Take control of your printing system

Jerome Alet
alet@librelogiciel.com

Original conference made in French for
Mediterranean Free Software Day
November 15th 2008
Sophia Antipolis

Some Free Software bricks to really benefit from CUPS

- pkipplib
- pkpgcounter
- Tea4CUPS
- PyKota

Reminders about CUPS

- Network printing server, GPL and LGPL
- Since 1997 (Easy Software Products)
- More advanced than LPD/LPR
- Easier to configure than LPRng (web interface http://localhost:631)
- Internet Printing Protocol Server
- Autodetection of servers and printers
- Load balancing
- No need for client side drivers
- Works on any *nix system
  - ➔ Standard on :
    - GNU/Linux
    - Mac OS X

pkipplib

Presentation of pkipplib
IPP support for Python

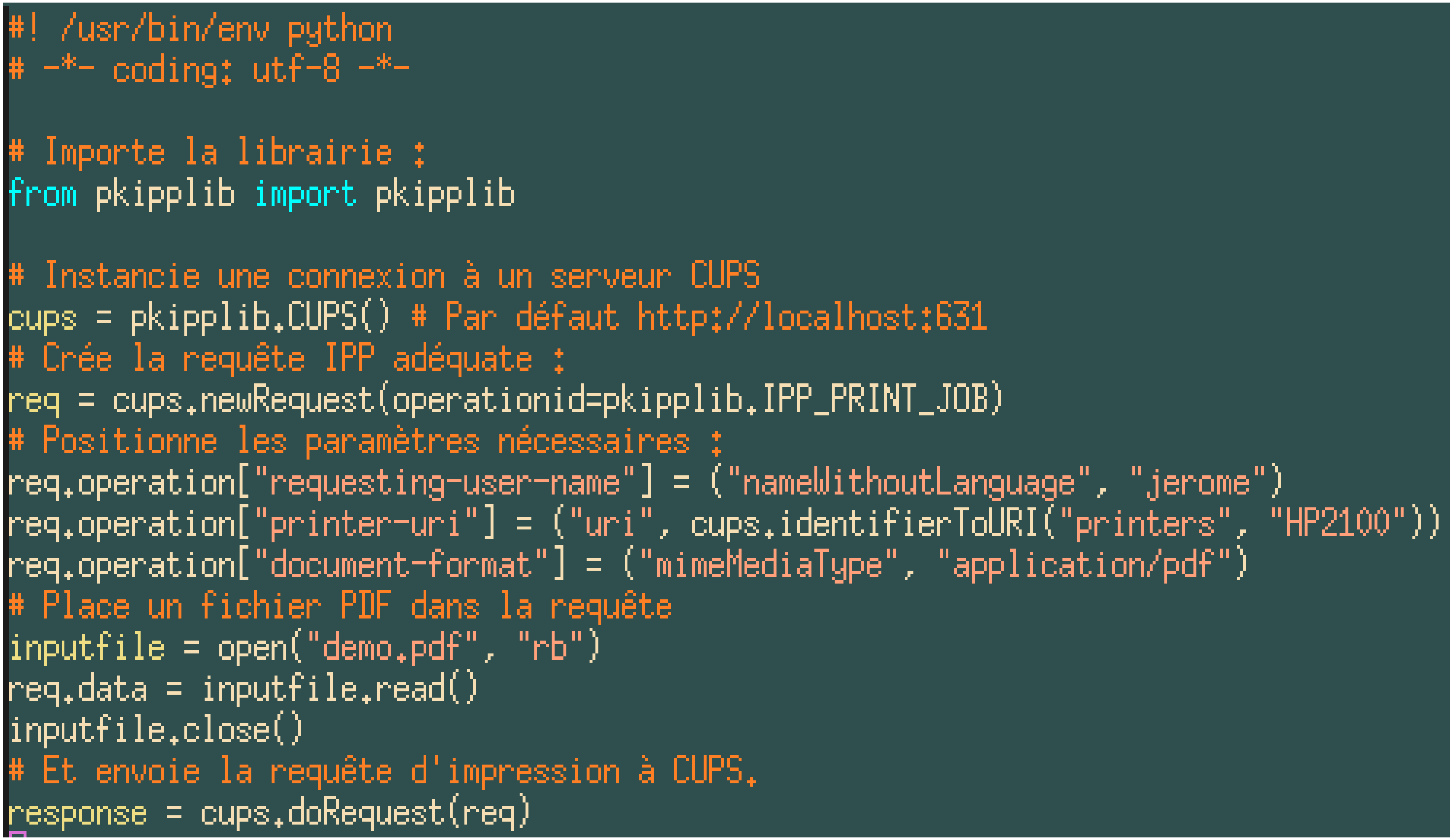
Pkipplib's description

- Library for the Python language
- 'pk' => comes from PyKota
- Internet Printing Protocol :
  - Novell + Xerox + IETF (RFC2910, 2911...)
  - « powerful » printing protocol :
    - Uses HTTP 1.1 as transport :
      - No limited to LANs
      - Authorization
      - Authentication
      - Encryption
      - ...
    - Allows one to submit or cancel print jobs
    - Allows one to query printers (or servers)

Pkipplib's features

- Encoding and decoding of IPP requests
  - IPP is a « binary » protocol :
    1. IPP Version (usually 1.1)
    2. Operation code (ex: 2 to print)
    3. Request identifier, or 1
    4. Mandatory attributes (type + value)
    5. Optional attributes (type + value)
    6. End of attributes marker
    7. Datas, for example a PostScript document
- Interact with IPP servers :
    - Printers
    - Novell iPrint
    - CUPS (all *nix)
    - Others...

Example :
Print a PDF document



```python
#! /usr/bin/env python
# -*- coding: utf-8 -*-

# Importe la librairie :
from pkipplib import pkipplib

