The life and times of

PuppetDB

Clojure/West 2013

deepak giridharagopal

deepak@puppetlabs.com
@grim_radical [github twitter freenode]

Let’s talk about…

Sorry

NO

CHANGE

Immutability is great!

Immutability allows for invariants, which help you reason about correctness

Immutability prevents spooky action at a distance

Immutability fosters modular, composable abstractions

(not a tough sell to Clojurists)*

**clojurians?*

That’s great for develoment, but how about operations?

Immutability for infrastructure?

Because operations is in the same boat as development

Everyone who's got their app running on a fleet of servers has experienced spooky action at a distance

Known, good state is critical for reliable upgrades

A lack of predictability in your systems ruins automation and abstraction

The problem is that:

Systems are inherently mutable!

But ideally:

Systems should behave as though they weren't!

façade of immutability

Computer systems are in many ways open systems, providing the keys to the vault if one is so inclined to grab them. But in order to foster an air of immutability in our own systems, it's of utmost importance to create a façade of immutability. Immutability requires that we layer over and abstract the parts of our system that provide unrestrained mutability.

-- The Joy of Clojure


puppet
labs®

Describe how you'd like your systems to look, and Puppet does all the hard work for you!

```
file { “/etc/issue”:
  content => “Got an issue? Here’s a tissue!”,
}
```

```
file { “/etc/motd”:
  content => template(“Welcome to $hostname!”),
}
```

```
file { "/etc/sudoers":
  owner  => root,
  group  => root,
  mode   => 440,
  source => "puppet:///modules/sudo/sudoers"
}
```

```
package { 'ntp':
  ensure => installed,
}

service { 'ntpd':
  ensure    => running,
  enable    => true,
  subscribe => File['/etc/ntp.conf'],
}

file { '/etc/ntp.conf':
  ensure  => file,
  require => Package['ntp'],
  source  => "puppet:///modules/ntp/ntp.conf",
}
```

```
class ntp {

    package { 'ntp':
      ensure => installed,
    }

    service { 'ntpd':
      ensure     => running,
      enable     => true,
      subscribe => File['/etc/ntp.conf'],
    }

    file { '/etc/ntp.conf':
      ensure  => file,
      require => Package['ntp'],
      source  => "puppet:///modules/ntp/ntp.conf",
    }

}
```

```
node “webserver.mydomain.com” {
   include ntp
}

node “appserver.mydomain.com” {
   include ntp
}

node “database.mydomain.com” {
   include ntp
}
```

```
class ssh {

    @@sshkey { $hostname:
      type => dsa,
      key => $sshdsakey
    }

    Sshkey <<|  |>>

}
```

File “/tmp/foo/bar”
User “deepak”
Dir “/tmp/foo”
Dir “/tmp”

Dir “/tmp”

User “deepak”

Dir “/tmp/foo”

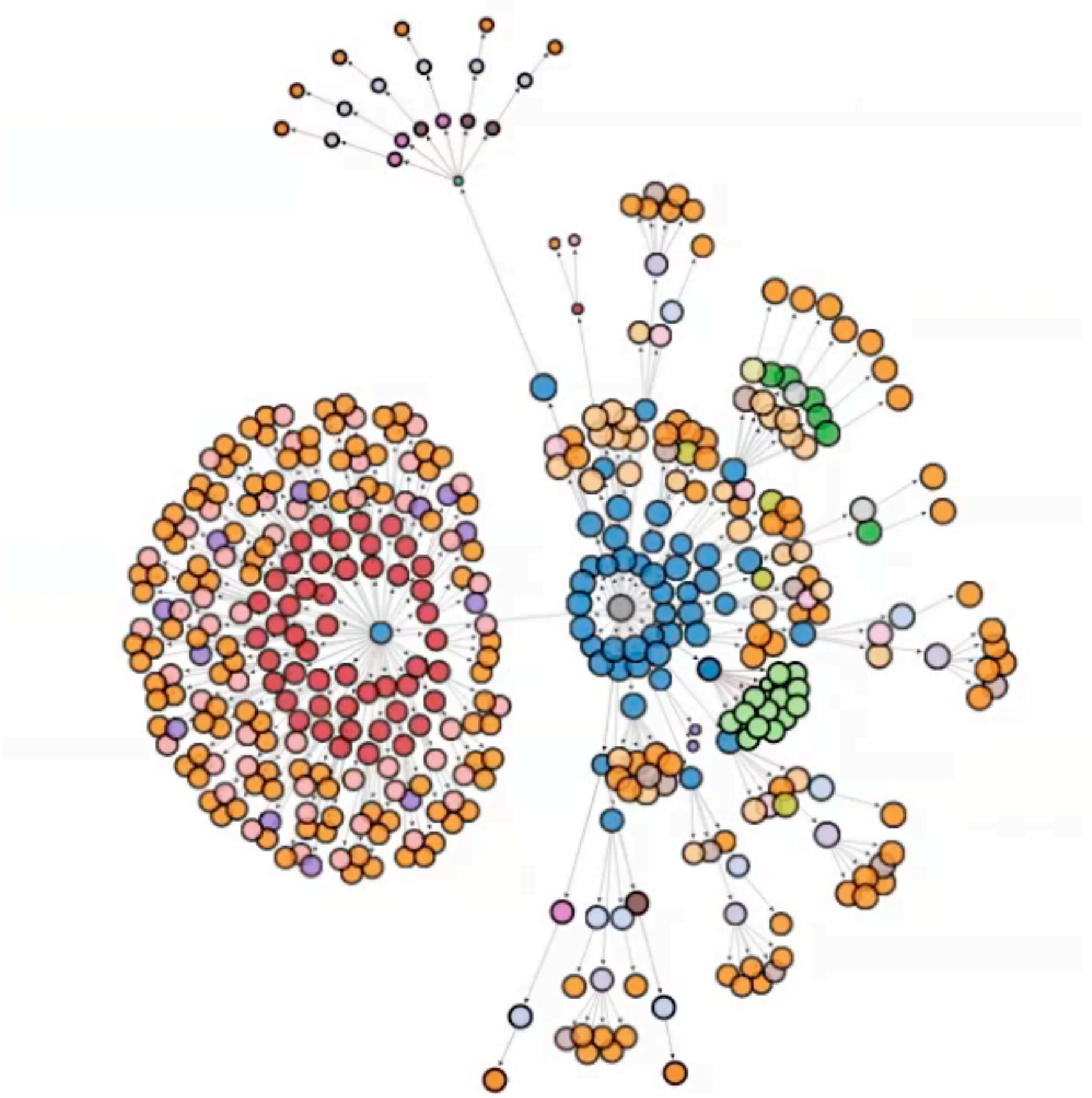
File “/tmp/foo/bar”

