PgDay Paris 2017

Can't imagine what a newbie can do with PostgreSQL on Linux in a month

SA Infoconsulting

iprec

Formations informatiques depuis 1971

More than 20 years working on databases administration and tuning

Consulting for major companies

Teaching IT to people in professional retraining

SA INFOCONSULTING

Database Consulting and Training

AGENDA

- **Project genesis**
- **The whole story**
- **Things and tips to remember**

PROJECT GENESIS

iprec

Formations informatiques depuis 1971

FOUNDING ELEMENTS

TRUST :

IT Training institute giving blank cheques to trainers

MOTIVATION :

Friendly students with strong technical orientation

INNOVATION :

Teacher with a lot of (progressive) ideas

THE BEGINNINGS : should have been better !

SPECIFICATIONS :

- Install and deploy a FREE **inventory management tool** for IPREC whereas leveraging students technical skills
- Present the first version of the tool **a month later**

HUMAN RESSOURCES :

- A project manager with **no time to help**
- 4 **newbies** in IT with some skills on Linux (and only two really motivated by the challenge)
- An IT Trainer, with good Linux and databases skills

But not so bad … we had big ASSETS

- Every IT staff member involved knows how to install and use Linux (they learned it at IPREC)
- The more motivated student is the more high-achieving one
- The trainer is a database expert (and loves PostgreSQL)

So here comes the idea : **we can do it ourselves !**

And then started the IT TALE !

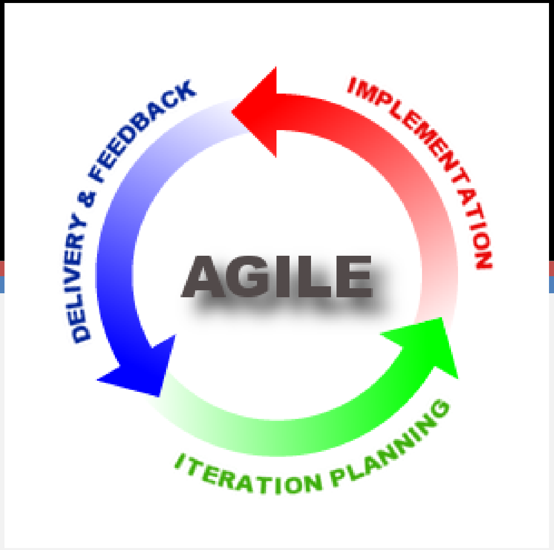
5 people in a room during a month with 5 PCs, internet, galions of coffee and with their personnal goals ⇒ The Dream Team

2 Motivated STUDENTS that wanted to develop a software on their own to put it on their CV and be proud of themselves

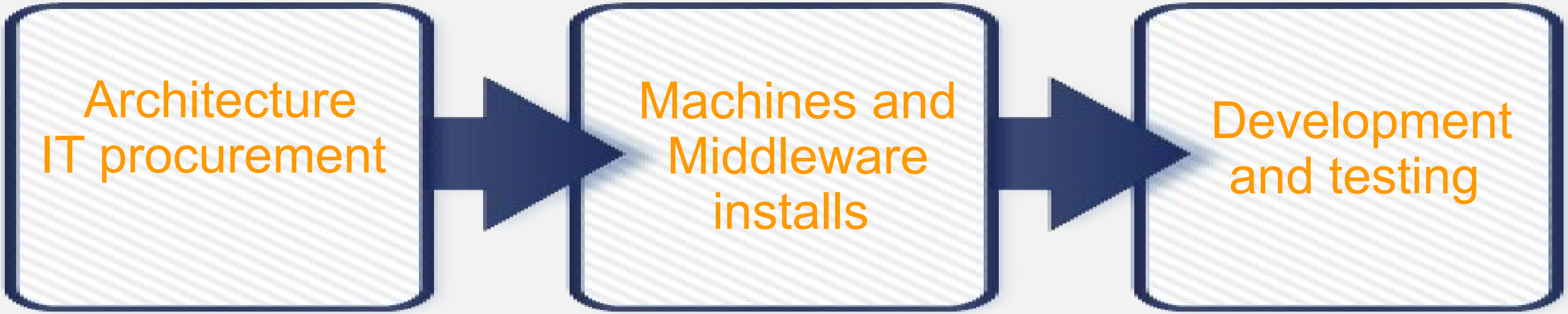
1 TRAINER and « technical supervisor » that wanted to consolidate students skills on Linux and develop their IT talents by learning how to install, use and maintain a PostgreSQL database

3 Others : well … loving breaks and surfing !

How could we make it in a month ?



