HAILS

Protecting Data Privacy in Untrusted Web Apps

Amit Levy

with Deian Stefan & David Mazieres

Stanford SCS Lab

- Formal work
  - Language support for IFC
  - Side-channels
- Mostly focused on building systems
  - tcpcrypt, HiStar, Cinder, CoralCDN...
- Today - Hails

Our Private Data is Everywhere

github:develop

It's Connected!

Google health

Basecamp API

foursquare DEVELOPERS

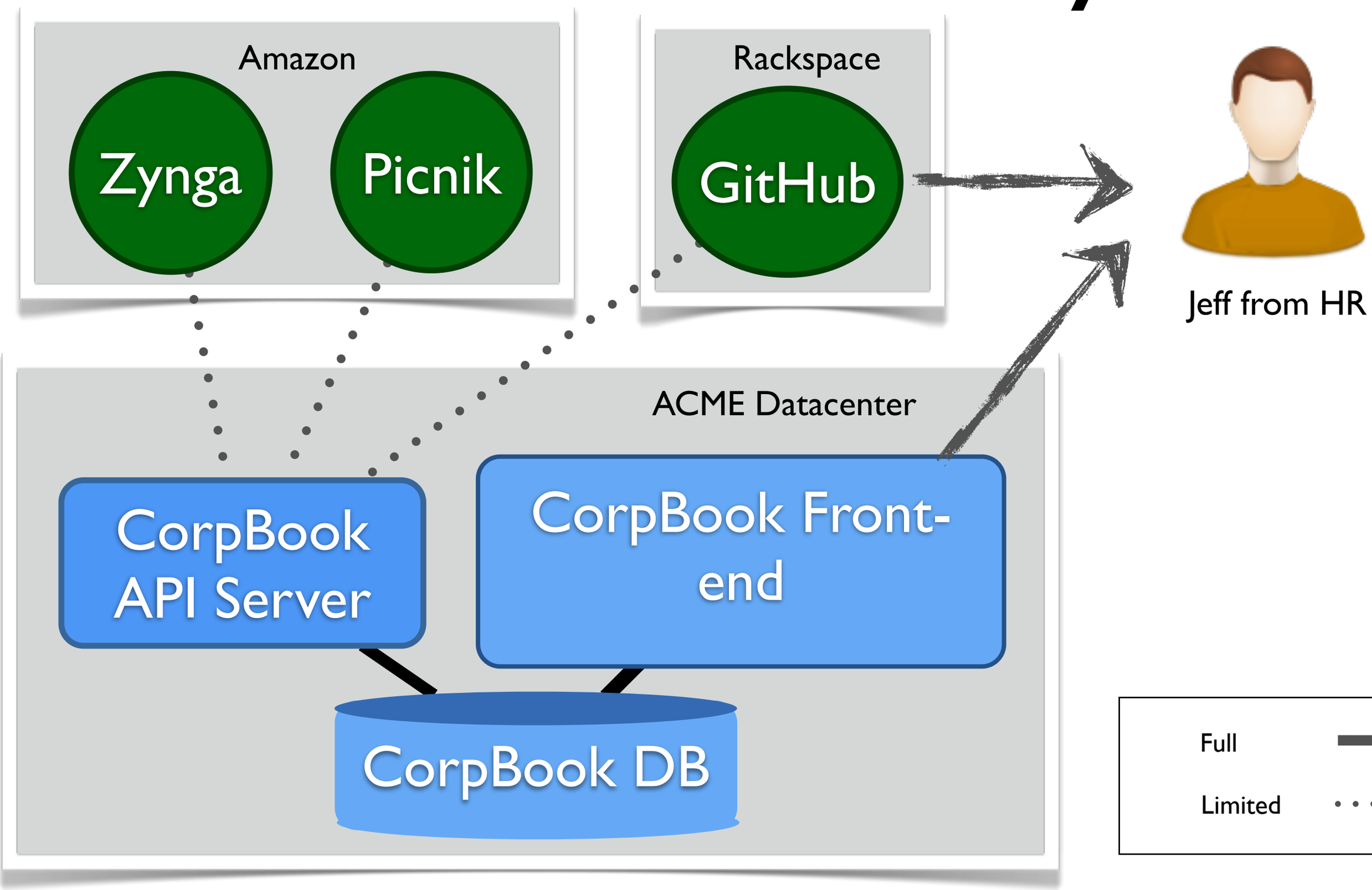
This is great!