Dir “/tmp”

User “deepak”

Dir “/tmp/foo”

File “/tmp/foo/bar”



Idempotent, and only does what's necessary

Compensates for the inherent mutability of systems

Combats spooky action at a distance with automatic repair

Brings predictability to your systems

A foundation of predictability and reliability lets you perform higher-level operations on your infrastructure

VOLUME

every resource
every parameter
every relationship
every fact
for every node
updated all the time

Users leverage this data to do higher-order things with their infrastructure

key distribution
monitoring
clustered services
master/slave replication
load balancers
shared filesystems
firewall rules
...

Infrastructure as code

Infrastructure as data

User demand:

Store as much data as we can!
Much better queryability!

Oh yeah, but:

Don’t slow down the system!
Don’t compromise reliability!

We can rebuild it, we have the technology!

Speed is important

Parsing, validating, and manipulating incoming data is computationally expensive

Speed is important

The slower central storage is, the less agile sysadmins can be. That can cost money and uptime!

Reliability is important

If it's a critical part of managing infrastructure, it's got to be solid!

Deployment is important

Should be easy to package, install, configure, use, monitor, and upgrade.

Wait, isn't Puppet written in Ruby?

Lessons learned from writing the rest of our software in Ruby

It's...not speedy. Object creation, method calls, garbage collection, etc. "Magical" APIs amplify the problem.

Lessons learned from writing the rest of our software in Ruby

Can only use one core!
Workarounds compromise performance or simplicity

Lessons learned from writing the rest of our software in Ruby

Mutable state all over the damn place!

Lessons learned from writing the rest of our software in Ruby

Many struggles with the runtime. :(

-- Jeff Gagliardi

1. It’s fast

2. JVM libraries & Tools

3. State & Parallelism

PuppetDB

PuppetDB

Definitely Better!

Fast, safe storage of catalogs, facts, and events

*like, *way* faster!*

HTTP APIs

for resource, fact, node, report retrieval

plenty of data, just a "curl" away!

Storage & Querying

Storage & Querying

CORS

Command
Query
Responsibility
Separation

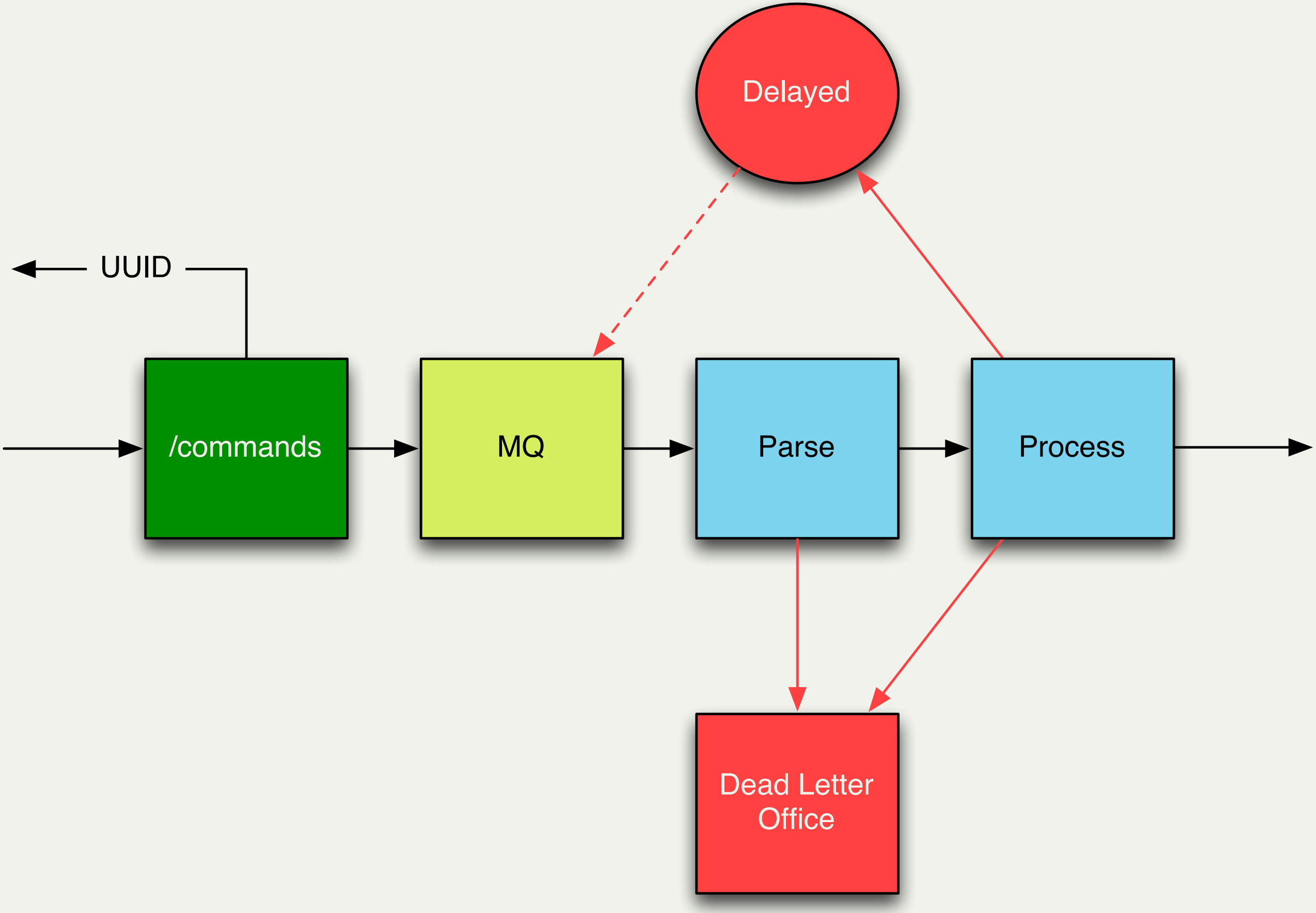
use a different model to update information than the model you use to read information

