Table of Contents

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'"'"';"'

# and exit
\q
```

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. 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
```

3. NAMING CONVENTIONS

3.1. 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 ..

3.1.1. 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

4. DYNAMIC SCENARIOS

4.1. Load issue txt files to db

This scenario implements the "Load issues file from file system to db" user story.

4.1.1. 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
```

4.1.2. Run the create db and create issue table scripts

Run the create db and create issue table scripts with the following call

```
# run the create db and create table scrpt
# define the sql_dir where the issue tracker sql scripts are stored
export sql_dir=/opt/csitea/issue-tracker/issue-tracker.0.0.5.dev.ysg/src/sql/pgsql/dev_issue_tracker

# run the scripts
bash /opt/csitea/pgsql-runner/src/bash/pgsql-runner.sh  -a run-pgsql-scripts

# ensure from the STDOUT msgs that both the db and the table were created
```

4.1.3. 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
```

4.1.4. 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
```

5. BUSINESS LOGIC

5.1. Business entities

5.2. 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
```

5.3. 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
```

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.