Pipe[r] Dreams: Making NoSQL Great ~~Again~~

Kathryn Dahlgren
DisorderlyLabs@UCSC

HPTS, 10Oct2017

TL;DR

- NoSQL is great, but could be better.

- **<u>Problem:</u>**
  The k-implementation of database management functionality across the total set of instances of NoSQL products around the world is sub-optimal.

- **<u>Goals:</u>**
  1. Motivation?
  2. Articulate solution qualities.
  3. Present **Piper** as a model.

BACKGROUND

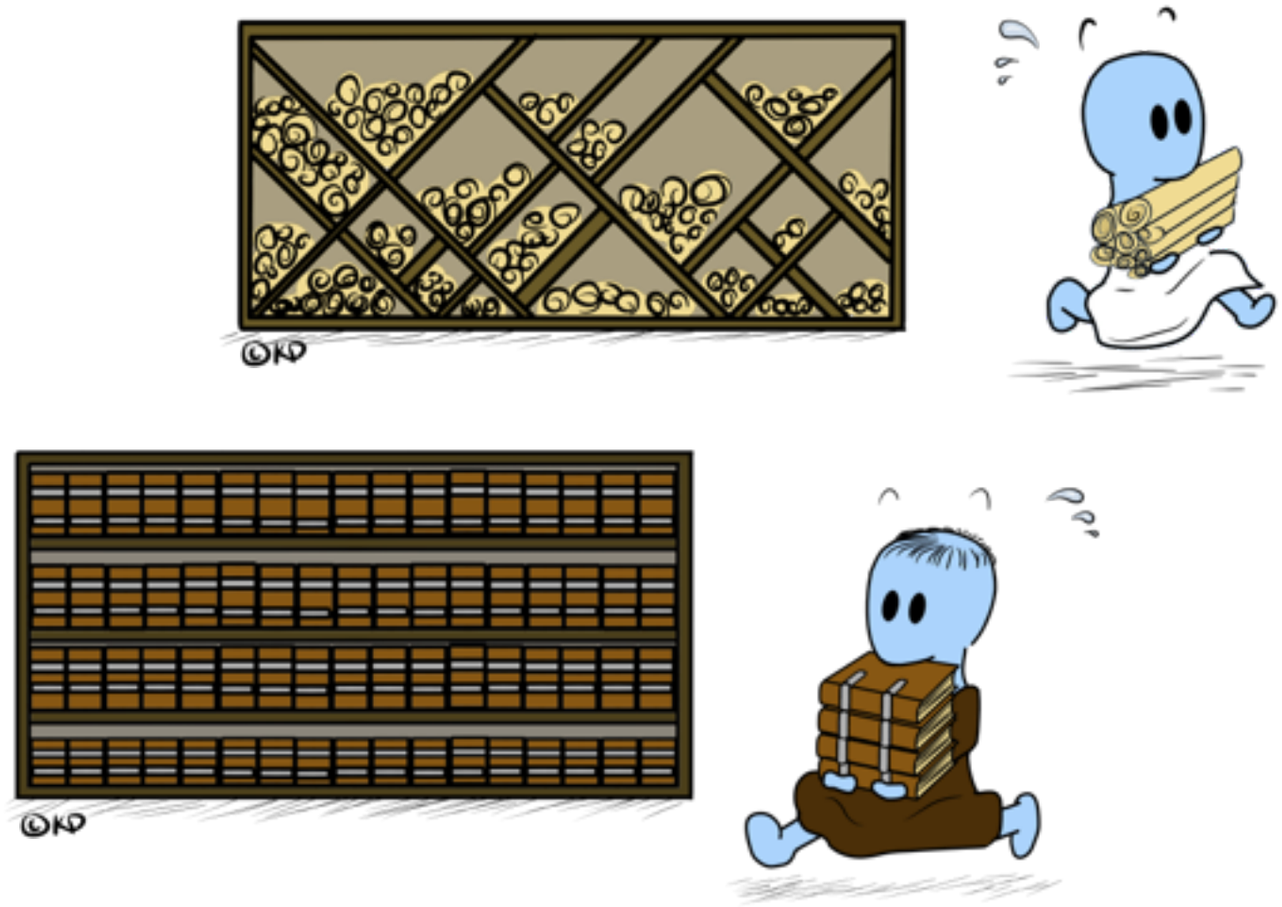
+

MOTIVATION

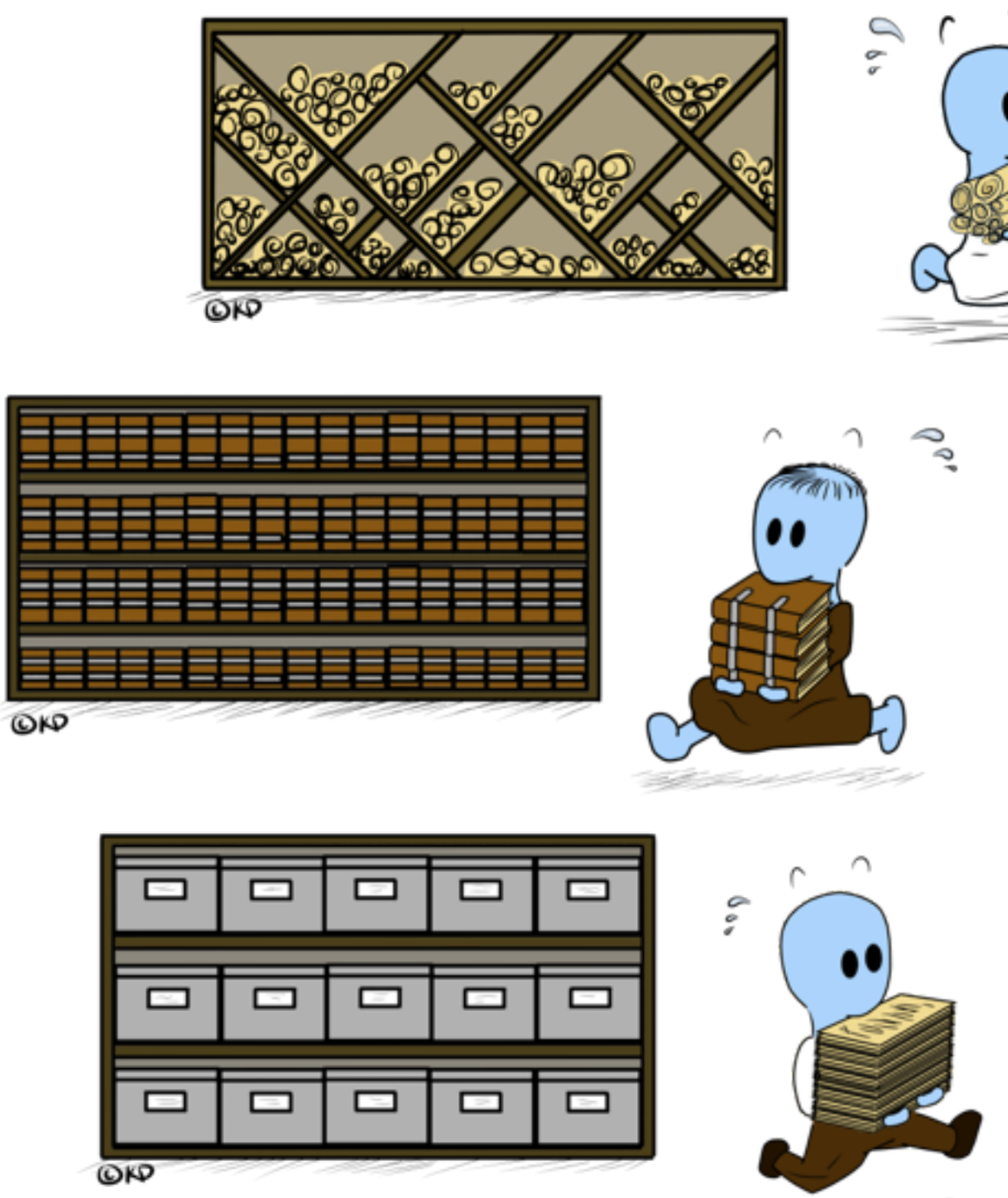
Ancestral Database Systems

Ancestral Database Systems

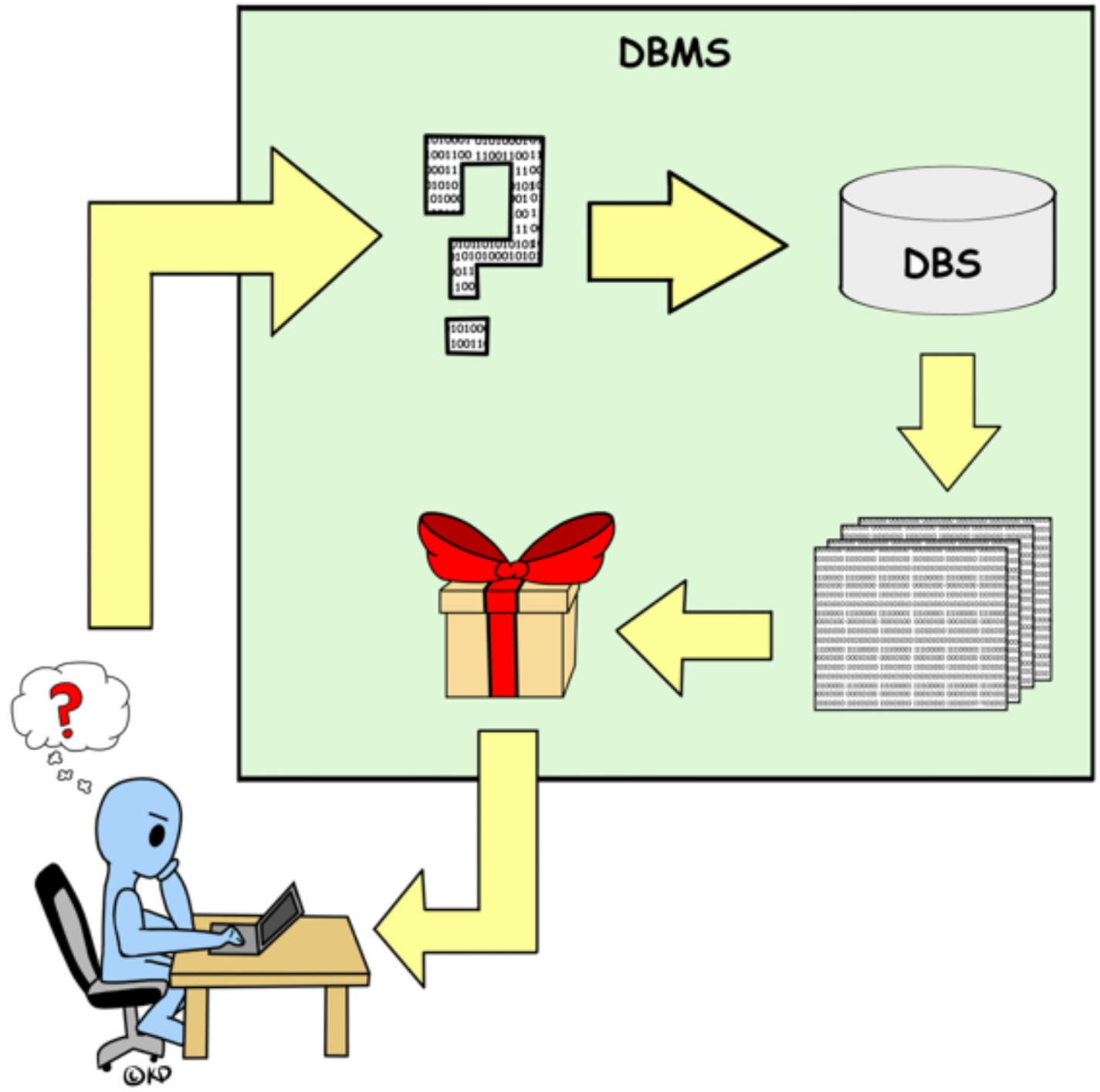
Ancestral Database Systems

