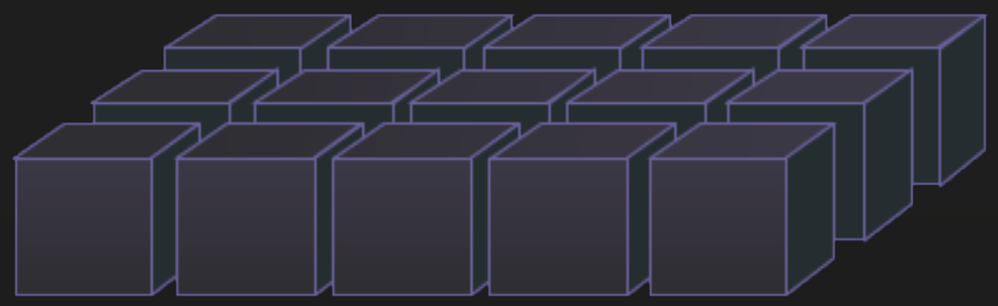
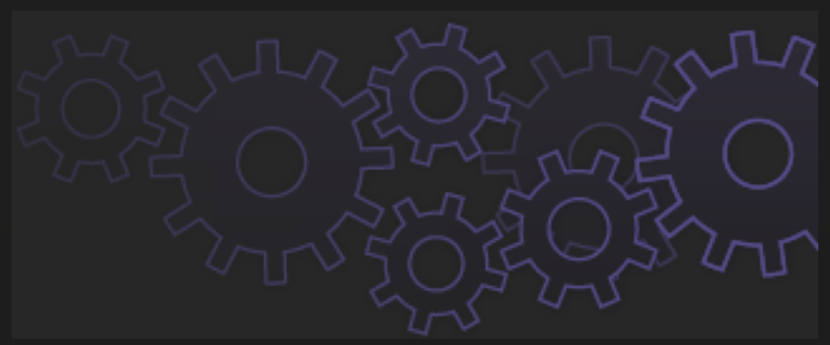


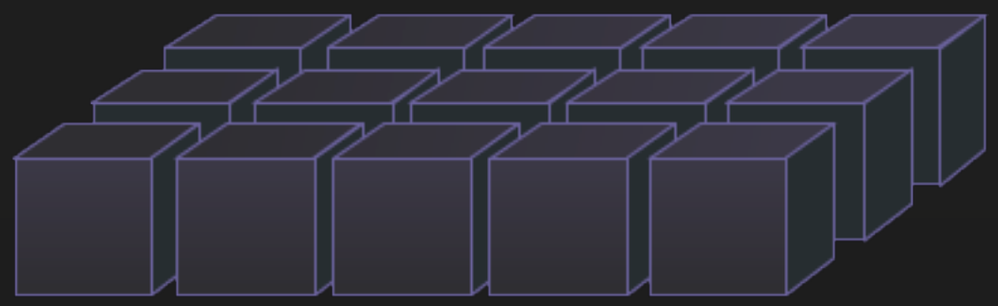
Modern Architecture

Today I am talking about building apps for the cloud.



The Worker Pattern

But wait, weren’t you supposed to talk about the worker pattern? Yes. But I am not going to do that now.



Modern Architecture

To that end, I am talking about building apps for the cloud. Some of the points I will make should be obvious in hindsight. However, some points I will make have not yet proven to be true so look out that!

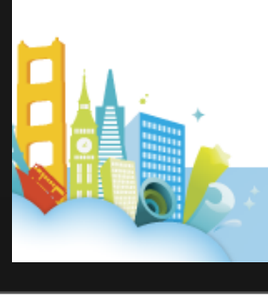
Safe Harbor

Safe harbor statement under the Private Securities Litigation Reform Act of 1995:

This presentation may contain forward-looking statements that involve risks, uncertainties, and assumptions. If any such uncertainties materialize or if any of the assumptions proves incorrect, the results of salesforce.com, inc. could differ materially from the results expressed or implied by the forward-looking statements we make. All statements other than statements of historical fact could be deemed forward-looking, including any projections of product or service availability, subscriber growth, earnings, revenues, or other financial items and any statements regarding strategies or plans of management for future operations, statements of belief, any statements concerning new, planned, or upgraded services or technology developments and customer contracts or use of our services.

The risks and uncertainties referred to above include – but are not limited to – risks associated with developing and delivering new functionality for our service, new products and services, our new business model, our past operating losses, possible fluctuations in our operating results and rate of growth, interruptions or delays in our Web hosting, breach of our security measures, the outcome of intellectual property and other litigation, risks associated with possible mergers and acquisitions, the immature market in which we operate, our relatively limited operating history, our ability to expand, retain, and motivate our employees and manage our growth, new releases of our service and successful customer deployment, our limited history reselling non-salesforce.com products, and utilization and selling to larger enterprise customers. Further information on potential factors that could affect the financial results of salesforce.com, inc. is included in our annual report on Form 10-Q for the most recent fiscal quarter ended April 30, 2011. This documents and others containing important disclosures are available on the SEC Filings section of the Investor Information section of our Web site.

Any unreleased services or features referenced in this or other presentations, press releases or public statements are not currently available and may not be delivered on time or at all. Customers who purchase our services should make the purchase decisions based upon features that are currently available. Salesforce.com, inc. assumes no obligation and does not intend to update these forward-looking statements.

