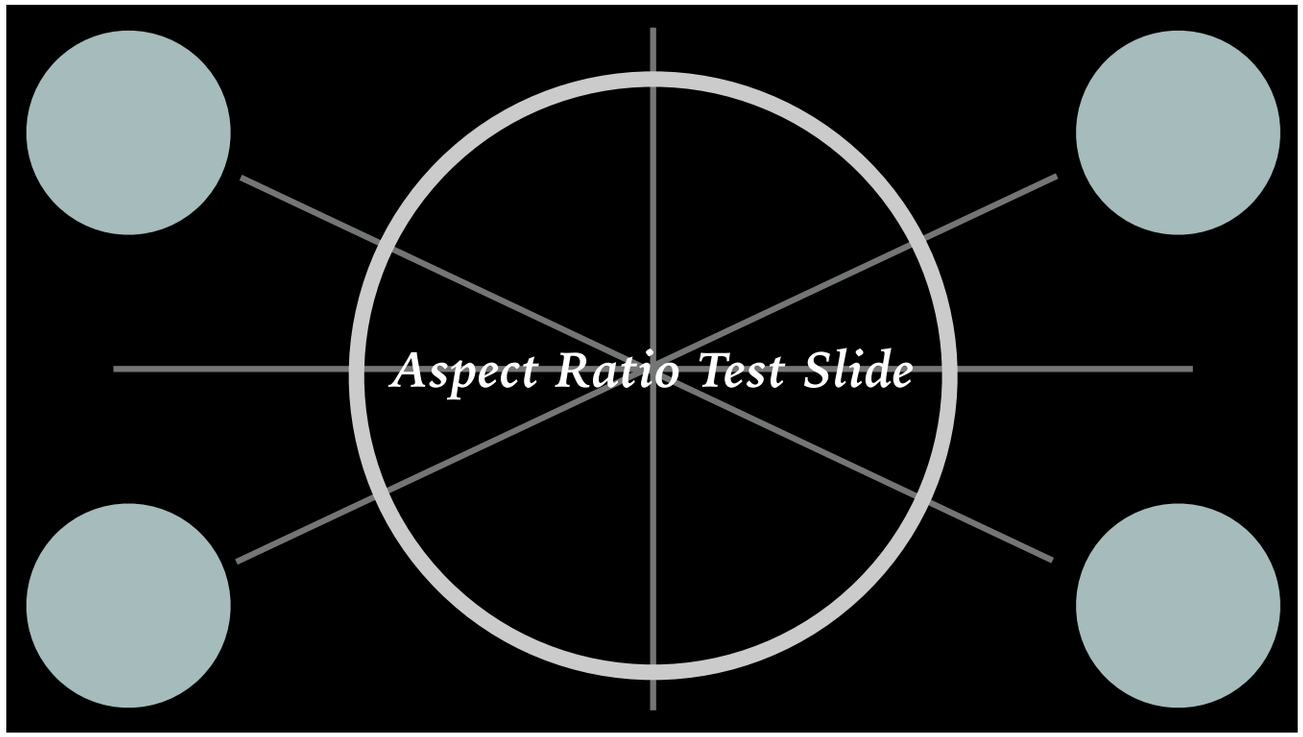

Aspect Ratio Test Slide

A 2.0 IS NOT GOING TO KILL YOU BUT IT WILL TRY

by Jan Lehnardt at ApacheCon EU 2016 in Sevilla

Hi, my name is Jan Lehnardt, I’m a developer and business person from Berlin, Germany

JAN LEHNARDT

- **CouchDB since 2006**
- **Apache CouchDB since 2008**
- **PMC Chair & VP of CouchDB since 2011**
- **Longest active contributor**
- **CEO at Neighbourhoodie Software in Berlin**

Joined CouchDB in 2006, longest standing contributor

Have done everything from evangelising, community work, core engineering.

Still do all of the above

* * *

Blog post by Michael Lopp who is good at identifying patters in software engineering management

HE SAID: “SHIPPING A 1.0 PRODUCT ISN’T GOING TO KILL YOU, BUT IT WILL TRY”

— http://randsinrepose.com/archives/1-0/

Great article about the dynamics of trying to ship a 1.0 product as a startup and all the ways it can go wrong.

I’ve grown to not be a big fan of the whole startup culture and ecosystem, but the things outlined in the article are applicable to just about any endeavour.

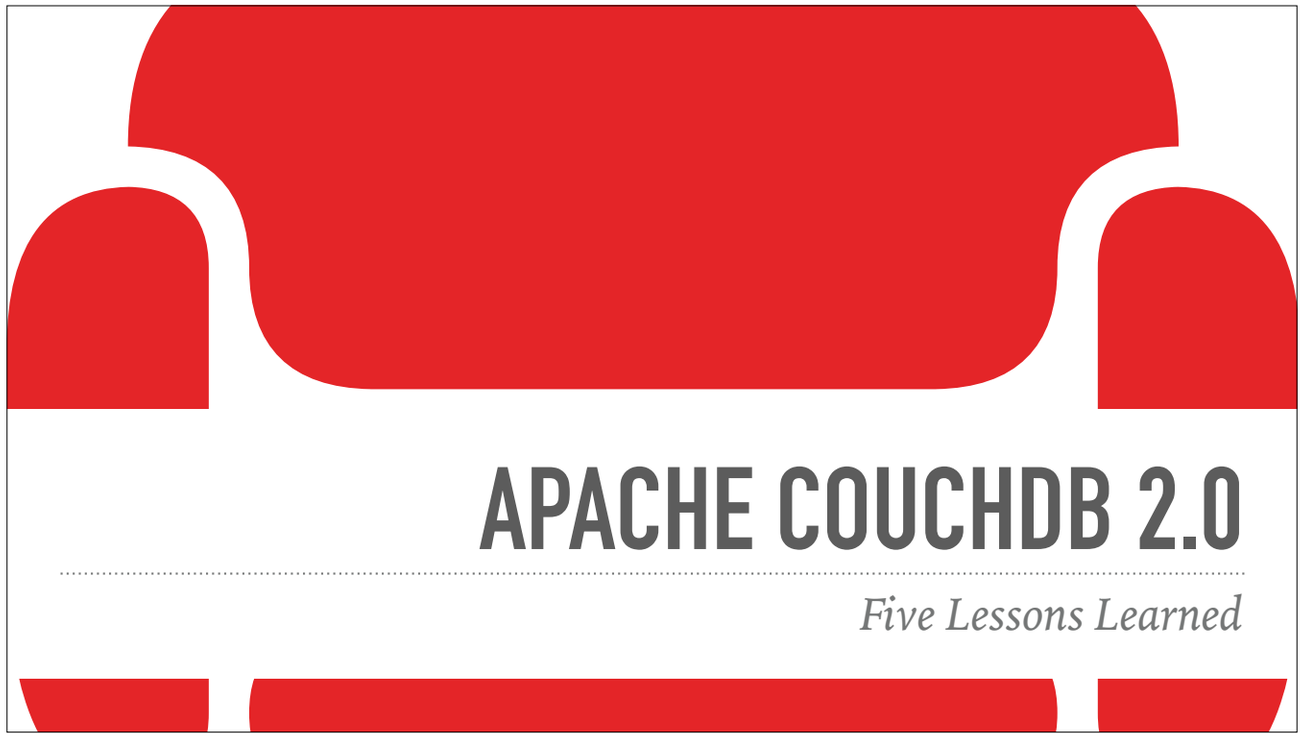
Now that we’ve shipped CouchDB 2.0, I’ve seen that in retrospective, there is a whole different set of things that can go wrong when building on an existing project, with an existing community, with existing success.