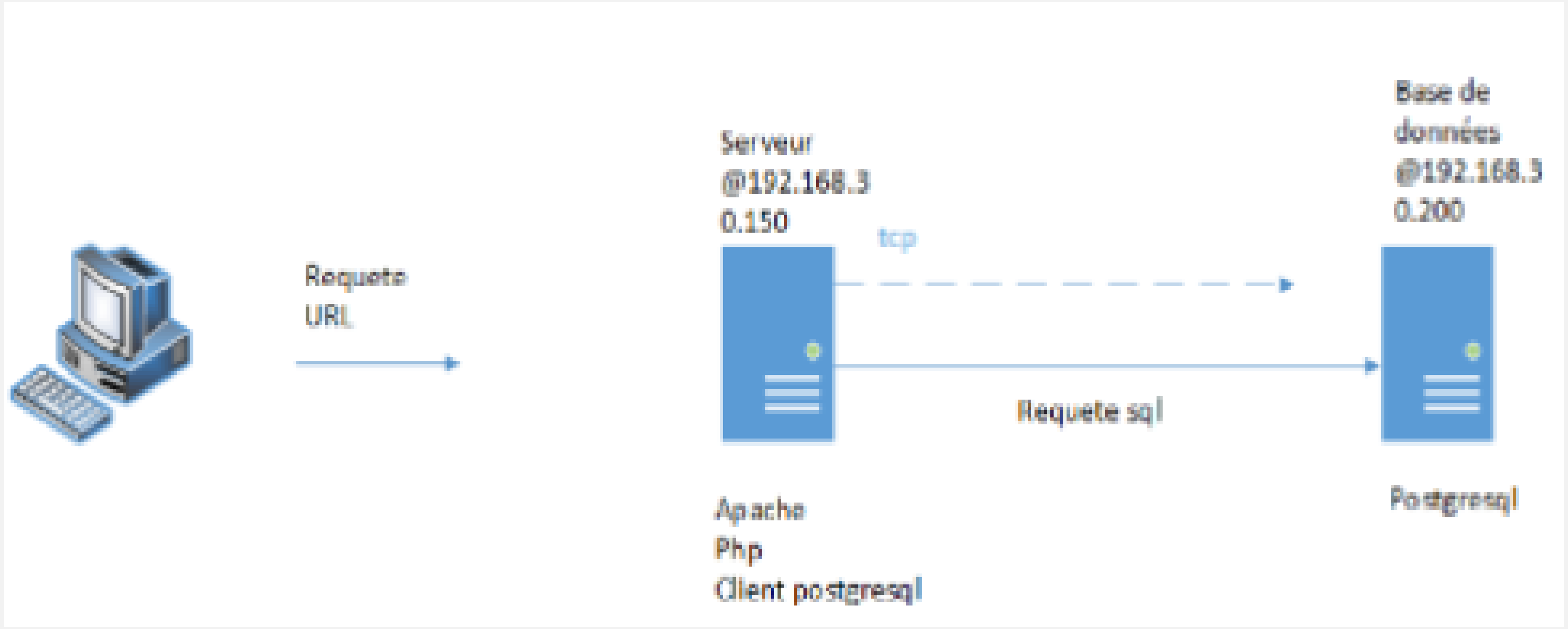
- Been AGILE
- Been PROACTIVE (alternative (inverted) learning)



3 days 5 days 15 days

Technical architecture

LAPP architecture



Installations of PostgreSQL Servers

2 identical environments (system and middleware installs)

- DEVELOPMENT: installed from scratch on a new Virtual Machine (VMWare) with :
  - * 1Gb of memory
  - * 20 Gb of Disk
  - * 8 file systems :
    - / : source directory
    - /root : administrator home directory
    - /bin : system (binaries)
    - /var : includes /var/log with system log files
    - **/opt** : middleware code (e.g. postgresql)
    - /tmp : for everything that is not mandatory
    - /usr : system softwares
    - /home : users home directories

- DEMO / PRODUCTION : new Virtual Machine generated by cloning the development once all installations and custimizations were done and successfully tested

Installation of PostgreSQL code

Once the VM is successfully installed, backed up and started,
PostgreSQL code and dependencies can be downloaded and installed :