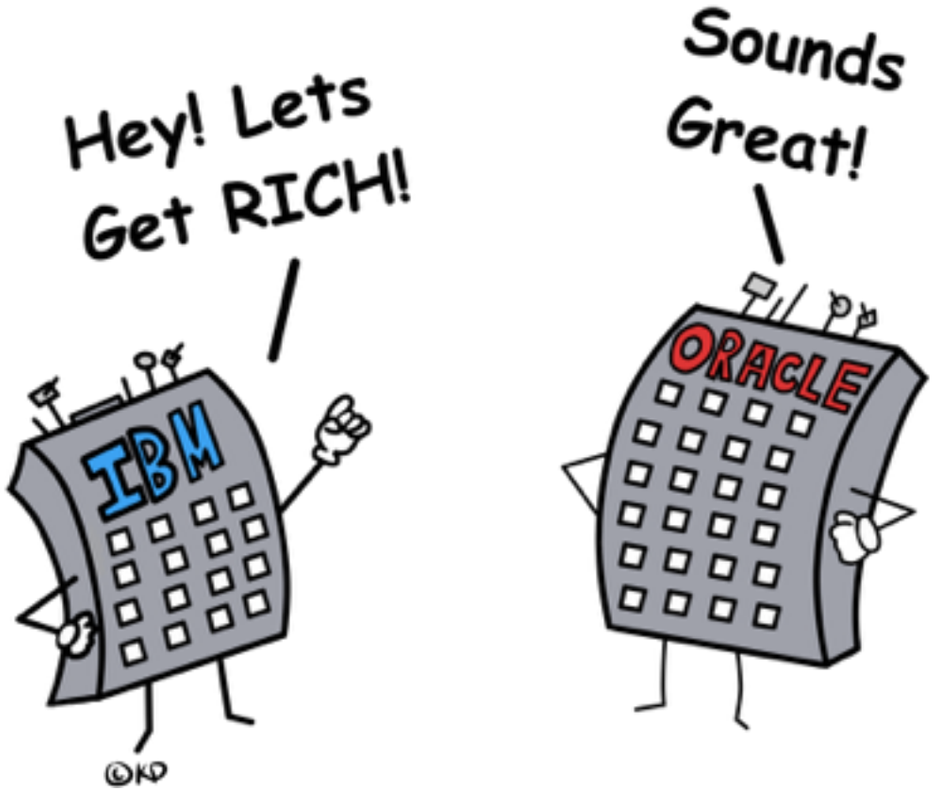


Ancestral Database Systems

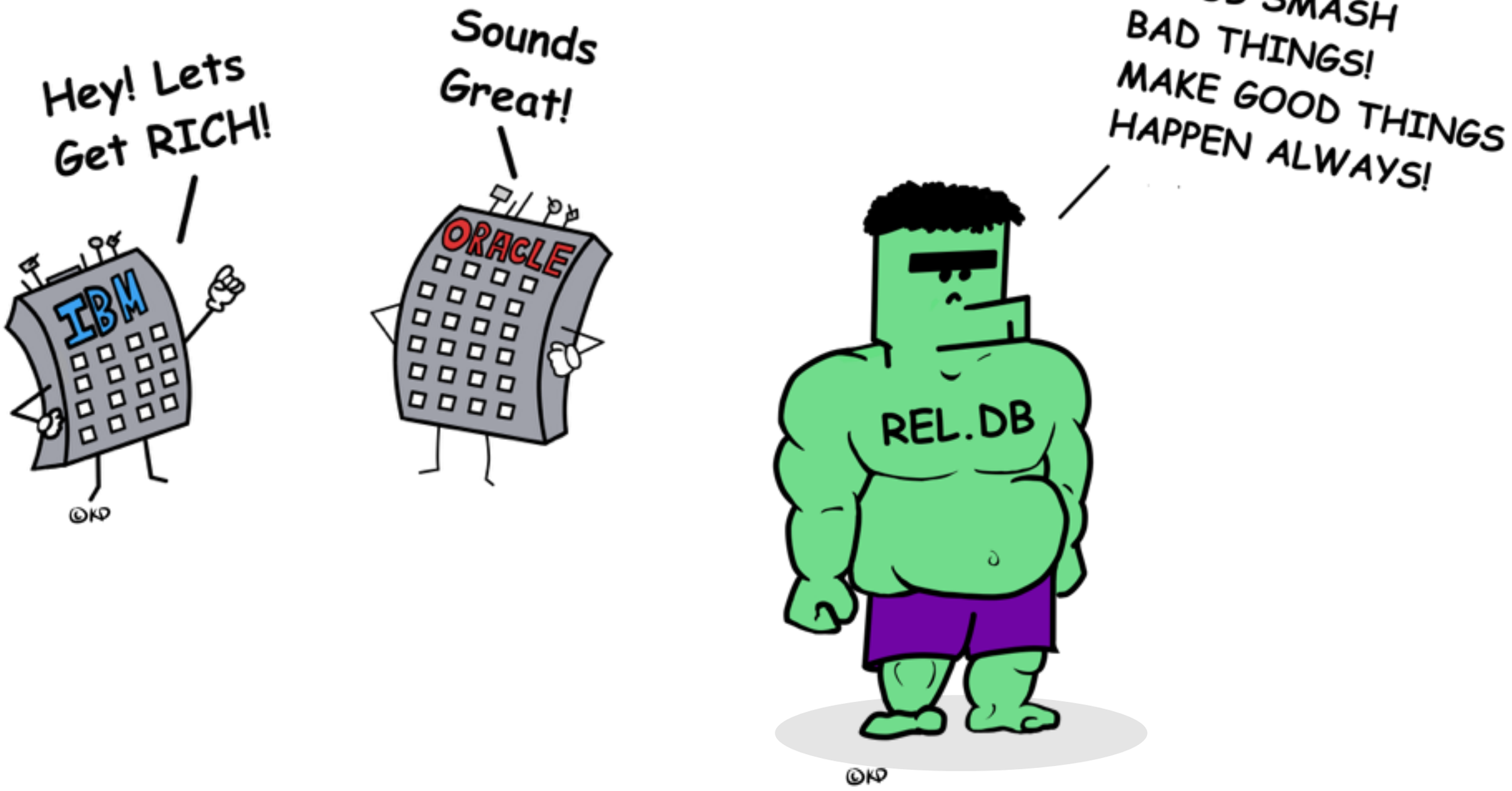


Database *Management* Systems

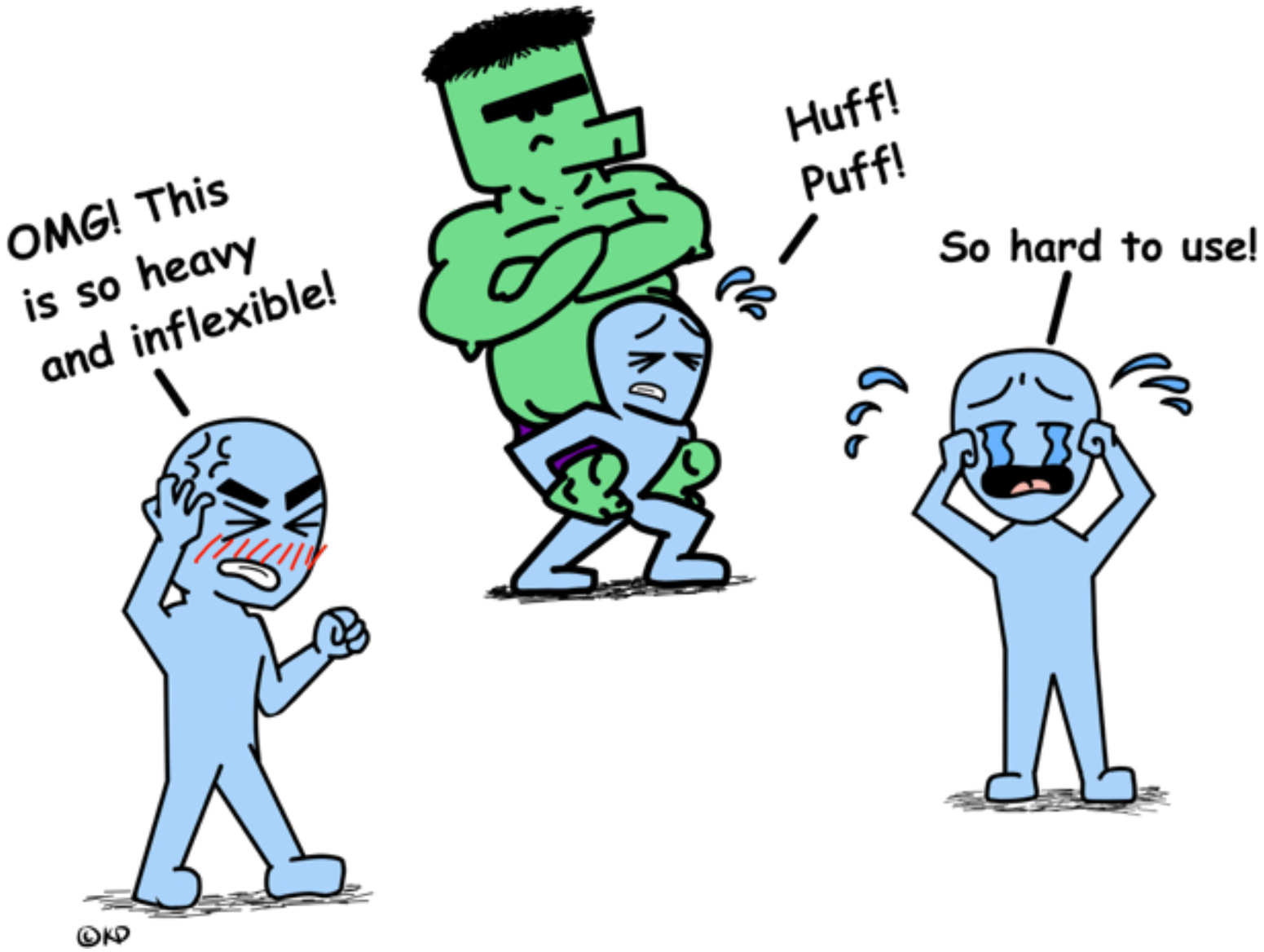




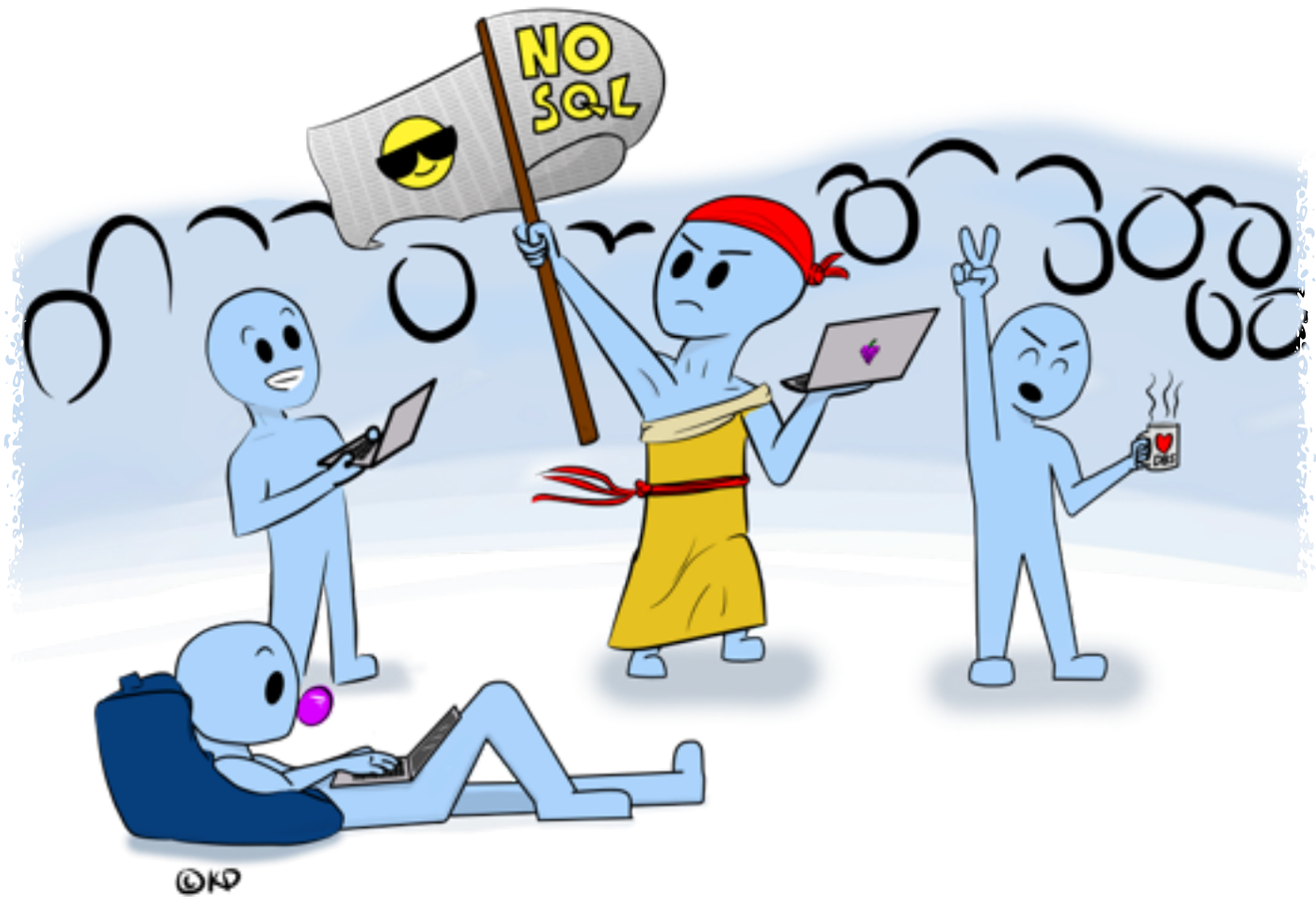
1980s-1990s


Hey! Lets Get RICH!
IBM
©KD
Sounds Great!
ORACLE
RELDB SMASH BAD THINGS! MAKE GOOD THINGS HAPPEN ALWAYS!
REL.DB
©KD

1980s-1990s


OMG! This is so heavy and inflexible!
Huff! Puff!
So hard to use!
©KD

1990s

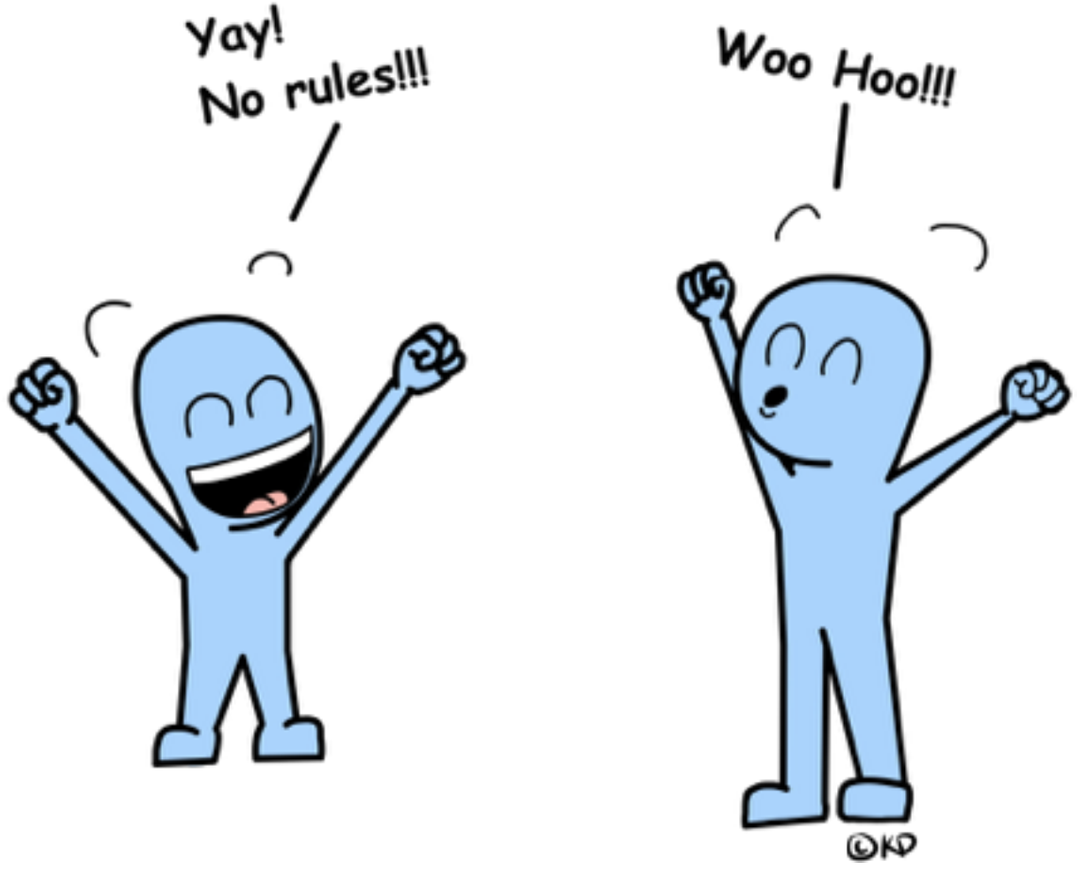


- Revolution!
- Coincides w/ rise of Internet + code repos (SourceForge, Github, …)
- 200+ NoSQL tools in existence (http://nosql-database.org/index.html)