SHIPPING A 2.0 PRODUCT ISN'T GOING TO KILL YOU, BUT IT WILL TRY

SHIPPING A 2.0 PRODUCT ISN'T GOING TO KILL YOU, BUT IT WILL TRY
. . . IN ENTIRELY NEW WAYS"

I'm using "Product" here in the broadest possible sense. Whatever endeavour you are on:

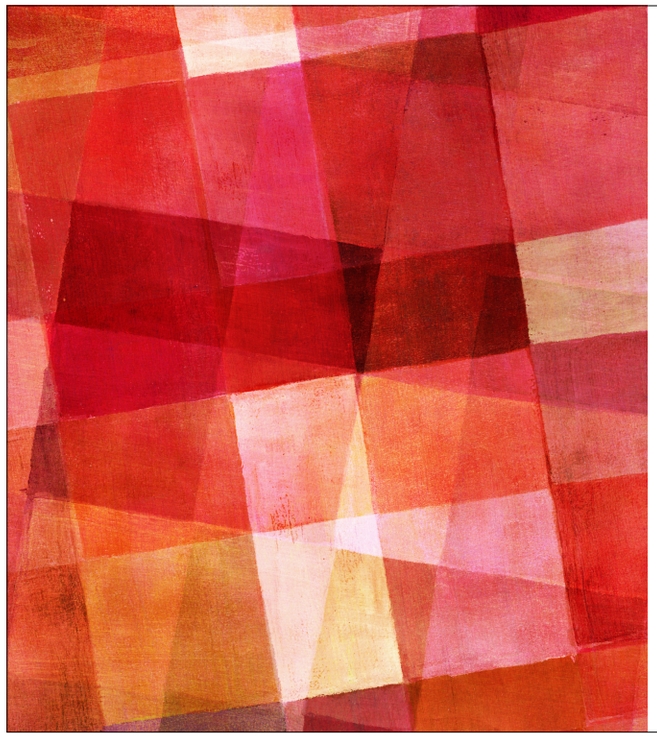
- a software product
- an open source project
- running a conference like this one
- starting a non-profit



This talk is not as universal as the blog post it was inspired by.

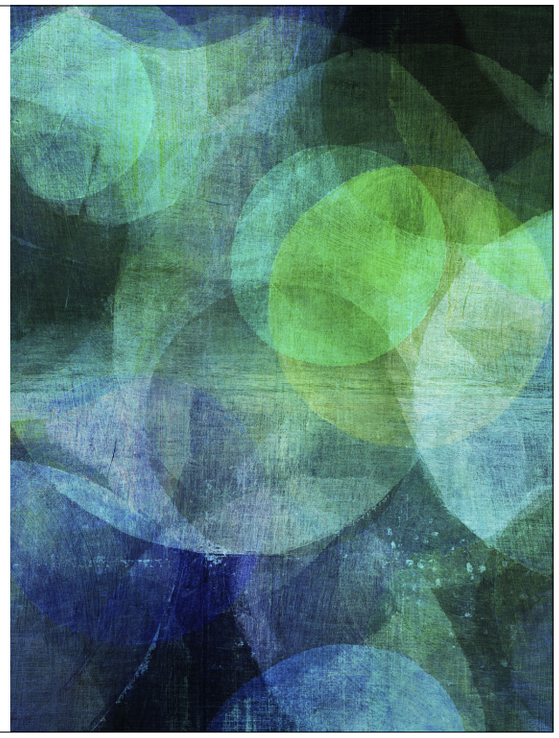
Since every 1.0 is different, every 2.0 starts from a different place.

Throughout this talk, I'll be presenting five stories, of how things could have gone wrong, what we did to save the day, and what you can learn from that for your projects.



1. Community Struggle
2. Feature Deprecation
3. Trademark Disputes
4. The Trade-Offs of Shipping
5. Sustainable Open Source

COMMUNITY STRUGGLE



COMMUNITY STRUGGLE: 2012

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves
  - and bad-mouths the project

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves
  - and bad-mouths the project
- Two high-profile projects drop CouchDB publicly

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves
  - and bad-mouths the project
- Two high-profile projects drop CouchDB publicly
  - despite the issues being not with CouchDB

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves
  - and bad-mouths the project
- Two high-profile projects drop CouchDB publicly
  - despite the issues being not with CouchDB
- The Result:

All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

Now, everywhere I go I hear “isn’t CouchDB dead?”

COMMUNITY STRUGGLE: 2012

- The Inventor leaves
  - and bad-mouths the project
- Two high-profile projects drop CouchDB publicly
  - despite the issues being not with CouchDB
- The Result:
  - “CouchDB is Dead” is the meme du-jour for five years.