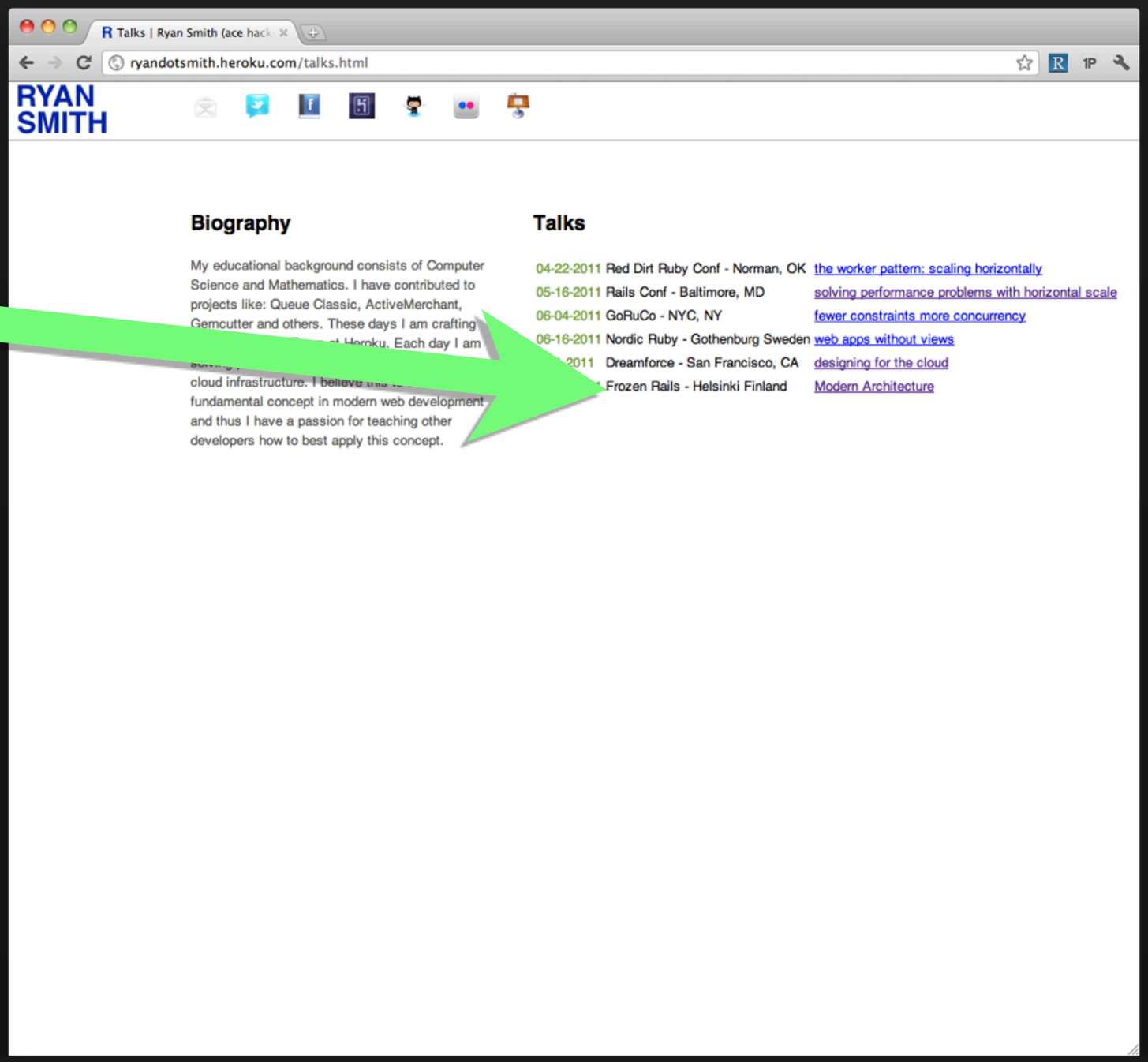


Ryan Smith

@ryandotsmith

My name is Ryan Smith, you can find me on twitter via @ryandotsmith.

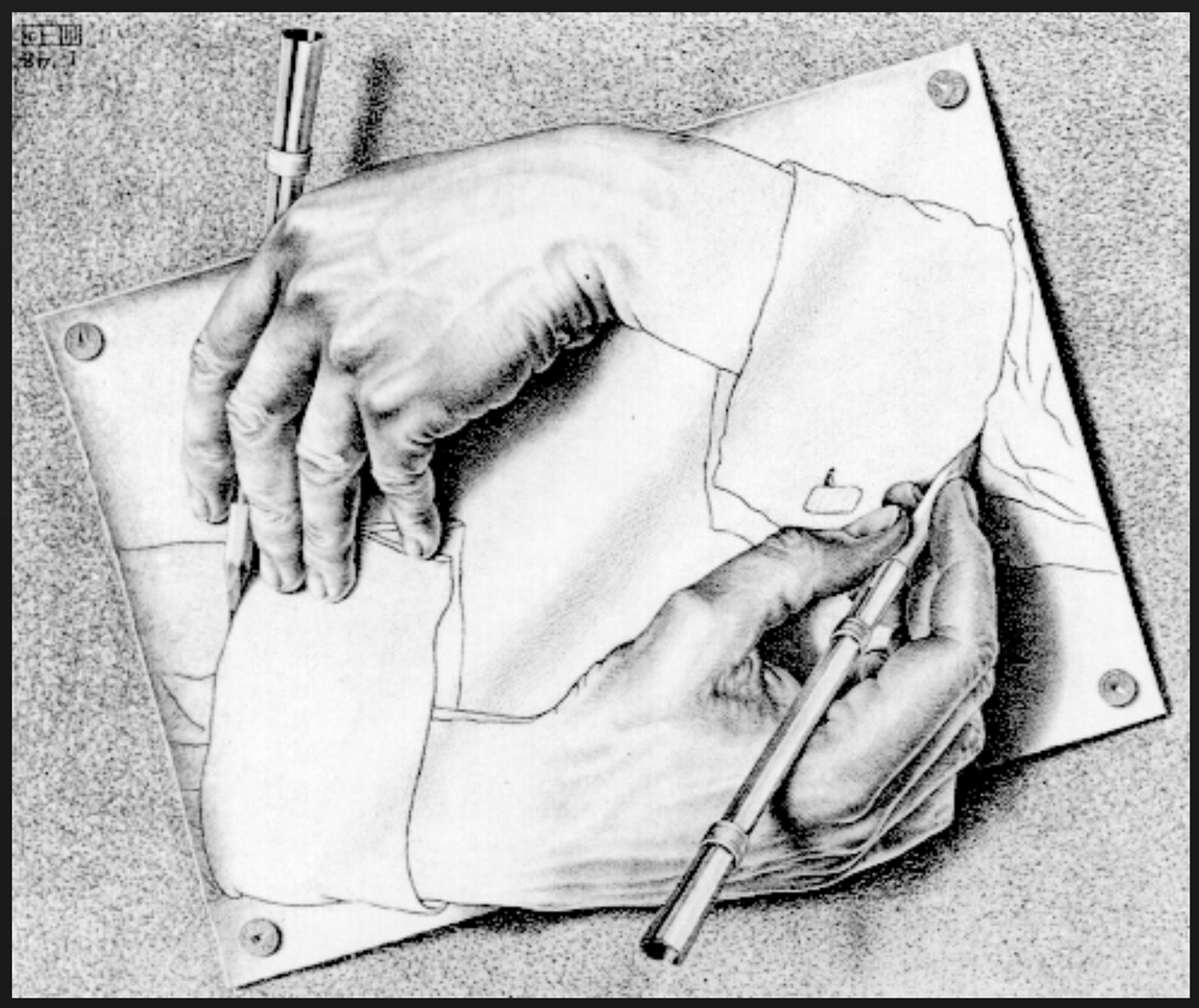
ryandotsmith.heroku.com



I am an engineer at Heroku.