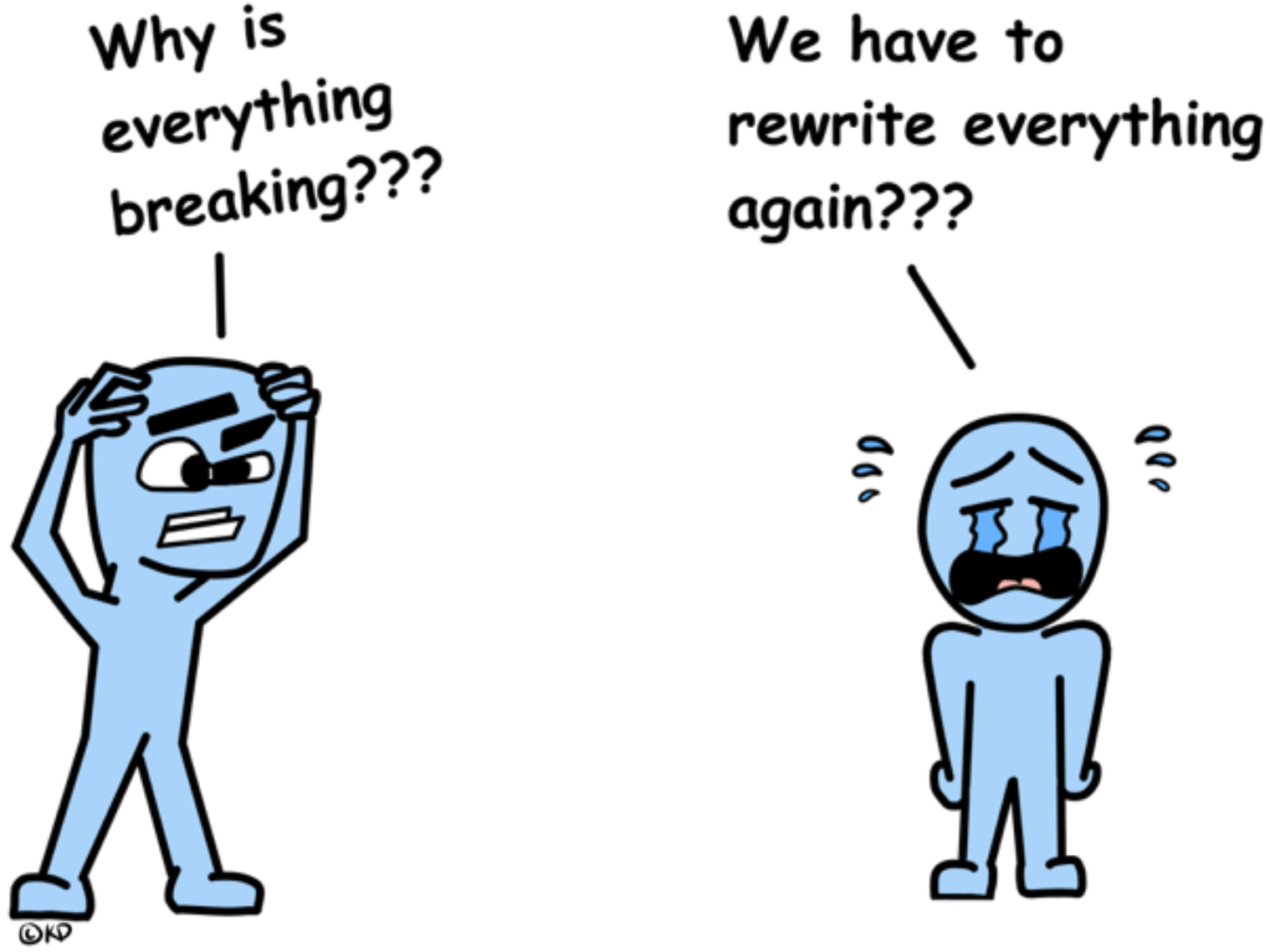
Early 2000s

What makes NoSQL Great?

- Flexibility
- Customizability

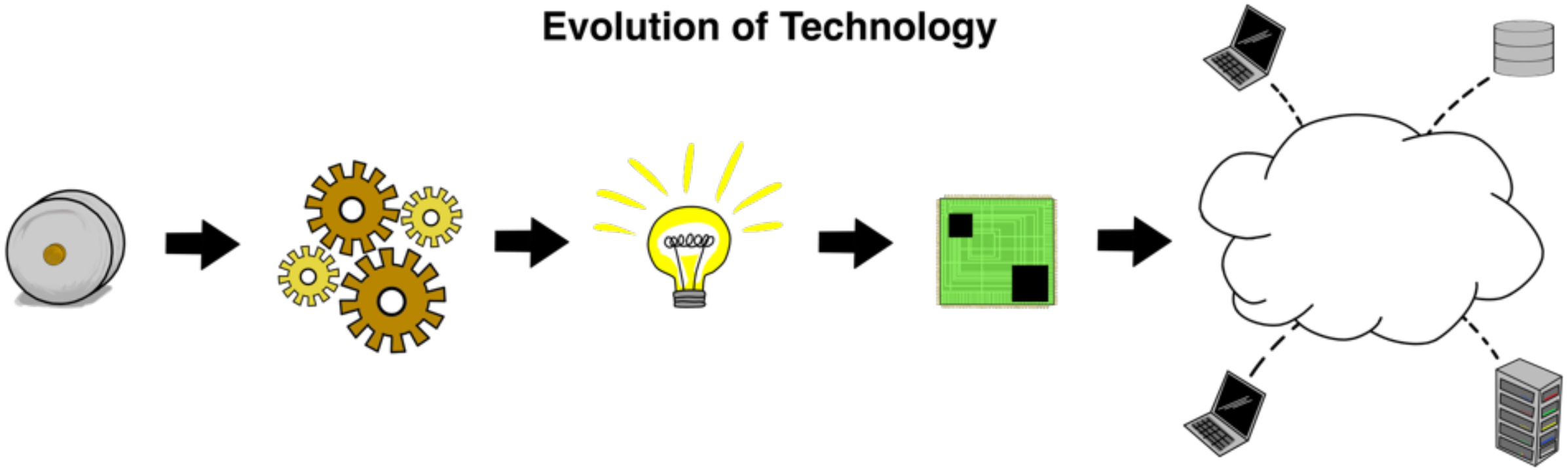


2000s

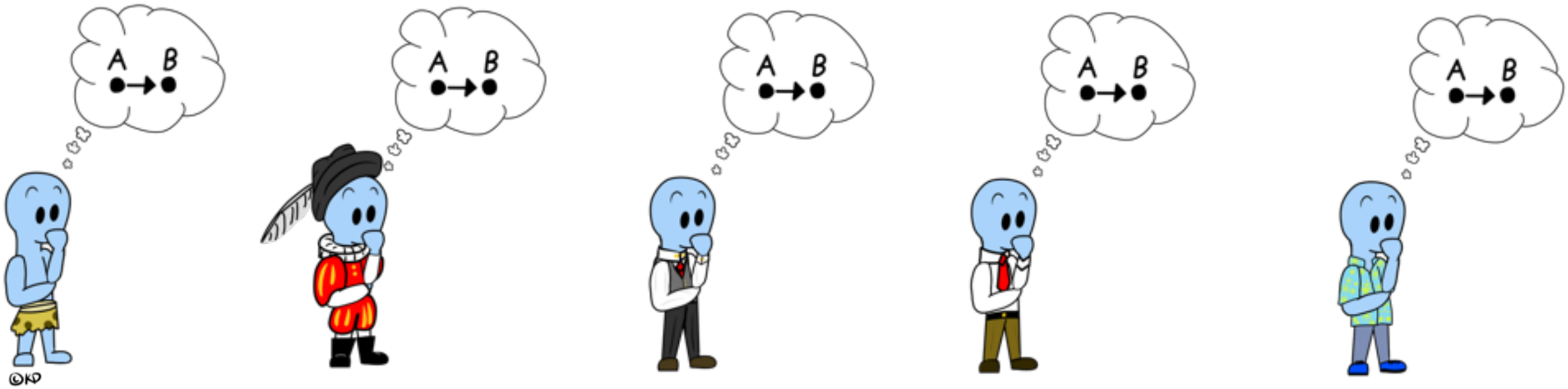

Why is everything breaking???
We have to rewrite everything again???
©KD

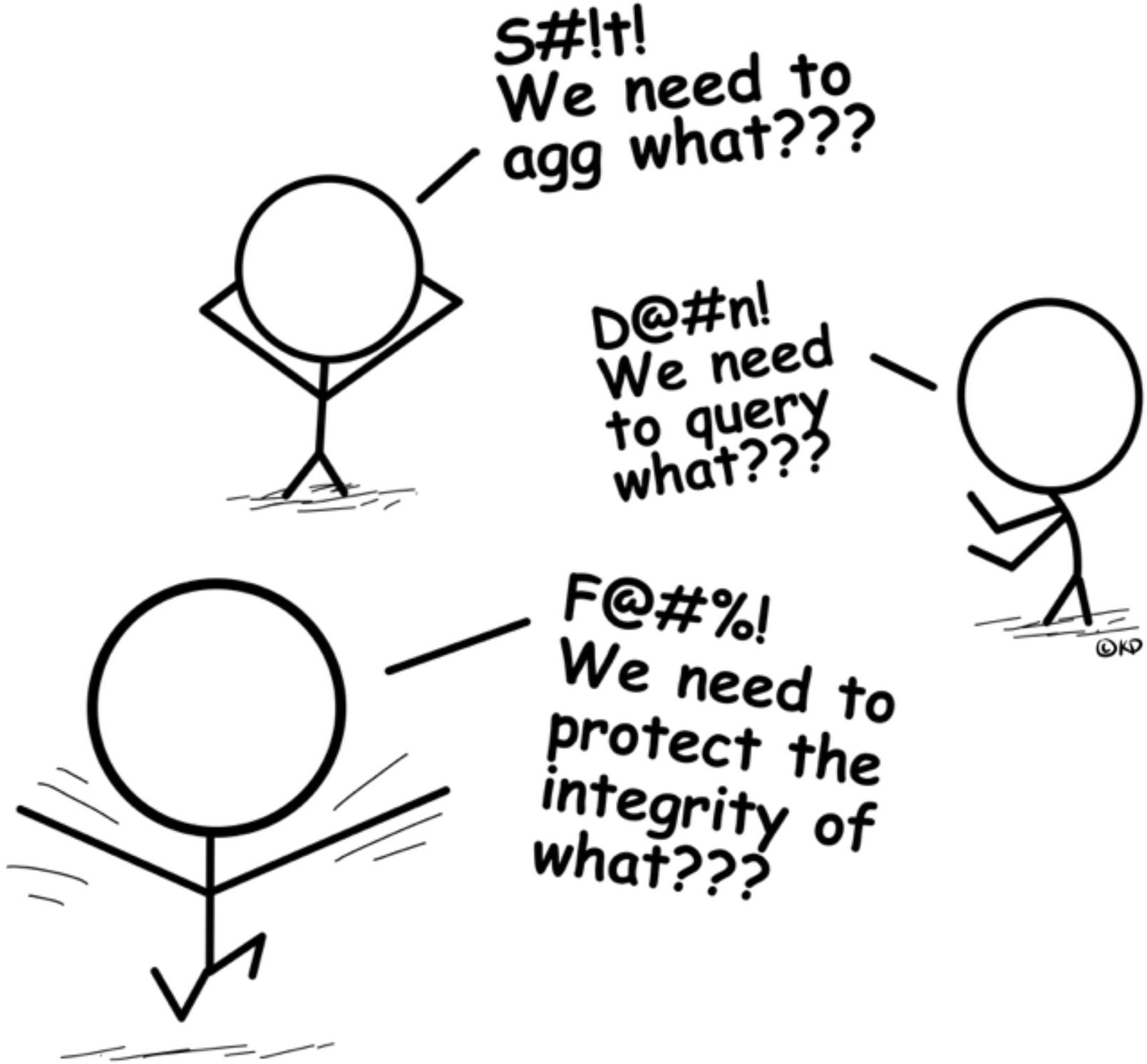
2010s

Evolution of Technology



Evolution of Needs





#DBManagementProblems

Problem

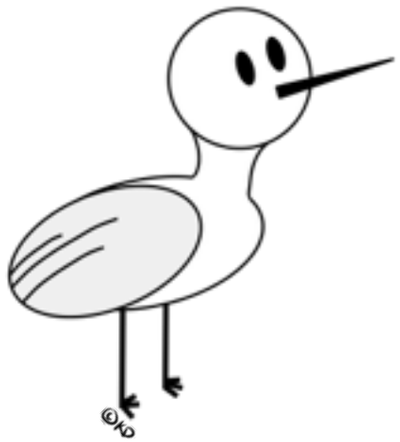
- Raw NoSQL systems are ***Database Systems*** (in the common case)

- Transforming a NoSQL product into a ***Database Management System*** necessitates engaging in a ***k-implementation*** nightmare with all the other NoSQL users in the world.

Valid Solution Qualities

Thou Shalt Not :

1. ***impede flexibility***
2. ***jeopardize customizability***
3. ***hamper dissemination***



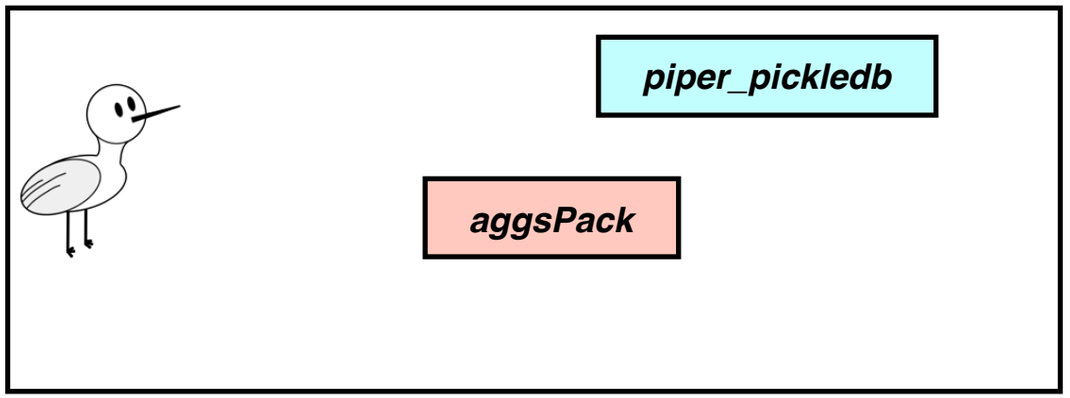
Piper

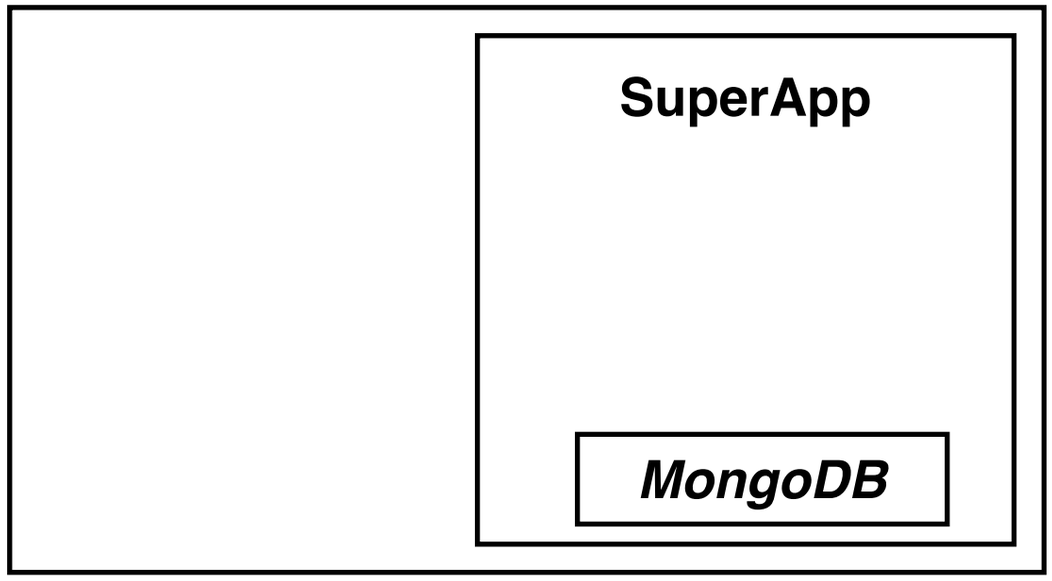
Definition :
Noun
1. A package index and management system exclusively dedicated to database management functionality.