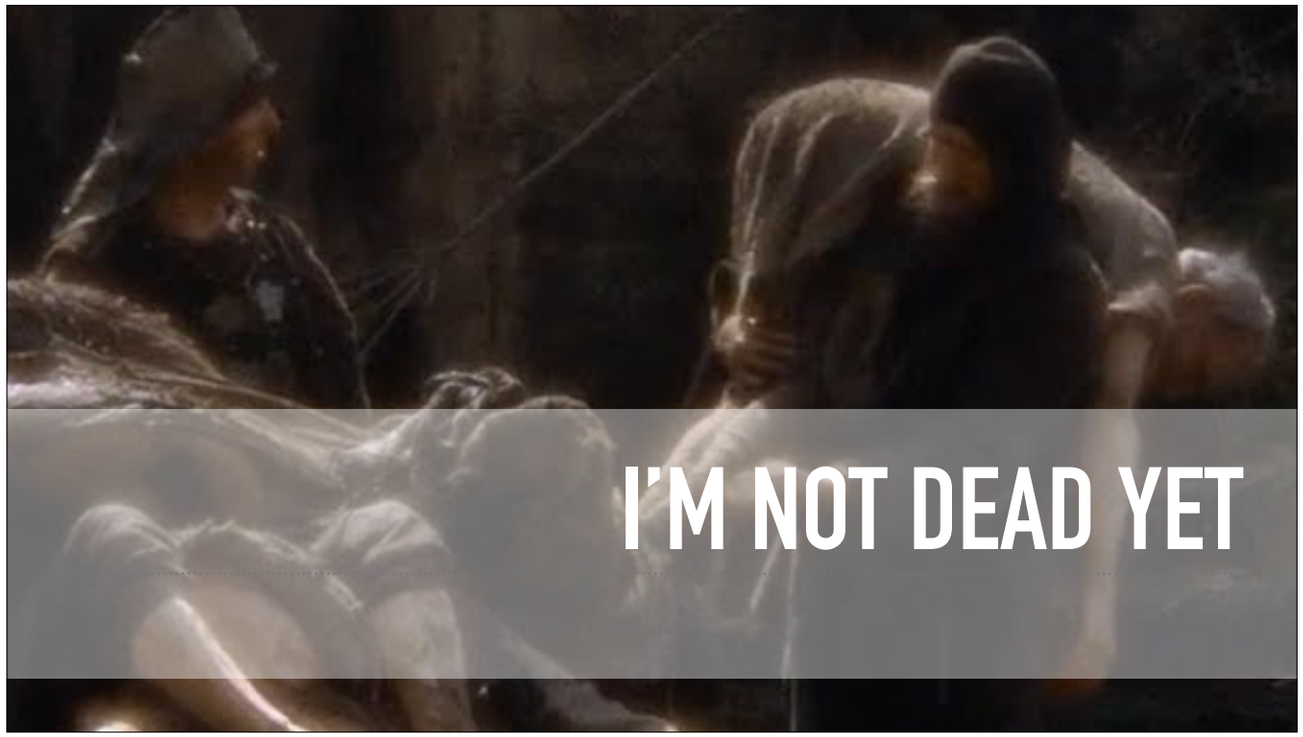
All of this happened at the beginning of 2012

Inventor had founded a startup around CouchDB, direction changed with new investors, “new product” not compatible, but communicated as “the future”.

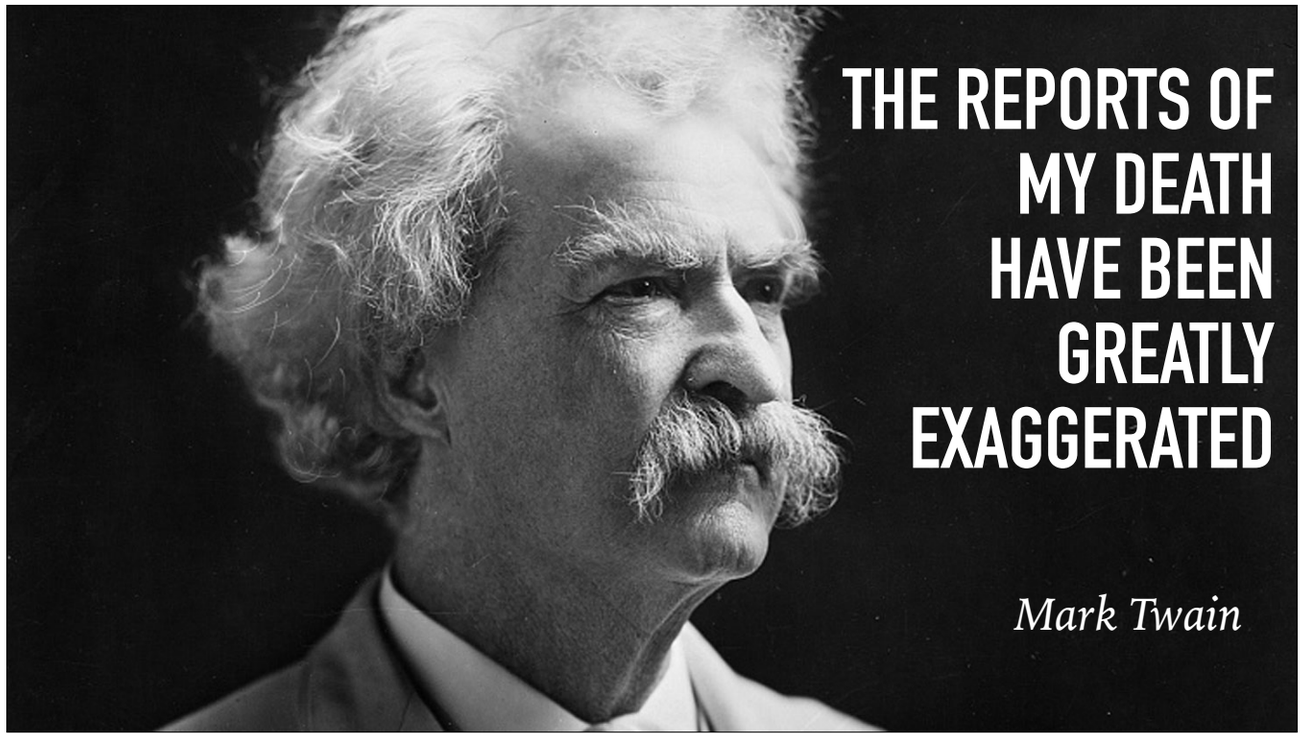
Project 1 we had worked with closely, even wrote custom code for them. Management issues, two people to build and run a multi-million user cloud infra + client software => doomed, shelved now.

Project 2 technology requirements changed. Early on CouchDB was a good fit. Changed to not be a good fit, they tried to make it work despite (bless them), but had to switch eventually.

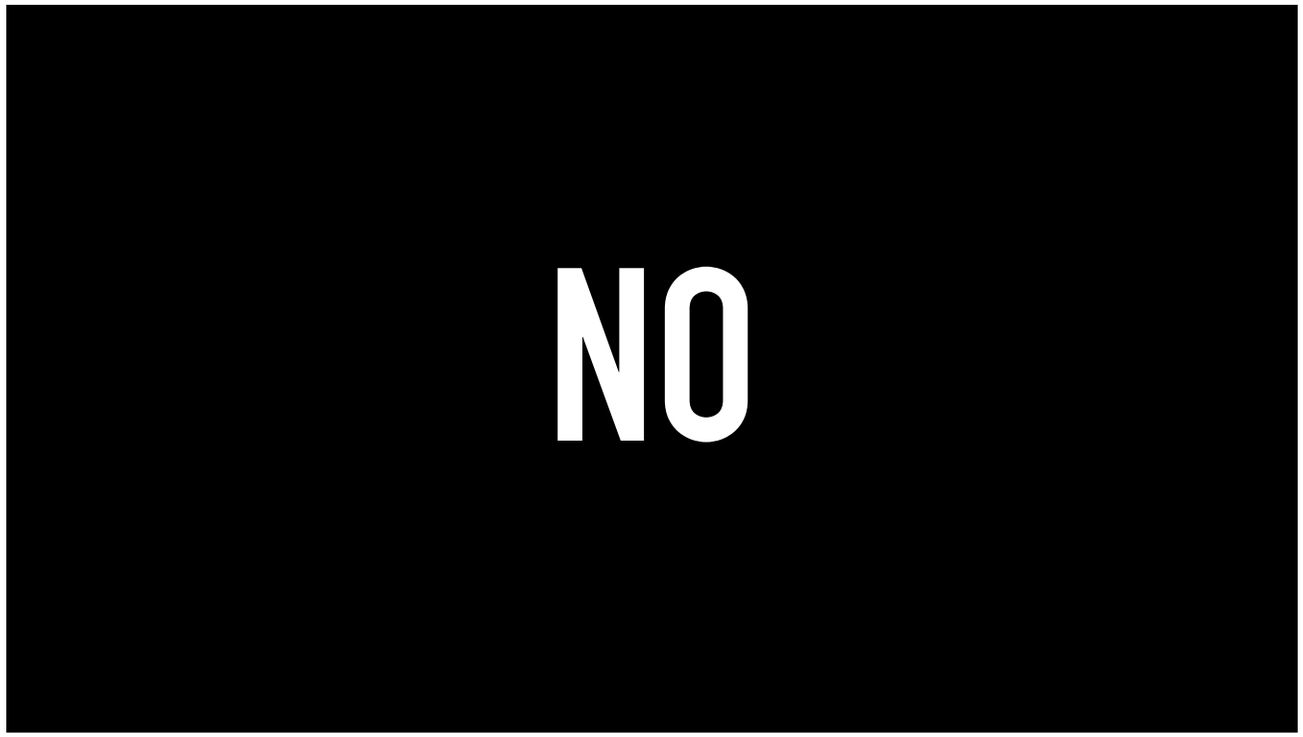
Now, everywhere I go I hear “isn’t CouchDB dead?”



What should we do?



We considered publishing a blog post stating that we are very much alive



But no, we thought this was more of a desperate move and we decided instead to:

buckle down and continue to work, and show, rather than tell

SHOW, NOT TELL

buckle down and continue to work, and show, rather than tell

COMMUNITY STRUGGLE: REMOVE POISONOUS PEOPLE

c.f. Google I/O 2008 - Open Source Projects and Poisonous People: https://www.youtube.com/watch?v=-F-3E8pyjFo

COMMUNITY STRUGGLE: REMOVE POISONOUS PEOPLE

- 5 years to realise a committer was poisonous / toxic

c.f. Google I/O 2008 - Open Source Projects and Poisonous People: https://www.youtube.com/watch?v=-F-3E8pyjFo

COMMUNITY STRUGGLE: REMOVE POISONOUS PEOPLE

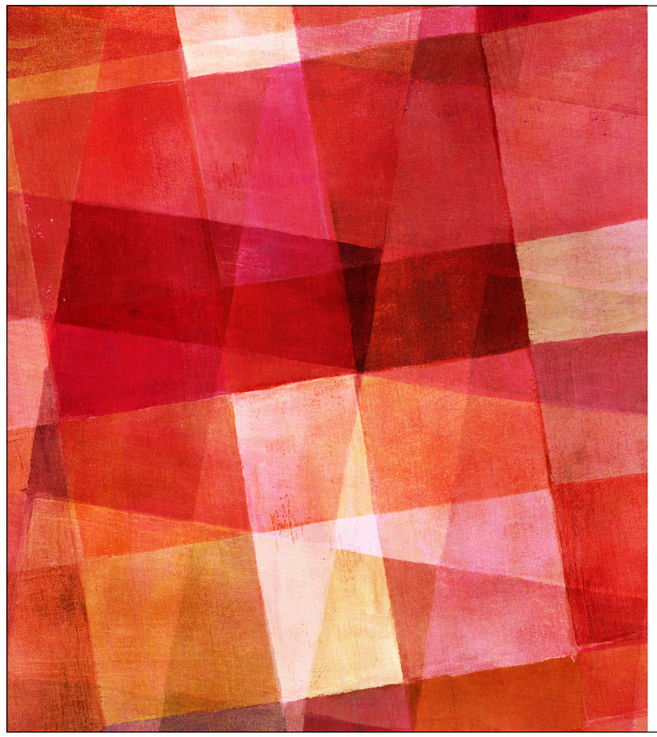
- 5 years to realise a committer was poisonous / toxic
- Another year to set up the process to remove them

c.f. Google I/O 2008 - Open Source Projects and Poisonous People: https://www.youtube.com/watch?v=-F-3E8pyjFo

COMMUNITY STRUGGLE: REMOVE POISONOUS PEOPLE

- 5 years to realise a committer was poisonous / toxic
- Another year to set up the process to remove them
- Hardest thing I've ever done in open source

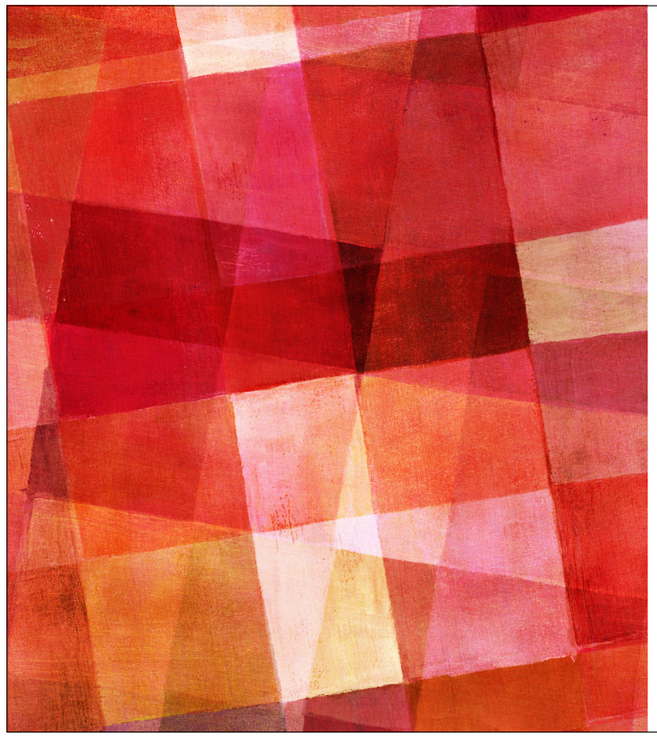
c.f. Google I/O 2008 - Open Source Projects and Poisonous People: https://www.youtube.com/watch?v=-F-3E8pyjFo



1. More frequent and more regular releases
2. New website
3. Regular online meetings
4. Weekly news blog posts about everything that's going on
   - Since adopted by a number of other ASF projects

This helped stem the tide, but took a year or two to arrive in the immediate communities, more in the larger tech world.