Writes

CQRS
write pipeline

async, parallel, MQ-based, with automatic retry

```
{
 :command "replace catalog"
 :version 2
 :payload {...}
}
```


Delayed
UUID
/commands
MQ
Parse
Process
Dead Letter
Office

```clojure
(defmulti process-command!
  (fn [{:keys [command version] :or {version 1}} _]
    [command version]))

(defmethod process-command! ["replace catalog" 1]
  [command options]
  (replace-catalog* command options))
```

```clojure
(defmulti process-command!
  (fn [{:keys [command version] :or {version 1}} _]
    [command version]))

(defmethod process-command! ["replace catalog" 2]
  [command options]
  (replace-catalog* command options))

(defmethod process-command! ["replace catalog" 1]
[command options]
(-> command
    (update-in [:payload] catalog-v1->v2)
    (replace-catalog* options))
```

Command processors must be retry-aware

*expect failure, because it *will* happen.*

(demo)

Reads

Queries
are expressed in their own “language”

domain specific, AST-based query language

```
["and",
["=", "type", "User"],
["=", "title", "nick"]]
```

```
["and",
["=", ["fact", "operatingsystem"], "Debian"],
["<", ["fact", "uptime_seconds"], 10000]]
```

```
["and",
  ["=", "name", "ipaddress"],
  ["in", "certname",
   ["extract", "certname", ["select-resources",
                            ["and",
                              ["=", "type", "Class"],
                              ["=", "title", "Apache"]]]]
```

```
["or",
  ["=", "certname", "foo.com"],
  ["=", "certname", "bar.com"],
  ["=", "certname", "baz.com"]]
```

We use core.match to walk the tree, compiling it to SQL

AST-based API lets users write their own languages

ah, you've got to love open source!

```
(Package[httpd] and country=fr)
or country=us


Package["mysql-server"]
and architecture=amd64
```


Erik Dalén, Spotify

https://github.com/dalen/puppet-puppetdbquery

AST-based API lets us more safely manipulate queries

```clojure
(def query-app
  (app
    [&]
    {:get (fn [{:keys [params] :as request}]
            (perform-query (params "query")))}))

(def resources-app
  (app
   []
   query-app

   [type title &]
   (comp query-app
         (partial http-q/restrict-resource-query-to-type type)
         (partial http-q/restrict-resource-query-to-title title))

   [type &]
   (comp query-app
         (partial http-q/restrict-resource-query-to-type type))))
```

```clojure
(defn restrict-resource-query-to-type
  [type query]
  (conj ["and"
         ["=" "type" type]]
        query))
```