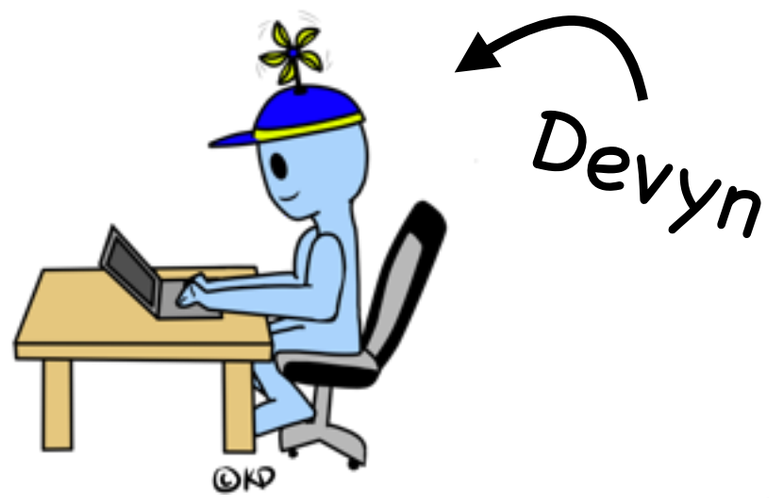
Usage :
"Damn! Why isn't this function already in ***Piper***!?!?"

- Developers working with NoSQL systems can:
  1. ***install*** management packages from the index.
  2. ***publish*** management packages to the index.
- Usage standards impose strict regulations on the ease of package installation + deinstallation.
- Inspired by
  - Package indexes (PyPI, NPM, …)
  - Source code repositories (Github, BitBucket, …)

Piper in action!



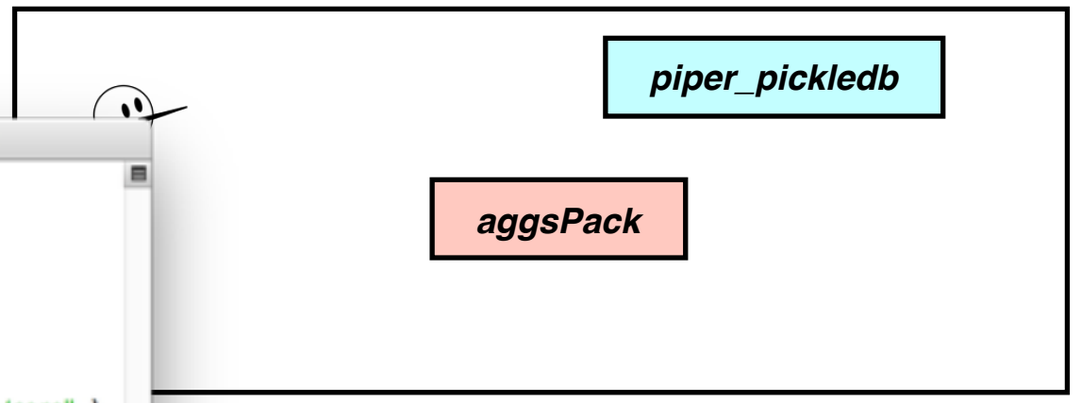


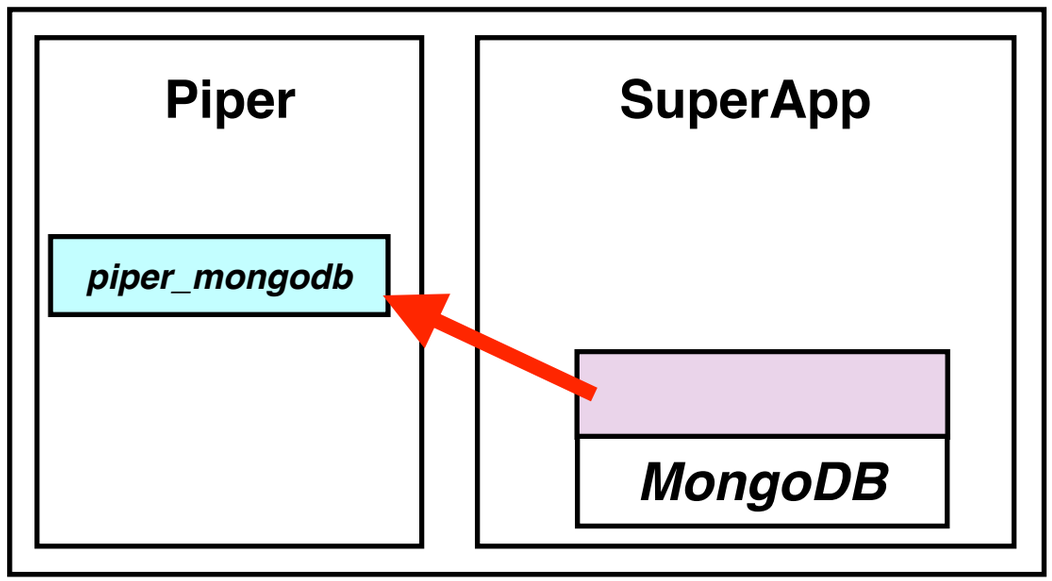


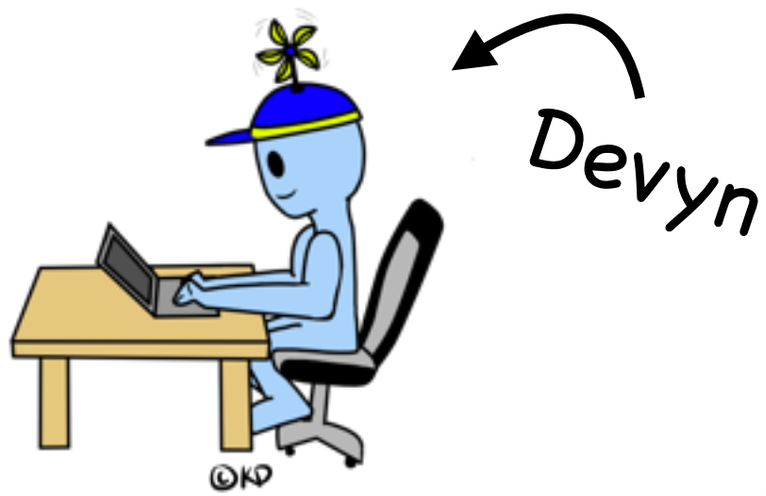
Piper in action!

adapters — vi piper_mongodb.py — 60×32

```
#!/usr/bin/env python


# ------------------------------------ #

import os, sys
from pymongo import *

# settings dir
settingsPath  = os.path.abspath( __file__ + "/../../core" )
sys.path.append( settingsPath )
import settings

# ------------------------------------ #

DEBUG = settings.DEBUG


#########
#  GET  #
#########
# get data on id
def get( ID, cursor ) :
  if DEBUG :
    print " >>> running piper_mongodb get "
  return cursor.find_one( { "_id" : ID } )


#########
#  EOF  #
#########
```





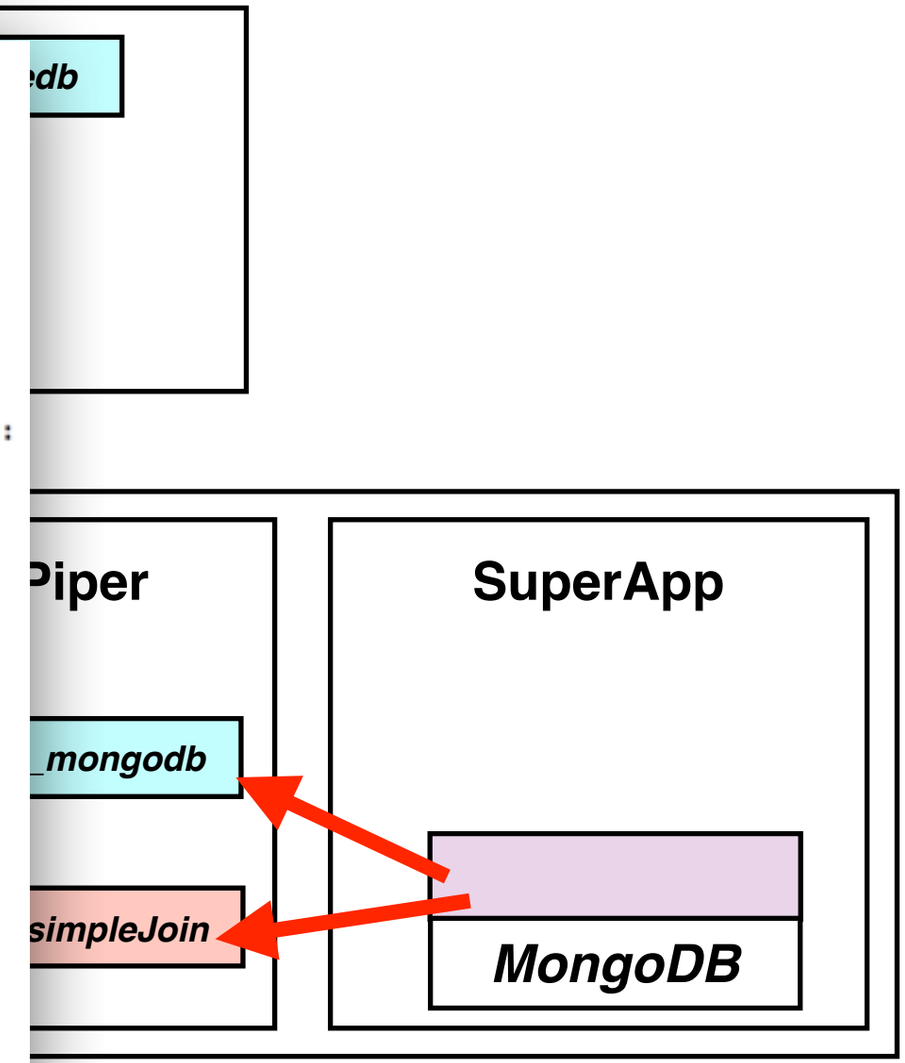


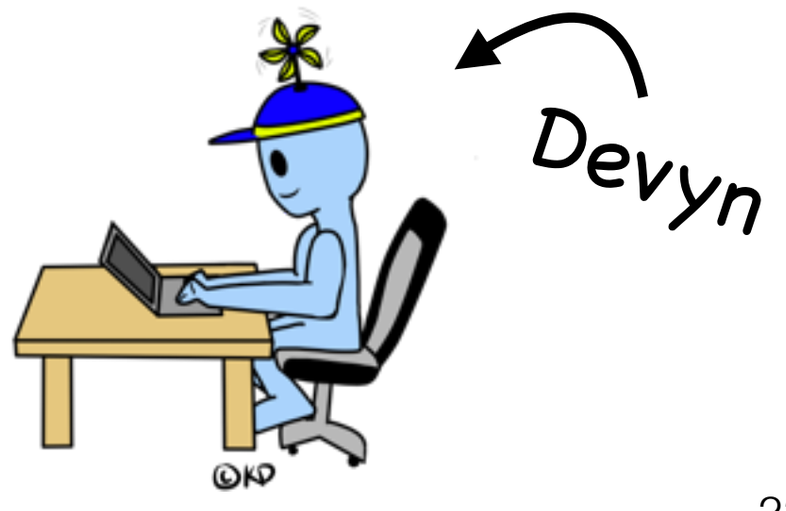
Piper in action!

adapters — vi piper_mongodb

```
#!/usr/bin/env python


# ------------------------------------

import os, sys
from pymongo import *

# settings dir
settingsPath  = os.path.abspath( __file_
sys.path.append( settingsPath )
import settings

# ------------------------------------

DEBUG = settings.DEBUG


#########
#  GET  #
#########
# get data on id
def get( ID, cursor ) :
  if DEBUG :
    print " >>> running piper_mongodb ge
  return cursor.find_one( { "_id" : ID }


#########
#  EOF  #
#########
```

```
#################
#  DICT MERGE   #
#################
def dictMerge( a, b ) :
  c = a.copy()
  c = c.update(b)
  return c


###################
#  SIMPLE JOIN   #
###################
def simpleJoin( nosql_type, cursor, idLists, joinAttr, pred ) :

  ad = Adapter.Adapter( nosql_type )

  # assume all ids in db are unique
  # ids per joinAttr
  idDict = {}
  for currList in idLists :
    for currID in currList :
      res1   = ad.get( currID, cursor )
      attVal = res1[ joinAttr ]
      idDict[ currID ] = attVal

if DEBUG :
  print "idDict = " + str(idDict)

# get ids with identical joinAttr
targetIDs = []
for k1 in idDict :
  att = idDict[ k1 ]
  for k2 in idDict :
    if not k1 == k2 :
      if idDict[k2] == att :
        targetIDs.append( k2 )

if DEBUG :
  print "targetIDs = " + str(targetIDs)

# grab all vals per joined id
currResDictList = []
for i in targetIDs :
  res = ad.get( i, cursor )
  currResDictList.append( res )

return currResDictList
```