# Instancie une connexion à un serveur CUPS
cups = pkipplib.CUPS() # Par défaut http://localhost:631
# Crée la requête IPP adéquate :
req = cups.newRequest(operationid=pkipplib.IPP_PRINT_JOB)
# Positionne les paramètres nécessaires :
req.operation["requesting-user-name"] = ("nameWithoutLanguage", "jerome")
req.operation["printer-uri"] = ("uri", cups.identifierToURI("printers", "HP2100"))
req.operation["document-format"] = ("mimeMediaType", "application/pdf")
# Place un fichier PDF dans la requête
inputfile = open("demo.pdf", "rb")
req.data = inputfile.read()
inputfile.close()
# Et envoie la requête d'impression à CUPS.
response = cups.doRequest(req)
```

Pkipplib's Pros/Cons

- Pros :
  - 100% Python :
    - No need for CUPS' client library
    - ➔ Works on any operating system with Python
  - Easy to use
  - Allows one to manage IPP subscriptions

- Cons :
  - Only supports IP sockets, not unix domain ones
  - API still in the works. Version 0.07, contributions welcome :-)

pkpgcounter

Presentation of pkpgcounter
Documents analyzer

Pkpgcounter's description

- Command line tool
- Library for the Python language
- 'pk' => comes from PyKota

- Two operating modes :
  - Count pages
  - Compute percent of ink coverage

Page counting mode

1. Autodetection of the Page Description Language (PDL)
2. Activation of the matching analyzer module
3. Analyze of content :
   - ±20 Internal parsers
   - « Special » case for PostScript
   - Special case for « .doc »
4. Returns the number of pages computed

Computation of percent of ink coverage mode

1. Autodetection of the PDL
2. Activation of the matching analyzer module
3. Conversion to TIFF (external tools)
4. Analyze wrt colorspace :
   - BW
   - RGB
   - CMY
   - CMYK
   - GC : « in-house » colorspace
5. For each page, result like :

