, eventualmente (R)ELK, con Redis o Rabbitmq per modellare i canali con memoria -
<https://www.elastic.co>
- ▶ Heka -
<https://github.com/mozilla-services/heka/>
- ▶ Graylog2 - <https://www.graylog.org/>

Alternative per il log management

VISTA L'ABBONDANZA DI ALTERNATIVE, tutte di qualità, è molto probabile che le vostre necessità di log management siano soddisfatte da uno dei seguenti progetti:

- ▶ ELK, eventualmente (R)ELK, con Redis o Rabbitmq per modellare i canali con memoria - <https://www.elastic.co>
- ▶ Heka - <https://github.com/mozilla-services/heka/>
- ▶ Graylog2 - <https://www.graylog.org/>

Measure ! Measure ! Measure everywhere !

PER DEFINIRE l'atto del controllare, dobbiamo definire *cosa* intendiamo controllare, ovvero *cosa intendiamo misurare*.

Measure ! Measure ! Measure everywhere !

PER DEFINIRE l'atto del controllare, dobbiamo definire *cosa* intendiamo controllare, ovvero *cosa intendiamo misurare*.

- Una misura è un **valore numerico con un nome** e il **momento** in cui essa è stata effettuata.
- Una successione di misure è pertanto **una serie temporale di valore numerici** associati ad un'etichetta (o nome).

Measure ! Measure ! Measure everywhere !

PER DEFINIRE l'atto del controllare, dobbiamo definire *cosa* intendiamo controllare, ovvero *cosa intendiamo misurare*.

- Una misura è un **valore numerico con un nome** e il **momento** in cui essa è stata effettuata.
- Una successione di misure è pertanto **una serie temporale di valore numerici** associati ad un'etichetta (o nome).

Visualizzare **tutti** i dati

- ▶ Una caratteristica chiave di un sistema di monitoring è la **visualizzazione dei dati**, ovvero rendere immediatamente **esplicite** le informazioni.
- ▶ In generale i sistemi di monitoraggio usati in ambito *Ops* mostrano dei dati di funzionamento dei server: ovvero quantità come l'uso della CPU, della RAM eccetera.
- ▶ È **fondamentale** integrare questi dati con i dati di funzionamento delle applicazioni, ovvero **i dati applicativi**.

Visualizzare **tutti** i dati

- Una caratteristica chiave di un sistema di monitoring è la **visualizzazione dei dati**, ovvero rendere immediatamente **esplicite** le informazioni.
- In generale i sistemi di monitoraggio usati in ambito *Ops* mostrano dei dati di funzionamento dei server: ovvero quantità come l'uso della CPU, della RAM eccetera.
- È **fondamentale** integrare questi dati con i dati di funzionamento delle applicazioni, ovvero **i dati applicativi**.

Visualizzare **tutti** i dati

- Una caratteristica chiave di un sistema di monitoring è la **visualizzazione dei dati**, ovvero rendere immediatamente **esplicite** le informazioni.
- In generale i sistemi di monitoraggio usati in ambito *Ops* mostrano dei dati di funzionamento dei server: ovvero quantità come l'uso della CPU, della RAM eccetera.
- È **fondamentale** integrare questi dati con i dati di funzionamento delle applicazioni, ovvero **i dati applicativi**.

- In un test si effettua un'operazione e si verifica che **una certa quantità** (tipicamente il risultato) **sia consistente** con un determinato **valore atteso**.
- Il valore atteso denota che **una certa proprietà è valida**.
- La validità della proprietà è **lo scopo del test**.
- Nei sistemi di misura si disaccoppia **la raccolta delle quantità** dalla **verifica delle proprietà**.

...e oltre

- Raccogliere **continuamente** delle misure corrisponde quindi ad effettuare ripetutamente una parte delle azioni svolte da un test.
- Pertanto un sistema di misura contiene, in certo senso, **un'evoluzione** dell'approccio basato sui test.
- Perché si trasforma un evento che si compie **solo durante lo sviluppo**, in un'attività che si svolge **sempre, quando il sistema è operativo**.