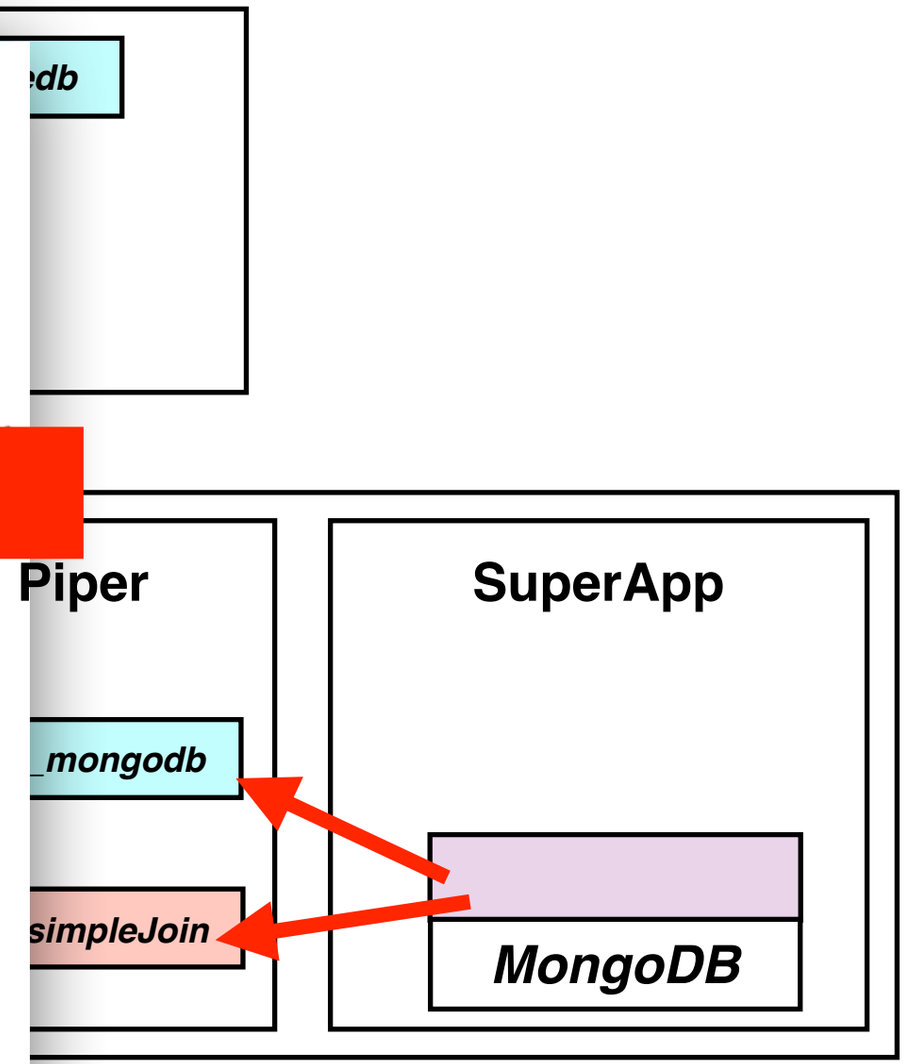


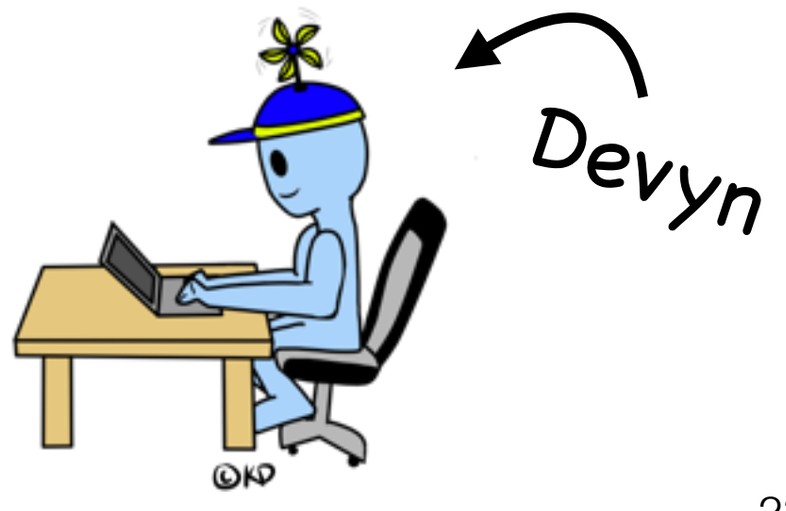
Piper in action!

adapters — vi piper_mongodb

```
#!/usr/bin/env python


# ------------------------------------

import os, sys
from pymongo import *

# settings dir
settingsPath  = os.path.abspath( __file_
sys.path.append( settingsPath )
import settings

# ------------------------------------

DEBUG = settings.DEBUG


#########
#  GET  #
#########
# get data on id
def get( ID, cursor ) :
  if DEBUG :
    print " >>> running piper_mongodb ge
  return cursor.find_one( { "_id" : ID }


#########
#  EOF  #
#########
```

```
#################
#  DICT MERGE   #
#################
def dictMerge( a, b ) :
  c = a.copy()
  c = c.update(b)
  return c


###################
#  SIMPLE JOIN   #
###################
def simpleJoin( nosql_type, cursor, idLists, jo
  ad = Adapter.Adapter( nosql_type )

  # assume all ids in db are unique
  # ids per joinAttr
  idDict = {}
  for currList in idLists :
    for currID in currList :
      res1   = ad.get( currID, cursor )
      attVal = res1[ joinAttr ]
      idDict[ currID ] = attVal

if DEBUG :
  print "idDict = " + str(idDict)

# get ids with identical joinAttr
targetIDs = []
for k1 in idDict :
  att = idDict[ k1 ]
  for k2 in idDict :
    if not k1 == k2 :
      if idDict[k2] == att :
        targetIDs.append( k2 )

if DEBUG :
  print "targetIDs = " + str(targetIDs)

# grab all vals per joined id
currResDictList = []
for i in targetIDs :
  res = ad.get( i, cursor )
  currResDictList.append( res )

return currResDictList
```



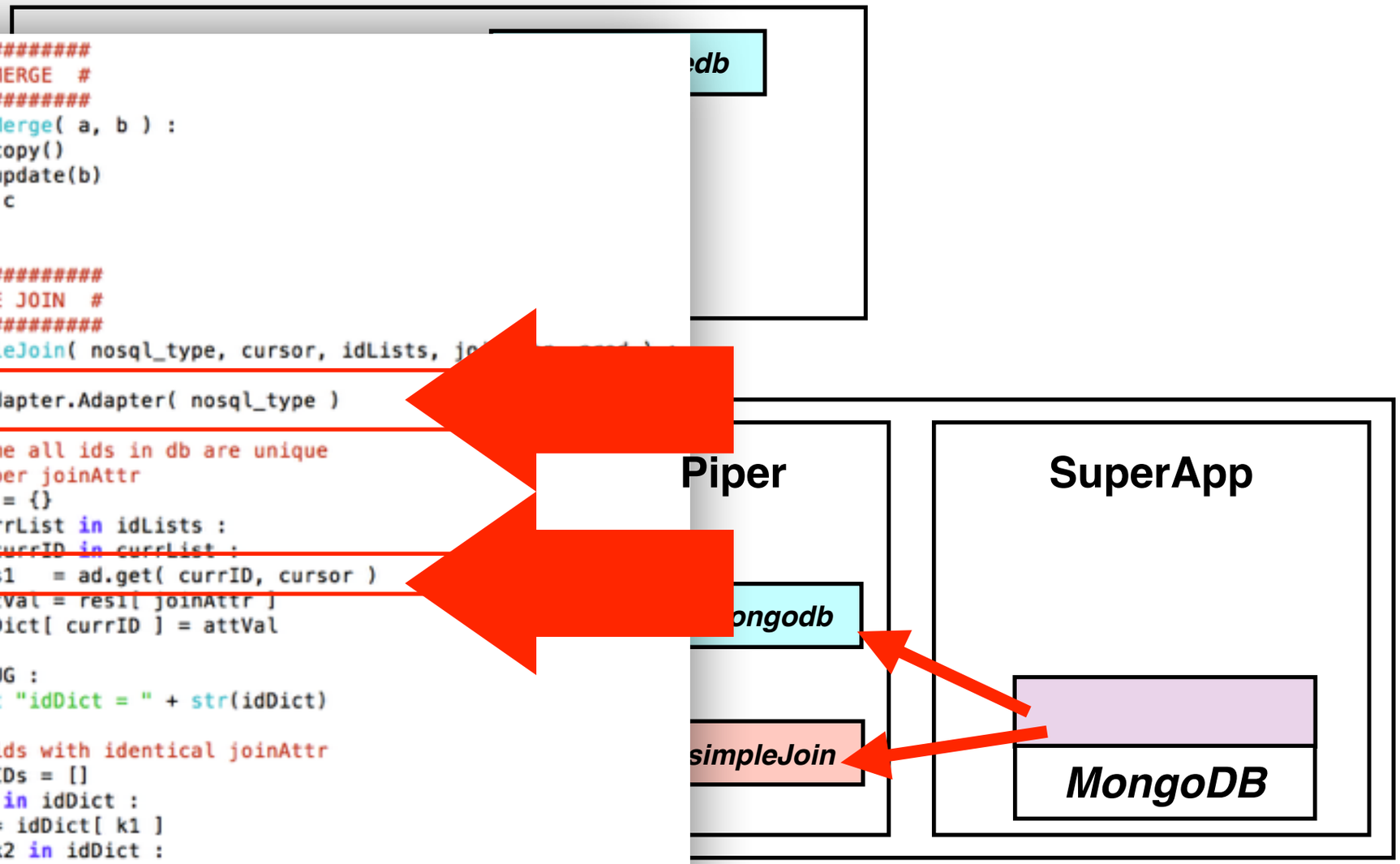


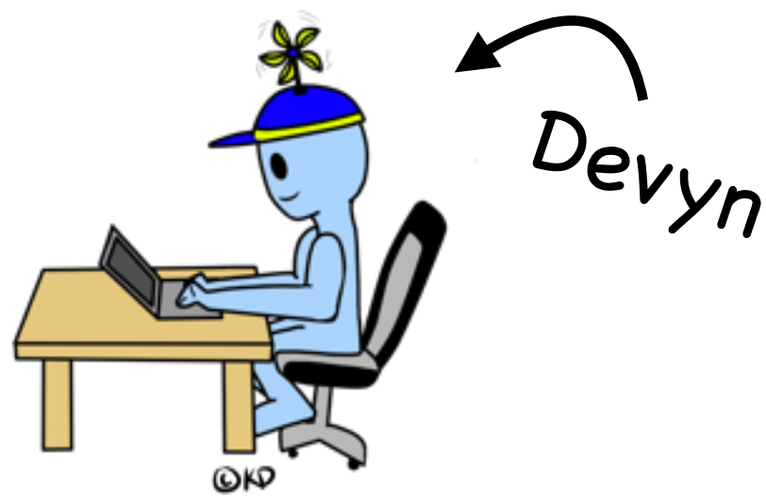
Piper in action!

adapters — vi piper_mongodb

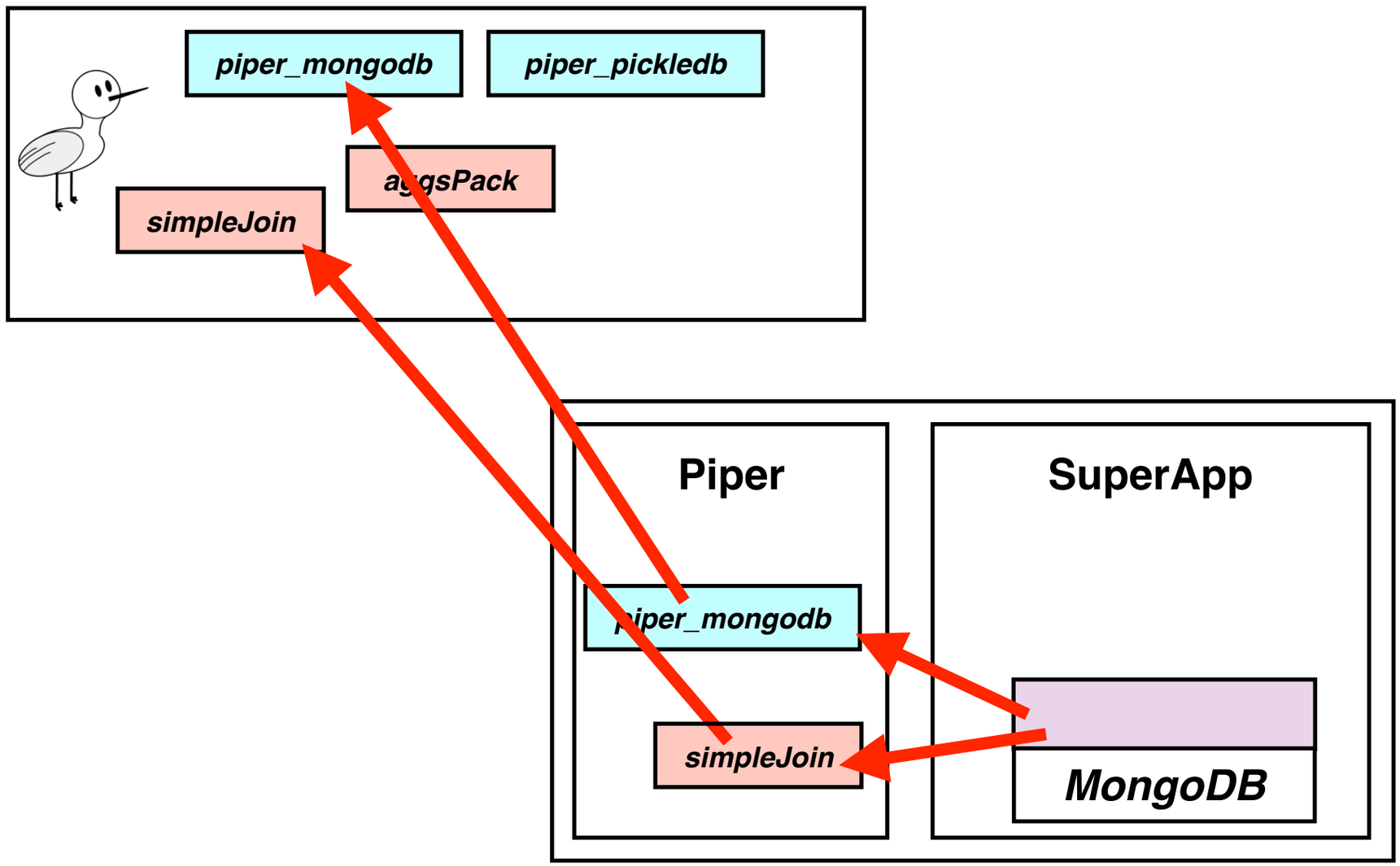
```
#!/usr/bin/env python


# ------------------------------------

import os, sys
from pymongo import *

# settings dir
settingsPath  = os.path.abspath( __file_
sys.path.append( settingsPath )
import settings

# ------------------------------------

DEBUG = settings.DEBUG


#########
#  GET  #
#########
# get data on id
def get( ID, cursor ) :
  if DEBUG :
    print " >>> running piper_mongodb ge
  return cursor.find_one( { "_id" : ID }


#########
#  EOF  #
#########
```

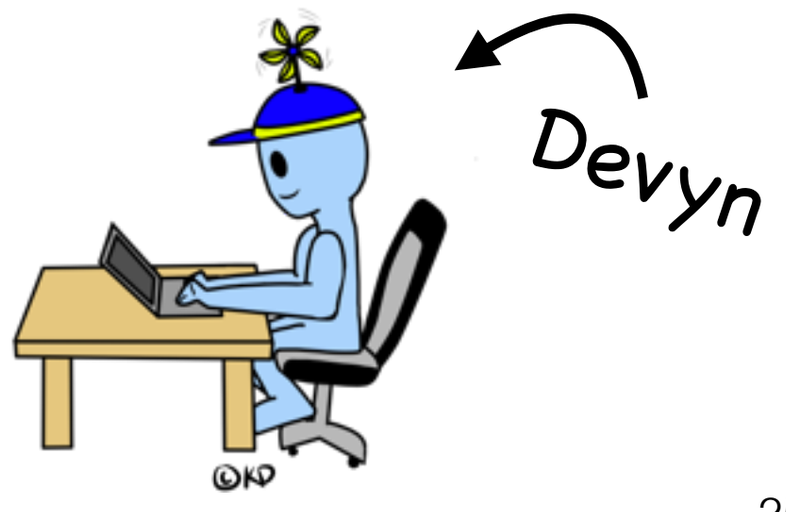
```
################
#  DICT MERGE  #
################
def dictMerge( a, b ) :
  c = a.copy()
  c = c.update(b)
  return c


################
#  SIMPLE JOIN  #
################
def simpleJoin( nosql_type, cursor, idLists, jo
  ad = Adapter.Adapter( nosql_type )

  # assume all ids in db are unique
  # ids per joinAttr
  idDict = {}
  for currList in idLists :
    for currID in currList :
      res1   = ad.get( currID, cursor )
      attVal = res1[ joinAttr ]
      idDict[ currID ] = attVal

  if DEBUG :
    print "idDict = " + str(idDict)

  # get ids with identical joinAttr
  targetIDs = []
  for k1 in idDict :
    att = idDict[ k1 ]
    for k2 in idDict :
      if not k1 == k2 :
        if idDict[k2] == att :
          targetIDs.append( k2 )

  if DEBUG :
    print "targetIDs = " + str(targetIDs)

  # grab all vals per joined id
  currResDictList = []
  for i in targetIDs :
    res = ad.get( i, cursor )
    currResDictList.append( res )

  return currResDictList
```