- Richer experience
  - Mashups
  - Personalization
- Cheap to innovate
- Lots of options for consumers

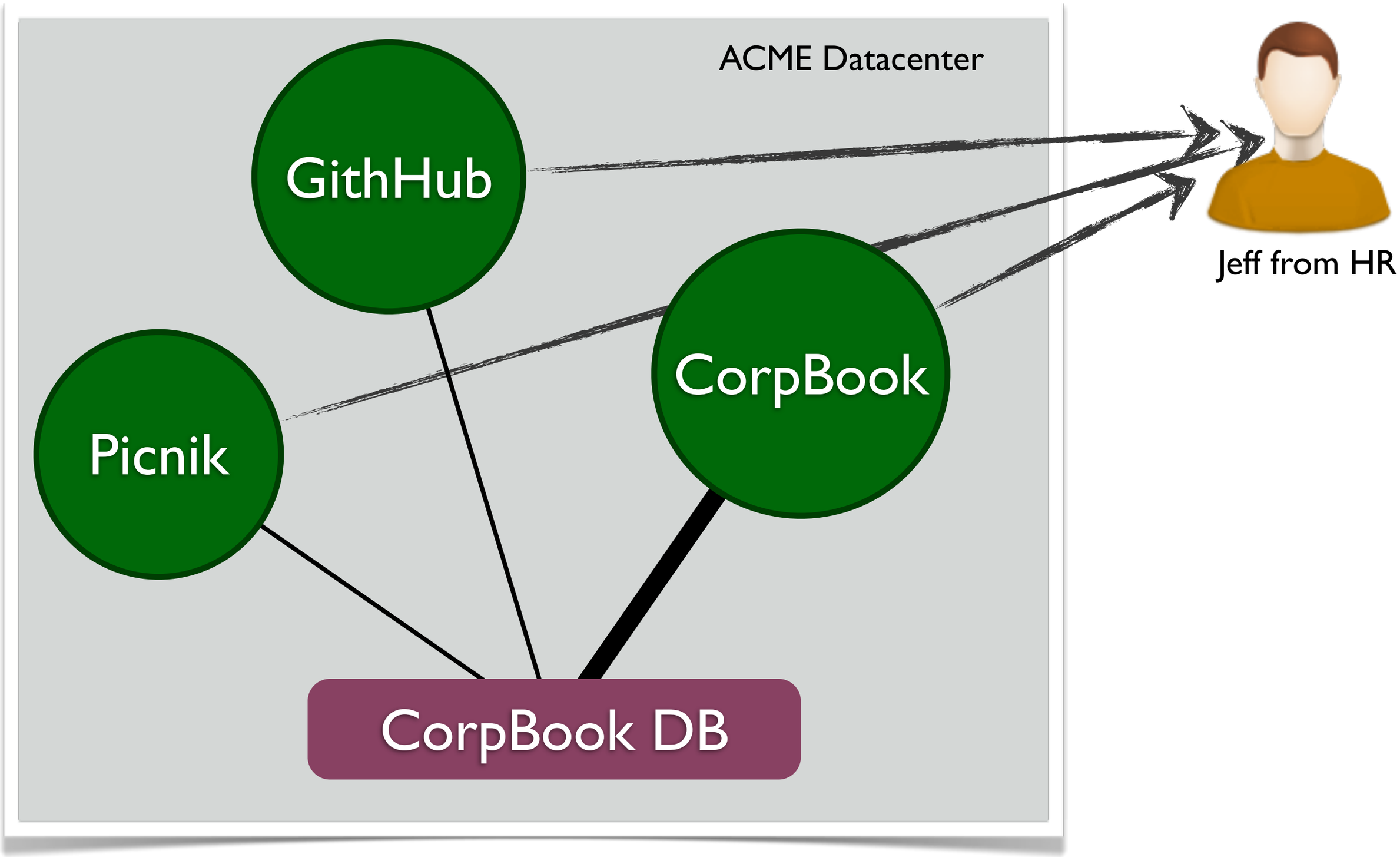
But it's also a problem

- Can't enforce policies on other applications
- Must resort to coarse grain access control
  - Yelp ***can*** access my Facebook data and do it's bidding
  - or Yelp ***cannot*** access data at all
- Over-share but under-deliver
  - Forced to choose between privacy and features
  - Get neither