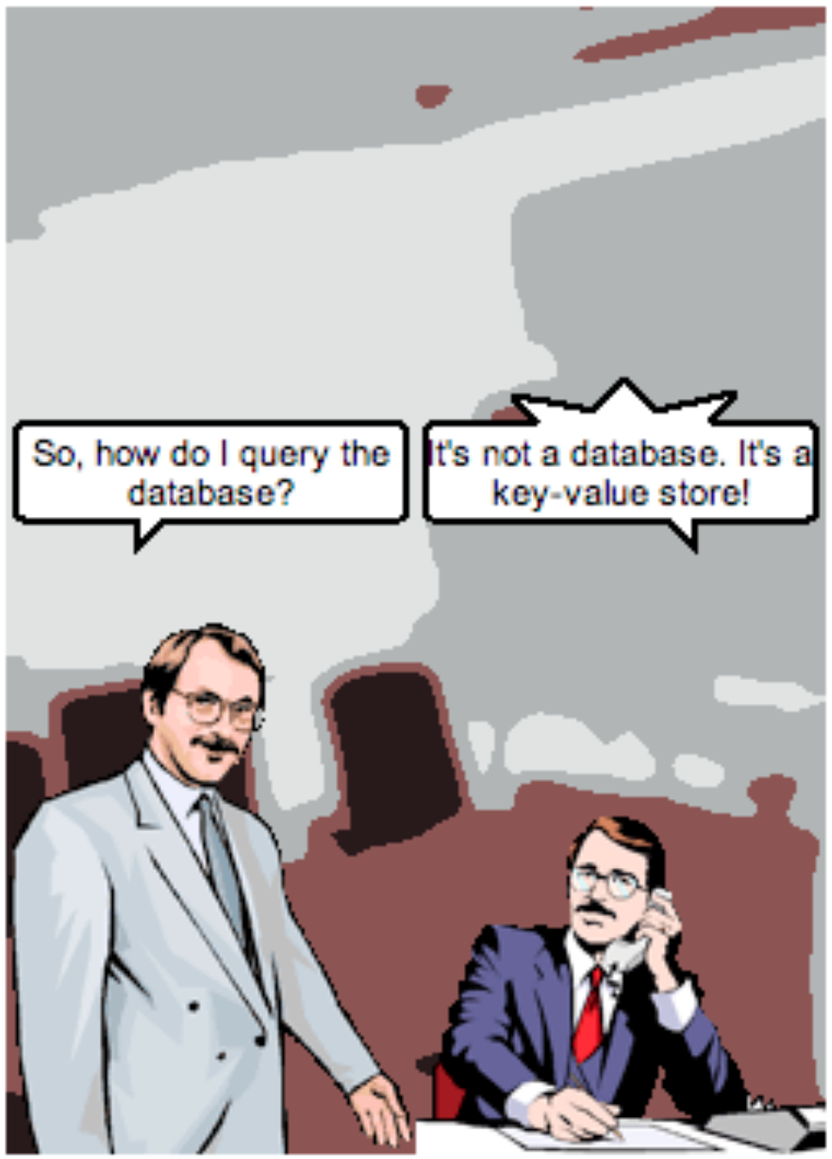
Storage

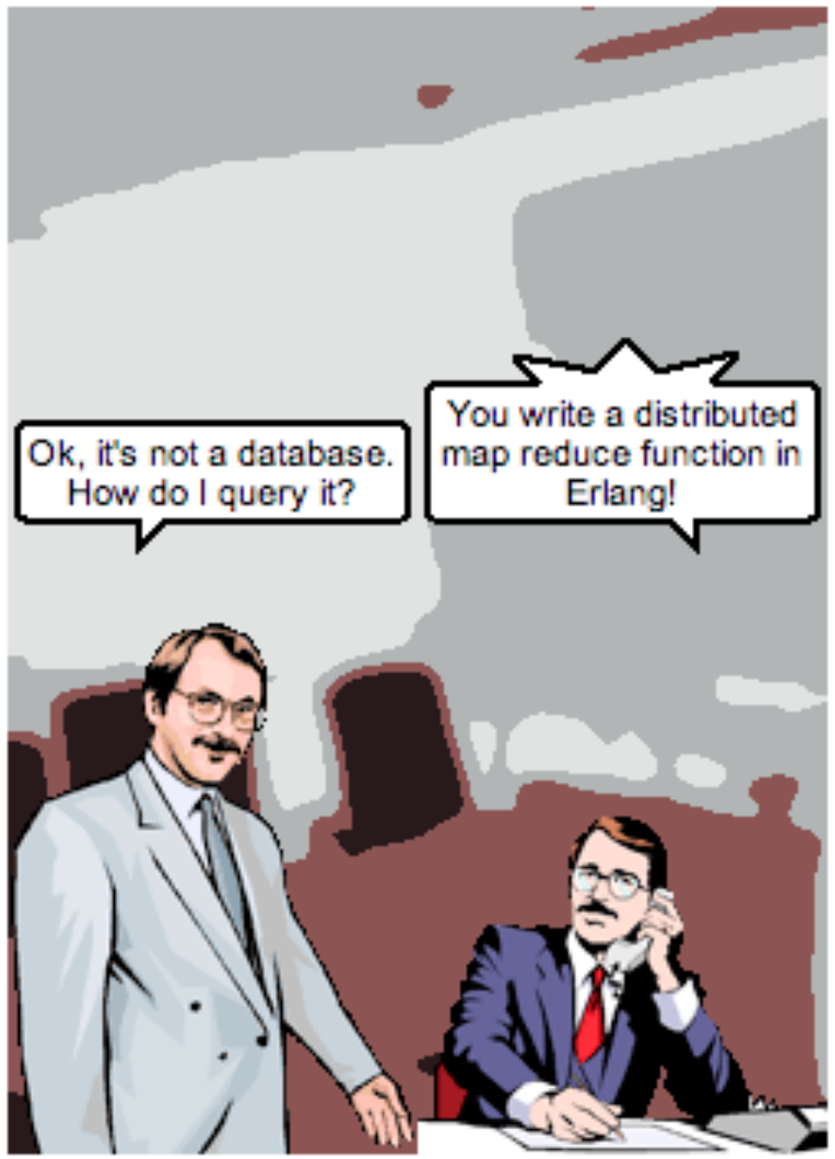
Relational Database, embedded or PostgreSQL

because they're actually pretty fantastic at ad-hoc queries, aggregation, windowing, etc. while maintaining safety

