Table of Contents

Table of Contents 1
ISSUE-TRACKER DEVOPS GUIDE 2
1. INSTALLATIONS AND CONFIGURATIONS 2
1.1. Configure the Ubuntu repositories 2
1.2. Add the media keys 2
1.3. Install the postgre package with apt 2
1.4. Change the postgre user password 2
1.4.1. start the postgreSQL 2
1.4.2. Start the psql client as the postgres shell user 2
1.4.3. Create the pgsql user 3
1.4.4. add the uuid generation capability enabling extension 3
1.5. Install the perl modules (optional) 3
2. MAINTENANCE AND OPERATIONS 4
2.1. RDBMS Runstate management 4
2.1.1. To check the status of the postgreSql 4
2.1.2. To stop the postgreSql 4
2.1.3. To start the postgreSql 4
2.1.4. to check the port on which it is listening 4
2.1.5. Check the postgres status 4
2.2. Application Layer runstate management 5
2.2.1. start the application layer 5
2.2.2. stop the application layer 5
3. USAGE SCENARIOS 5
3.1. Shell based actions usage 5
3.1.1. Run increase-date action 5
3.1.2. Run xls-to-db action 5
3.1.3. Run db-to-txt action 5
3.1.4. Load xls issues to db and from db to txt files 5
3.1.5. Run the issue-tracker file to db load 6
3.1.6. Verify the inserted data from the db 6
3.2. web based routes usage 6
3.2.1. Run the http://<<web-host>>:<<web-port>>/<<proj-db>>/get/<<table>>/<<guid>> route 6
4. BUSINESS LOGIC 6
4.1. Projects management 6
4.2. Categories 7
4.2.1. Issues / Issue items / items 7
5. NAMING CONVENTIONS 7
5.1. Dirs naming conventions 7
5.2. Root Dirs naming conventions 7
6. SOURCE CODE MANAGEMENT 7
6.1. The meaning of the used brances 7

ISSUE-TRACKER DEVOPS GUIDE

1. INSTALLATIONS AND CONFIGURATIONS

1.1. Configure the Ubuntu repositories

Configure the Ubuntu repositories

```
# creae the following repo list file
sudo vim /etc/apt/sources.list.d/pgdg.list

# add the following line in it:
deb http://apt.postgresql.org/pub/repos/apt/ trusty-pgdg main
```

1.2. Add the media keys

Add the media keys as follows:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc |   sudo apt-key add -
```

1.3. Install the postgre package with apt

Install the postgre package with apt

```
# update your repos
sudo apt-get update

# install the postgresql binary
sudo apt-get install postgresql postgresql-contrib

# enable postgre
sudo update-rc.d postgresql enable
```

1.4. Change the postgre user password

Configure the Ubuntu repositories

```
sudo passwd postgres
# Type a pw - add to your password manager !!!

# and verify
su - postgres
```

1.4.1. start the postgreSQL

Start the postgreSQL by issuing the following command

```
sudo /etc/init.d/postgresql start
```

1.4.2. Start the psql client as the postgres shell user

Start the psql client as the postgres shell user
source:
http://dba.stackexchange.com/a/54253/1245

```
sudo su - postgres
# start the psql client
psql

# the psql prompt should appear as
# postgres=#

# list the databases
\l
#and quit
\q
```

1.4.3. Create the pgsql user

Create the pgsql user and grant him the privileges to create dbs and to connect to the postgres db.
You could alternatively configure different way of authenticatio according to the options provided in this stackoverflow answer:
http://stackoverflow.com/a/9736231/65706

```
# create the pgsql user to be the same as the shell
# user you are going to execute the scripts with
sudo su - postgres  -c "psql -c 'CREATE USER '$USER' ;'"

# grant him the priviledges
sudo su - postgres  -c "psql -c 'grant all privileges on database postgres to '$USER' ;'"

# grant him the privilege to create db's
sudo su - postgres  -c "psql -c 'ALTER USER '$USER' CREATEDB;'"

sudo su - postgres  -c 'psql -c "select * from information_schema.role_table_grants
 where grantee='"'"$USER"'";'"
```

1.4.4. add the uuid generation capability enabling extension

add the uuid generation capability enabling extension

```
sudo su - postgres  -c "psql template1 -c 'CREATE EXTENSION IF NOT EXISTS \"uuid-ossp\";'"

sudo su - postgres  -c "psql template1 -c 'CREATE EXTENSION IF NOT EXISTS \"pgcrypto\";'"
```

1.5. Install the perl modules (optional)

Install the perl module by first installing the server development package

```
# check which server development packages are available
sudo apt-cache search postgres | grep -i server-dev | sort
```

```
# install it
sudo apt-get install -y postgresql-server-dev-9.6

# install the DBD::Pg module
sudo perl -MCPAN -e 'install DBD::Pg'

sudo perl -MCPAN -e 'Tie::Hash::DBD'
```

2. MAINTENANCE AND OPERATIONS

2.1. RDBMS Runstate management

2.1.1. To check the status of the postgreSql

To check the status of the postgreSql issue:

```
sudo /etc/init.d/postgresql status
```

2.1.2. To stop the postgreSql

To stop the postgreSql issues:

```
sudo /etc/init.d/postgresql stop
```

2.1.3. To start the postgreSql

To start the postgreSql issues:

```
sudo /etc/init.d/postgresql start
```

2.1.4. to check the port on which it is listening

To check the port on which it is listening issue:

```
sudo netstat -tulntp | grep -i postgres
# tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      8095/postgres
```

2.1.5. Check the postgres status

Check the postgres status.

Check the port to which the postres is running with this command:

```
sudo /etc/init.d/postgresql status

# restart if needed
sudo /etc/init.d/postgresql restart

# check on which ports it is runnning
sudo netstat -plunt |grep postgres
```

2.2. Application Layer runstate management

2.2.1. start the application layer

To start the application layer in development mode use the morbo command (debug output will be shown) , to start it in production mode use the hypnotoad pattern

```
bash src/bash/issue-tracker/issue-tracker.sh -a mojo-hypnotoad-start

bash src/bash/issue-tracker/issue-tracker.sh -a mojo-morbo-start
```

2.2.2. stop the application layer

To stop the application layer in development mode use the morbo command (debug output will be shown) , to start it in production mode use the hypnotoad pattern

```
bash src/bash/issue-tracker/issue-tracker.sh -a mojo-hypnotoad-stop

bash src/bash/issue-tracker/issue-tracker.sh -a mojo-morbo-stop
```

3. USAGE SCENARIOS

3.1. Shell based actions usage

3.1.1. Run increase-date action

You track the issues of your projects by storing them into xls files in "daily" proj_txt dirs.
Each time the day changes by running the increase-date action you will be able to clone the data of the previous date and start working on the currnent date.

```
bash src/bash/issue-tracker/issue-tracker.sh -a increase-date
```

3.1.2. Run xls-to-db action

You insert the date of the daily , weekly , monthly or yearly issues from the daily input excel file(s) by running the xls-to-db action.

```
bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db
```

3.1.3. Run db-to-txt action

3.1.4. Load xls issues to db and from db to txt files

to load xls issues to db and from db to txt files

```
bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db -a db-to-txt

# or run for all the periods
for period in `echo daily weekly monthly yearly`; do export period=$period ;
```

```
bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db -a db-to-txt ; done ;
```

3.1.5. Run the issue-tracker file to db load

Run the issue-tracker file to db load

```
# ensure the following actions will be tested
cat src/bash/issue-tracker/tests/run-issue-tracker-tests.lst | grep -v '#'
# output should be if not correct
check-perl-syntax
run-issue-tracker

# test those uncommented actions
bash src/bash/issue-tracker/test-issue-tracker.sh
```

3.1.6. Verify the inserted data from the db

Verify the inserted data from the db as follows:

```
# check that the rows where inserted
echo 'SELECT * FROM issue ; ' | psql -d dev_issue_tracker
```

3.2. web based routes usage

3.2.1. Run the http://<<web-host>>:<<web-port>>/<<proj-db>>/get/<<table>>/<<guid>> route

Load a table with guid's.
Check a single item with your browser, for example:
http://doc-pub-host:3000/dev_stockit_issues/get/company_eps/727cf807-c9f1-446b-a7fc-65f9dc53ed2d

```
# load the items
while read -r f; do export xls_file=$f; bash src/bash/issue-tracker/issue-tracker.sh -a xls-to-db  ; done < <(find $proj_txt_dir -type f)

# verify the data
psql -d $db_name -c "SELECT * FROM company_eps "
```

4. BUSINESS LOGIC

4.1. Projects management

You can manage multiple projects with the issue-tracker tool. Each project has its own data directories, database storage and configurations. You could also have different envornments named dev,tst,prd for each project separately.
As the tool is backwards compatible you could have differrrent instances of the issue-tracker projects with different versions (and set of features) operatiing against differrent project (each one in its own version).
You must pre-set the configuration variables of an issue-tracker project each time you start working on a project from the shell

```
doParseIniEnvVars /vagrant/csitea/cnf/projects/isg-pub/isg-pub.issue-tracker.doc-pub-host.conf
```

4.2. Categories

Each issue item could be categorized under one and only one category. One category might have 1 or more issues.

The categories could contain letters ,numbers, dashes

Examples: organisation-it organisation-it-operations

4.2.1. Issues / Issue items / items

Issue item is the shortest possible description of task , activity , note or anything requiring distinguishable and prerferable measurable action or producing verfifiable outcome.

Issues could be of different types - tasks, activities, notes etc.

Examples: go get the milk do the homework procurement e-mail discussion follow-up

5. NAMING CONVENTIONS

5.1. Dirs naming conventions

The dir structure should be logical and a person navigating to a dir should almost understand what is to be find in thre by its name ..

5.2. Root Dirs naming conventions

The root dirs and named as follows:

bin - contains the produced binaries for th project

cnf - for the configuration

dat - for the data of the app

lib - for any external libraries used

src - for the source code of the actual projects and subprojects

6. SOURCE CODE MANAGEMENT

The issue-tracker is a derivative of the wrapp tool - this means that development and deployment process must be integrated into a single pipeline.

6.1. The meaning of the used brances

In almost all development projects there are slightly or even quite big differences between what type of code in which branch is situated.

The ideology of issue tracker is that the code which is under active development is in the dev branch , the code which is under testing in the tst branch , the code which is in production in the prd branch.

Only after the code in production has been successfully operated and prooved working it could be moved to the master branch and the version increased.

Once you wanto to start adding new feature branch from the master branch.