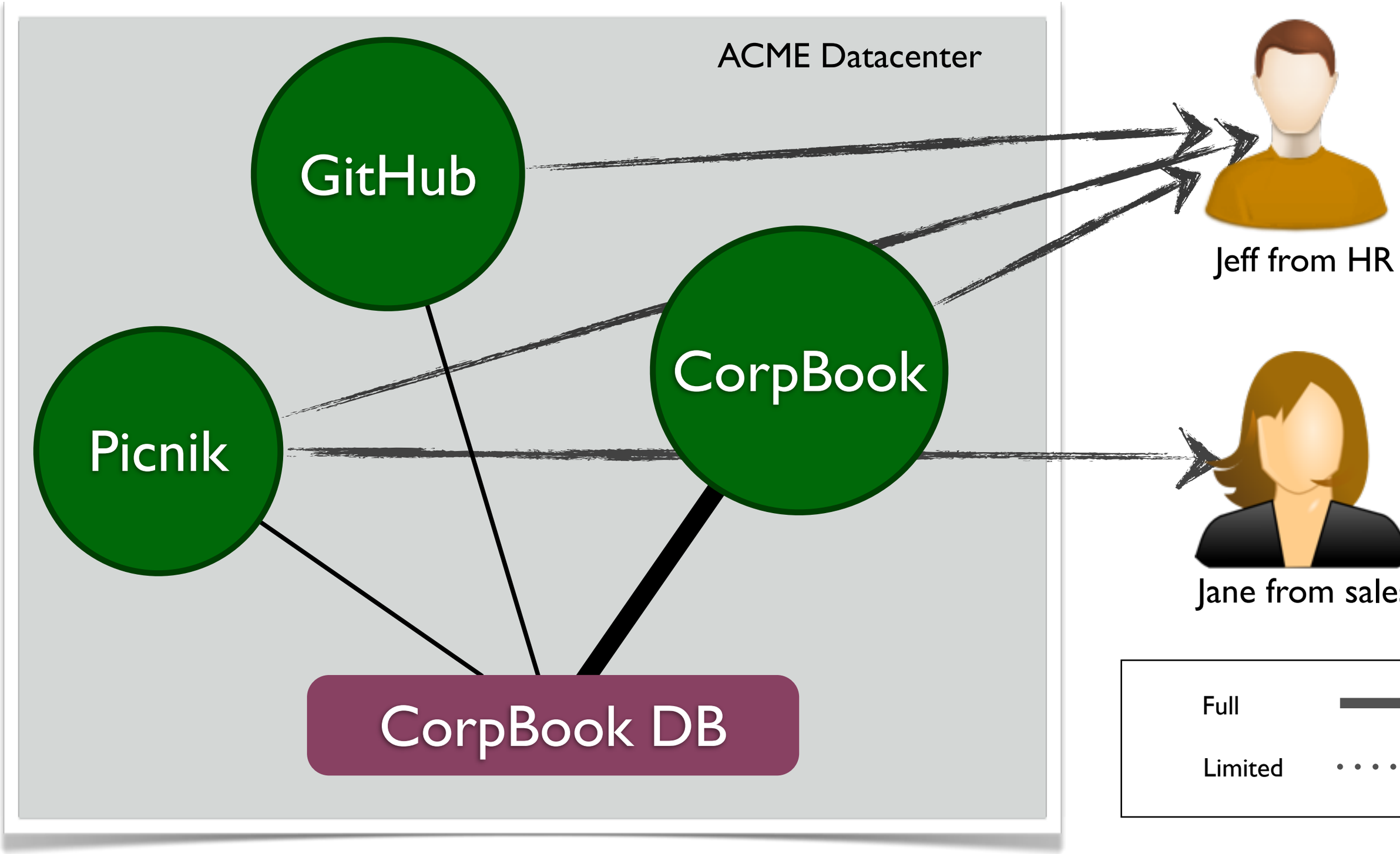
Web APIs Today



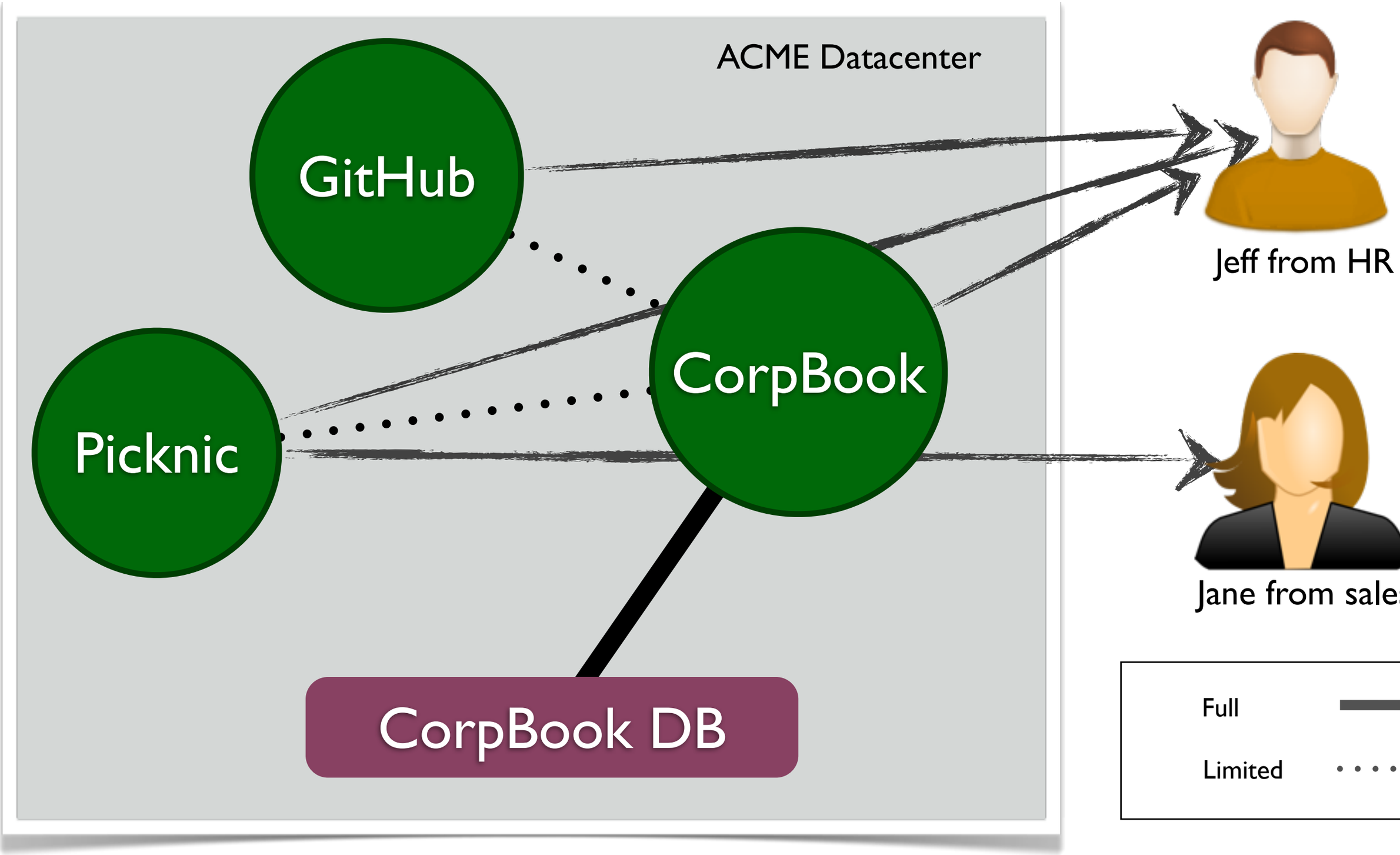
What about centralizing?