Fault-tolerance

by @jrecursive





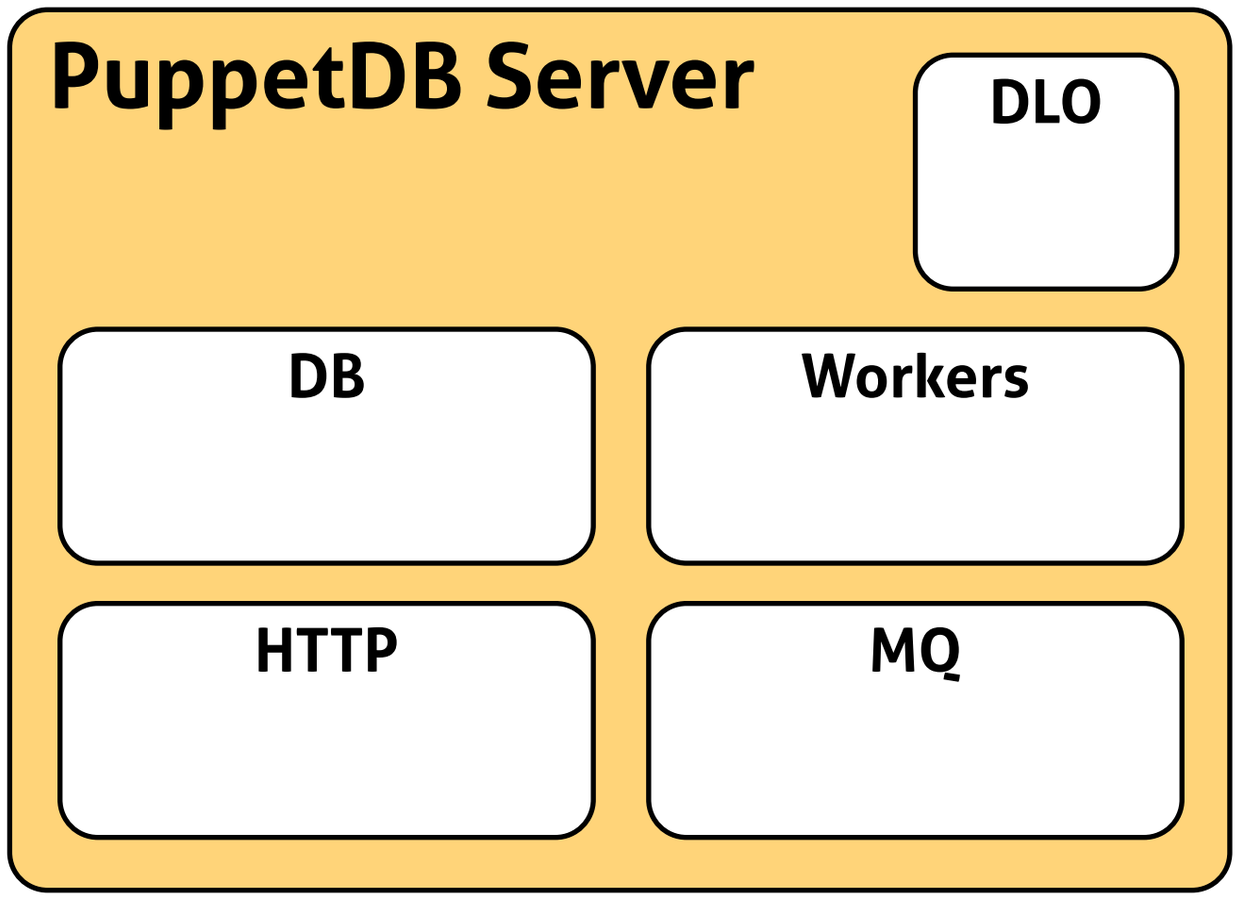
Relational Database, embedded or PostgreSQL