Tesi provocatoria

UN SISTEMA DI MISURA BEN STRUTTURATO può efficacemente sostituire un insieme di test, col vantaggio che le misure permangono anche col sistema **in produzione**, mentre i test vengono eseguiti solo nelle fasi antecedenti al *deployment*.

I componenti di un'architettura di monitoraggio

Route è la parte che concretamente raccoglie la misura e fa sì che essa arrivi sui server preposti alla gestione del dato.

Store di solito è un database di tipo *round robin* (RRD) o specializzato per le *time series*.

Aggregate è un componente che si può realizzare in vari modi e che si occupa di accorpare insieme dei dati, o delle misure (ad esempio per effettuare uno *scaling* temporale).

Visualize è il *frontend* del sistema, il modo con cui si accedono interattivamente le misure, sotto forma di grafici (tipicamente).

I componenti di un'architettura di monitoraggio

Route è la parte che concretamente raccoglie la misura e fa sì che essa arrivi sui server preposti alla gestione del dato.

Store di solito è un database di tipo *round robin* (RRD) o specializzato per le *time series*.

Aggregate è un componente che si può realizzare in vari modi e che si occupa di accorpare insieme dei dati, o delle misure (ad esempio per effettuare uno *scaling* temporale).

Visualize è il *frontend* del sistema, il modo con cui si accedono interattivamente le misure, sotto forma di grafici (tipicamente).

I componenti di un'architettura di monitoraggio

Route è la parte che concretamente raccoglie la misura e fa sì che essa arrivi sui server preposti alla gestione del dato.

Store di solito è un database di tipo *round robin* (RRD) o specializzato per le *time series*.

Aggregate è un componente che si può realizzare in vari modi e che si occupa di accorpare insieme dei dati, o delle misure (ad esempio per effettuare uno *scaling* temporale).

Visualize è il *frontend* del sistema, il modo con cui si accedono interattivamente le misure, sotto forma di grafici (tipicamente).

I componenti di un'architettura di monitoraggio

Route è la parte che concretamente raccoglie la misura e fa sì che essa arrivi sui server preposti alla gestione del dato.

Store di solito è un database di tipo *round robin* (RRD) o specializzato per le *time series*.

Aggregate è un componente che si può realizzare in vari modi e che si occupa di accorpare insieme dei dati, o delle misure (ad esempio per effettuare uno *scaling* temporale).

Visualize è il *frontend* del sistema, il modo con cui si accedono interattivamente le misure, sotto forma di grafici (tipicamente).

I componenti di un'architettura di monitoraggio

Route è la parte che concretamente raccoglie la misura e fa sì che essa arrivi sui server preposti alla gestione del dato.

Store di solito è un database di tipo *round robin* (RRD) o specializzato per le *time series*.

Aggregate è un componente che si può realizzare in vari modi e che si occupa di accorpare insieme dei dati, o delle misure (ad esempio per effettuare uno *scaling* temporale).

Visualize è il *frontend* del sistema, il modo con cui si accedono interattivamente le misure, sotto forma di grafici (tipicamente).

I componenti in uso in BioDec

DOPO LUNGHI PATIMENTI, fatiche inenarrabili, false partenze, vicoli ciechi — ed in generale avere guadagnato un miliardo di anni di purgatorio — al momento usiamo questo *stack*:

Route collectd, statsd.

Store graphite (whisper), influxdb.

Aggregate graphite (carbon), influxdb.

Visualize graphite-web, grafana, graph-explorer.

I componenti in uso in BioDec

DOPO LUNGHI PATIMENTI, fatiche inenarrabili, false partenze, vicoli ciechi — ed in generale avere guadagnato un miliardo di anni di purgatorio — al momento usiamo questo *stack*:

Route collectd, statsd.

Store graphite (whisper), influxdb.

Aggregate graphite (carbon), influxdb.

Visualize graphite-web, grafana, graph-explorer.

Il punto chiave è **raccogliere le metriche dei dati operativi dei server** e mandarli ad un server Graphite o ad un server che parla il protocollo di Graphite (come ad esempio Influxdb).

