Local-first software
Resilient and secure collaboration

MARTIN KLEPPMANN
TU MUNICH

Bluesky: @martin.kleppmann.com
Mastodon: @martin@nondeterministic.computer

TUM  VolkswagenStiftung  Ink & Switch
Collaborative Applications

- Google Docs
- Office 365
- Overleaf
- Trello
- Figma

Wide range of domain-specific collaboration software, e.g. for investigative journalism, medical records, data analysis, engineering/CAD, ...
CLOUD SOFTWARE

e.g. Google

amazon web services

"The cloud" storage

User Alice

User Bob
CLOUD SOFTWARE

e.g. Google

amazon web services

"The cloud" storage

User Alice

User Bob
CLOUD SOFTWARE

e.g. Google

amazon web services

“The cloud” storage

User Alice

cloud not accessible?
lose access to all the documents you ever created — your life’s work!

User Bob
A mysterious message is locking Google Docs users out of their files

(Loic Venance/AFP/Getty Images)

By Brian Fung
October 31, 2017

Imagine you're working on a Google Doc when, seemingly out of nowhere, your ability to edit the online file gets revoked. What you see instead is an error message indicating that you've violated Google's terms of service.

Google refuses to reinstate man's account after he took medical images of son's groin

Experts say case highlights dangers of automated detection of child sexual abuse images

Google has refused to reinstate a man's account after it wrongly flagged medical images he took of his son's groin as child sexual abuse material (CSAM), the New York Times first reported. Experts say it's an inevitable pitfall of trying to apply a technological solution to a societal problem.
SUSPENDED ACCOUNTS

Microsoft: 182,000 in 2020
of which 730 (0.4%) were restored!

Google: “disables tens of millions of accounts per year”
“The vast majority are spammers and bots”

A precarious situation!

Source: https://tweakers.net/reviews/9094/all/account-geblokkeerd-wat-nu.html [in Dutch]
Cuneiform script on clay tablet, ca. 3000 BCE. Image from Wikimedia Commons. 
https://commons.wikimedia.org/wiki/File:Early_writing_tablet_recording_the_allocation_of_beer.jpg
WEB APP ARCHITECTURE THROUGH THE AGES

ca. 2000-2010

web browser → client

HTML ↑ ↓ submit forms

Python/Ruby/PHP/Node

Database → server
WEB APP ARCHITECTURE THROUGH THE AGES

ca. 2000-2010

```
web browser

HTML ↑ ↓ submit forms

Python/Ruby/PHP/Node

↑ ↓

Database
```
HTML DOM

User input

Render e.g. React

JS app state

e.g. Redux
HTML DOM → JS app state (e.g. Redux) → JSON REST API → model objects

User input → render (e.g. React)

RPC request → response

Request → response
HTML DOM → JS app state (e.g. React) → JSON REST API → model objects (as in "MVC") → ORM → database (e.g. SQL)

User input → JS app → JSON REST API → model objects → database

RPC request → JSON REST API → response

Response → JS app state (e.g. Redux)
HTML DOM → JS app state → JSON REST API → model objects → ORM → database

- User input:
  - HTML DOM
  - React
  - E.g. Redux

- JS app state:
  - Request
  - Response
  - E.g. Redux

- JSON REST API:
  - Request
  - Response

- Model objects:
  - As in "MVC"

- ORM:
  - I/O
    - Disk & Network
  - E.g. SQL

- Database:
  - Persistent storage
Six different representations of app state?! 

HTML DOM → JS app state (e.g. Redux) → JSON REST API → model objects (as in "MVC") → ORM → database (e.g. SQL) 

Lots of code is just converting data from one representation to another.
Pros:
- Real-time collaboration
- Access from any device, anywhere

Cons:
- Service shuts down?
  => everything lost
- Weak offline support
- Server, not user, owns data
CLOUD APPS / WEB APPS / SAAS

**Pros:**
- Real-time collaboration
- Access from any device, anywhere

**Cons:**
- Service shuts down?
  ⇒ everything lost
- Weak offline support
- Server, not user, owns data

"OLD-FASHIONED" SOFTWARE

**Pros:**
- Data & software on your own device: works offline, full control, backups, run software forever (in VM)

**Cons:**
- Manual file copying between devices & collaborators
- Only one person at a time can edit file (or manual merge)
CLoud Apps / WEB Apps / SaaS

Best of both worlds?