# using a sudo account : install code
sudo yum install postgresql-server postgresql-contrib
# USE SYSTEM STORAGE MANAGER for convenience
sudo yum install system-strorage-manager
# using sudo : create XFS fs to store data and indexes (things are stored in /etc/fstab)
sudo ssm create -s 1G -n cronos_data --fstype xfs -p cronospool /dev/sdb /data/cronos
# give necessary permissions to postgres user
chown postgres.postgres /cronos

Start / Stop PostgreSQL

# using sudo : **create new db cluster**

postgresql-setup initdb

# postgres HOME should be here : /var/lib/pgsql

# customize postgresql.conf (e.g. change default port)

Basic configuration in the **postgresql.conf** file

- Binaries directory : PGENGINE = /usr/pgsql-9.5/bin
- Data directory: PGDATA = /var/lib/pgsql/9.5/data
- Engine log file : PGLOG = /var/lib/pgsql/9.5/pgstartup.log

# using sudo : startup & enable postgres

systemctl start postgresql

systemctl enable postgresql

Basic security on PostgreSQL

Network side basics :

- **change default port (5432) in /etc/services file**
- **custom pg_hba.conf** file to allow only php server and PC admin Client to connect directly to postgresql server (see next page for details)
- **custom postgresql.conf** :
  - port TCPIP used by the engine : PGPORT = 5432 by default
  - IP adresses listening on the postgreSQL port : LISTEN_ADRESSES='localhost,@ip server ' (ou '*')

Command to check if the listener is running : netstat -an ! grep 5432

Basic security on PostgreSQL

Custom pg_hba.conf file to allow only php server and PC admin client to connect directly to postgresql server

# Every local connected user can access to everything from everywhere !

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             127.0.0.1/32            trust
```

# All users with valid authentication on 192.xxx.xx.10 host can connect to database my_db

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    my_db           all             192.xxx.xx.10/32        md5
```

# All users with valid authentication in PostgreSQL database on a machine belonging to the test.com domain (!! password sent in clear ; prefer ldap ou kerberos authentication)

```
# TYPE  DATABASE        USER            ADDRESS                 METHOD
host    all             all             .test.com               password
```

Basic security on PostgreSQL

And if it’s not enough for you ...

it’s posible to **ACTIVATE SSL INTEGRATED AUTHENTICATION**

* Open SSL has to be installed on both postgresql server and its clients

* set **SSL=ON** in the postgresql.conf file and correlated parameters

Reminder : users have to be declared in postgresql befor using this

See https://www.postgresql.org/docs/9.5/static/ssl-tcp.html for more inforation

Creating the Database

```
# login as postgres user for everything else:

# create directory for data and indexes
cd /cronos ; mkdir data ; mkdir indexes

# create db owner
CREATE USER cronos_owner WITH LOGIN PASSWORD 'XXX' ;
# create db user
CREATE USER cronos WITH LOGIN PASSWORD 'XXX' ;

# create tablespaces for db (data & indexes)
CREATE TABLESPACE data_cronos OWNER cronos_owner LOCATION '/cronos/data'
CREATE TABLESPACE index_cronos OWNER cronos_owner LOCATION '/cronos/indexes';
# create db
CREATE DATABASE cronos_db OWNER cronos_owner TABLESPACE data_cronos
```

Extensions : we did not need them but ...

... this should be done to secure the database :
EXTENSIONS (contribs) add more functionnalities to postgresql including pg_crypto(PGP included)

Check installed contribs :
Select name from pg_available_extensions
Where installed_version is not null;

Add an extension to an existng database :
System install : yum *install package_extension*
PostgreSql install : *CREATE EXTENSION pgcrypto*

PostgreSQL Backups : the basic way

Taking a Database Backup with pg_dump

⇒ **create a sql script**

```
pg_dump cronos_db > cronos_db.sql
```

⇒ **create a compressed dump file**

```
pg_dump -Fc -f cronos_db.dump cronos_db
```