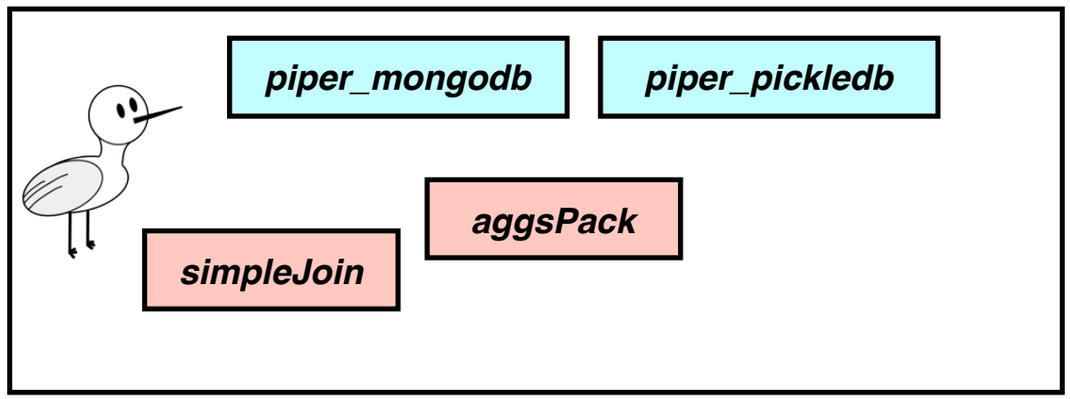


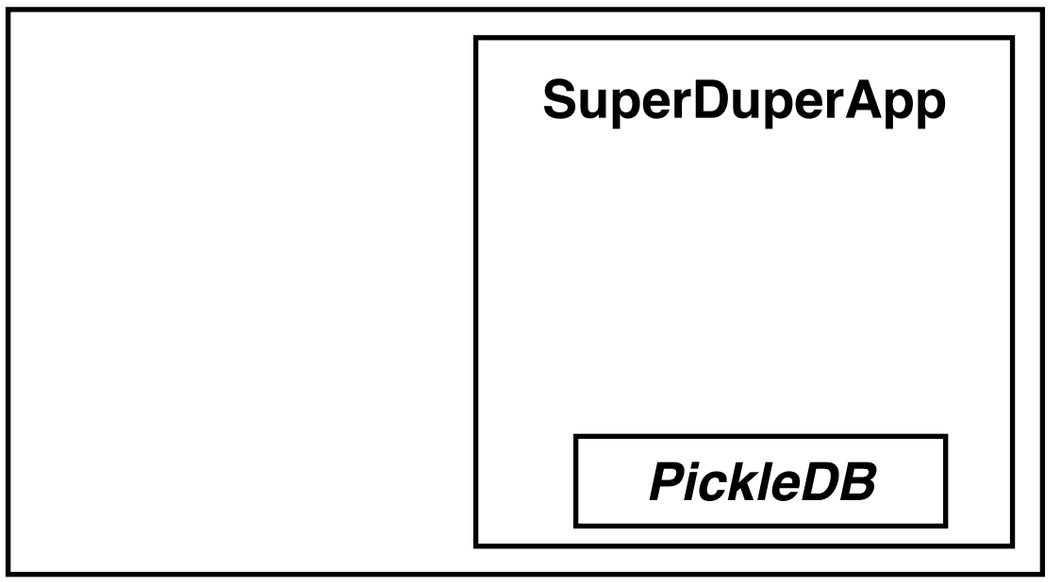
Piper in action!

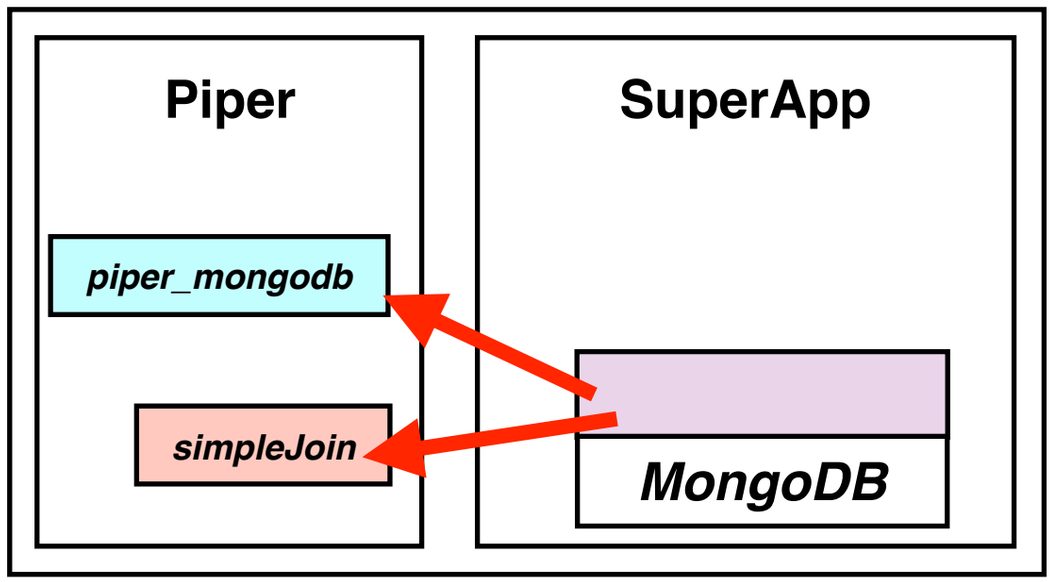


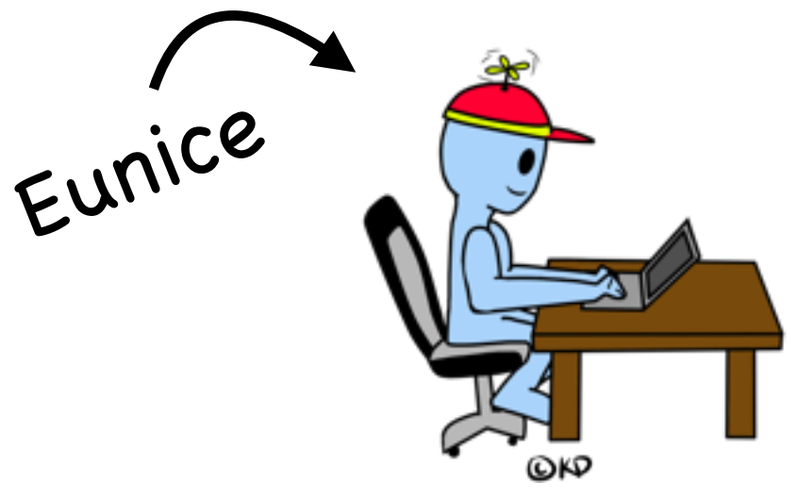


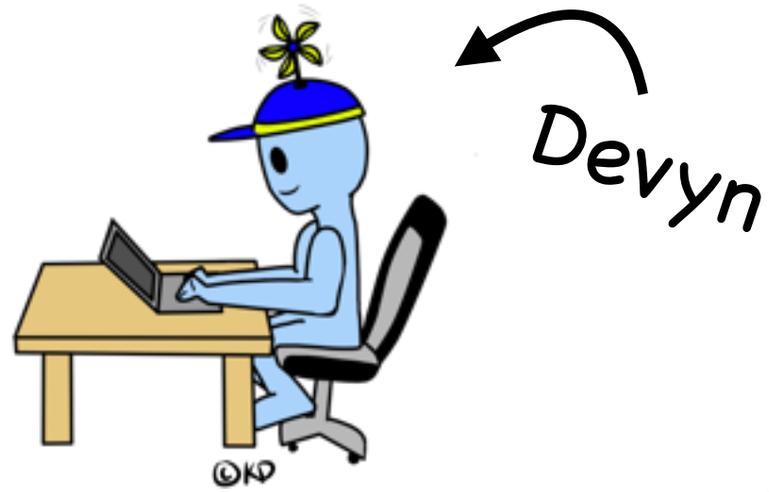
Piper in action!



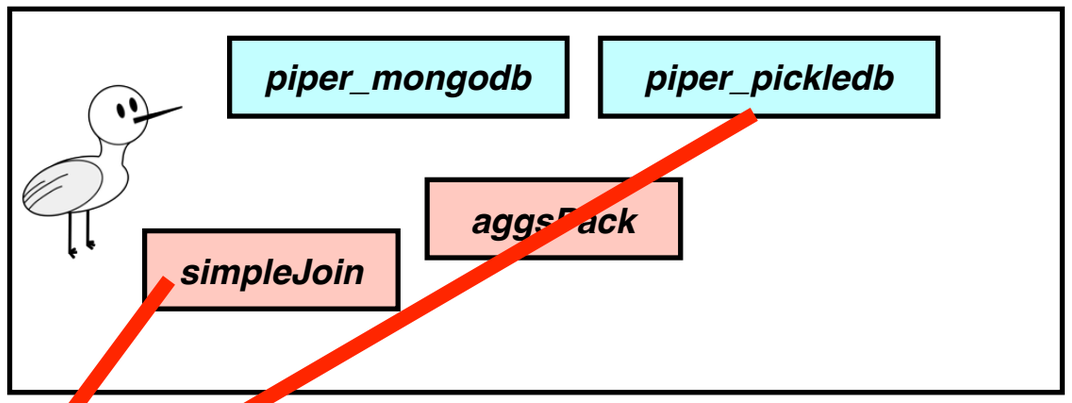


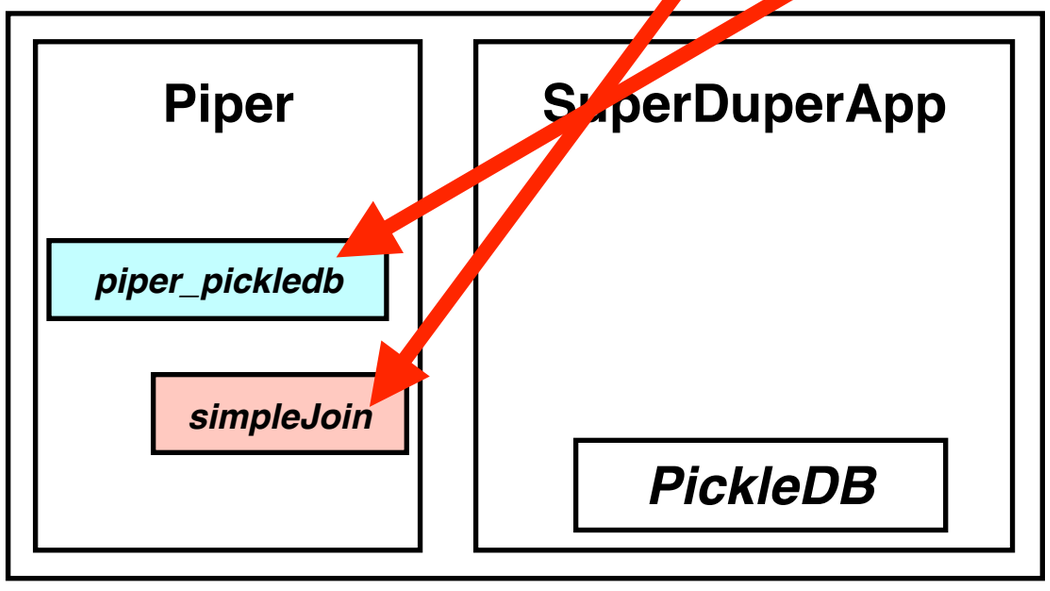


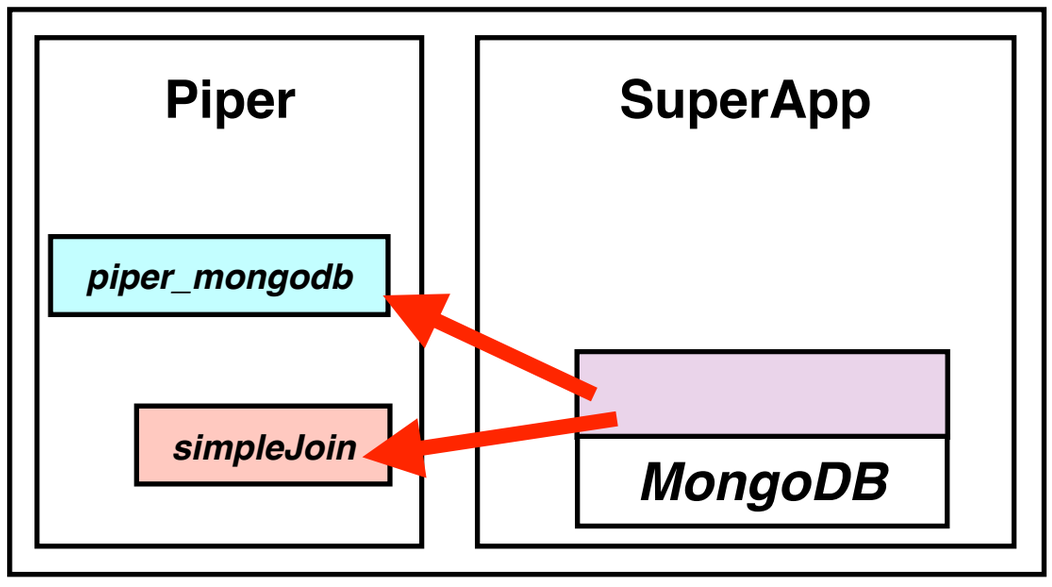


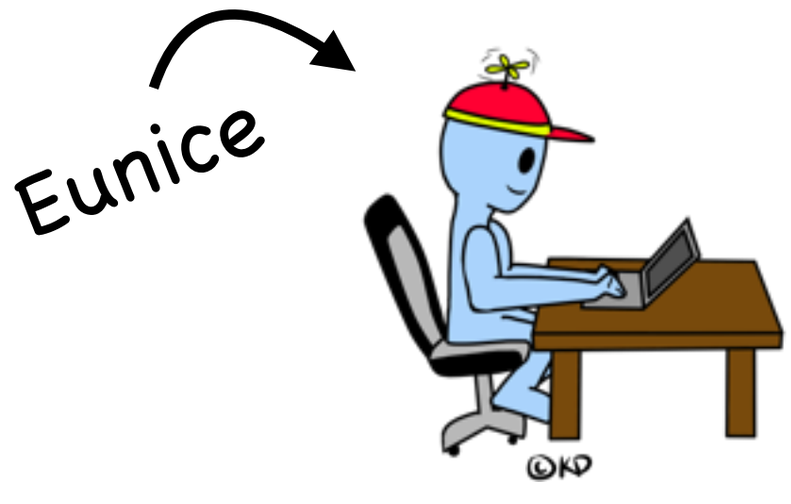


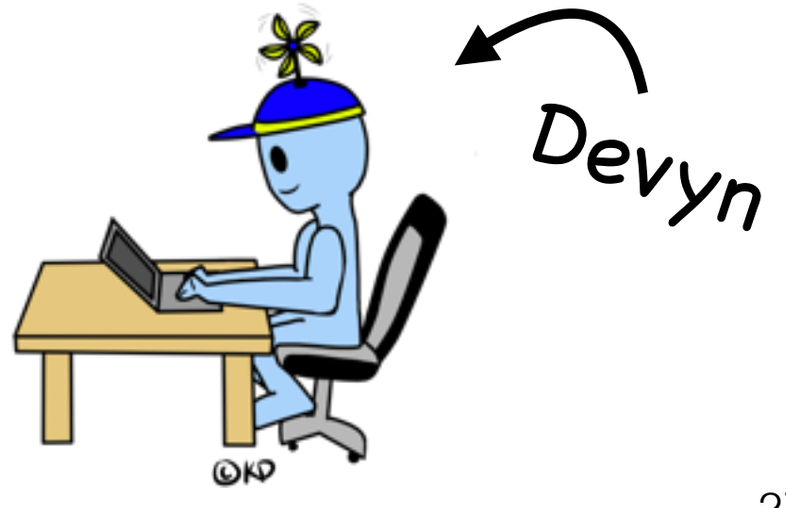
Piper in action!