The Vault

I work in The Vault. The Vault is a team within Heroku that is responsible for innovating the usage & billing components of our business. We track all of the events in our cloud an figure out how to make an invoice out of the events. We do all of this in realtime and with large amounts of data.



One interesting note. All of The Vault’s components are hosted on Heorku. We have been doing all sorts of strange things related to self hosting these days. Personally, I love the idea. Isn’t self hosting a mind bend? I mean the wrote the C compiler in C.

1

You would think that such an operation requires a large engineering team. However, we get by with a team of 1.

Designed for the Cloud

Our team's size would not be feasible if we maintained our own platform. Knowing this, The Vault's apps were optimized for a cloud based deployment and therein lies the core ideas of this case study. I want to talk about the engineering practices that allowed us a cloud deployment.

Goals

{ engineering concepts }

concept => implementation

My goal for this talk is to present a set of engineering concepts that we have maintained in The Vault that have directly led to successful development. I will introduce the concept then follow it up by providing example of how we implement the concept. Lets get started with the first concept.

UNIX

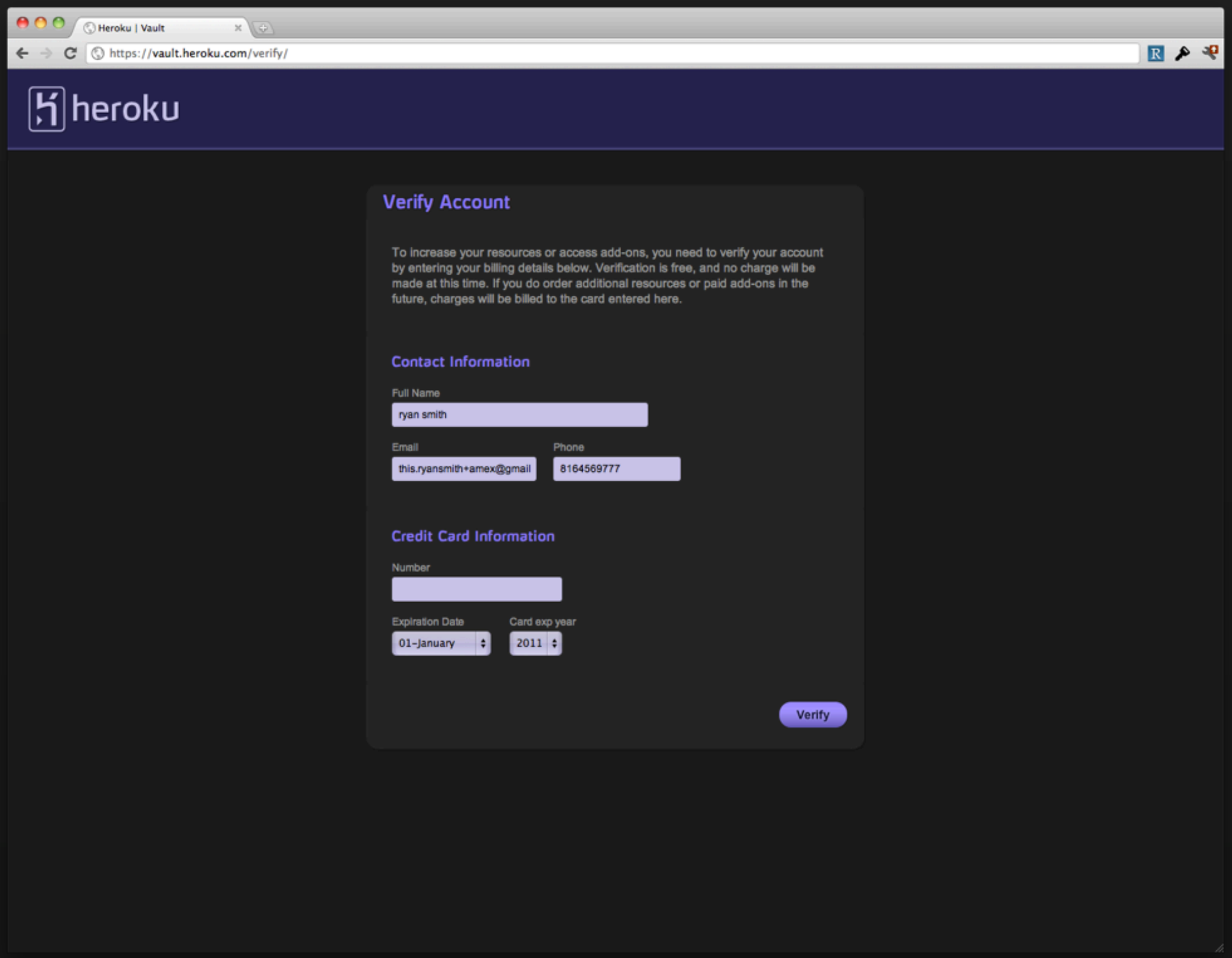
There is a lot to learn from the UNIX philosophy.

“Write programs that do one thing and do it well.”