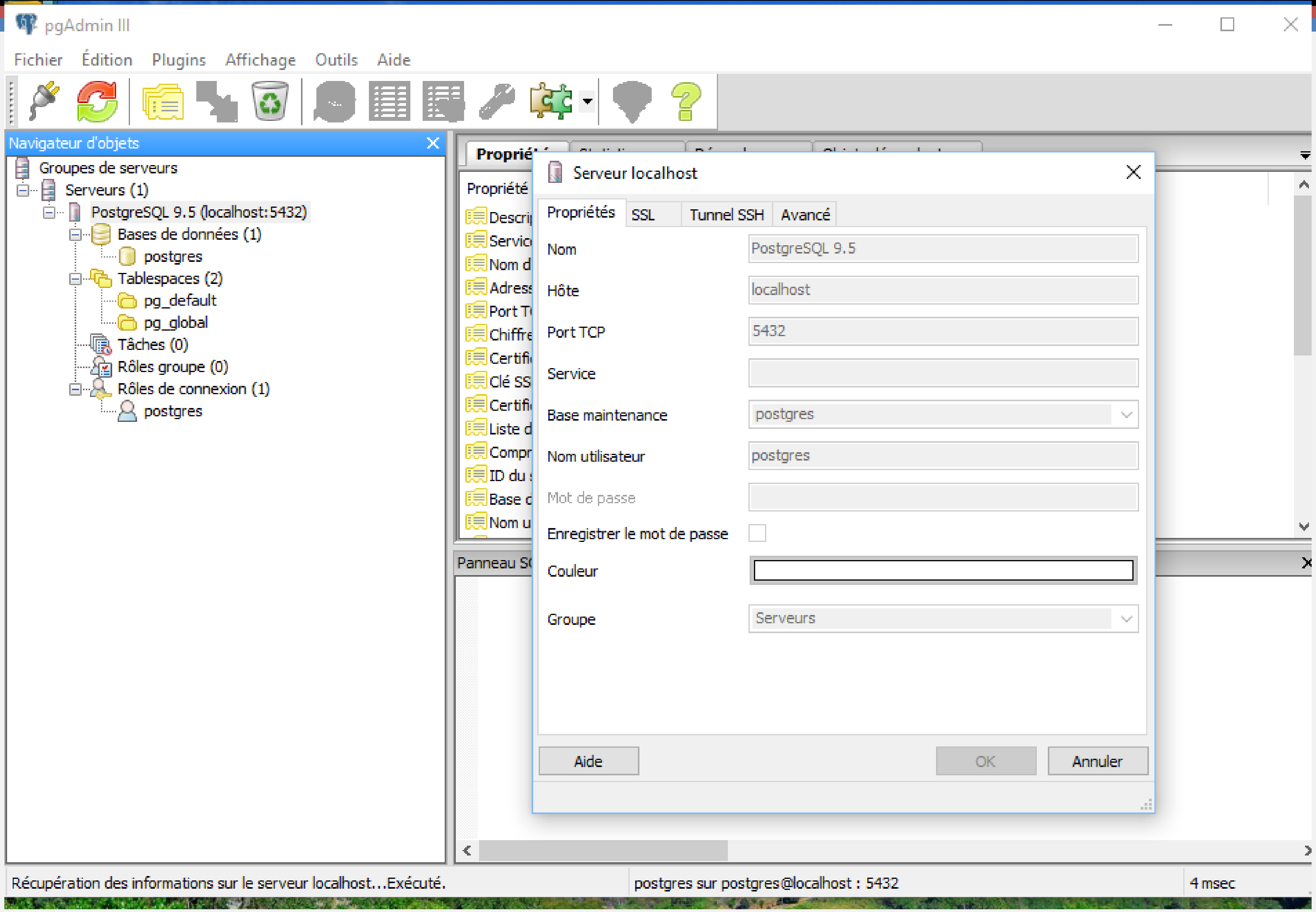
PostgreSQL Backups … should be restored

We did not need it but it should have been necessary

☐restore a database from its backup sql script
psql -d mydb -f cronos_db.sql

☐restore a database from its compressed backup file
pg_restore -C -d postgres cronos_db.dump

Admintool Installation (pgAdmin III)



Groupes de serveurs
- Serveurs (1)
  - PostgreSQL 9.5 (localhost:5432)
    - Bases de données (1)
      - postgres
    - Tablespaces (2)
      - pg_default
      - pg_global
    - Tâches (0)
    - Rôles groupe (0)
    - Rôles de connexion (1)
      - postgres

Serveur localhost

Propriétés | SSL | Tunnel SSH | Avancé

Nom	PostgreSQL 9.5
Hôte	localhost
Port TCP	5432
Service	
Base maintenance	postgres
Nom utilisateur	postgres
Mot de passe	
Enregistrer le mot de passe	☐
Couleur	
Groupe	Serveurs