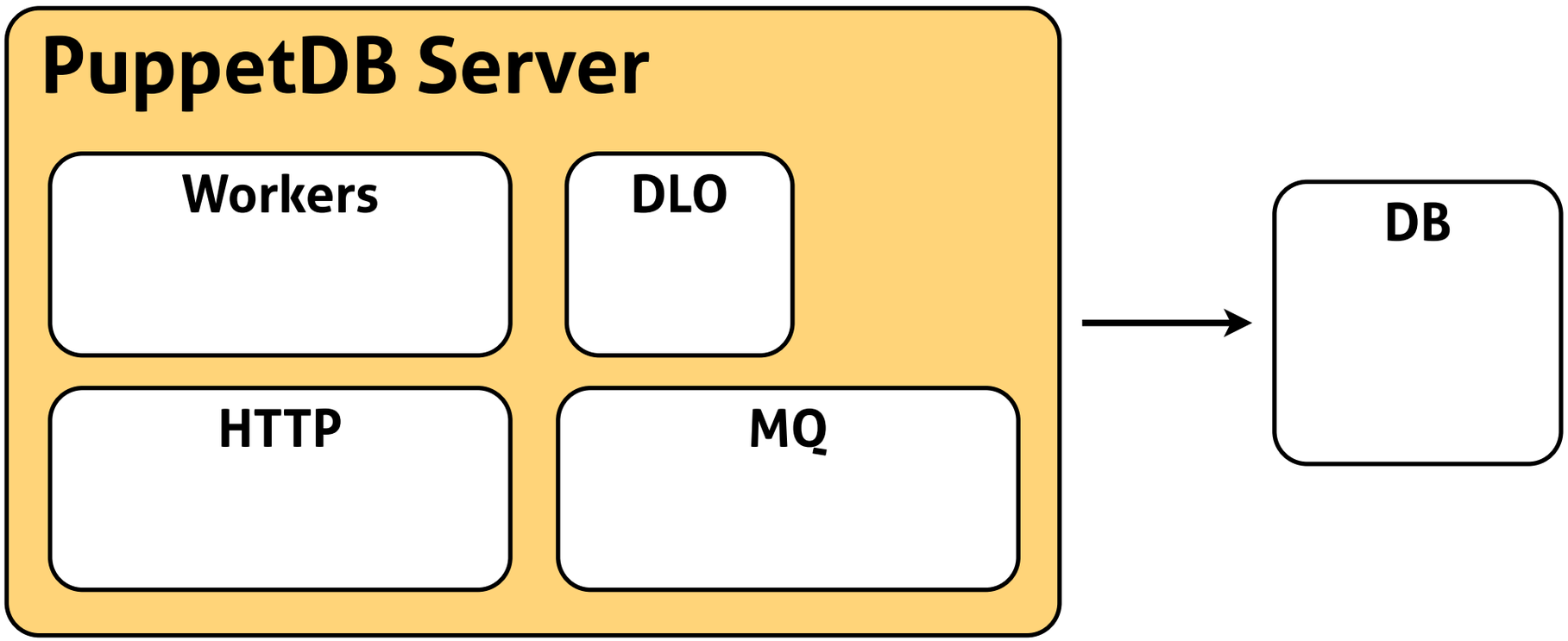
we use arrays, recursive queries, indexing inside complex structures

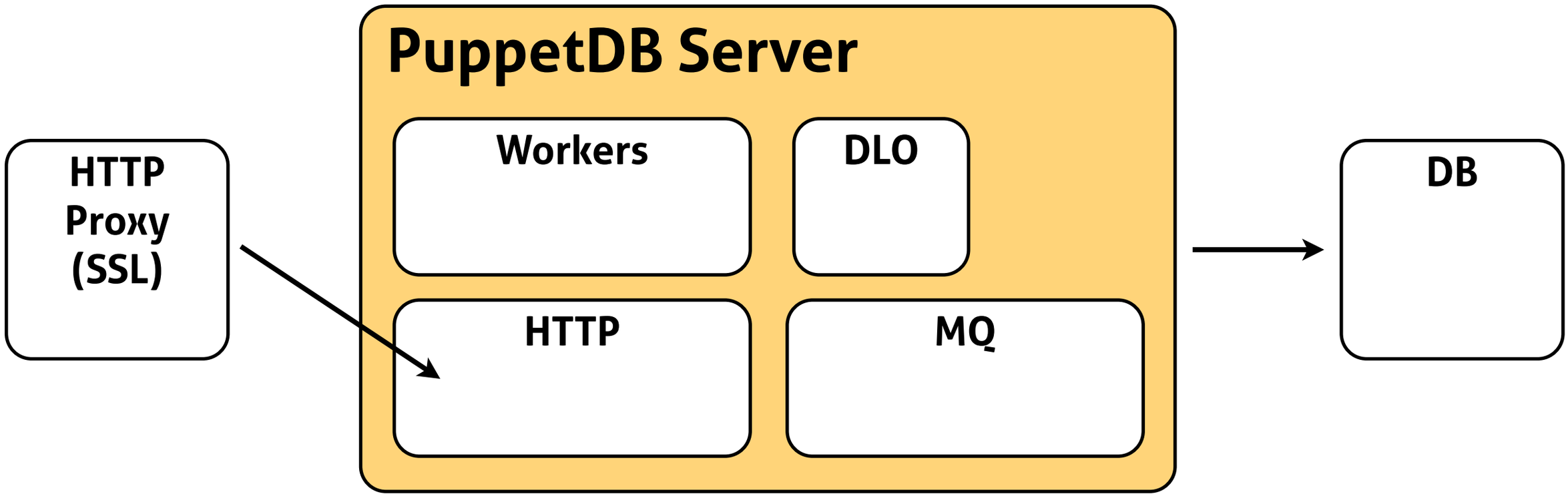
Relational Database, embedded or PostgreSQL