C: 0.873%   M: 0.775%   Y: 0.576%   K: 2.883%

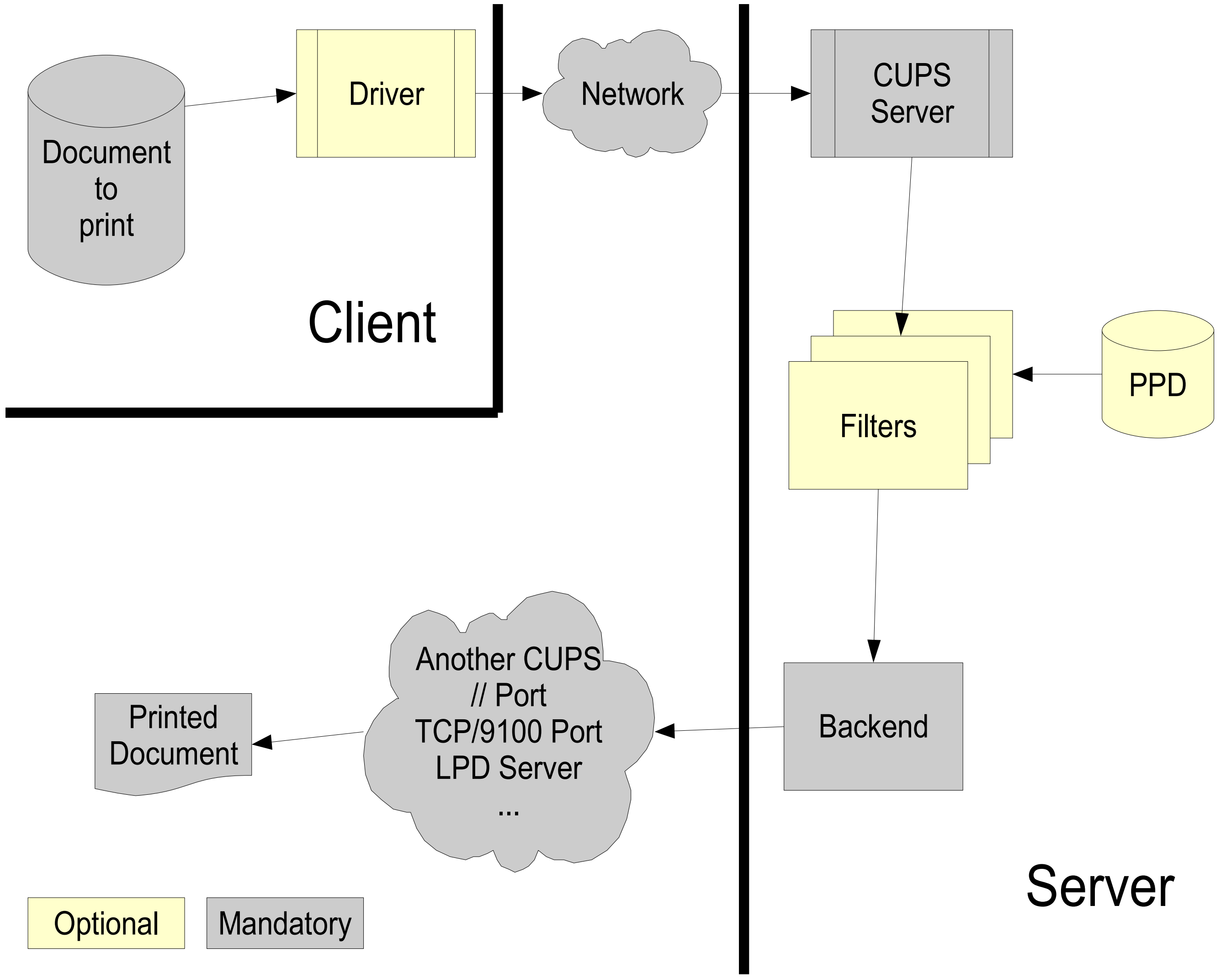
Missing features

- Document formats :
  - Most well known are already supported
  - Entirely proprietary formats
  - Partially proprietary formats
- Detection of printing mode :
  - Duplex vs Simplex
  - Page size
  - N-up
  - Paper length (plotters)
- Ink coverage : ***really*** needed for Epson drivers