Récupération des informations sur le serveur localhost...Exécuté. | postgres sur postgres@localhost : 5432 | 4 msec

Web server basic configuration to access PostgreSQL

Install apache and php servers : nothing special, only the basics in the httpd.conf file

```
# DirectoryIndex: name of the « head » file used by Apache
<IfModule dir_module>
    DirectoryIndex index.html
</IfModule>
# Utilisation de PHP 5.x:
LoadModule php5_module        modules/libphp5.so
AddHandler php5-script php
# Adding the « head » php »  file index.php to DirectoryIndex :
DirectoryIndex index.html index.php
AddType application/x-httpd-php .php
```

Then do the same thing in the php.conf file

```
<FilesMatch \.php$>
    SetHandler application/x-httpd-php
</FilesMatch>
AddType text/html .php
DirectoryIndex index.php
php_value session.save_handler "files"
php_value session.save_path    "/var/lib/php/session"
```

Time to code : connecting to the database

Database connection php file (in **/var/www/html** directory)
base.php

```
<?php

$dbconn = pg_connect("host=192.xxx.xx.200 port=5432
dbname=cronos user=cronos_owner password=xxxx")
    or die('Connexion impossible : ' . pg_last_error($conn));

// Close connection
pg_close($dbconn);
?>
```

Time to code : retrieving data from the database

```
// Executing a Select
$query = 'SELECT nom, prenom, role FROM login';
$result = pg_query($query) or die('Échec de la requête : ' . pg_last_error());

// Displaying results in HTML from php
echo "<table>\n";
while ($line = pg_fetch_array($result, null, PGSQL_ASSOC)) {
    echo "\t<tr>\n";
    foreach ($line as $col_value) {
        echo "\t\t<td>$col_value</td>\n";
    }
    echo "\t</tr>\n";
}
echo "</table>\n";
// Release memory
pg_free_result($result);
```

Time to code : errors management

```
// Build SQL
$sql="INSERT INTO users  VALUES ($prenom,$nom, $login.)"

// Execute SQL
@$result=pg_query($dbconn,$sql);

// Deal with error message
if(!$result)
{ $error= pg_last_error($dbconn);
   if($error==='ERROR: duplicate key value violates unique constraint "pk_user"')
   $outputmessage="utilisateur existant";
}
```

Time to code : the tricks we faced with

BOOLEANS

PHP deals with PostgreSQL booleans as characters strings

⇒ ‘F’ for FAUX and ‘t’ for VRAI because those are the values stored by PostgreSQL