Dati di sistema

NELLA LORO CONFIGURAZIONE MINIMA questi strumenti permettono **praticamente con la configurazione di default** di raccogliere le metriche **di sistema**.

Questo implica che con uno sforzo minimo è possibile **consolidare in un unico posto** tutte le metriche della parte server della vostra infrastruttura.

Dati di servizi o di middleware

A SECONDA DEI SERVIZI — ovvero dei vari middleware che utilizzate — può essere più o meno complesso aggiungere le metriche relative: nel senso che può essere necessario configurare dei *plugin*, come doverseli scrivere.

Dipende dal software in oggetto: tutti i servizi più diffusi sotto Linux sono di solito **ben supportati** (postgresql, mysql, apache, postfix, eccetera).

Enter *statsd*

POICHÉ COLLECTD RACCOGLIE SOLO LE METRICHE DI SISTEMA è necessario avere uno strumento diverso per raccogliere le metriche applicative. Statsd è un:

(...) network daemon that (...) listens for statistics, like counters and timers, sent over UDP and sends aggregates to one or more pluggable backend services (e.g., Graphite).

I concetti chiave di statsd

buckets È in pratica il nome, o etichetta, della misura. La convenzione è che il punto '.' separi i campi in modo da organizzarli ad albero, ad esempio:

- ▶ `collectd.server1.load`
- ▶ `collectd.server2.load`

sono due misure distinte, ma gli strumenti di navigazione mostreranno due children chiamati `server1` e `server2` sotto la radice `collectd`.

Ogni valore ha un modificatore.

Ogni valore ha un modificatore.

flush È l'intervallo di tempo dopo il quale le statistiche sono aggregate e mandate *upstream*.

I modificatori

Counting `libri:3|c` indica di **aggiungere 3** al contatore 'libri'. Al *flush* il valore corrente è spedito, e 'libri' viene posto a 0.

È uno dei contatori più semplici: conta quante volte accade un determinato evento nel periodo di campionamento.

E quindi l'unica operazione consentita è una **add** di un valore intero al valore precedente.

I modificatori

Counting `libri:3|c` indica di **aggiungere 3** al contatore 'libri'. Al *flush* il valore corrente è spedito, e 'libri' viene posto a 0.

È uno dei contatori più semplici: conta quante volte accade un determinato evento nel periodo di campionamento.

E quindi l'unica operazione consentita è una **add** di un valore intero al valore precedente.

I modificatori

Counting `libri:3|c` indica di **aggiungere 3** al contatore 'libri'. Al *flush* il valore corrente è spedito, e 'libri' viene posto a 0.

È uno dei contatori più semplici: conta quante volte accade un determinato evento nel periodo di campionamento.

E quindi l'unica operazione consentita è una **add** di un valore intero al valore precedente.

I modificatori

Timing **page:511|ms** indica che una certa azione è stata svolta in 511 (millisecondi); statsd in maniera automatica calcola, in base ai valori ricevuti nell'intervallo di campionamento:

1. i percentili,
2. la media,
3. la deviazione standard,
4. la somma,
5. gli estremi.

I modificatori

Gauges `accelerazione:3|g` è una misura del parametro indicato da 'accelerazione': se durante un intervallo di campionamento il valore non cambia, il *flush* spedisce il valore precedente (si noti la differenza dal counter).

I gauge consentono anche modifiche incrementali, come `accelerazione:+1|g` oppure `accelerazione:-2|g`.

I modificatori

Gauges `accelerazione:3|g` è una misura del parametro indicato da 'accelerazione': se durante un intervallo di campionamento il valore non cambia, il *flush* spedisce il valore precedente (si noti la differenza dal counter).
I gauge consentono anche modifiche incrementali, come `accelerazione:+1|g` oppure `accelerazione:-2|g`.

I modificatori

Sets `bag:34 | s` è un po' controintuitivo, in quando si aggiunge il valore '34' ad un insieme chiamato 'bag', e si ritorna la cardinalità di 'bag'.