schema is oriented towards the types of queries we encounter

Deployment


PuppetDB Server
DLO
DB
Workers
HTTP
MQ


PuppetDB Server
Workers
DLO
HTTP
MQ
DB


PuppetDB Server
HTTP Proxy (SSL)
Workers
DLO
HTTP
MQ
DB

Codez

```clojure
(defn process-commands!
  "Connect to an MQ an continually, sequentially process commands.

  If the MQ consumption timeout is reached without any new data, the
  function will terminate."
  [connection endpoint discarded-dir options-map]
  (let [producer   (mq-conn/producer connection)
        publish    (partial mq-producer/publish producer endpoint)
        on-message (produce-message-handler publish discarded-dir options-map)
        mq-error   (promise)
        consumer   (mq-conn/consumer connection
                                     {:endpoint   endpoint
                                      :on-message on-message
                                      :transacted true
                                      :on-failure #(deliver mq-error (:exception %))})]
    (mq-cons/start consumer)

    ;; Block until an exception is thrown by the consumer thread
    (try
      (deref mq-error)
      (throw @mq-error)
      (finally
        (mq-cons/close consumer)))))
```

```clojure
(defmacro with-error-delivery
  "Executes body, and delivers an exception to the provided promise if one is
  thrown."
  [error & body]
  `(try
     ~@body
     (catch Exception e#
       (deliver ~error e#))))
```

```clojure
(defn filter-mbean
  "Converts an mbean to a map. For attributes that can't be converted to JSON,
  return a string representation of the value."
  [mbean]
  {:post [(map? %)]}
  (into {} (for [[k v] mbean]
             (cond
              ;; Nested structures should themselves be filtered
              (map? v)
              [k (filter-mbean v)]

              ;; Cheshire can serialize to JSON anything that
              ;; implements the JSONable protocol
              (satisfies? JSONable v)
              [k v]

              :else
              [k (str v)]))))
```

```clojure
(defmacro with-test-broker
  "Constructs and starts an embedded MQ, and evaluates `body` inside a
  `with-open` expression that takes care of connection cleanup and MQ
  tear-down.

  `name` - The name to use for the embedded MQ

  `conn-var` - Inside of `body`, the variable named `conn-var`
  contains an active connection to the embedded broker.

  Example:

      (with-test-broker \"my-broker\" the-connetion
        ;; Do something with the connection
        (prn the-connection))
  "
  [name conn-var & body]
  `(let [dir#                    (fs/absolute-path (fs/temp-dir))
         broker-name#            ~name
         conn-str#               (str "vm://" ~name)
         ^BrokerService broker# (mq/build-embedded-broker broker-name# dir#)]

     (.setUseJmx broker# false)
     (.setPersistent broker# false)
     (mq/start-broker! broker#)

     (try
       (with-open [~conn-var (mq/connect! conn-str#)]
         ~@body)
       (finally
         (mq/stop-broker! broker#)
         (fs/delete-dir dir#)))))
```

Results

Thousands of production deployments

Small shops with a dozen hosts, large shops with thousands of hosts, intercontinental deployments...

Dean Wilson @unixdaemon 7h

Did some basic performance testing of #**puppetdb** and reduced our compile times from 180ish second to mid 20s. #massivewin

Expand

Greg Mason @greg_mason 25 Sep

according to #**puppetdb**, I'm currently managing 71,777 resources with Puppet.