ARRAY type

Not supported by PHP

What we shouldn’t have done

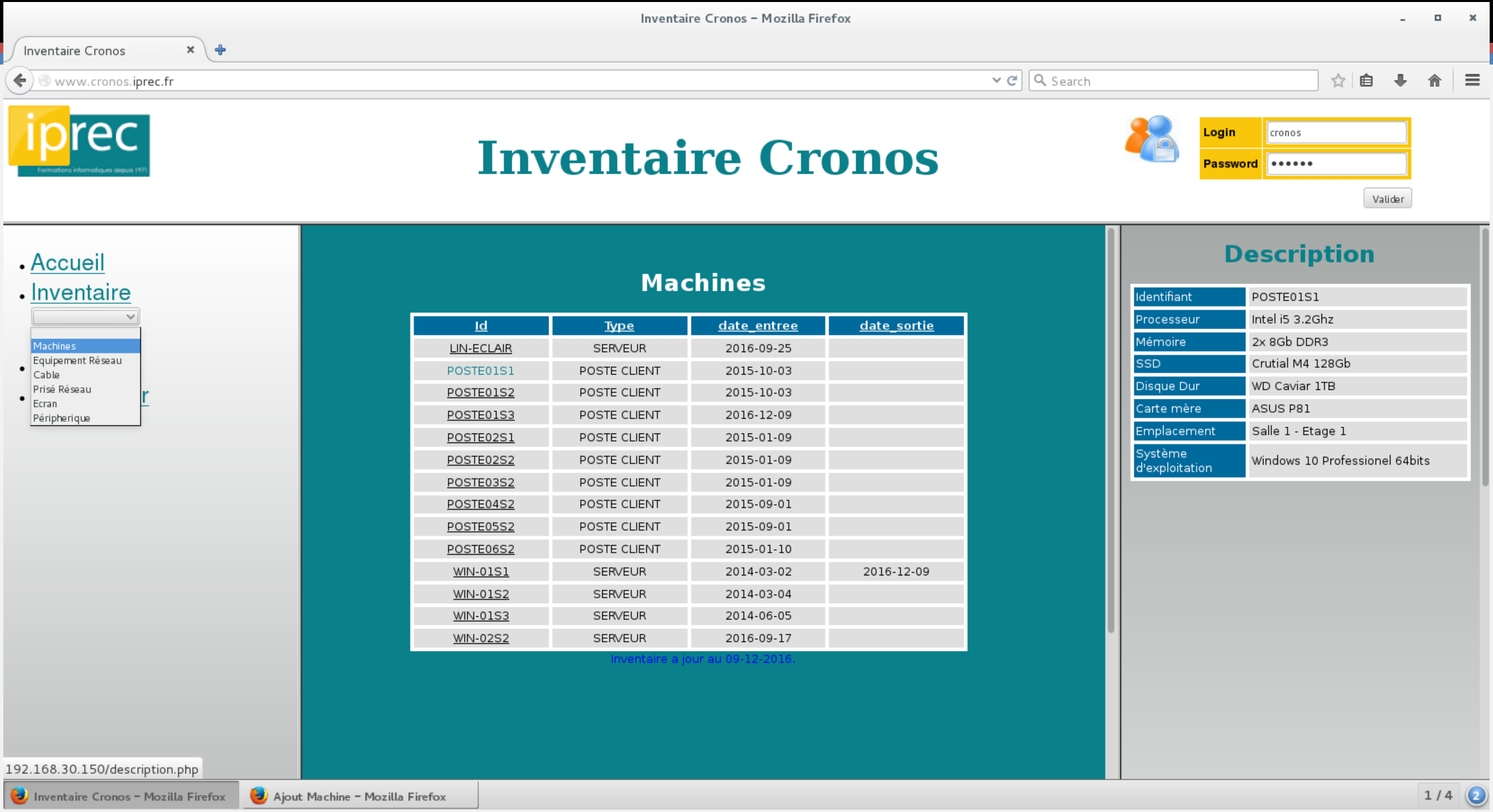
Be too impatient to code and **searching the internet to get php code samples before reading PostgreSQL documentation**

Think that coding with php on mysql and with php on postgresql is the same : **error when trying to execute some updates** :

```
UPDATE machines, screens SET machines.id_screen=screens.id WHERE
screens.id_machine = machines.id;
```

⇒ works in mysql but not in Postgresql

And after 3 weeks we get this



Inventaire Cronos

Machines

Id	Type	date_entree	date_sortie
LIN-ECLAIR	SERVEUR	2016-09-25	
POSTE01S1	POSTE CLIENT	2015-10-03	
POSTE01S2	POSTE CLIENT	2015-10-03	
POSTE01S3	POSTE CLIENT	2016-12-09	
POSTE02S1	POSTE CLIENT	2015-01-09	
POSTE02S2	POSTE CLIENT	2015-01-09	
POSTE03S2	POSTE CLIENT	2015-01-09	
POSTE04S2	POSTE CLIENT	2015-09-01	
POSTE05S2	POSTE CLIENT	2015-09-01	
POSTE06S2	POSTE CLIENT	2015-01-10	
WIN-01S1	SERVEUR	2014-03-02	2016-12-09
WIN-01S2	SERVEUR	2014-03-04	
WIN-01S3	SERVEUR	2014-06-05	
WIN-02S2	SERVEUR	2016-09-17	

Inventaire a jour au 09-12-2016.

Description

Identifiant	POSTE01S1
Processeur	Intel i5 3.2Ghz
Mémoire	2x 8Gb DDR3
SSD	Crutial M4 128Gb
Disque Dur	WD Caviar 1TB
Carte mère	ASUS P81
Emplacement	Salle 1 - Etage 1
Système d'exploitation	Windows 10 Professionel 64bits

3 things and tips to remember

- There's **no need to have DBA skills to install and maintain basic PostgreSQL** databases
- **Always go first to the PosgreSQL website** before looking everywhere else
- and last but not least :
- “**It is not because things are difficult that we do not dare; it is because we do not dare that things are difficult**.”
  -Seneca-

We used these books

Basics are all in them, and also some more information to go further :

Online documentation : http://www.postgresql.org/docs/manuals/

PostgreSQL Up & Running (R. Obe & L. Hsu / O'Reilly)