Centralizing Not Enough



Centralizing Not Enough



~~*How to reduce unemployment with...*~~

~~*How to cure cancer with...*~~

~~*How to alleviate hunger with...*~~

How to protect web data privacy with HAILS

What is HAILS?

- Multi-Application Web Platform
- Information Flow Control (IFC) to enforce policies on data
- Leverages LIO framework in Haskell
- DCLabels for policies
- Enforces fine-grained policies on untrusted apps with high performance

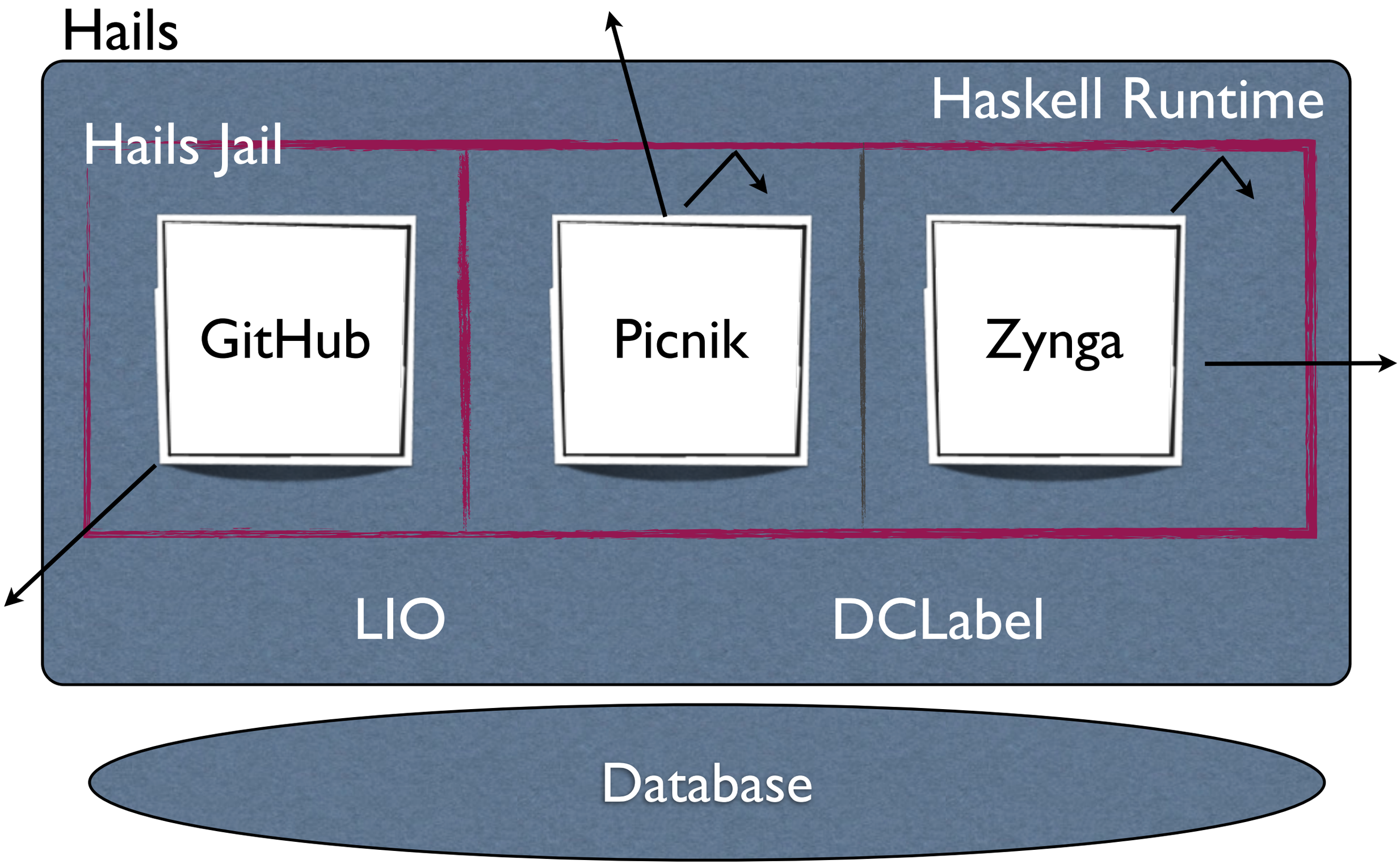
Goal

To allow untrusted web applications access to users' entire data while ensuring that they do not violate policies set on that data and without sacrificing functionality.

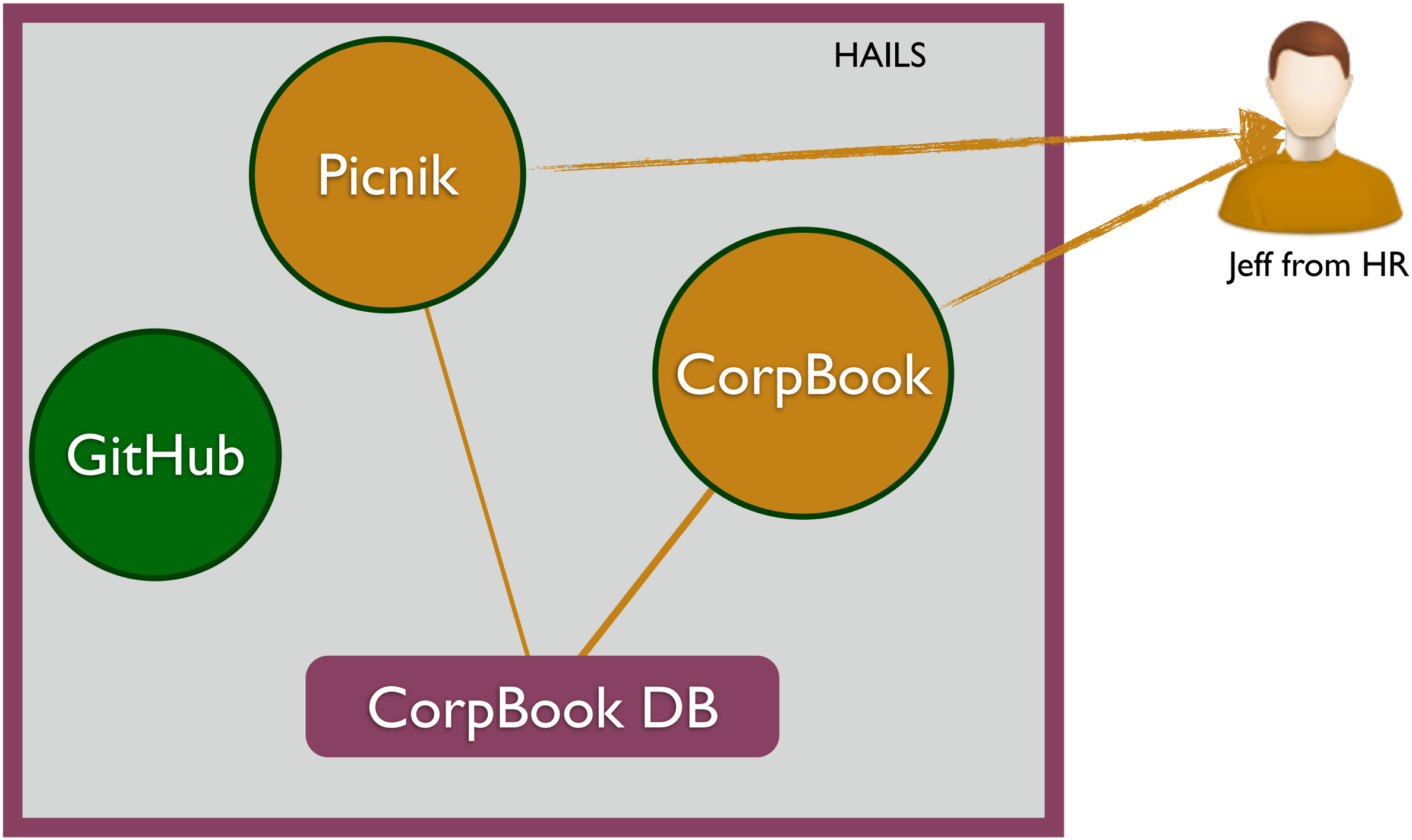
Why IFC?