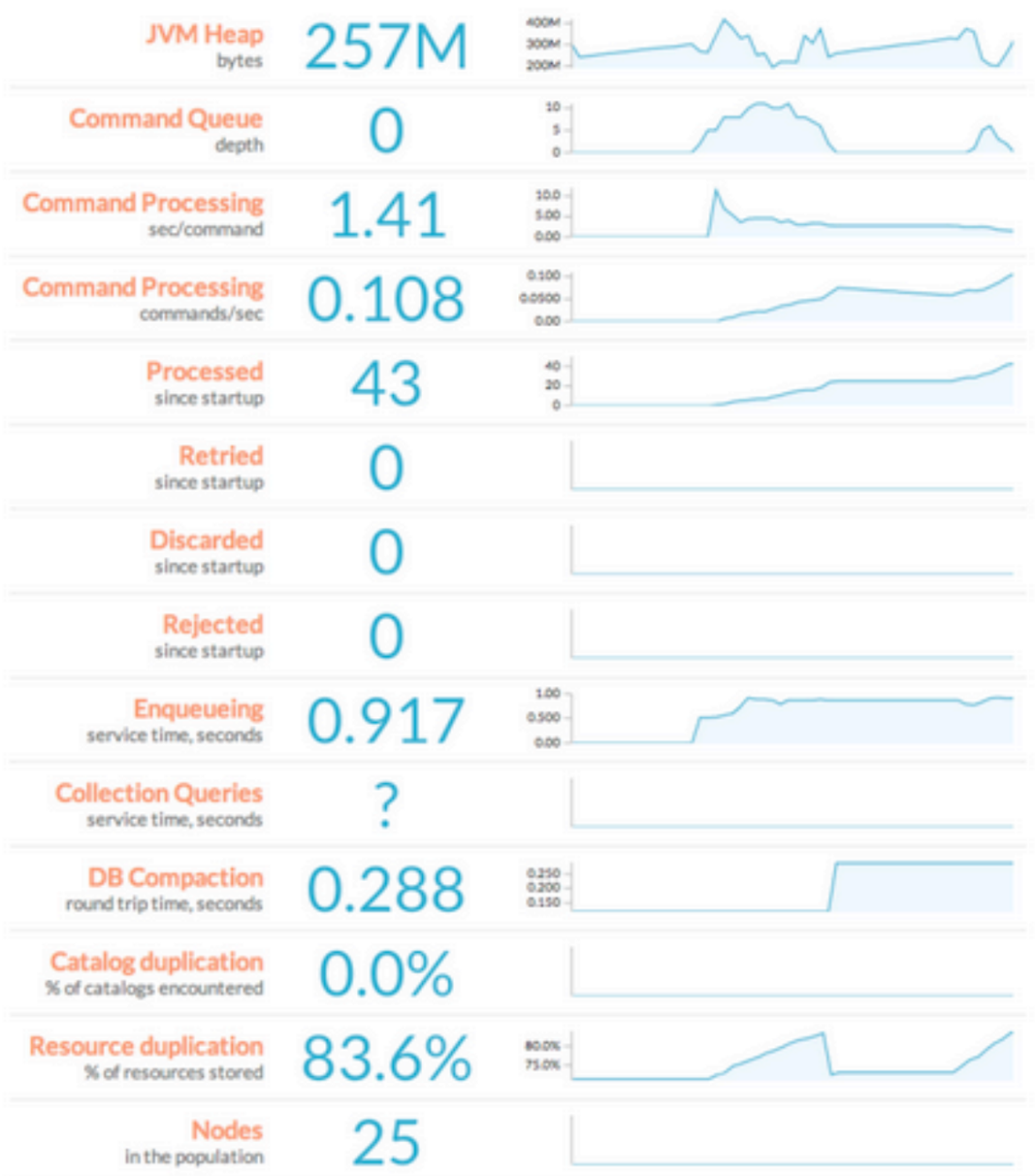
Expand

ripienaar @ripienaar 13 Jun

puppetdb + 25 masterless nodes with embedded database twitpic.com/9vz7y2

Retweeted by D Giridharagopal

Hide photo Reply Retweeted Favorite



Matt Nicholson @sjoeboo 20 Jul

@glarizza @grim_radical two pic.twitter.com/mvJuOiJF

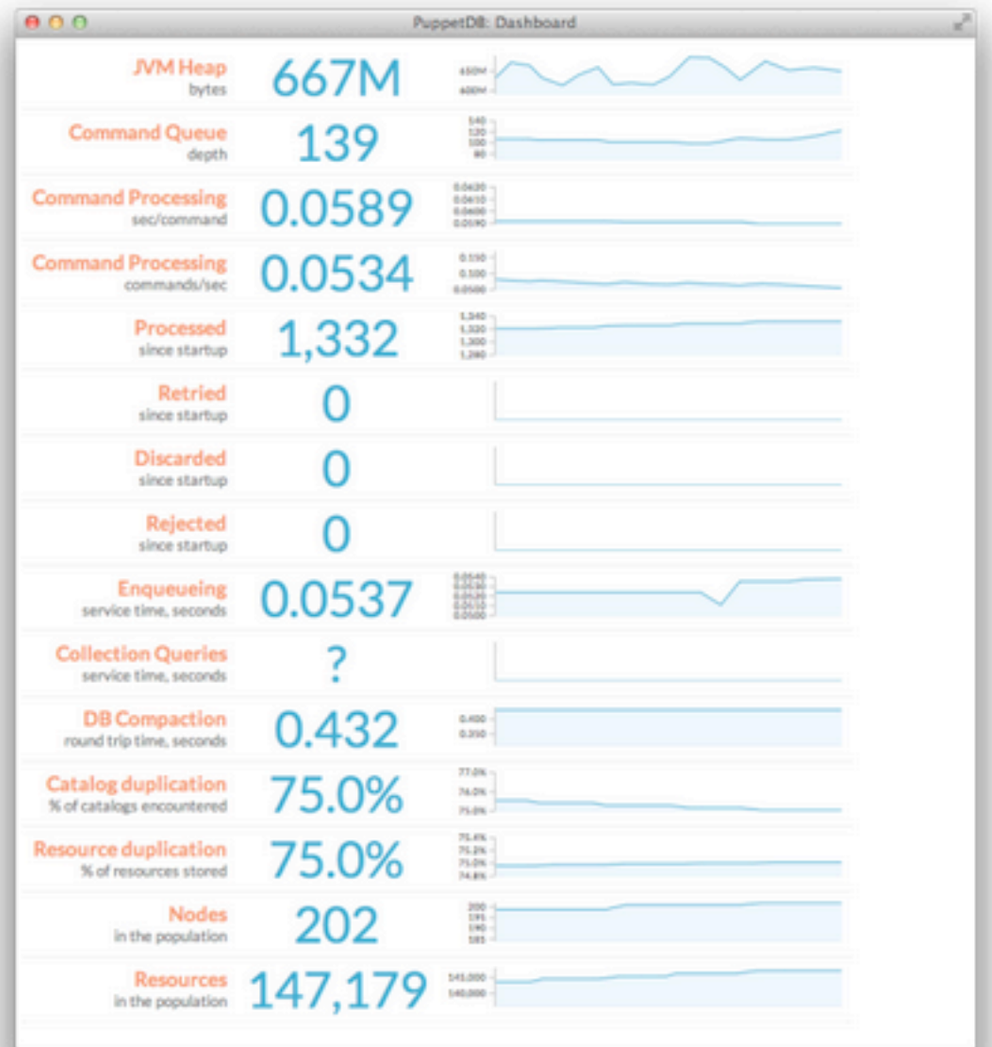
Hide photo

Lars Fronius @LarsFronius 23 May

This #puppetdb dashboard thingy is actually so awesome. Oh, and puppetdb is such a big step in performance! #puppetize twitpic.com/9oi290

Retweeted by D Giridharagopal

Hide photo Reply Retweeted Favorite



Thousands of deployments,
Hundreds of threads per install,
Zero deadlocks,
Zero bugs involving clojure state

companion Ruby code has
~10x the defect rate

Users dig the uberjar, makes deployment straightforward and less error-prone

Nobody cares that it's Clojure. Users just want stuff to work.

Monitoring and instrumentation is a big deal. Users want easy ways to consume metrics.

If you build it, they will come.

Open source

http://github.com/puppetlabs/puppetdb

deepak giridharagopal

deepak@puppetlabs.com
@grim_radical [github twitter freenode]

We're hiring!