"Old-fashioned" Software
Cloud Apps / Web Apps / SaaS

Best of both worlds? \(\Rightarrow\) "Local-first"

"Old-fashioned" Software
LOCAL-FIRST SOFTWARE

- "The cloud" storage
  - Direct data sync between local storage and cloud
  - User Alice
  - User Bob
TRADITIONAL WEB APP MODEL:

"If it’s not stored in the server database, it didn’t really happen"

Can’t reach the server? Can’t do anything!
TRADITIONAL WEB APP MODEL:

"If it's not stored in the server database, it didn't really happen"

Can't reach the server? Can't do anything!

LOCAL-FIRST MODEL:

"The client's local storage is what matters — the server is just for multi-user sync and backup"

Don't care if we're online or offline right now!
TRADITIONAL WEB APP MODEL:

HTML DOM

user input

render e.g. React

JS app state

e.g. Redux

RPC request

response

JSON REST API

request

response

model objects

as in "MVC"

ORM

database e.g. SQL

I/O disk & network

persistent storage
TRADITIONAL WEB APP MODEL:

HTML DOM → JS app state → JSON REST API → model objects → ORM → database

User input → render e.g. React → e.g. Redux → request → response → request → response → as in "MVC" → I/O → disk & network → persistent storage

LOCAL-FIRST MODEL:

HTML DOM → JS app state → storage & sync

User input → render e.g. React → e.g. Automerge → I/O → disk & network

Self-contained!
End-to-end encryption for billions of users

WhatsApp

Signal

iMessage
End-to-end encryption:

encrypted

unencrypted

encrypted

untrusted

trusted

untrusted
Real-time collaboration is hard to implement

Built for real-time sync
<table>
<thead>
<tr>
<th>CLOUD SOFTWARE</th>
<th>LOCAL-FIRST SOFTWARE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real-time collaboration is hard to implement</td>
<td>Built for real-time sync</td>
</tr>
<tr>
<td>Does not work offline</td>
<td>Works offline</td>
</tr>
<tr>
<td>CLOUD SOFTWARE</td>
<td>LOCAL-FIRST SOFTWARE</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---------------------------------------</td>
</tr>
<tr>
<td>🙁 Real-time collaboration is hard to implement</td>
<td>☺ Built for real-time sync</td>
</tr>
<tr>
<td>🙁 Does not work offline</td>
<td>☺ Works offline</td>
</tr>
<tr>
<td>🙁 Service shuts down? Lose everything!</td>
<td>☺ Users can continue using local copy of software + data</td>
</tr>
<tr>
<td>CLOUD SOFTWARE</td>
<td>LOCAL-FIRST SOFTWARE</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>🙁 Real-time collaboration is hard to implement</td>
<td>😊 Built for real-time sync</td>
</tr>
<tr>
<td>🙁 Does not work offline</td>
<td>😊 Works offline</td>
</tr>
<tr>
<td>🙁 Service shuts down? Lose everything!</td>
<td>😊 Users can continue using local copy of software + data</td>
</tr>
<tr>
<td>🙁 Custom service for each app (infra, ops, on-call rotation, ...)</td>
<td>😊 Sync service is generic ⇒ outsource to cloud vendor</td>
</tr>
<tr>
<td>Local-first is a good fit for:</td>
<td>Local-first is a bad fit for:</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>&quot;File editing&quot; software</td>
<td></td>
</tr>
<tr>
<td>(text editor, word processor,</td>
<td></td>
</tr>
<tr>
<td>spreadsheet, presentation</td>
<td></td>
</tr>
<tr>
<td>slides, graphics editor,</td>
<td></td>
</tr>
<tr>
<td>video editing, music production,</td>
<td></td>
</tr>
<tr>
<td>CAD software for engineering,</td>
<td></td>
</tr>
<tr>
<td>Jupyter notebooks, ...)</td>
<td></td>
</tr>
<tr>
<td>Local-first is a good fit for:</td>
<td>Local-first is a bad fit for:</td>
</tr>
<tr>
<td>-------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>&quot;File editing&quot; software</td>
<td></td>
</tr>
<tr>
<td>(text editor, word processor, spreadsheet, presentation slides,</td>
<td></td>
</tr>
<tr>
<td>graphics editor, video editing, music production, CAD software</td>
<td></td>
</tr>
<tr>
<td>for engineering, Jupyter notebooks, ...)</td>
<td></td>
</tr>
<tr>
<td><strong>Productivity software</strong></td>
<td></td>
</tr>
<tr>
<td>(notes, to-do lists, issue trackers, calendar, time tracking,</td>
<td></td>
</tr>
<tr>
<td>group messaging, bookkeeping, ...)</td>
<td></td>
</tr>
<tr>
<td>Basically, apps where the user can edit the data however they</td>
<td></td>
</tr>
<tr>
<td>like.</td>
<td></td>
</tr>
<tr>
<td>Local-first is a good fit for:</td>
<td>Local-first is a bad fit for:</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>&quot;File editing&quot; software</td>
<td>Managing a real-world resource, e.g.</td>
</tr>
<tr>
<td>(text editor, word processor, spreadsheet, presentation slides, graphics editor, video editing, music production, CAD software for engineering, Jupyter notebooks, ...)</td>
<td>- money (bank account, payments, ad impressions)</td>
</tr>
<tr>
<td>Productivity software</td>
<td>- physical products (e-commerce, warehouse inventory)</td>
</tr>
<tr>
<td>(notes, to-do lists, issue trackers, calendar, time tracking, group messaging, bookkeeping, ...)</td>
<td>- vehicles (car-sharing/rental, freight/logistics)</td>
</tr>
</tbody>
</table>