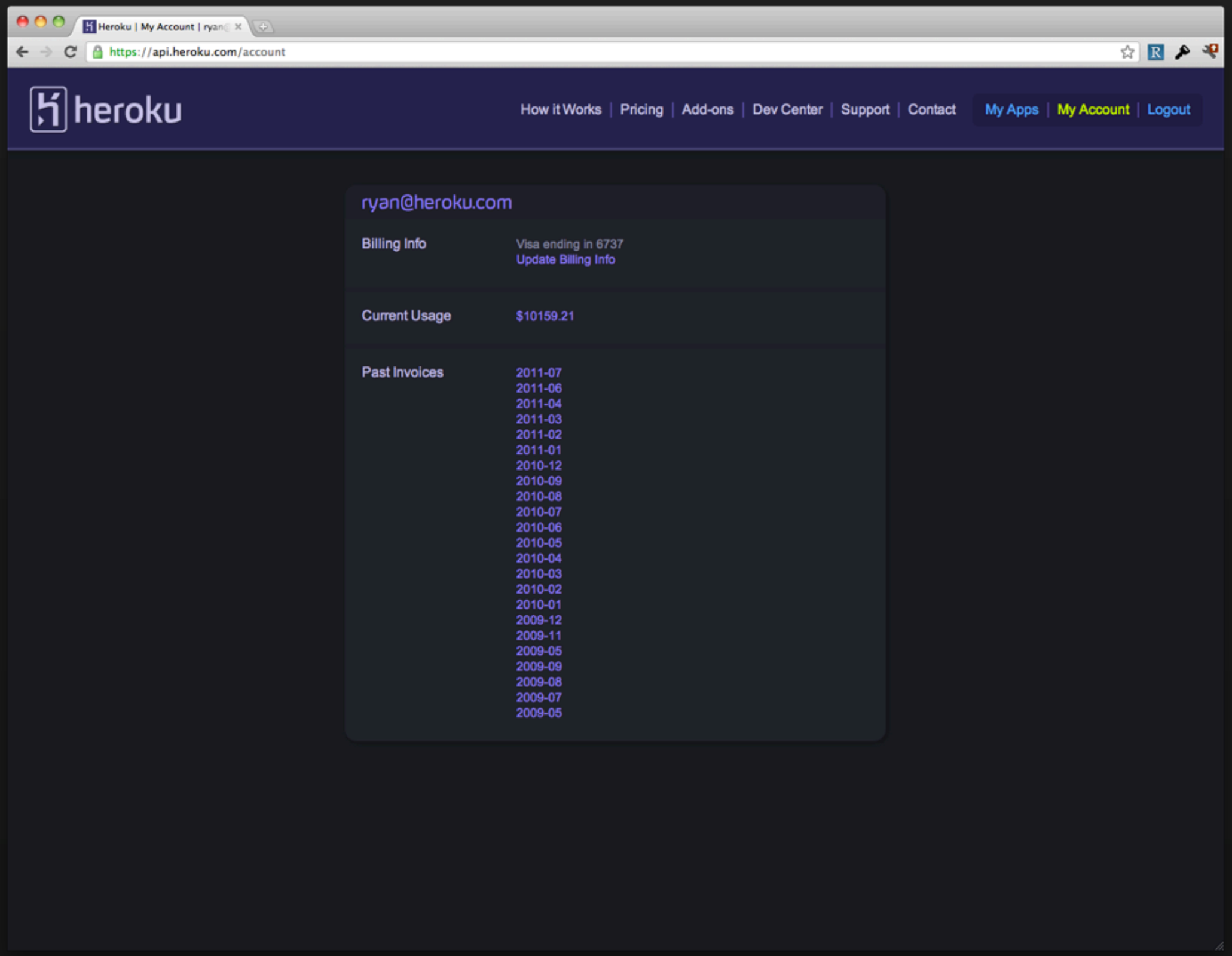
-- Doug McIlroy

heroku

One of the things we picked early on was the need to write many small apps instead of one large app. We have apps for all kinds of things.



Web apps for credit card forms.



Web apps for invoice archiving and current usage

Background Apps

```
Aug 29 16:56:21 vault-kicker app/shovel.11: V14 account=33912 stop invoice process. memory=100.0MB
Aug 29 16:56:21 vault-kicker app/shovel.16: V14 account=33808 stop invoice process. memory=100.0MB
Aug 29 16:56:21 vault-kicker app/shovel.5: V14 account=33934 stop invoice process. memory=100.0MB
Aug 29 16:56:21 vault-kicker app/shovel.4: V14 account=33925 stop invoice process. memory=100.0MB
Aug 29 16:56:21 vault-kicker app/shovel.14: V14 account=33905 stop invoice process. memory=100.0MB
Aug 29 16:56:21 vault-kicker app/shovel.10: V14 account=33923 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.9: V14 account=33765 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.7: V14 account=34003 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.8: V14 account=33950 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.2: V14 account=34005 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.6: V14 account=33954 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.16: V14 account=34009 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.11: V14 account=34023 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.13: V14 account=33968 stop invoice process. memory=100.0MB
Aug 29 16:56:22 vault-kicker app/shovel.5: V14 account=34099 stop invoice process. memory=100.0MB
Aug 29 16:56:23 vault-kicker app/shovel.14: V14 account=34122 stop invoice process. memory=100.0MB
Aug 29 16:56:23 vault-kicker app/shovel.4: V14 account=34109 stop invoice process. memory=100.0MB
Aug 29 16:56:23 vault-kicker app/shovel.9: V14 account=34195 stop invoice process. memory=100.0MB
Aug 29 16:56:23 vault-kicker app/shovel.15: V14 account=34063 stop invoice process. memory=100.0MB
```

Background apps with no Web UI. Apps that process gigabytes of data.

There was a time when all of our code was in 1 large monolithic app and it was quite painful. When I go to write a feature for an app I not only have to load the code into my editor, I also have to load it into my brain. At that point, I become the bottleneck.

To that end, I started breaking down the apps into simpler components.

small > large

you can also horizontally scale your app independently of other components

Informal agreement on pre/post conditions.

I would be remiss if I didn’t inform you of the pitfalls in deploying many small apps in your system.
There will be problems in maintaining consistency across dependent APIs. However, The Vault has had great success in overcoming this challenge by setting up inter-team contracts.

If 1 or more components within our business are dependent upon our API then we draft an informal document describing what we promise to uphold. We typically keep this document in the doc directory of the applications code base.

Dependencies

So the first concept was small apps and the next concept is dependency management.

Common

HTTP

JSON

Queueing

Scheduling

Most of our apps share a set of dependencies. Although some may vary by version, they are generally solutions to common problems. We use libraries that help us make HTTP requests, pase JSON, publish and consume on message queues

Formal Declaration

We keep a machine readable list of dependencies in each one of our apps.

Easy install for new developers.

This allows new developers a wonderful way to bootstrap the app.

Consistency in Production.

This also ensures that our production platform knows the exact version of dependency to install.

Java	Maven
Ruby	Bundler
Python	Pip
Node.js	NPM
Clojure	Lein

heroku

Popular languages support this. It is good. I can’t imagine not using one of these tools with my apps.

Even if your project has 0 dependencies, still include a Gemfie. At the end of the day, you are communicating to other developers what a project needs. If you project needs nothing, communicate that.

Environment

Production

Staging

Development

For each app, we have both a staging and production deployment. The two deployments are identical. They are on the same platform, the same OS, they have the same codebase, the same dependencies... IDENTICAL.

We try and get our development environment identical as well. We develop BSD and Linux and execute the same commands that our server executes.

Creating apps should be trivial.

It would be too much work if I had to create servers for each environment. Heroku allows me to quickly create apps.

I don’t have time to provision hardware and setup my runtime.

Caveat

Staging Credit Card gateway

$VERBOSE='true'

Of course we need to tweak a few settings between environments.

We use the sandbox API provided by our credit card gateway.

We use staging for experiments. On heroku it is very easy to load our production dataset into a staging database. So we take the opportunity to test the performance of new algorithms, tune SQL queries, etc....

And in development we turn on all of the logging features in our libraries.

The point is that all of tweaks are defined in the environment via shell variables.

Application Code

```
if ENV['VERBOSE'] == 'true'
  log('message')
end
```

Production Environment

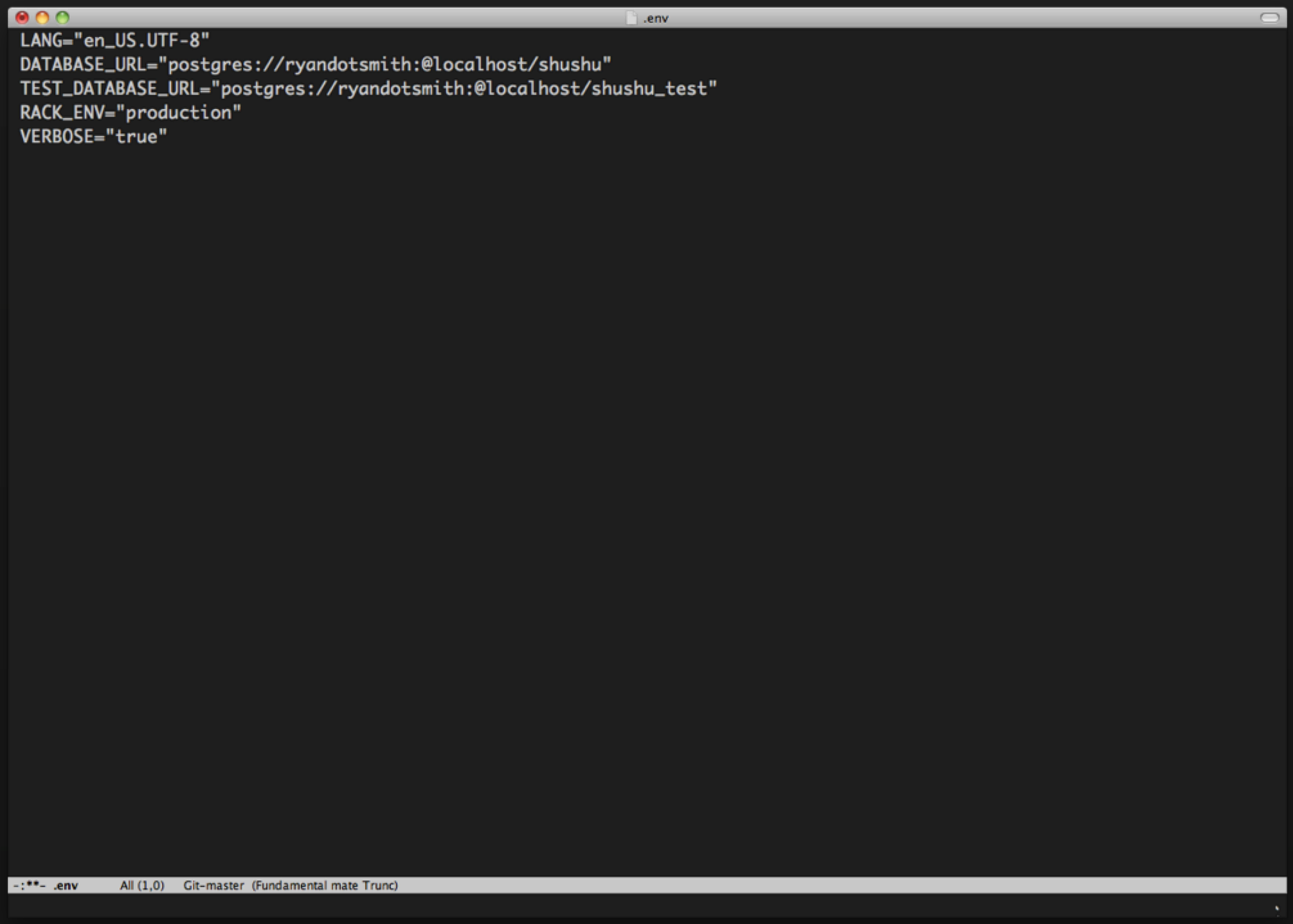
```
$ export VERBOSE='false'
```

Development Environment

```
$ export VERBOSE='true'
```

heroku

Our code base remains constant while our environment determines how the system should react.



I keep a file in my project directory and name it .env
This is how I communicate what environment setting my app requires

```
cat .env | xargs env
```

I keep a file in my project directory and name it .env
This is how I communicate what environment setting my app requires.
If you use foreman, this file will be loaded into your shell environment upon it's startup.

Services

Everything is a service.

Database

Cache

Message Bus

Log Drain

Heartbeat Monitor

Exception Logger

Everything is a service.

Everything is a service. I don’t have time to configure and maintain all of the systems. I am a team of 1.

New Features :)

DBA :(

The Vault, relies on Postgres. This service is provided by Heroku's data team. This is definitely a feature we could not do without. I can't imagine developing new features AND maintaining a database.

CREATE INDEX

The only thing remotely close to DBA work that I do is: CREATE INDEX.

PRO TIP

CREATE INDEX CONCURRENTLY

typically when you create an index it will lock your table. This is bad. Postgres will allow you to create an index without locking the table. Super useful for debugging production performance problems.

Provision a new DB in < 60s

Just like apps, cloud services can be brought online in seconds, not days.

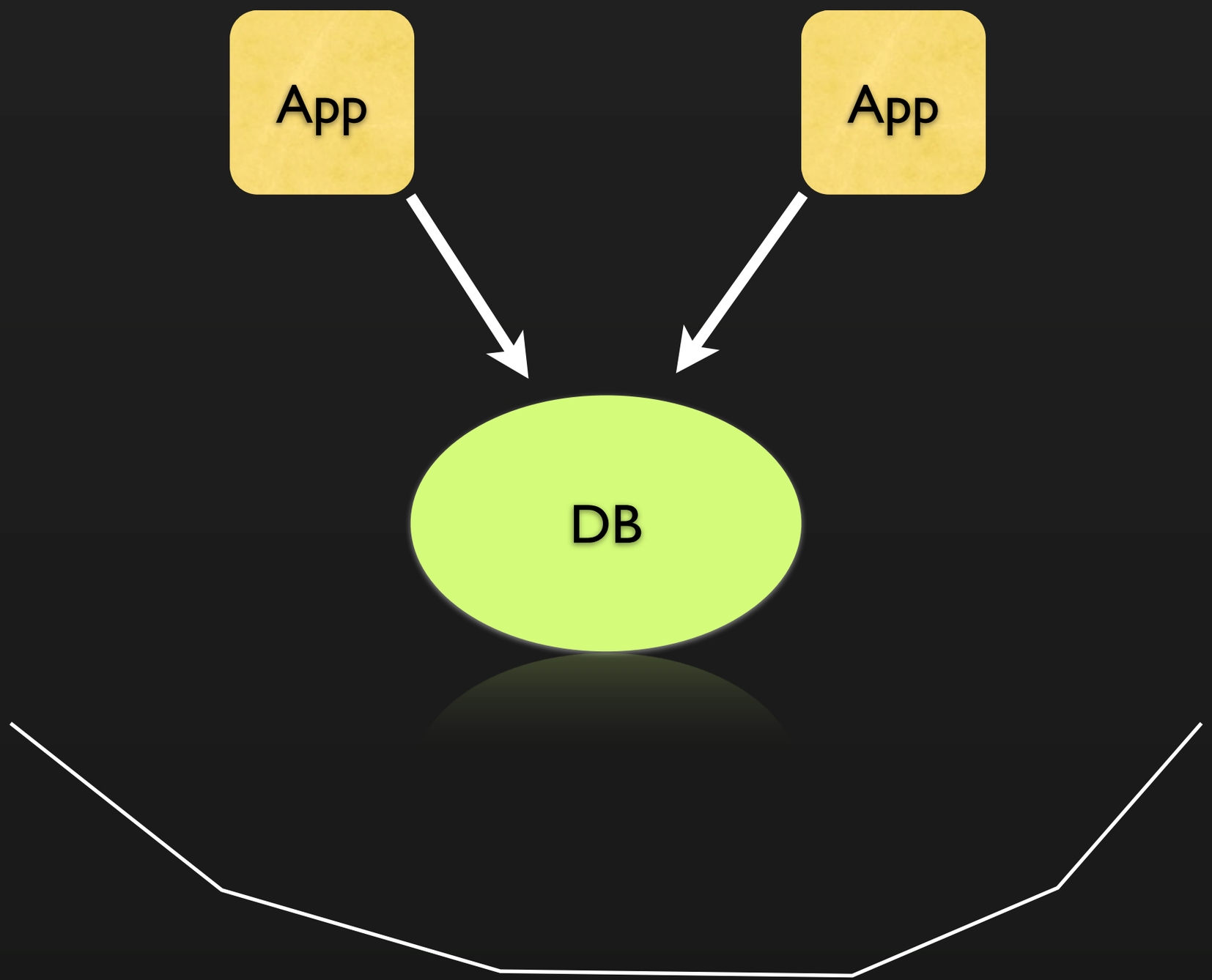
Configured in environment.

When we provision a database, our provider gives us a URI which we set as a shell variable in our environment. There are no configuration files to track, just set the proper variable in the environment.

```
database = DB.connect(
    ENV["DATABASE_URL"]
);
```

heroku

In my code, I reference a URL that is set in my environment.



Because my database is just a URL, I can easily share this database among other apps in the cloud.

This makes me happy.

Deploy

heroku

Developer Deploys Code

As a team of 1, I develop the feature, test the system and roll it out to my customers. I have great insight into all of the moving parts of my system.

I want for my deploy.

This allows me to quickly ship features into production. When I fix a bug or ship a new UI improvement. I want this in production right away. I don’t want to be blocked by anything! I want prime for my deployment.

Incremental Rollout

Even though I have never shipped a bug into production, I like to know that I have a safety net. Incremental Rollout --one of the most valuable practices I have incorporated into The Vault's repertoire.

```
SELECT ... WHERE MOD(id, 4) = 0;
```

I can choose to enable a new feature for a portion of our customers. I deploy the change, watch my logs and if there is a problem, I quickly rollback to the last known commit.

Rollback

heroku

When I encounter an unusual amount of exceptions in my app, I rely on the ability to rollback.

Continuous Deployment

Developers deploying and incremental rollout describe a more general concept. And that is: Continuous deployment. As we fix bugs, develop features, we deploy the code. Since it takes < 30 seconds to release code to production, it makes a lot of sense for us to push changes at will.

The days are long over when we deploy hundreds of commits.

Visibility

heroku

Visibility is quite important to a team of 1. I don’t have time to manually monitor all of my systems. So I rely on cloud technologies to alert me when interesting things happen.

Science

On a side note. I have been on a science kick lately. When troubleshooting complex routines or reasoning about the performance of my system, I start by employing the scientific method.

Hypothesis

Conclude Implications

Observe Reality*

heroku

The 3rd step is what I need help with. I need an easy way to make observations.