And we still got some of this when we shipped CouchDB 2.0 two months ago.



1. Community Struggle
2. **Feature Deprecation**
3. Trademark Disputes
4. The Trade-Offs of Shipping
5. Sustainable Open Source

FEATURE DEPRECATION: COUCHAPPS

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage
- It works really well with web browsers

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage
- It works really well with web browsers
- You can build apps that live inside CouchDB and run in the web browser; we call them CouchApps

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage
- It works really well with web browsers
- You can build apps that live inside CouchDB and run in the web browser; we call them CouchApps
- They would sync with the data in a P2P fashion

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage
- It works really well with web browsers
- You can build apps that live inside CouchDB and run in the web browser; we call them CouchApps
- They would sync with the data in a P2P fashion
- They are a bad idea

FEATURE DEPRECATION: COUCHAPPS

- CouchDB is a database that uses HTTP for transport and JSON for data storage
- It works really well with web browsers
- You can build apps that live inside CouchDB and run in the web browser; we call them CouchApps
- They would sync with the data in a P2P fashion
- They are a bad idea
  - It just took us a couple of years to figure that out

FEATURE DEPRECATION: COUCHAPPS

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called
- hand-coded C-wrapper around a specific version of a specific JavaScript runtime

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called
- hand-coded C-wrapper around a specific version of a specific JavaScript runtime
  - no upgrades, missing features

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called
- hand-coded C-wrapper around a specific version of a specific JavaScript runtime
  - no upgrades, missing features
- missing flexibility on the server-side

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called
- hand-coded C-wrapper around a specific version of a specific JavaScript runtime
  - no upgrades, missing features
- missing flexibility on the server-side
  - you can’t actually build many useful classes of apps

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

- They are a bad idea
- cgi-bin *fork per concurrent request*-model
  - 1995 called
- hand-coded C-wrapper around a specific version of a specific JavaScript runtime
  - no upgrades, missing features
- missing flexibility on the server-side
  - you can’t actually build many useful classes of apps
- No active development

That isn’t to say you can’t do anything useful, but it is not really convenient.

cgi-bin: used for another thing where the model is okay

In the meantime, Node.js came, saw, and won that space of web development and CouchDB should just follow suit.

FEATURE DEPRECATION: COUCHAPPS

They didn't step up to move the project forward, yet they continued to boycott any attempts to sideline CouchApps.

We eventually gave them their own mailing-list. They discussed for another 100 emails and there is silence since then.

FEATURE DEPRECATION: COUCHAPPS

- When it came to discuss the scope for 2.0, the core team said CouchApps won't be a focus (no deprecation yet, no removing yet, just not a focus).

They didn't step up to move the project forward, yet they continued to boycott any attempts to sideline CouchApps.

We eventually gave them their own mailing-list. They discussed for another 100 emails and there is silence since then.

FEATURE DEPRECATION: COUCHAPPS

- When it came to discuss the scope for 2.0, the core team said CouchApps won't be a focus (no deprecation yet, no removing yet, just not a focus).
- Vocal minority (3-5 people, who we've never heard of before) went on a protest, spent weeks dominating any discussions

They didn't step up to move the project forward, yet they continued to boycott any attempts to sideline CouchApps.

We eventually gave them their own mailing-list. They discussed for another 100 emails and there is silence since then.

FEATURE DEPRECATION: COUCHAPPS

- When it came to discuss the scope for 2.0, the core team said CouchApps won't be a focus (no deprecation yet, no removing yet, just not a focus).
- Vocal minority (3-5 people, who we've never heard of before) went on a protest, spent weeks dominating any discussions
- I worked out a transition plan including a design for a modern technical foundation, and asked the 3-5 to get cracking.

They didn't step up to move the project forward, yet they continued to boycott any attempts to sideline CouchApps.

We eventually gave them their own mailing-list. They discussed for another 100 emails and there is silence since then.

FEATURE DEPRECATION: COUCHAPPS

- When it came to discuss the scope for 2.0, the core team said CouchApps won't be a focus (no deprecation yet, no removing yet, just not a focus).
- Vocal minority (3-5 people, who we've never heard of before) went on a protest, spent weeks dominating any discussions
- I worked out a transition plan including a design for a modern technical foundation, and asked the 3-5 to get cracking.
- No cracking. Crickets.

They didn't step up to move the project forward, yet they continued to boycott any attempts to sideline CouchApps.

We eventually gave them their own mailing-list. They discussed for another 100 emails and there is silence since then.

FEATURE DEPRECATION: LESSONS LEARNED

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project
- Lesson 2: make deprecation decisions swiftly

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project
- Lesson 2: make deprecation decisions swiftly
  - be prepared to leave people behind

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project
- Lesson 2: make deprecation decisions swiftly
  - be prepared to leave people behind
- Lesson 3: move vocal minorities to sub-communities

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project
- Lesson 2: make deprecation decisions swiftly
  - be prepared to leave people behind
- Lesson 3: move vocal minorities to sub-communities
  - give them a chance to step up

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won’t shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.