Basically, apps where the user can edit the data however they like. For these apps, a centralised cloud/server model works best.
AUTOMERGE: "Git for your app's data"

```json
{"todos": [
  {
    "title": "buy milk", "done": false
  },
  {
    "title": "water plants", "done": false
  }
]}
```
AUTOMERGE: “Git for your app’s data”

```json
{
  "todos": [
    {
      "title": "buy milk", "done": false
    },
    {
      "title": "water plants", "done": false
    }
  ]
}
```

```javascript
after = Automerge.change(before, "mark item as done", doc => {
  doc.todos[1].done = true;
});
```
AUTOMERGE: "Git for your app's data"

```json
{ "todos": [
  { "title": "buy milk", "done": false },
  { "title": "water plants", "done": true }
]
}
```

```javascript
after = Automerge.change(before, "mark item as done", doc => {
  doc.todos[1].done = true;
});
```
AUTOMERGE: "Git for your app's data"

```json
{ "todos": [
    { "title": "buy milk", "done": false },
    { "title": "water plants", "done": true }
]
}
```

```javascript
after = Automerge.change(before, "mark item as done", doc => {
    doc.todos[1].done = true;
});
```
AUTOMERGE: “Git for your app’s data”

```json
{
  "todos": [
    {
      "title": "buy milk", "done": false
    },
    {
      "title": "water plants", "done": false
    }
  ]
}
```

reflects updated state

immutable

"commit message" (optional)

```javascript
after = Automerge.change(before, "mark item as done", doc => {
  doc.todos[1].done = true;
});
```
AUTOMERGE: "Git for your app's data"

```json
{
    "todos": [
        {
            "title": "buy milk",  "done": false
        },
        {
            "title": "water plants",  "done": true
        }
    ]
}
```

- reflects updated state
- immutable
- "commit message" (optional)

```javascript
after = Automerge.change(before, "mark item as done", doc => {
    doc.todos[1].done = true;
});
```

record exactly what changed
AUTOMERGE: “Git for your app's data”

```json
{
  "todos": [
    {
      "title": "buy milk", "done": false
    },
    {
      "title": "water plants", "done": false
    }
  ]
}```
AUTOMERGE: "Git for your app’s data"