Ergo, se si inserisce di nuovo il valore '34', il valore di 'bag' non cambia.

Un tipico caso d'uso è inserire le UID di oggetti come le login, per sapere in un determinato momento quante persone sono presenti sul server.

I modificatori

Sets `bag:34 | s` è un po' controintuitivo, in quando si aggiunge il valore '34' ad un insieme chiamato 'bag', e si ritorna la cardinalità di 'bag'.
Ergo, se si inserisce di nuovo il valore '34', il valore di 'bag' non cambia.

Un tipico caso d'uso è inserire le UID di oggetti come le login, per sapere in un determinato momento quante persone sono presenti sul server.

I modificatori

Sets `bag:34 | s` è un po' controintuitivo, in quando si aggiunge il valore '34' ad un insieme chiamato 'bag', e si ritorna la cardinalità di 'bag'.
Ergo, se si inserisce di nuovo il valore '34', il valore di 'bag' non cambia.
Un tipico caso d'uso è inserire le UID di oggetti come le login, per sapere in un determinato momento quante persone sono presenti sul server.

Un esempio d'uso di statsd

```
@stats.DebugTimer('export_riak_img')  
def export_img(img_binary, img_file_name):  
    """ Export image from riak and save as filename."""  
    with file(img_file_name, 'wb') as img_file_obj:  
        img_file_obj.write(img_binary)
```

Un semplice decoratore

```
class DebugTimer():
    """ Decorator for statsd.timing().
    If statsd is disabled the function will not be altered."""
    def __init__(self, stat_name):
        self.stat_name = stat_name
    def __call__(self, original_func):
        @wraps(original_func)
        def wrapped_f(*args, **kw):
            if STATSD_ENABLED:
                # MEASURE THE EXECUTION TIME
                with STATSD_HANDLER.timer(self.stat_name):
                    ret = original_func(*args, **kw)
            else:
                # CALL ONLY THE ORIGINAL FUNCTION
                ret = original_func(*args, **kw)
            return ret
        return wrapped_f
```


I pezzi che compongono Graphite

Ci sono tre componenti fondamentali, tutte scritte in Python:

carbon È il componente che eroga il servizio di collettore delle serie numeriche.

whisper È un'implementazione di un RRD, sul quale vengono rese persistenti le serie.

graphite webapp È il frontend del sistema, ovvero un'applicazione web che permette di visualizzare le serie.

Qualche parola in più per carbon

- ▶ Esistono tre modi di usare carbon: quello base prevede di lanciare il demone `carbon-cache.py` — è il sistema di default.
- ▶ Esistono però altre due modalità: `carbon-relay.py` e `carbon-aggregator.py` che soddisfano la necessità, rispettivamente, di fare replicazione (e *sharding*) e di fare *buffering*.

- ▶ Graphite è uno stack completo, contenente anche un sistema di visualizzazione (graphite-web); i dati di Influxdb devono essere visualizzati con uno strumento terzo.
- ▶ Influxdb permette di costruire un *cluster* vero e proprio, che usa Raft per gestire il consenso; Carbon da questo punto di vista è più semplice ma più primitivo.
- ▶ Influxdb è interrogabile con un linguaggio *simil-SQL*; i file Whisper sono leggibili molto semplicemente da un programma Python, ad esempio.