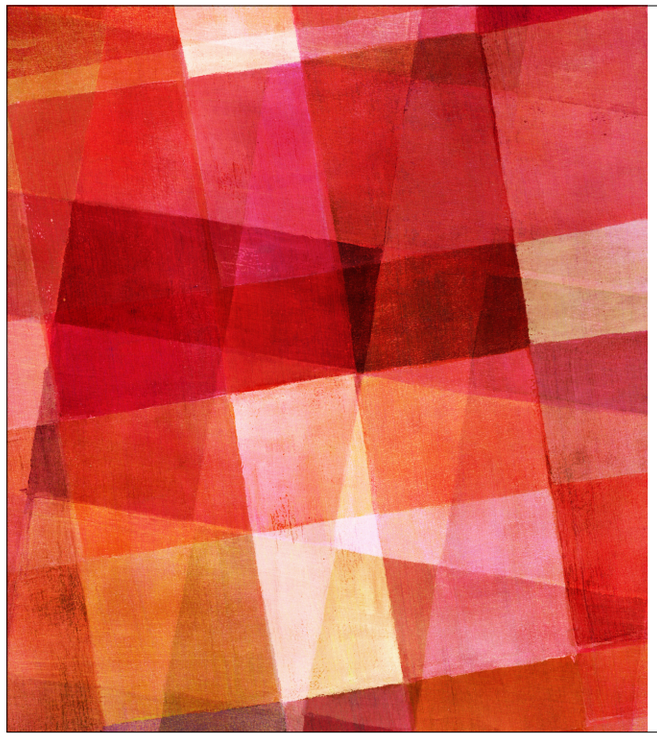
FEATURE DEPRECATION: LESSONS LEARNED

- Lesson 1: it takes a while until you understand the nature of your project
- Lesson 2: make deprecation decisions swiftly
  - be prepared to leave people behind
- Lesson 3: move vocal minorities to sub-communities
  - give them a chance to step up
  - if they don't, you can ignore them

Lesson 1: this will give some people the wrong idea about your project and course correction is going to take time and effort

Lesson 2: otherwise people will take all the fun out of the project, they drain resources by binding everyone into discussions that lead nowhere.

Lesson 3: if they won't shut up, get them out of the critical path, give them clear criteria for re-joining the mainstream. If they step up: great, if not: also great.



1. Community Struggle
2. Feature Deprecation
3. **Trademark Disputes**
4. The Trade-Offs of Shipping
5. Sustainable Open Source

TRADEMARK DISPUTES: SCENARIO 1

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

- ASF was built to avoid this
- ASF needs to document itself better, and “read the mailing list archive” is not documentation
- Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 1

- Fail to give CouchDB™ to ASF during incubation 🙈 (Sorry Shane!)

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

ASF was built to avoid this

ASF needs to document itself better, and “read the mailing list archive” is not documentation

Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 1

- Fail to give CouchDB™ to ASF during incubation 🙈 (Sorry Shane!)
- Inventor does startup, startup ends up with CouchDB™

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

ASF was built to avoid this

ASF needs to document itself better, and “read the mailing list archive” is not documentation

Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 1

- Fail to give CouchDB™ to ASF during incubation 🙈 (Sorry Shane!)
- Inventor does startup, startup ends up with CouchDB™
- Mutual-existence paperwork still WIP

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

ASF was built to avoid this

ASF needs to document itself better, and “read the mailing list archive” is not documentation

Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 1

- Fail to give CouchDB™ to ASF during incubation 🙈 (Sorry Shane!)
- Inventor does startup, startup ends up with CouchDB™
- Mutual-existence paperwork still WIP
  - Startup now preparing IPO 🙊

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

ASF was built to avoid this

ASF needs to document itself better, and “read the mailing list archive” is not documentation

Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 1

- ➤Fail to give CouchDB™ to ASF during incubation 🙈 (Sorry Shane!)
- ➤Inventor does startup, startup ends up with CouchDB™
- ➤Mutual-existence paperwork still WIP
  - ➤Startup now preparing IPO 🙉
- ➤Discussions about renaming 🐒

We failed to do incubation right. ASF has a lot of rules for projects just coming in. They are all there for a reason. It is not easy to find out those reasons.

ASF was built to avoid this

ASF needs to document itself better, and “read the mailing list archive” is not documentation

Nobody has got time to do this

Renaming: **CouchDB** is too good a name, we can’t possibly rename, but some in the PMC argued the benefits. And I agree with the benefits, but the downsides outweigh the benefits.

TRADEMARK DISPUTES: SCENARIO 2

He never contributed a single thing

No surprise: one of the vocal “CouchApp” minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they’d even accept a sponsor that violates the TM

Some people’s ego so big. 🙄

TRADEMARK DISPUTES: SCENARIO 2

➤“Well-meaning egomaniac”

He never contributed a single thing

No surprise: one of the vocal “CouchApp” minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they’d even accept a sponsor that violates the TM

Some people’s ego so big. 🙄

TRADEMARK DISPUTES: SCENARIO 2

- "Well-meaning egomaniac"
- Says they wants to build "open source company" with CouchDB technology at its core, contribute back, and also become ASF sponsor.

He never contributed a single thing

No surprise: one of the vocal "CouchApp" minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they'd even accept a sponsor that violates the TM

Some people's ego so big. 🙄

TRADEMARK DISPUTES: SCENARIO 2

- "Well-meaning egomaniac"
- Says they wants to build "open source company" with CouchDB technology at its core, contribute back, and also become ASF sponsor.
- Want to use "Couch*"-prefix as the name.

He never contributed a single thing

No surprise: one of the vocal "CouchApp" minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they'd even accept a sponsor that violates the TM

Some people's ego so big. 🙄

TRADEMARK DISPUTES: SCENARIO 2

- "Well-meaning egomaniac"
- Says they wants to build "open source company" with CouchDB technology at its core, contribute back, and also become ASF sponsor.
- Want to use "Couch*"-prefix as the name.
    - After Scenario 1, PMC decided to defend against commercial "Couch*"-naming.

He never contributed a single thing

No surprise: one of the vocal "CouchApp" minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they'd even accept a sponsor that violates the TM

Some people's ego so big. 🙄

TRADEMARK DISPUTES: SCENARIO 2

- "Well-meaning egomaniac"
- Says they wants to build "open source company" with CouchDB technology at its core, contribute back, and also become ASF sponsor.
- Want to use "Couch*"-prefix as the name.
    - After Scenario 1, PMC decided to defend against commercial "Couch*"-naming.
- They never contributed a single thing.

He never contributed a single thing

No surprise: one of the vocal "CouchApp" minority

Only wasted hours and hours of time.

Never bothered to talk to the ASF whether they'd even accept a sponsor that violates the TM

Some people's ego so big. 🙄

TRADEMARK DISPUTES: LESSONS LEARNED

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)
  - Actively enforce trademark violations

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)
  - Actively enforce trademark violations
- Document trademark rules (Couch*)

TRADEMARK DISPUTES: LESSONS LEARNED

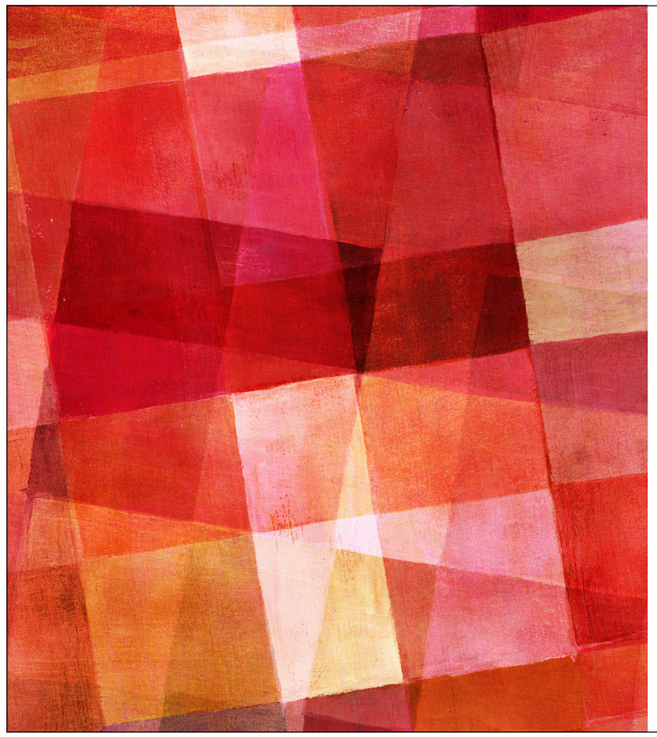
- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)
  - Actively enforce trademark violations
- Document trademark rules (Couch*)
- Contributors need to earn trust first, and cash in later

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)
  - Actively enforce trademark violations
- Document trademark rules (Couch*)
- Contributors need to earn trust first, and cash in later
- Doing PMC emails like these take a long time and all the fun out of working on the project

TRADEMARK DISPUTES: LESSONS LEARNED

- Follow the ASF rules
- You need to learn about US and intl. trademark law and enforcement (Shane helped us a lot)
  - Actively enforce trademark violations
- Document trademark rules (Couch*)
- Contributors need to earn trust first, and cash in later
- Doing PMC emails like these take a long time and all the fun out of working on the project
- Startups will get reckless



1. Community Struggle
2. Feature Deprecation
3. Trademark Disputes
4. **The Trade-Offs of Shipping**
5. Sustainable Open Source

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

➤The old one slowed us down

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

- The old one slowed us down
  - covered installation very well (`make install` worked everywhere)

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

- The old one slowed us down
  - covered installation very well (`make install` worked everywhere)
- The 2.0 build system

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

- The old one slowed us down
  - covered installation very well (`make install` worked everywhere)
- The 2.0 build system
  - Didn't finish `make install` in time

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

- The old one slowed us down
  - covered installation very well (`make install` worked everywhere)
- The 2.0 build system
  - Didn't finish `make install` in time
  - Ubuntu Snap, Docker, Mac & Windows native app

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: THE NEW BUILD SYSTEM

- The old one slowed us down
  - covered installation very well (`make install` worked everywhere)
- The 2.0 build system
  - Didn't finish `make install` in time
  - Ubuntu Snap, Docker, Mac & Windows native app
  - Should we ever add it?

Autotools

People, especially larger, slower orgs always want “Centos/Ubuntu” pkgs.

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

- CouchDB 2.0 documentation is minimal, despite major new features

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

- CouchDB 2.0 documentation is minimal, despite major new features
    - Setup & Migration instructions

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

- CouchDB 2.0 documentation is minimal, despite major new features
    - Setup & Migration instructions
    - Basic guides and API docs for new features

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

- CouchDB 2.0 documentation is minimal, despite major new features
    - Setup & Migration instructions
    - Basic guides and API docs for new features
    - Release Notes

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: DOCUMENTATION

- CouchDB 2.0 documentation is minimal, despite major new features
    - Setup & Migration instructions
    - Basic guides and API docs for new features
    - Release Notes
    - Good enough

You have to have some docs, but very little is good enough for a release, if you struggle about contributions.

Of course it’s not ideal, but what are you going to do :)

THE TRADE-OFFS OF SHIPPING: NOBODY TESTS A RELEASE CANDIDATE

Swag: my idea, ElasticSearch tried it, we adapted it

Worked somewhat, people are happy when they get the swag.

Will have to work on it sooner next time. Also: unique swag items

THE TRADE-OFFS OF SHIPPING: NOBODY TESTS A RELEASE CANDIDATE

➤ Nobody tests a release candidate

Swag: my idea, ElasticSearch tried it, we adapted it

Worked somewhat, people are happy when they get the swag.

Will have to work on it sooner next time. Also: unique swag items

THE TRADE-OFFS OF SHIPPING: NOBODY TESTS A RELEASE CANDIDATE

- Nobody tests a release candidate
- Everybody asks when the final version comes out

Swag: my idea, ElasticSearch tried it, we adapted it

Worked somewhat, people are happy when they get the swag.

Will have to work on it sooner next time. Also: unique swag items

THE TRADE-OFFS OF SHIPPING: NOBODY TESTS A RELEASE CANDIDATE

- Nobody tests a release candidate
- Everybody asks when the final version comes out
- “We release the final version once you’ve told us all the issues you’ve found in the release candidate” -> didn’t work.

Swag: my idea, ElasticSearch tried it, we adapted it

Worked somewhat, people are happy when they get the swag.

Will have to work on it sooner next time. Also: unique swag items

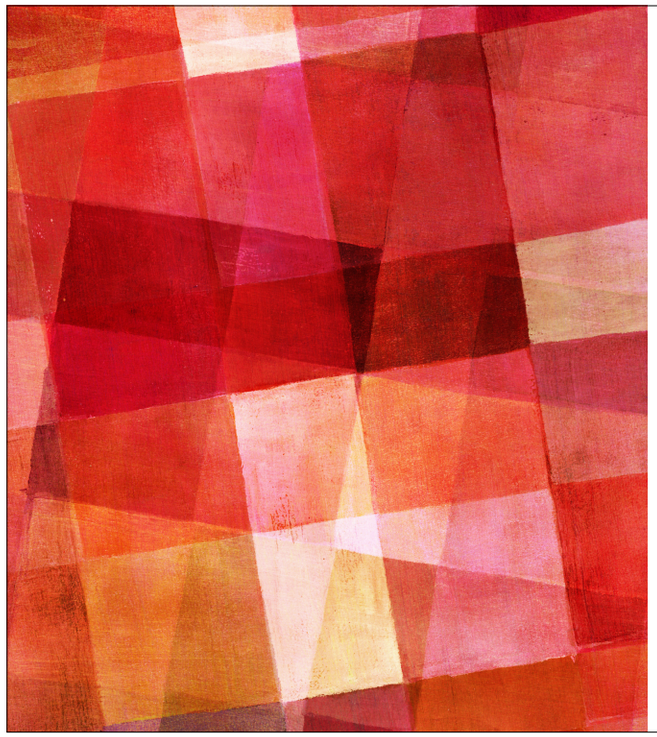
THE TRADE-OFFS OF SHIPPING: NOBODY TESTS A RELEASE CANDIDATE

- Nobody tests a release candidate
- Everybody asks when the final version comes out
- “We release the final version once you’ve told us all the issues you’ve found in the release candidate” -> didn’t work.
- SWAG programme

Swag: my idea, ElasticSearch tried it, we adapted it

Worked somewhat, people are happy when they get the swag.