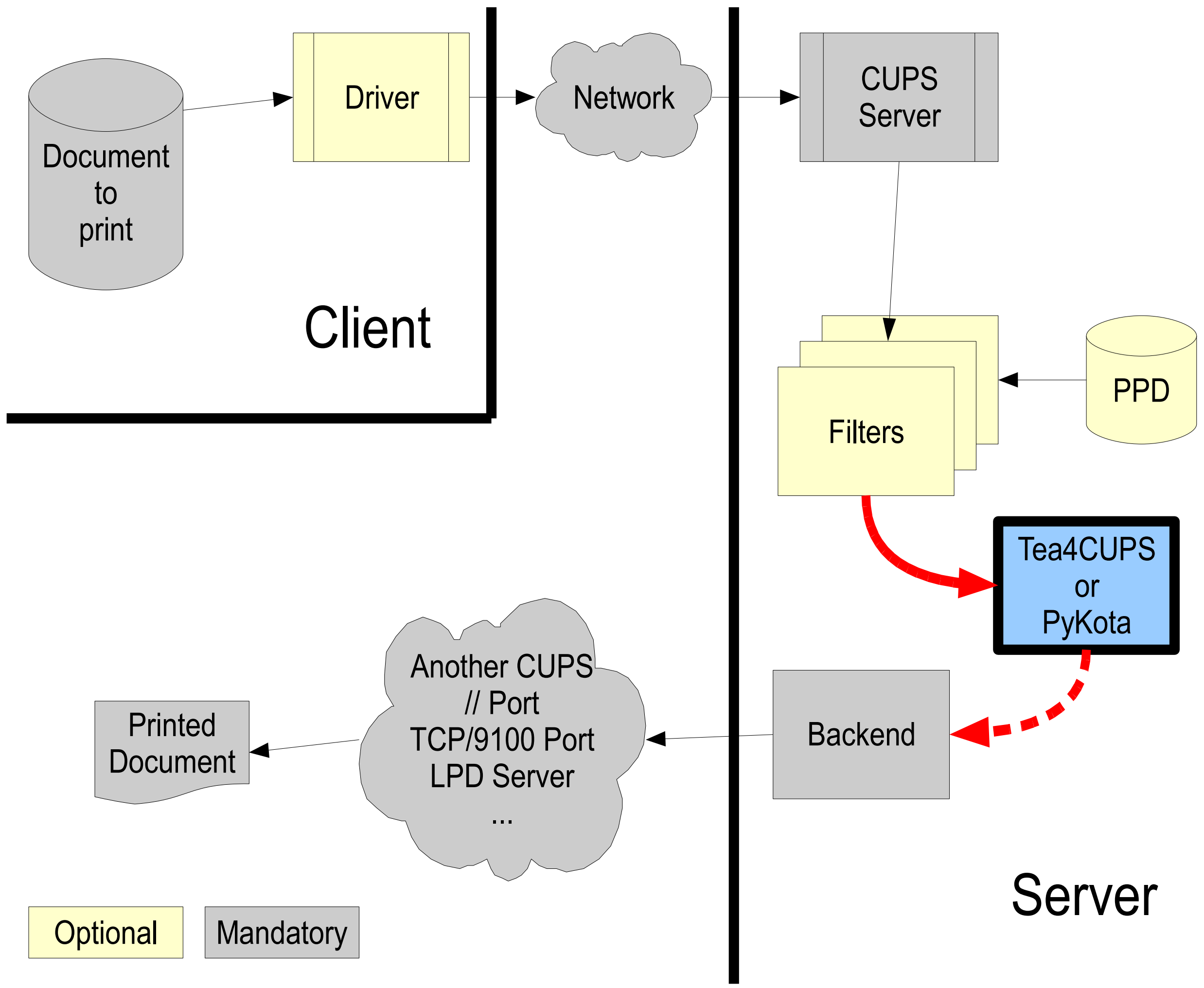
Tea4CUPS

Presentation of Tea4CUPS
« Swiss Army's knife » of the CUPS admin

Printing flow with CUPS



Printing flow with Tea4CUPS



Description de Tea4CUPS

- CUPS Backend (/usr/lib/cups/backend/)
- Inspired from the '*tee*' command line tool
- Captures printing flows
- Can filter content
- Set environment variables
- Executes « prehooks »
- Executes the original CUPS backend
- Executes « posthooks »