- Today's web policies restrict what data apps can see or who they can connect to
- Real concerns relate to where data can flow
- Replace "Picnik can see my photos because I trust Picnik not to show them to my boss."
- With "Any app can see my photos as long as my boss doesn't see them."

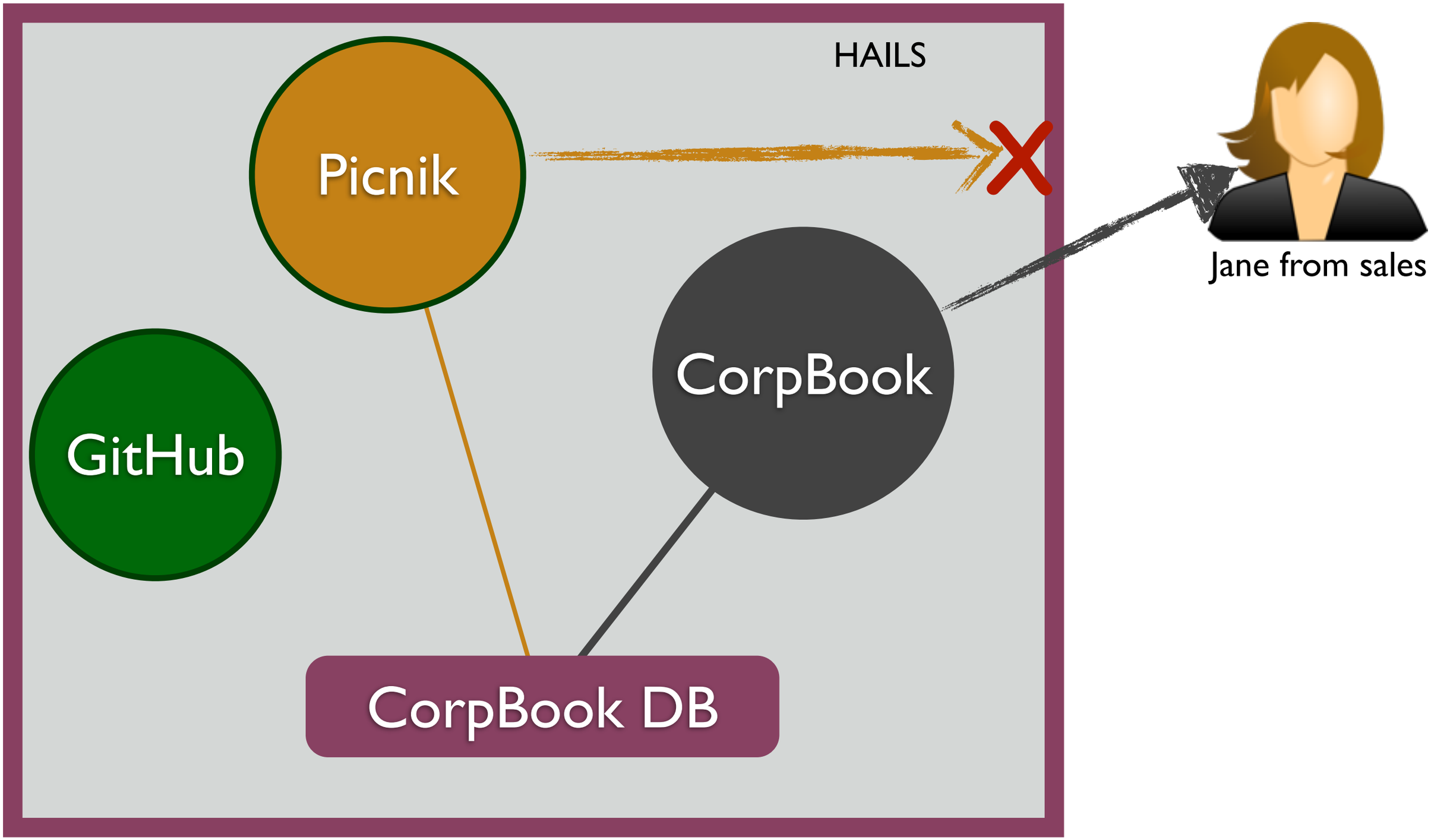
HAILS Web Stack



HAILS Architecture



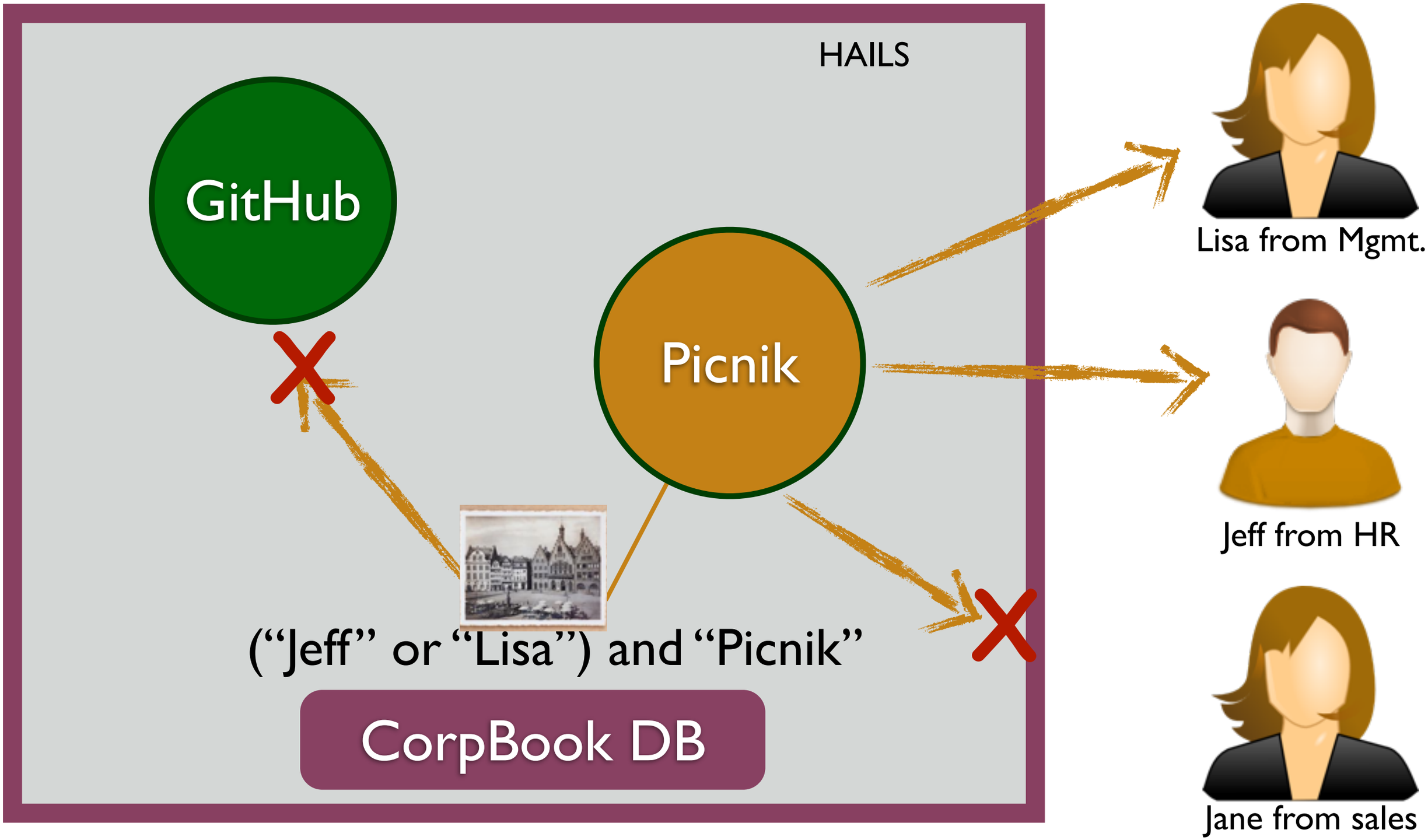
HAILS Architecture



DCLabels

- Disjunction Category Labels
  - (“amit” or “deian” or “david”)
    - Matches the kinds of policies we want to express in the web
- Powerful enough to express today’s polcies:
  - (“amit” or “deian” or “david”) and “corpbook”

DCLabels in Action



GitStar

- An extensible social code hosting application
- Like GitHub but ***better!***™
- How would we build if from many small mutually distrustful components
- Project management, issues, messaging, newsfeed, wiki etc' are all separate apps
- They rely on each other's data
- Launch by April 11th!