Will have to work on it sooner next time. Also: unique swag items



1. Community Struggle
2. Feature Deprecation
3. Trademark Disputes
4. The Trade-Offs of Shipping
5. **Sustainable Open Source**

SUSTAINABLE OPEN SOURCE: HOW PROJECTS DIE IN TWO ACTS

People leave the project:
- no more time
- no more interest
- no more energy to overcome project friction (like getting new features in, setting project direction, release chores, microaggressions)

No new people join:
- project not well presented (website, blogs, conferences, books)
- technically no longer relevant (maintenance mode)
- hard to get started (obscure programming language, no getting started docs, no internals handbook, no starter issues, confusing bug tracker, not on GitHub)

Let’s look at how we can fix both causes of project death.

I would argue that making sure people don’t leave should be the first priority, because when we only focus on people joining, they’ll leave as fast as we recruit them.

1. People leave the project

People leave the project:
- no more time
- no more interest
- no more energy to overcome project friction (like getting new features in, setting project direction, release chores, microaggressions)

No new people join:
- project not well presented (website, blogs, conferences, books)
- technically no longer relevant (maintenance mode)
- hard to get started (obscure programming language, no getting started docs, no internals handbook, no starter issues, confusing bug tracker, not on GitHub)

Let’s look at how we can fix both causes of project death.

I would argue that making sure people don’t leave should be the first priority, because when we only focus on people joining, they’ll leave as fast as we recruit them.

SUSTAINABLE OPEN SOURCE: HOW PROJECTS DIE IN TWO ACTS

1. People leave the project
2. No new people join

People leave the project:
- no more time
- no more interest
- no more energy to overcome project friction (like getting new features in, setting project direction, release chores, microaggressions)

No new people join:
- project not well presented (website, blogs, conferences, books)
- technically no longer relevant (maintenance mode)
- hard to get started (obscure programming language, no getting started docs, no internals handbook, no starter issues, confusing bug tracker, not on GitHub)

Let’s look at how we can fix both causes of project death.

I would argue that making sure people don’t leave should be the first priority, because when we only focus on people joining, they’ll leave as fast as we recruit them.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

➤ People > *

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
    - Lack of control

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward
  - Lack of community

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward
  - Lack of community
  - Absence of fairness

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward
  - Lack of community
  - Absence of fairness
  - Conflict in values

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward
  - Lack of community
  - Absence of fairness
  - Conflict in values
  - Work overload

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: RETAINING CONTRIBUTORS

- People > *
- Care about contributors, not contributions
- Avoid burnout at all cost (by Cate Houston / @catehstn)
  - Lack of control
  - Insufficient Reward
  - Lack of community
  - Absence of fairness
  - Conflict in values
  - Work overload
- Identify and remove micro aggressions

Community over code, people > *

Doesn’t have to be friendship, but professional care. Tell them it’s okay when they miss a deadline, or have stress at home, work. Appreciate them being part of the project, don’t define them by way of their contributions.

Lack of control: project things happen and contributors or even committers don’t know why. Document your processes.

Insufficient Reward: *cough* Merit *cough* contributor -> committer -> pmc member -> ASF member: not enough levels of recognition

Lack of community: people > *

Absence of fairness: tyranny of structurelessness, ASF equal playing field: there are always people/companies who are actively or passively trying to game the system

Conflict in values: have a clear mission statement, keep it up to date, talk about it regularly, frame everything you do as part of that mission statement: align people, outliers will leave (this is a good thing)

Work overload: Systematic: don’t rely on individuals for critical contributions, always make a process out of it (releases, docs, issue triage, user support, marketing, etc.). self-inflicted: ask people who contribute a lot to take regular breaks (force if necessary). If necessary, reduce the scope of the project until you can manage with the people you have contributing.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

➤ Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

- Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement
- Document all processes, make it easy for anyone to do anything on day one by following the process docs

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

- Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement
- Document all processes, make it easy for anyone to do anything on day one by following the process docs
- Have a discussion culture of “yes, and”, not “no, because”

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

- Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement
- Document all processes, make it easy for anyone to do anything on day one by following the process docs
- Have a discussion culture of “yes, and”, not “no, because”
  - smooth over cultural differences and language barriers

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

- Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement
- Document all processes, make it easy for anyone to do anything on day one by following the process docs
- Have a discussion culture of “yes, and”, not “no, because”
  - smooth over cultural differences and language barriers
- Be nice, science says so

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

SUSTAINABLE OPEN SOURCE: GAINING CONTRIBUTORS

- Have a Code of Conduct (Joan Touzet at CouchDB pioneered the ASF CoC) and Diversity Statement
- Document all processes, make it easy for anyone to do anything on day one by following the process docs
- Have a discussion culture of “yes, and”, not “no, because”
  - smooth over cultural differences and language barriers
- Be nice, science says so
- Money, Money, Money

CoC / Diversity Statement: signal that you welcome _everyone_, otherwise only white men will show up. This is the baseline, don’t start before you have this. CoC needs to be actionable, be prepared to enforce it.

Document everything: have a newcomer do a release: you win

“I think” considered weak in some cultures, leads to “rude”-sounding comments/mails, learn the source of the perceived rudeness / cultural difference, work together to find a solution.

Be nice: http://qz.com/625870/after-years-of-intensive-analysis-google-discovers-the-key-to-good-teamwork-is-being-nice/

Money: underrepresented people in tech tend to not have the same privilege of most of the people in this room: having enough spare time, or being paid to work on open source. We need programmes to support these people getting into Open Source. Luckily, these exist. Please ask your employers to support them: RailsGirls Summer of Code (not just Rails), Outreachy, etc.

THANK YOU!

CouchDB 2.0: Five Lessons Learned
Jan Lehnardt @janl jan@apache.org
Professional Support for Apache CouchDB: https://neighbourhood.ie