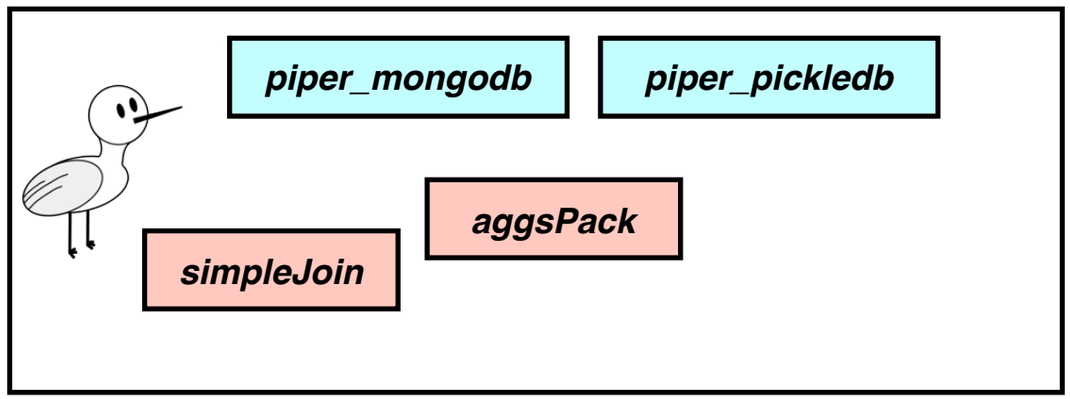


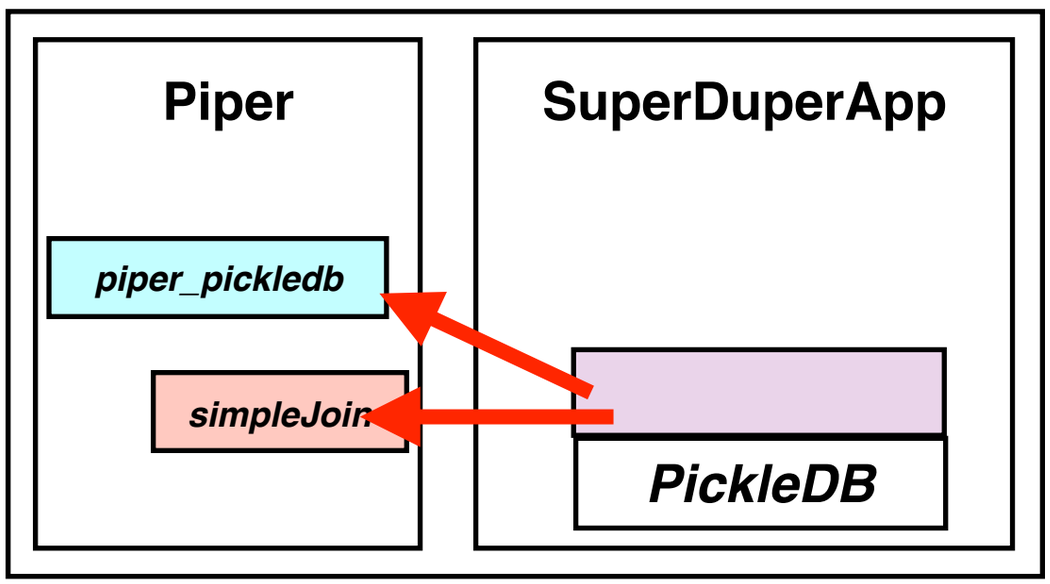


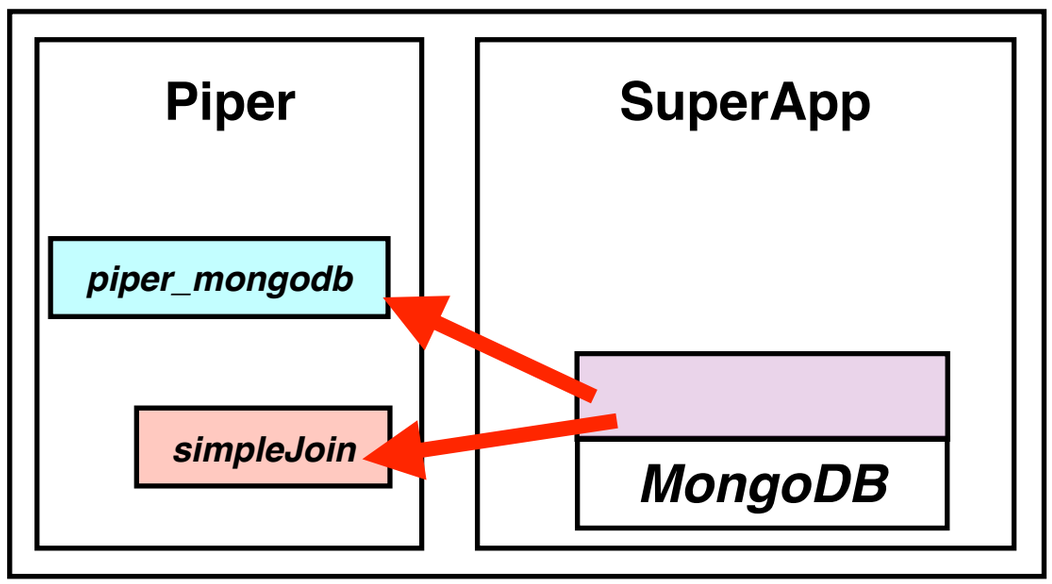


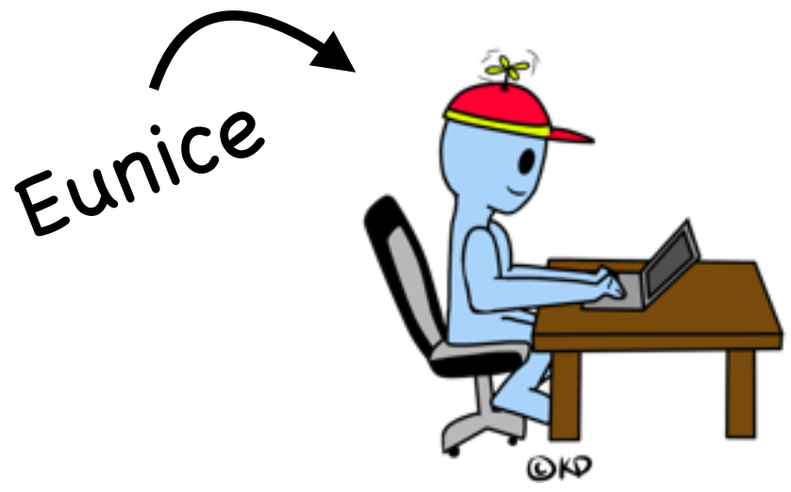


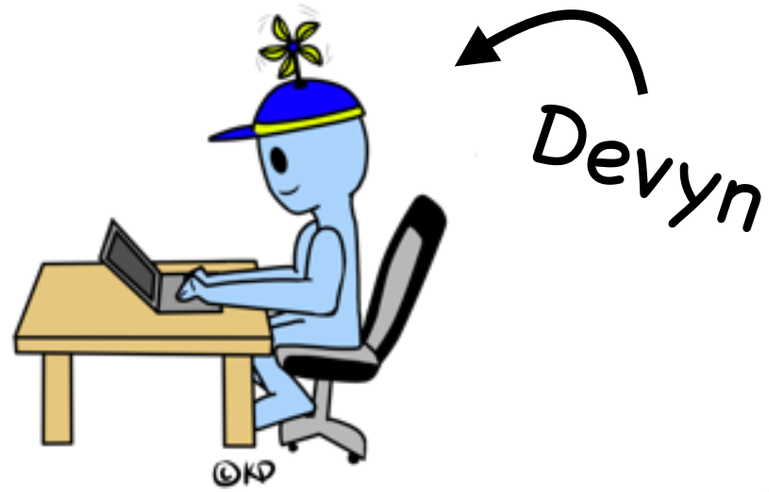
Piper in action!