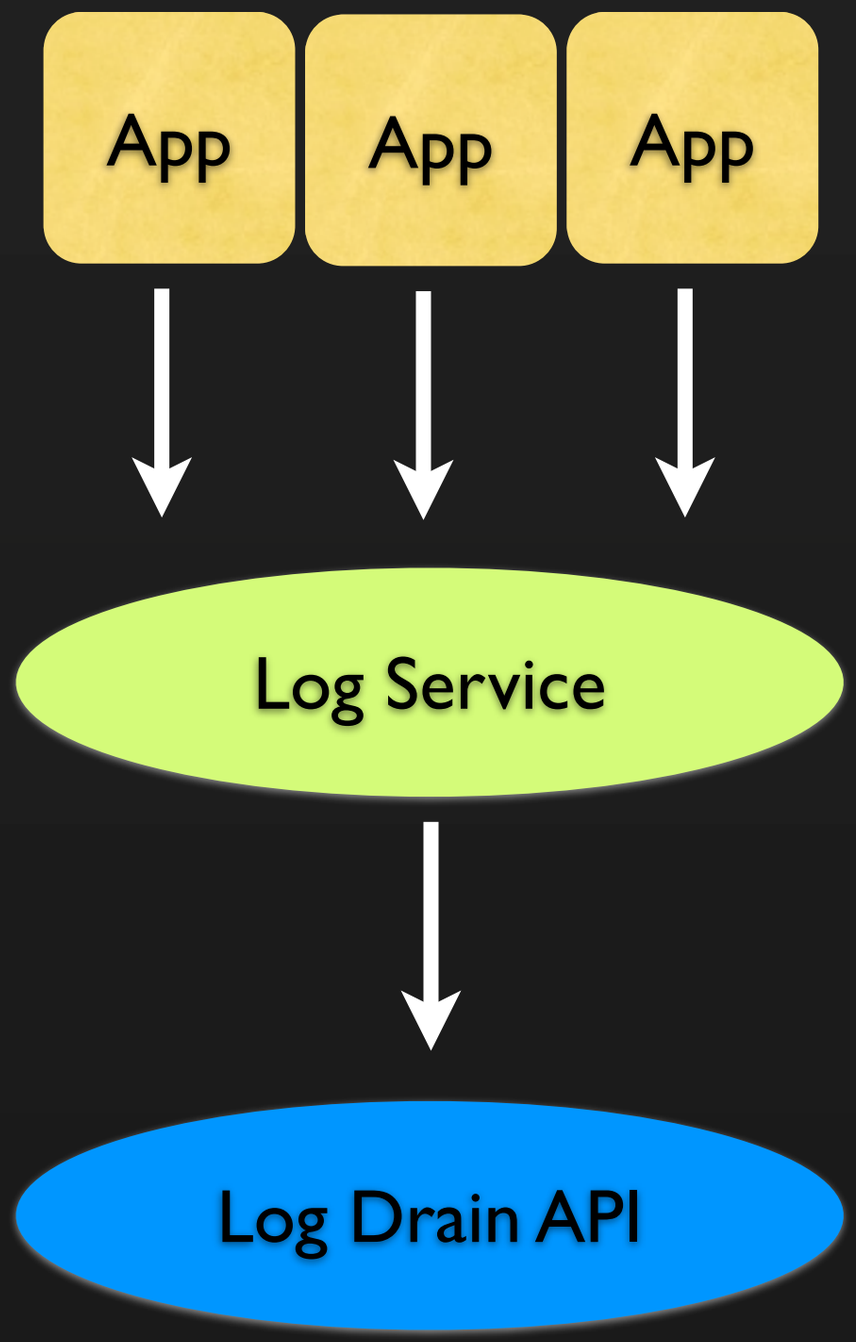
Logging

Logging is what helps us observe reality

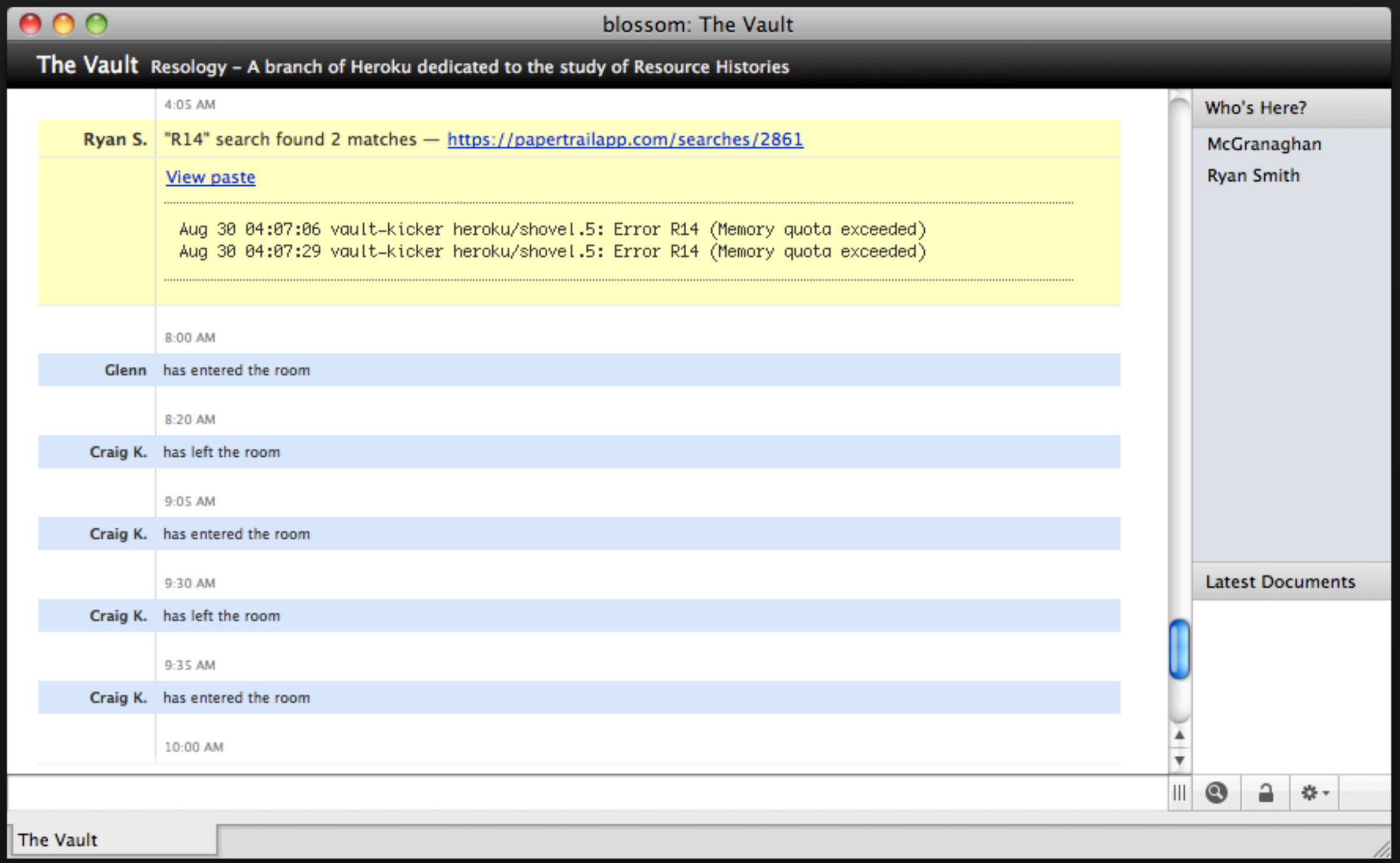
Think of your logs as streams of data.

heroku

I am not talking about log files. Forget about log files. Start treating your logs as first-class data. Data that streams from all of our apps.