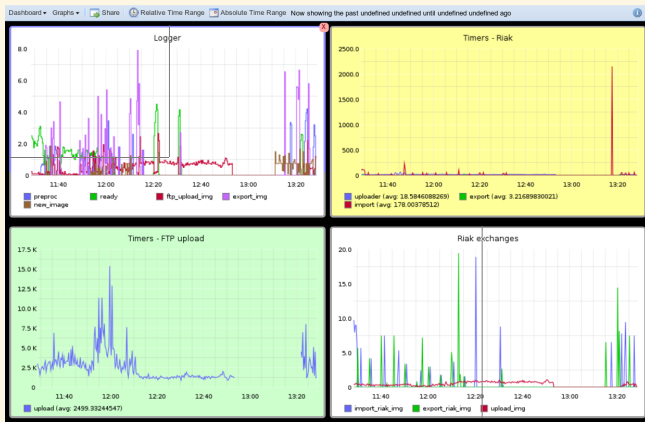
Differenze principali fra i due sistemi

- ▶ Graphite è uno stack completo, contenente anche un sistema di visualizzazione (graphite-web); i dati di Influxdb devono essere visualizzati con uno strumento terzo.
- ▶ Influxdb permette di costruire un *cluster* vero e proprio, che usa Raft per gestire il consenso; Carbon da questo punto di vista è più semplice ma più primitivo.
- ▶ Influxdb è interrogabile con un linguaggio *simil-SQL*; i file Whisper sono leggibili molto semplicemente da un programma Python, ad esempio.

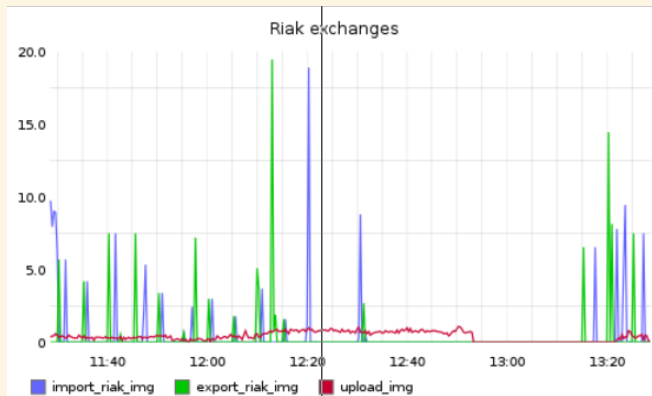
The leading graph and dashboard builder for visualizing time series metrics.

È UN'ALTERNATIVA (consigliata) all'uso di graphite-web. In particolare grafana può mostrare serie temporali che siano già presenti in graphite (e quindi permette, in un certo senso di usare graphite solo come *storage*), così come da altri *backend*.