tea4cups.conf

```
[global]
directory : /var/spool/tea4cups
debug : no
keepfiles : no
filter : /bin/grep -v "%%CreationDate:"

posthook_etat : echo Job $TEAJOBID printed (status $TEASTATUS)
                                   | smbclient -M $TEAUSERNAME
prehook_compta : echo $TEAPRINTERNAME $TEAJOBID $TEAUSERNAME
                        $TEABILLING `pkpgcounter $TEADATAFILE`
posthook_compta : /bin/cat >>/var/log/printaccounting.log

[MonImprimantePostScript]
prehook_rawpdf : /bin/cat $TEADATAFILE
       | /bin/su $TEAUSERNAME -c
       "ps2pdf - `/usr/bin/getent passwd $TEAUSERNAME
                  | /usr/bin/cut -f 6,6 -d :`/JOB-$TEAJOBID.pdf"
posthook_rawpdf : /bin/cat >>/tmp/logs_and_errors_ps2pdf.log
```

Capabilities (almost no limit)

- Archiving, Publishing
- Limitation & Multiplication
- Hold & Release
- Routing to most nearby printer
- Modification :
  - Add page backgrounds, logos, signatures, etc...
- And why not :
  - Home automation :

    lp -dcoffeemaker expresso.recipe
  - Strange things :

    lp -dscanner scanjob.description
    - Acquire, transform, print...
  - Everything you can think

PyKota

Presentation of PyKota
Print Accounting and Quotas Software

Why ???

- Accounting (entreprise world) :
  - Audit :

    `purchased – available – used = wasted`
  - Planning :
    - Approvisionning
    - Preventive maintenance
  - Invoicing :
    - Per department, per user, per client, etc...

- Limiting (education world) :
  - Costs :
    - Consummables : paper, ink
    - Hardware : purchase, maintenance
  - Environnemental impact :
    - Forests (+ or -), water++, chemical substances++

    http://feuille-erable.org/accueil_cycle.htm

Description of PyKota

- PyKota ***roughly*** combines :
  - pkipplib
  - pkpgcounter
  - Tea4CUPS
  - Data storage facilities :
    - PostgreSQL
    - MySQL
    - LDAP (OpenLDAP, SunOne)
    - SQLite
  - User interfaces :
    - Command line : administration and querying
    - Web : querying only
    - Native GUIs : OSD and/or dialog boxes : interaction with the end user
  - And...

Description of PyKota (cont)

- YOUR IMAGINATION :
  - All strategic points are SCRIPTABLE !
  - No hard link between printing account and system account : you can do accounting per IP or MAC address of the client host if you want, etc...
  - Complex interactions with the end user (with or without add-on) : information, confirmation, authentication, etc...

5 Ways to limit an « account » from printing

- Number of pages per printer :
  - 100 pages on HP LaserJet 2100
  - 30 pages on EPSON Stylus Color
- Number of credits to spend :
  - 1 page = x credits on HP 2100
  - 1 page = y credits on Stylus Color
  - + Cost of the job itself (optional)
  - + Induced costs (printers hierarchies)
  - * Account's overcharging factor (+ or -)
- No limit, but accounting done
- No limit, and no accounting
- Printing is forbidden

2 Accounting modes

- Hardware accounting (direct querying of printers) :
  - SNMP
  - HP PJL
  - External scripts