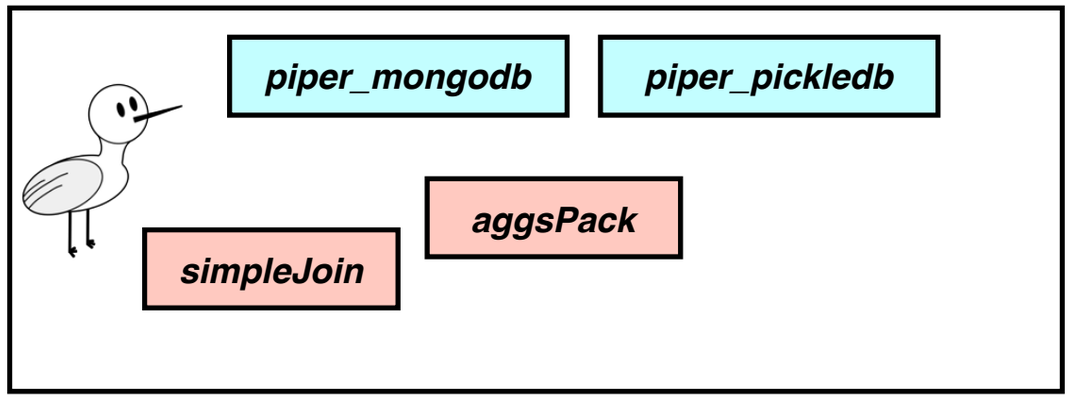


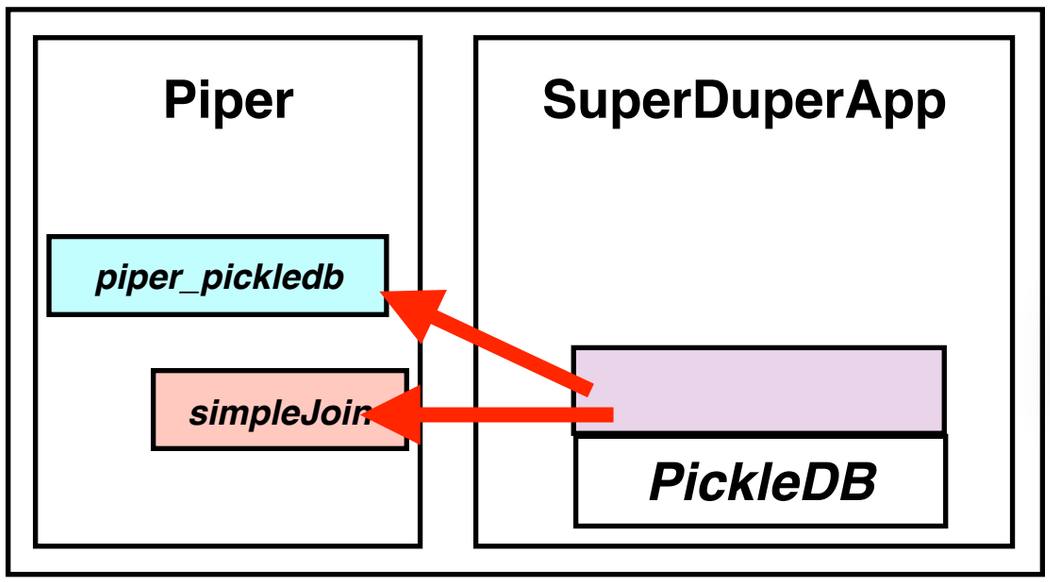


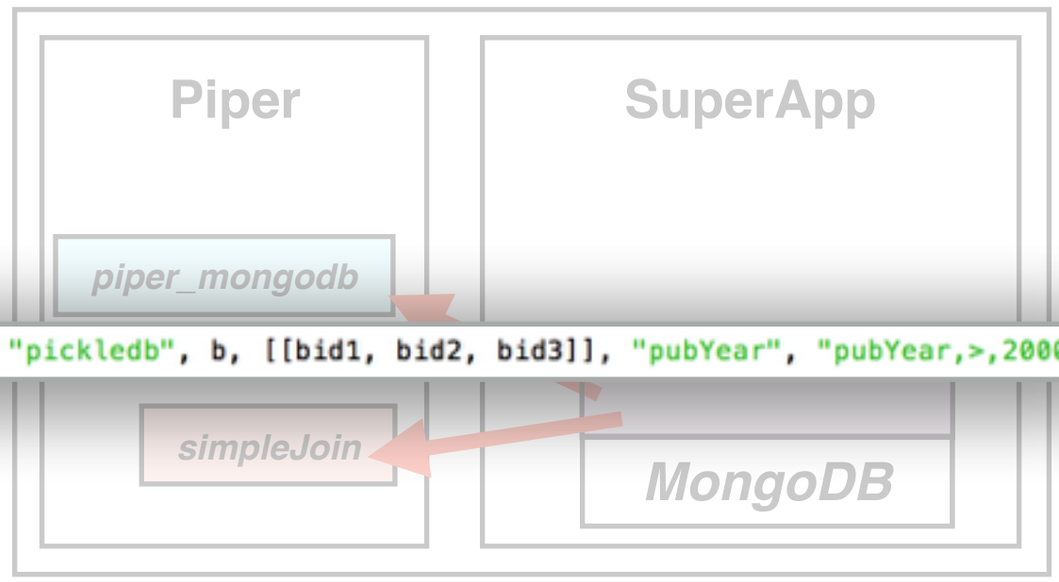


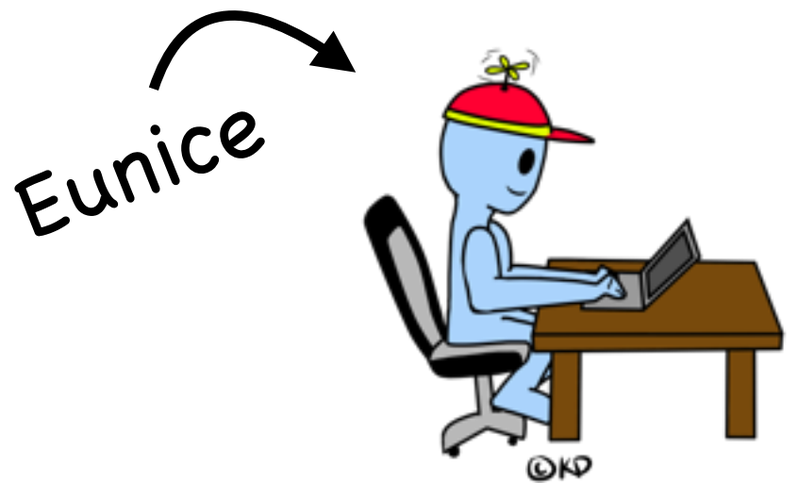


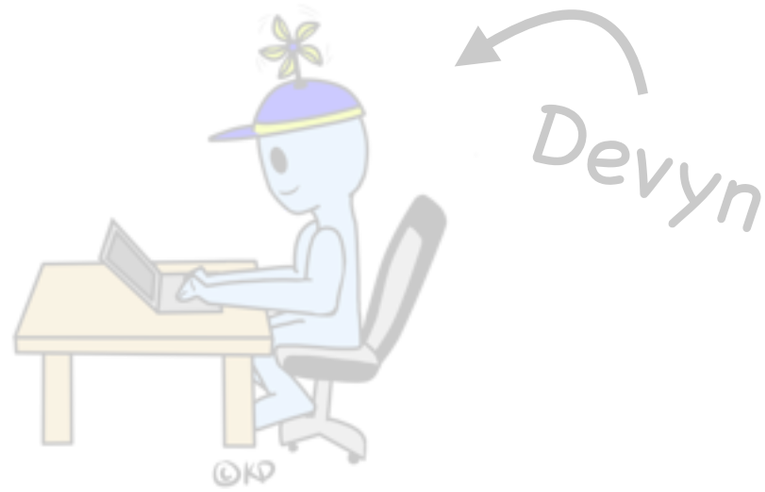
Piper in action!











Valid Solution!

WISH LIST :

1. ***Promotes flexibility***
2. ***Champions customizability***
3. ***Eases solution dissemination***

Valid Solution!

WISH LIST :

1. ***Promotes flexibility*** ✓
2. ***Champions customizability***
3. ***Eases solution dissemination***

Valid Solution!

WISH LIST :

1. ***Promotes flexibility*** ✓
2. ***Champions customizability*** ✓
3. ***Eases solution dissemination***

Valid Solution!

WISH LIST :

1. ***Promotes flexibility*** ✓
2. ***Champions customizability*** ✓
3. ***Eases solution dissemination*** ✓

Project Status

- Bare-bones proof-of-concept.

- Supports two NoSQL DBSs :
  - **MongoDB** (document store)
  - **PickleDB** (key-value store)

- Contains two packages :
  - **aggsPack**
  - **simpleJoin**

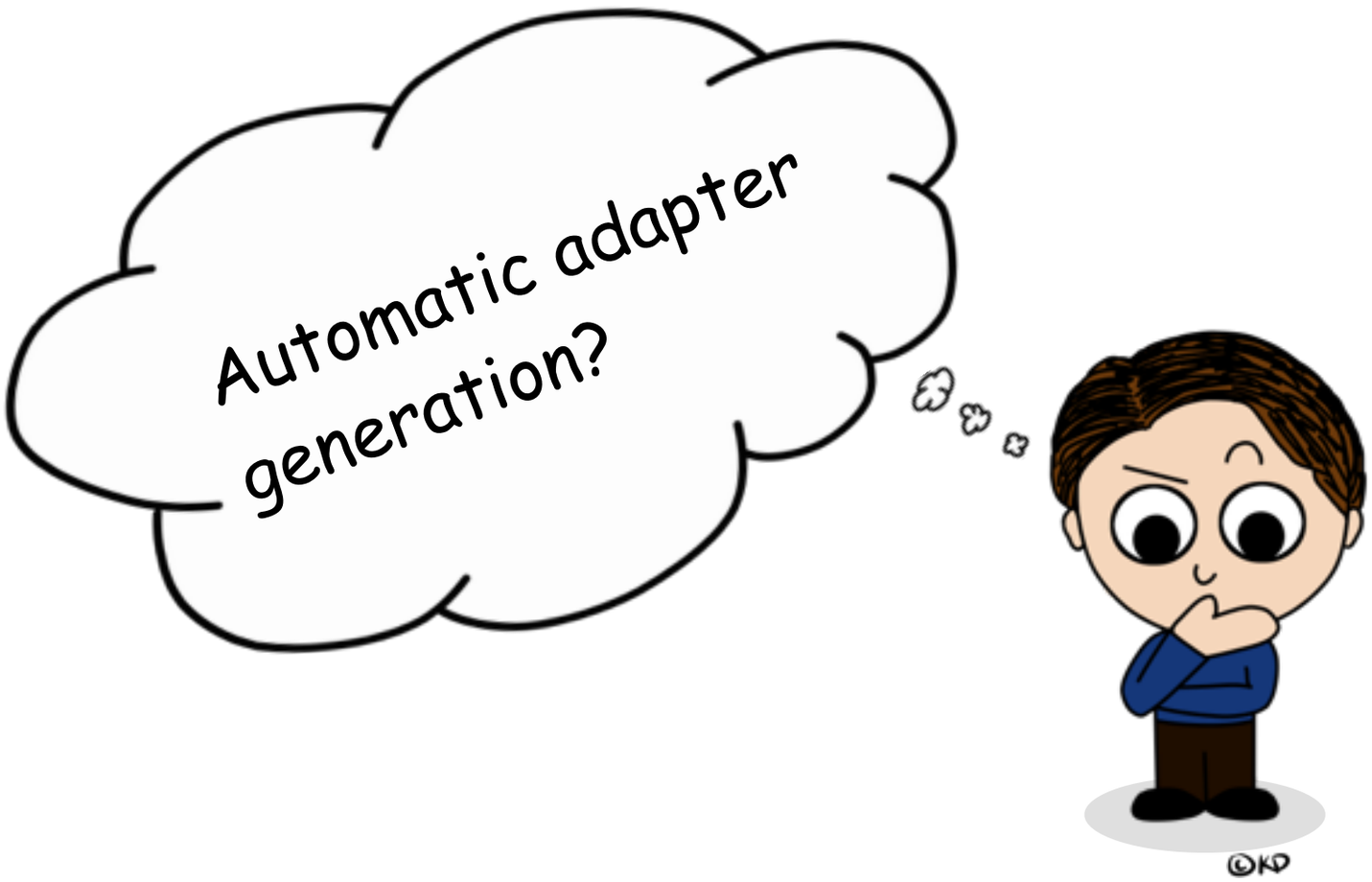
https://github.com/PiperProject

Research Directions

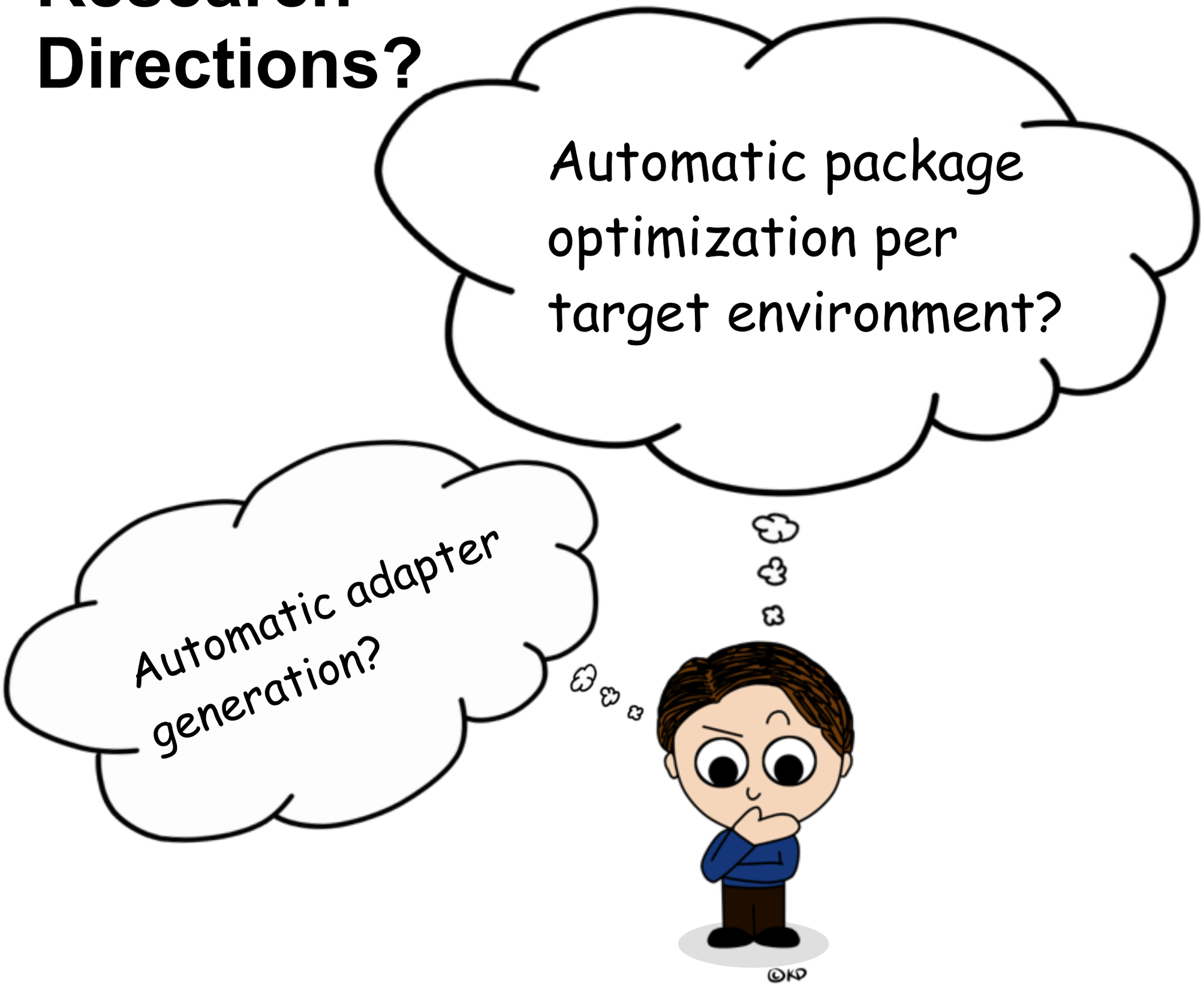
???

Research Directions?

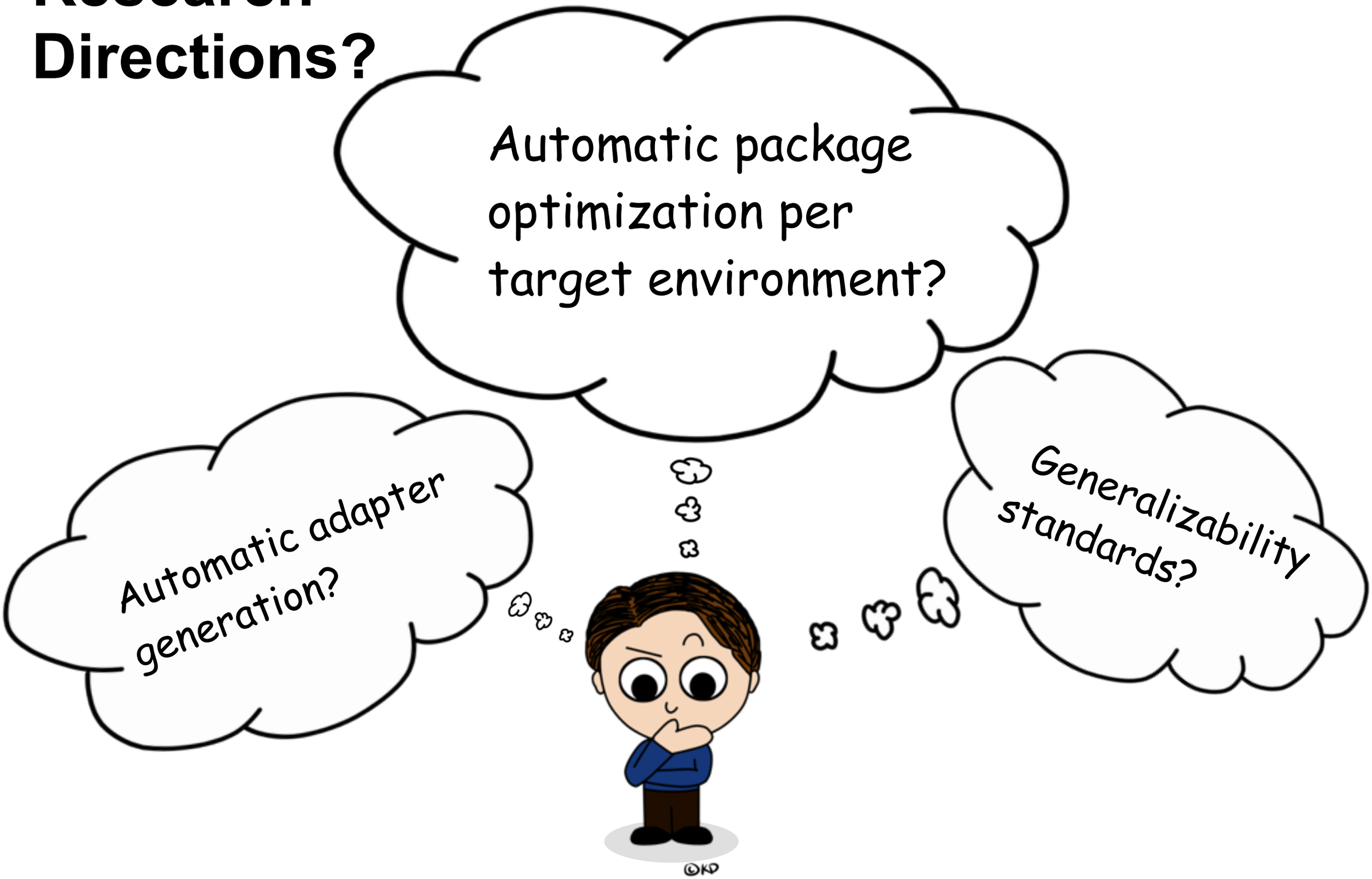
Research Directions?



Research Directions?



Research Directions?



Conclusion

- NoSQL is here to stay
- No rules != totally awesome
- *k*-implementation is sub-optimal
- **Piper** is a proof-of-concept tool modeling a valid solution

Conclusion

- NoSQL is here to stay
- No rules != totally awesome
- *k*-implementation is sub-optimal
- **Piper** is a proof-of-concept tool modeling a valid solution

Making NoSQL great

:-

Being stronger together!


Thanks!