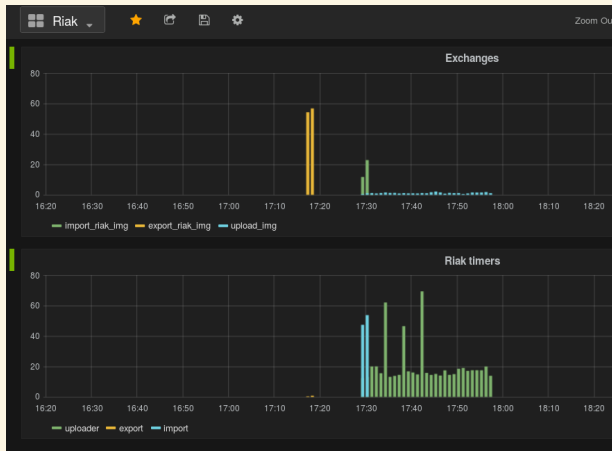
Graphite



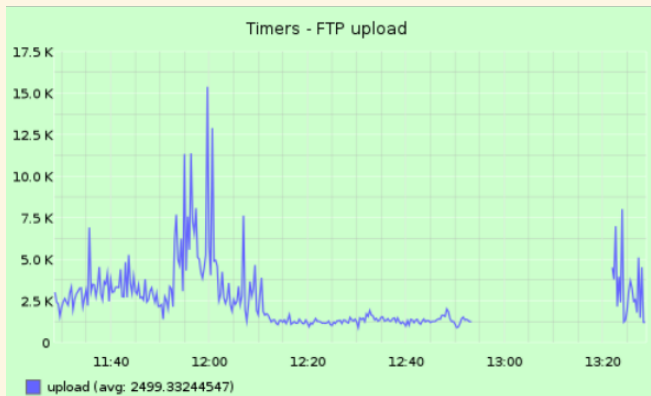
Graphite



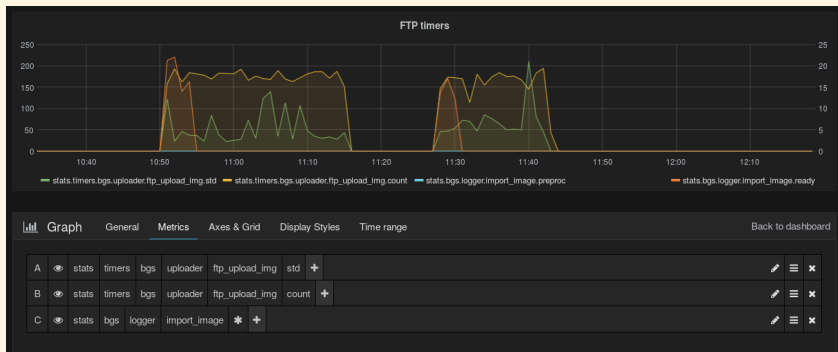
Grafana



Graphite

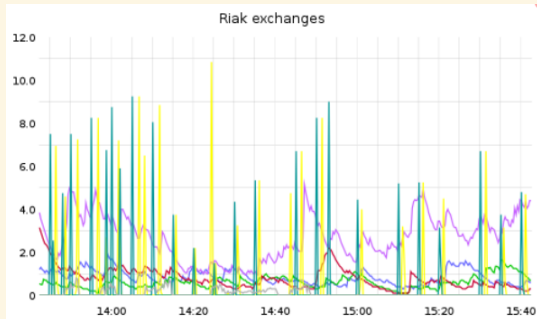


Grafana

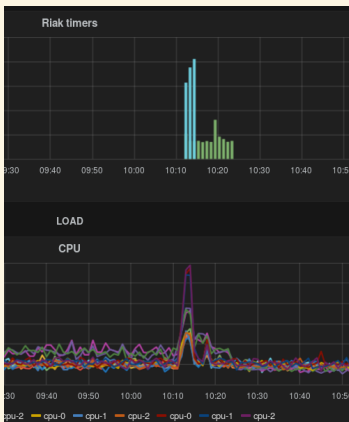


Integrazione di dati applicativi e di sistema

È lo stesso grafico di prima — un paio di ore dopo, per la precisione — ma col load di **ogni Riak server del cluster (5 server virtuali diversi)**.



Integrazione in Grafana



Le stesse informazioni (in un altro periodo) ma ancora più leggibili e navigabili, nonostante l'aumento del numero di metriche mostrate.

Cose che non funzionano come vorremmo

- ▶ Uno stack collectd / statsd / influxdb / grafana è una buona soluzione di monitoraggio ?
- ▶ In molti casi **si**.
- ▶ Di più, in parecchi casi **è una soluzione molto buona**.
- ▶ Ma ci sono delle cose che sono ancora problematiche (o incomplete).

Cose che non funzionano come vorremmo

- ▶ Uno stack collectd / statsd / influxdb / grafana è una buona soluzione di monitoraggio ?
- ▶ In molti casi **si**.
- ▶ Di più, in parecchi casi **è una soluzione molto buona**.
- ▶ Ma ci sono delle cose che sono ancora problematiche (o incomplete).

Il maledetto Docker

- Come si gestisce un'infrastruttura dove i nodi vanno e vengono ?
- Sarebbero necessarie delle funzionalità di *autodiscovery* che al momento **mancano**.
- In realtà non è per niente colpa di Docker: semplicemente l'uso dei *container* ha fatto esplodere un problema che già si presentava negli ambienti virtualizzati soggetti ad **elasticità**.

Gli allarmi

L'INTERAZIONE FRA IL SISTEMA DI MONITORAGGIO e il sistema di gestione degli allarmi è a dir poco “problematica”.

- ▶ In parte a causa della questione di un'infrastruttura elastica, per la quale non è facile dire se il fatto di avere un certo numero di servizi che rispondono sia uno stato corretto o di errore.
- ▶ In parte per il fatto che i sistemi attuali permettono sì di costruire dei *check* sulle metriche che vengono raccolte, ma questa operazione è sovente ...
- ▶ ... **un'operazione manuale.**

Gli allarmi


L'INTERAZIONE FRA IL SISTEMA DI MONITORAGGIO e il sistema di gestione degli allarmi è a dir poco “problematica”.

- In parte a causa della questione di un'infrastruttura elastica, per la quale non è facile dire se il fatto di avere un certo numero di servizi che rispondono sia uno stato corretto o di errore.
- In parte per il fatto che i sistemi attuali permettono sì di costruire dei *check* sulle metriche che vengono raccolte, ma questa operazione è sovente ...
- ... un'operazione manuale.

Gli allarmi

L'INTERAZIONE FRA IL SISTEMA DI MONITORAGGIO e il sistema di gestione degli allarmi è a dir poco “problematica”.

- In parte a causa della questione di un'infrastruttura elastica, per la quale non è facile dire se il fatto di avere un certo numero di servizi che rispondono sia uno stato corretto o di errore.
- In parte per il fatto che i sistemi attuali permettono sì di costruire dei *check* sulle metriche che vengono raccolte, ma questa operazione è sovente ...
- ... **un'operazione manuale.**

- 

Integrazione con i sistemi di alerting classici

- ▶ Con tutte le limitazioni del caso un software come Nagios, o Icinga o CheckMK o simili, viene tipicamente con una configurazione che senza troppa fatica permette di associare un certo numero di controlli ad un determinato server.
- ▶ Servirebbe un Nagios per Influxdb / Graphite. O meglio ancora un CheckMK Agent per le metriche.
- ▶ Al momento il massimo che si ottiene è usare Grafana come alternativa a PNP4Nagios.
- ▶ Meglio che niente, ma non cambia radicalmente la questione.

Integrazione con i sistemi di alerting classici

- Con tutte le limitazioni del caso un software come Nagios, o Icinga o CheckMK o simili, viene tipicamente con una configurazione che senza troppa fatica permette di associare un certo numero di controlli ad un determinato server.
- Servirebbe un Nagios per Influxdb / Graphite. O meglio ancora un CheckMK Agent per le metriche.
- Al momento il massimo che si ottiene è usare Grafana come alternativa a PNP4Nagios.
- Meglio che niente, ma non cambia radicalmente la questione.

- Con tutte le limitazioni del caso un software come Nagios, o Icinga o CheckMK o simili, viene tipicamente con una configurazione che senza troppa fatica permette di associare un certo numero di controlli ad un determinato server.
- Servirebbe un Nagios per Influxdb / Graphite. O meglio ancora un CheckMK Agent per le metriche.
- Al momento il massimo che si ottiene è usare Grafana come alternativa a PNP4Nagios.
- Meglio che niente, ma non cambia radicalmente la questione.

◀ ◻ ▶ ◀ ◻ ▶ ◀ ≡ ▶ ◀ ≡ ▶ ≡ ↺ 🔍 ↻

Domande



*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>

Grazie dell'attenzione !

*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>

Grazie dell'attenzione !

IDI2016 Incontro DevOps Italia 2016, 1 Aprile 2016 a.k.a.
“Non è uno scherzo”, a Bologna.

*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>

Grazie dell'attenzione !

IDI2016 Incontro DevOps Italia 2016, 1 Aprile 2016 a.k.a.
“Non è uno scherzo”, a Bologna.

WWW <http://www.incontrodevops.it/>

*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>

Grazie dell'attenzione !

IDI2016 Incontro DevOps Italia 2016, 1 Aprile 2016 a.k.a.
“Non è uno scherzo”, a Bologna.

WWW <http://www.incontrodevops.it/>

Twitter @incontrodevops

*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>

Thanks & see you soon ...

Grazie dell'attenzione !

IDI2016 Incontro DevOps Italia 2016, 1 Aprile 2016 a.k.a.
“Non è uno scherzo”, a Bologna.

WWW <http://www.incontrodevops.it/>

Twitter [@incontrodevops](https://twitter.com/incontrodevops)

Me m@biodec.com [@gaunilone](https://plus.google.com/+Gaunilone)

*licenza della presentazione:

<http://creativecommons.org/licenses/by-sa/4.0/>