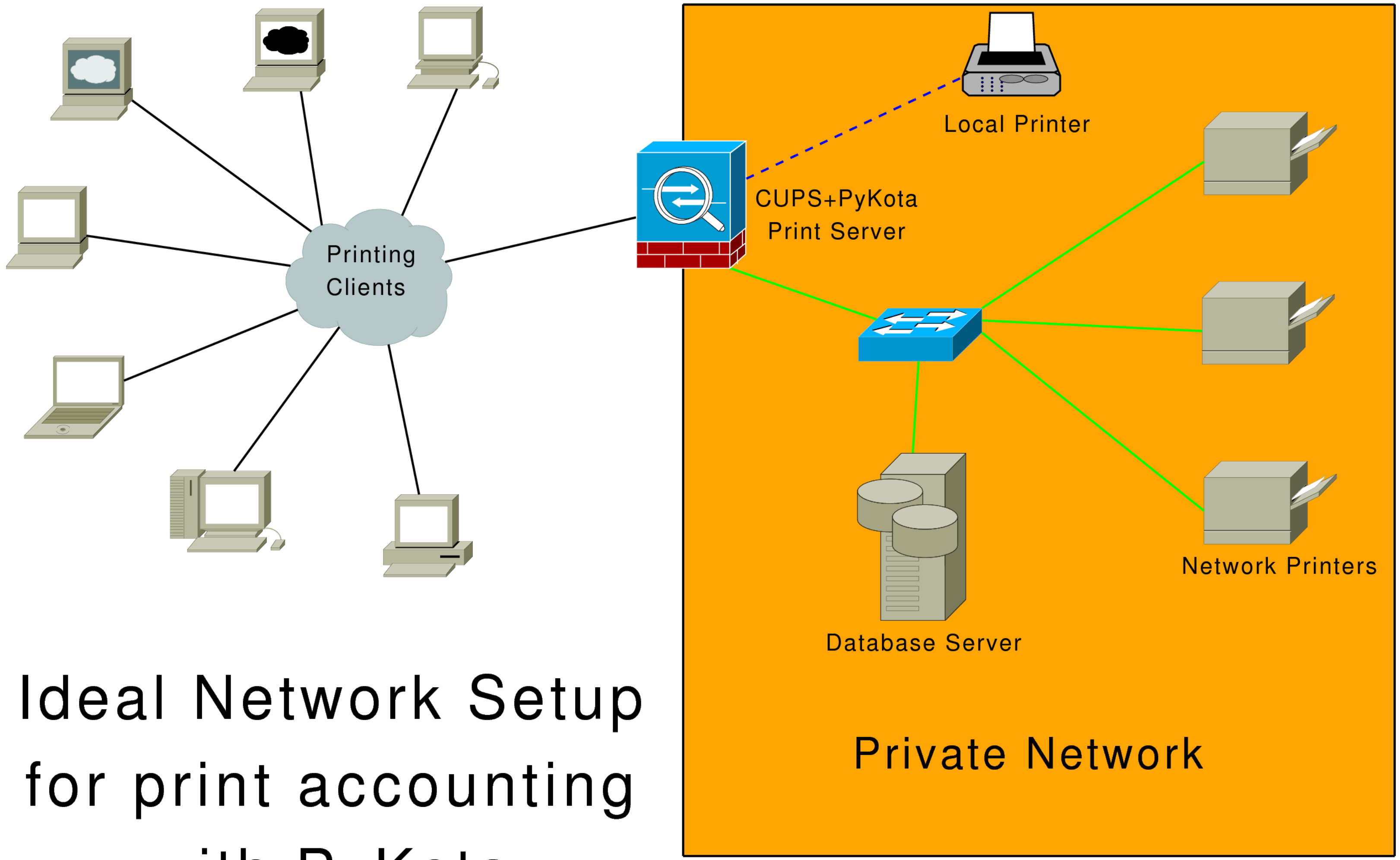
- Software accounting :
  - pkpgcounter :
    - Counting pages
    - Computing percent of ink coverage
  - External scripts :
    - Counting pages

Useful features

- Creation of printers and accounts « on the fly »
- Passthrough printers : during exams...
- Maximal size of a job per printer
- PDF invoices or refund certificates
- Export to CSV, XML...
- Users groups
- Nestable printers groups :

  Laser -> HP,DELL -> LJ2100, LJ2200, 5110CN

Recommended Setup



Ideal Network Setup
for print accounting
with PyKota

Distribution and Financing

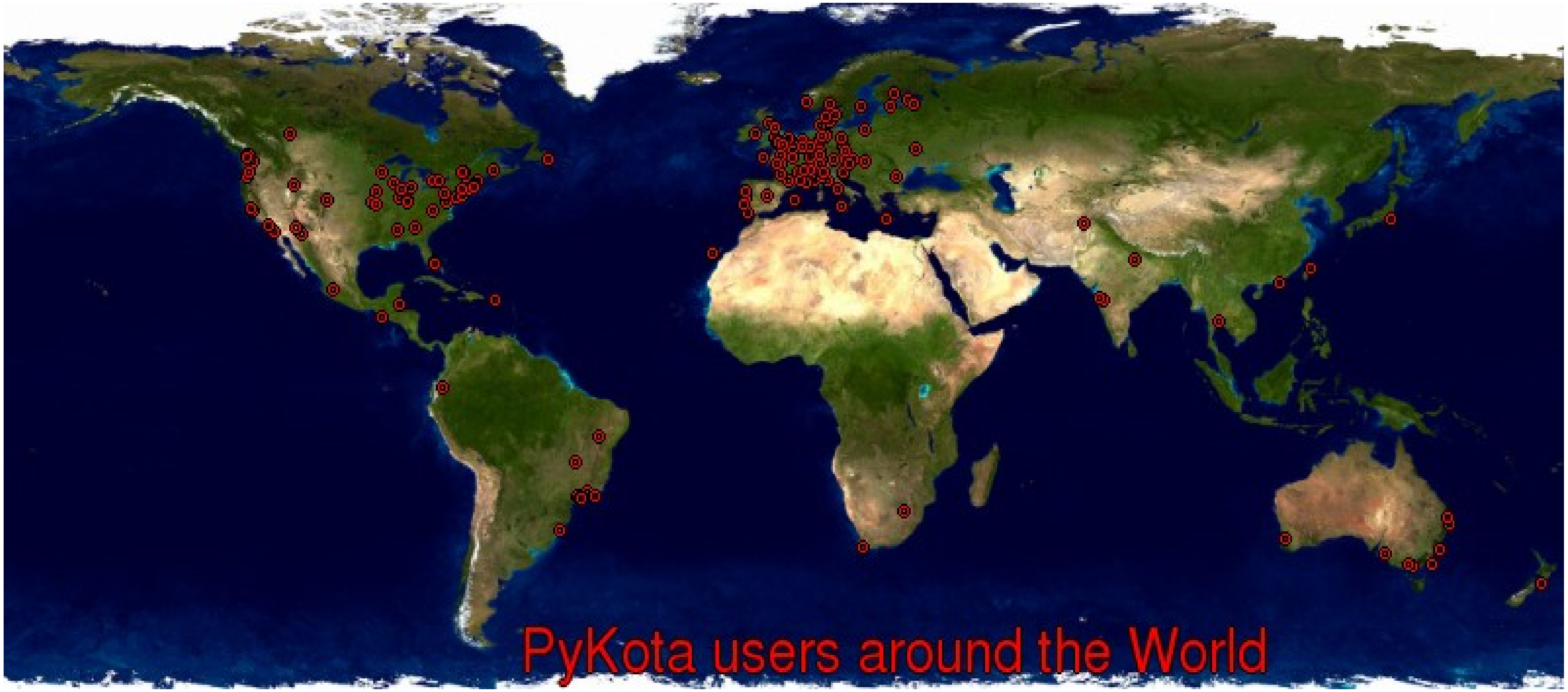
- Free Software
- GNU GPL v3 or + *(excepted Tea4CUPS under v2 or +)*
- Download from subversion or *.tar.gz*
- Excepted for PyKota and Tea4CUPS :
  - Free download through subversion
  - Download of « OFFICIAL » *.tar.gz*, *.deb*, *.rpm* with ***25 EUROS*** Entry Pass, unlimited duration.
  - But of course : redistribution and modification* are allowed.

  * I ask that any modified « official » release doesn't carry the « official » word anymore in the version number. **<u>This is not a legal obligation.</u>**
- Technical Support Services Agreements
- Guided installations

Impact of distribution method

- Community not very contributive in code, but mostly contributive in money
- It's possible to encourage contributions of code :
  - Various gifts : books, T-shirts...
  - Money
  - ???
  - ➔ BUT : how to stay fair ?
- Reactions :
  - Few criticism (1%)
  - No redistribution of « Official » packages by a third party, despite this being allowed
  - Some add-ons were developed

Who uses PyKota ?



Some links !

- Website :
  - *http://www.pykota.com*
- Bug Tracker :
  - *http://trac.pykota.com*
- IRC :
  - *#pykota* on *irc.freenode.net*
- Mailing lists *@lists.pykota.com* :
  - *pykota-devel* (25+) : mostly « commits »
  - *pykota* (300+) : user assistance
  - *pykota-support* (580+), readonly, registered users : announces, bugfixes, sécurity...