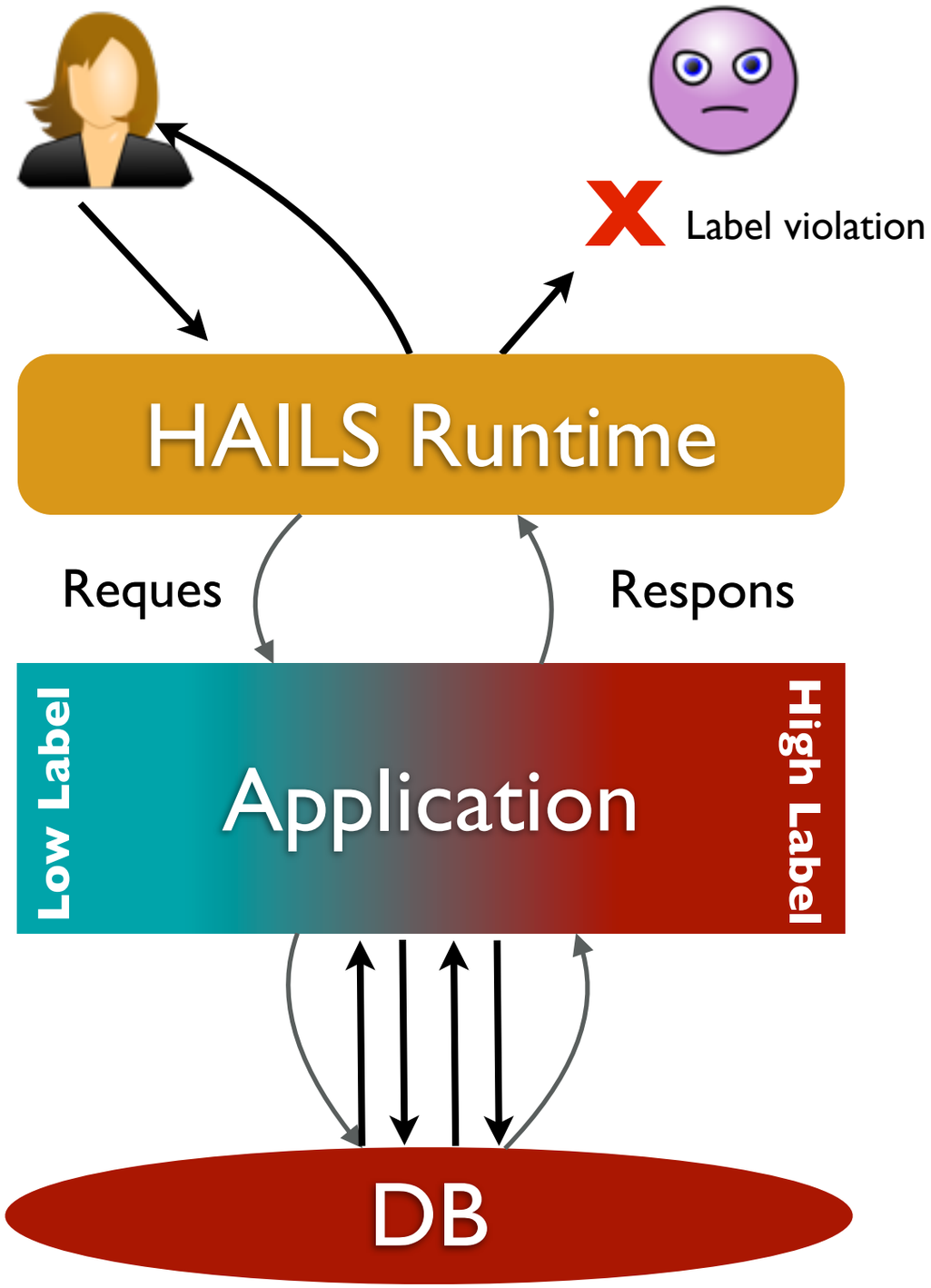
Check us out!

- http://github.com/scslab (soon to be at gitstar.com!)
  - hails
  - gitstar
  - dclabel
  - lio

Thanks! Questions?

Lifetime of a request

- TCB accepts HTTP request
- HAILS login
- Proxy to app with clearance based on user
- Label starts low
- Reading from database raises label
- Label check by HAILS on HTTP response



Hails DB Model

Database (“gitstar”)

DBLabel - (ALL) (ALL)

“messages”

(ALL) (ALL)

“news feeds”

(ALL) (“gitstar”)

“projects”

(ALL) (“gitstar”)

(ALL) (“#linux” ∨ “gitstar”)

(ALL) (“#hails” ∨ “gitstar”)

(“#ms_dos” ∨ “gitstar”) (“#ms_dos” ∨ “gitstar”)

•
•
•

Hails Policy Modules

- Policy modules moderate unlabeled DB with labeled apps
- Each policy module "owns" a single database
- Transforms unlabeled MongoDB documents (JSON)
- Defines which collections are available

Hails Policy Modules

```
lcollections = newDC (<>) ("gitstar" :: String)
lpub = newDC (<>) (<>)

projectsCollection :: DC (Collection DCLabel)
projectsCollection = collection "projects" lpub lpub $
  RawPolicy (\doc -> newDC (<>) (“#” ++ doc ! “name” .\/. “gitstar”))
            [("name", SearchableField)
             (“repo”, FieldPolicy $ (\doc ->
                newDC (“#” ++ doc ! “name” .\/. “gitstar”) (<>)))]

configDB :: DBConf -> DC (Database DCLabel)
configDB conf = do
  db <- labelDatabase conf lcollections lpub
  let priv = dbConfPriv conf
  myUsersCollection <- usersCollection
  assocCollectionP priv myUsersCollection db
```