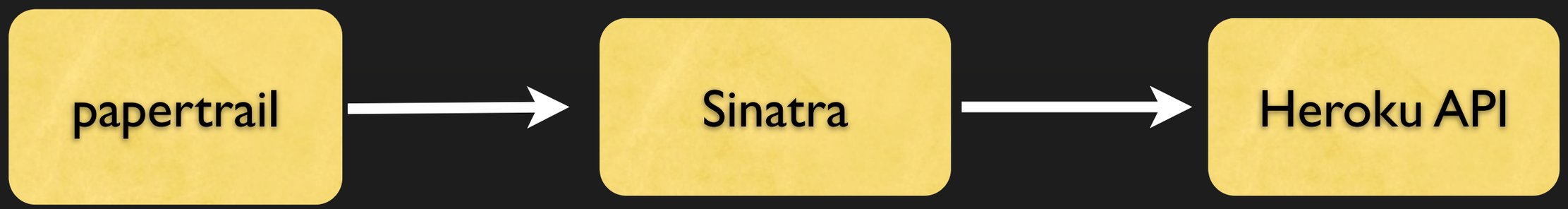


We point these streams at cloud logging services. Products like papertrailapp give you a web ui and APIs to do all sorts of analysis on your log data.



1 cool feature that we could not live without is log alerts. This service is pioneered by papertrailapp.com We setup search queries for our log data. Whenever a log message comes in that matches that criteria, papertrail will send a message into our group chat service.

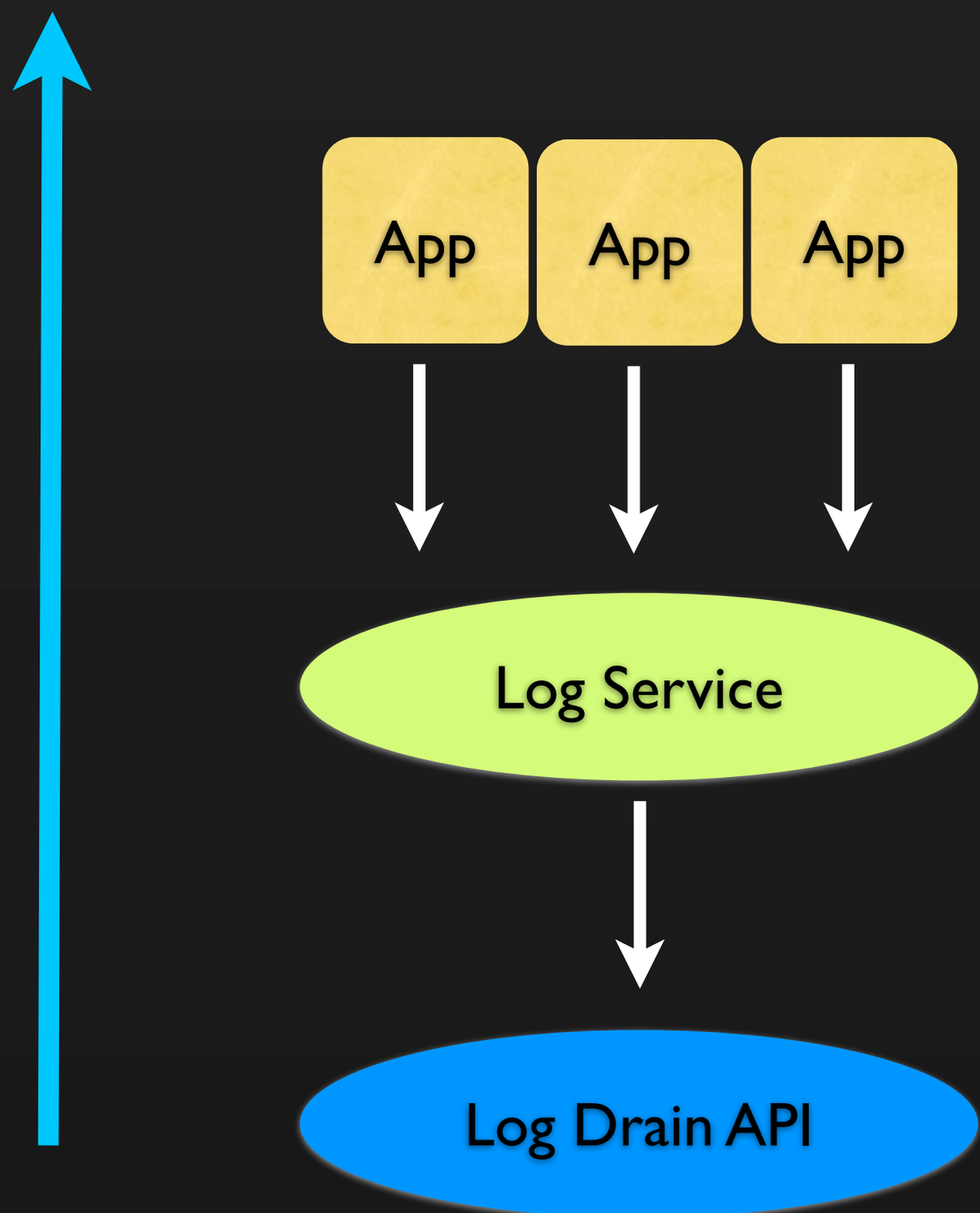
For instance, I log errors when my worker apps start to slow down. When this happens, I get notified immediately in my team's chat room.




```
heroku restart -a app_name -ps worker.5
```

Not only can i receive campfire messages about this worker problem. I can also write a service that repairs my system for me.

If we trust our logs to warn us, why not trust them with other tasks.



What if we used the connivence of logging as a transport for data back to our applications.

log = event stream

We started treating our logs as first-class data. Data that streams from all of our apps.

Event streams are the future. This area is ripe for innovation and standardization.

Administration

So after we realize that an app is misbehaving, we need some way of easily restarting the process. Jokingly, we call this activity: kicking the vault.

If you do it twice, automate it.

heroku

Conclusion

heroku

Small Apps

Dependencies

Environment Vars

Services

Deployment

Visibility

heroku

There are a million different ways to skin this cat. The Vault has found an approach that works reasonably well. The approach I have outlined here is optimized for rapid change and feedback --the agile process. It is also optimized for scale. The size of my dataset grows geometrically; the size of my team grows linearly at best. These practices maintain sanity in our chaotic environment.

The concepts we have discussed will help you do the same.

Focus on your application not your platform

heroku

finnish

heroku