```javascript

let todos = [{
    title: "buy milk", done: false,
    title: "water plants", done: false
},

after = Automerge.change(before, "add new item", doc => {
    doc.todos.push({
        title: "do laundry", done: false
    });
});
```
AUTOMERGE: "Git for your app's data"

```json
{"todos": [  {
    "title": "buy milk",  "done": false
  },
  {
    "title": "water plants",  "done": false
  },
  {
    "title": "do laundry",  "done": false
  }
]}
```

```javascript
after = Automerge.change(before, "add new item", doc => {
  doc.todos.push({title: "do laundry", done: false});
});
```
AUTOMERGE: "Git for your app's data"

```json
{ "todos": [ 
    { "title": "buy milk", "done": false },
    { "title": "water plants", "done": false },
    { "title": "do laundry", "done": false }
  ]
}
```

```javascript
after = Automerge.change(before, "add new item", doc => {
  doc.todos.push({ title: "do laundry", done: false });
});
```
AUTOMERGE: Branching and merging

```json
{"todos": [
  {"title": "buy milk",
    "done": false},
  {"title": "water plants",
    "done": false}]
}
```
AUTOMERGE: Branching and merging

USER A:

```
{ "todos": [
    { "title": "buy milk", "done": false },
    { "title": "water plants", "done": true }]
}
```
AUTOMERGE: Branching and merging

USER A:

```json
"todos": [
  {
    "title": "buy milk",
    "done": false
  },
  {
    "title": "water plants",
    "done": true
  }
]
```

USER B:

```json
"todos": [
  {
    "title": "buy milk",
    "done": false
  },
  {
    "title": "water plants",
    "done": false
  },
  {
    "title": "do laundry",
    "done": false
  }
]
```
AUTOMERGE: Branching and merging

USER A:

```json
{ "todos": [ 
  { "title": "buy milk", "done": false },
  { "title": "water plants", "done": true }
]
}
```

USER B:

```json
{ "todos": [ 
  { "title": "buy milk", "done": false },
  { "title": "water plants", "done": false },
  { "title": "do laundry", "done": false }
]
}
```

Automerger change

merge
Example: Text editing

“Hello!”

“Hello!”

(time)
Example: Text editing

"Hello!"  "Hello World!"

insert "World" after "Hello"

"Hello!"  "Hello! :-)"

insert ":-)" after "!"
Example: Text editing

"Hello!"  "Hello World!"  "Hello World! :-)"

insert "World" after "Hello"
insert ":-)" after "!"

"Hello!"  "Hello! :-)"  "Hello World! :-)"

insert...
Automerge guarantees:

- All changes are preserved
- If two users have seen the same set of changes (in any order), then they are in the same state
- Branches (=concurrent updates) can be merged automatically
- Can branch & merge arbitrarily, can inspect/compare versions

Automerge is a CRDT
(conflict-free replicated data type)
Rust API

Automerge (core algorithms in Rust)
Rust API
JS/Typescript
WebAssembly

Automerge (core algorithms in Rust)
Automerge (core algorithms in Rust)
Automerge (core algorithms in Rust)
Automerge (core algorithms in Rust)
Cross-platform apps

Rust API
JS/Typescript
WebAssembly
Go API
C API
Swift API
Kotlin/Android
Java API

Automerge (core algorithms in Rust)
Resources

Local-first https://www.inkandswitch.com/local-first/
Automerge https://automerge.org/
My work https://martin.kleppmann.com/
Email martin@kleppmann.com
Bluesky @martin.kleppmann.com
Mastodon @martin@nondeterministic.computer

Huge thanks to the Automerge community and contributors, especially Alex Good, Peter van Hardenberg, Orion Henry, Andrew Jeffery, Herb Caudill, Alex Currie-Clark, Jason Kankiewicz, Conrad Irwin, and many others!

Thank you to my Patreon backers and institutional supporters: