Table of Contents

ISSUE-TRACKER INSTALLATIONS AND CONFIGURATION GUIDE

1. INTRODUCTION

1.1. Purpose

The purpose of this document is to describe the tasks and activities to be performed in order to achieve a fully operational issue-tracker application instance.

1.2. Document status

This document is updated constantly in every release of the issue-tracker. Each version however is considered to be a complete version regarding the issue-tracker version it situated in.

1.3. Audience

This document is aimed for everyone, who shall, will or even would like to interact with an issue-tracker application instance. Although this guide is aimed to be fully implementable via copy paste by following top to bottom, you need to have basic understanding of networking, protocols and Linux in order to complete the full installation, as your mealige will vary...

1.4. Master storage and storage format

The master storage of this document is the master branch of the issue-tracker release you are interested in. The main storage format is Markdown.

1.5. Version control

The contents of this document MUST be updated according to the EXISTING features, functionalities and capabilities of the issue-tracker version, in which this document residues.

1.6. Process

The issue-tracker provides a mean for tracking of this documentation contents to the source code per feature/functionality, thus should you find inconsistencies in the behavior of the application and the content of this document you should create a bug issue and assign it to the owner of your product instance.

2. OS LEVEL ACCESS

The issue-tracker has been deployed on Ubuntu OS running virtual machines , physical hosts and aws instances.

2.1. OS level access via aws

This section will provide you with the instructions to instantiate an Ubuntu 16 micro instance in AWS. Should create an AWS account if you have not yet one ...

2.1.1. Launch a new instance

Go to your management console's by clicking Services-EC2 (note I am using the region=eu-west-1, you might be using a different one) :
https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=eu-west-1#Instances:sort=monitoring
Click on the blue "launch instance" button.

2.1.1.1. Choose an Ubuntu AMI

Scroll down the list. Choose the "Canonical, Ubuntu, 16.04 LTS, amd64 xenial image build on 2018-05-22" AMI. Click on select

2.1.1.2. Select the type of instance

Choose the type of instance - I am using t2.micro, which is supposed to be free "freetire eligible". Click on the review and Launch (you could continue configuring till the end of the wizard but you should be ok with the defaults by now

2.1.2. Open ports

In the AWS menu click Services - EC2. Click on the left on the security groups. If you configured an explicit security group, select it by clicking on the check box next to it, if you did not that click on the check box to the default security group created something like "launch-wizard-1 created 2018-07-03T14:39:25.644+03:00", where the latest string is the time when you launched the instance.
Click on Actions - edit inbound rules - open the 3000 and 8080 tcp ports (the default ones used by the mojo and hypnotoad

malicious web servers)

2.1.3. Create a key-pair to access the instance via ssh

Click on the key-pairs in the left. Create a new one. Download the file in a path you could navigate later on to. I am using for example the following path:

```
/opt/csitea/issue-tracker/dat/sec/key-pair-issue-tracker.pem
```

2.1.4. Access the running instance via ssh.

Run the following commands, by replacing the ssh_server with the value taken from the "Private DNS" of the running instance.

```
export ssh_user=ubuntu
export ssh_server=ec2-31-223-97-157.eu-west-1.compute.amazonaws.com
export key_pair_file=/opt/csitea/issue-tracker/dat/sec/key-pair-issue-tracker.pem

ssh -i $key_pair_file $ssh_user@$ssh_server
```

3. LIBS AND BINARIES INSTALLATIONS ON OS LEVEL

3.1. Check you can sudo with the ubuntu user

You should be able to sudo with your ubuntu user.

```
# set a nice informative prompt
export PS1="\h [\d \t] [\w] $ \n\n  "

sudo su -
```

3.2. Add the issue-tracker application OS user's group

The current "as simpliest as possible" setup is to use a separate OS group for the user of the issue-tracker application. The name of the group could be anything, as well it's gid , use those values if you do not want to make additional own configurations.

```
export group=grp_it
export gid=10001
sudo groupadd -g "$gid" "$group"
sudo cat /etc/group | grep --color "$group"
```

3.3. Add the issue-tracker application OS user

Add the user as shown bellow. The name of the user could be also anyting, as log as it belongs to the above configured group.

```
export user=usr_it
export uid=10001
export home_dir=/home/$user
export desc="the hadoop group"
#how-to add an user
sudo useradd --uid "$uid" --home-dir "$home_dir" --gid "$group" \
--create-home --shell /bin/bash "$user" \
--comment "$desc"
sudo cat /etc/passwd | grep --color "$user"
groups "$user"
```

3.4. Add the issue-tracker app user to the sudoers group

Edit the /etc/group

```
sudo:x:27:ubuntu,usr_it
```

3.5. Allow sudo without having to type password

Add the following text to the /etc/sudoers file:

```
# See sudoers(5) for more information on "#include" directives:
usr_it  ALL=(ALL) NOPASSWD: ALL
```

3.6. Install the git binary

You will need the git binary to fetch the issue-tracker source - you could alternatively download its zip fille.

```
sudo apt-get install build-essential git
```

3.7. Fetch the source

Fetch the source from git hub as follows:

```
# got to a dir you have write permissions , for example:
mkdir -p ~/opt/; cd ~/opt/

# fetch the source
git clone https://github.com/YordanGeorgiev/issue-tracker.git

# checkit
ls -la
```

3.8. run the boot-strap script

The bootstrap script will interpolate change the git deployment dir to a "product_instance_dir" (your instance of the issue-tracker, having the same version as this one, but running on a different host with different owner - your)

```
# defiine the latest and greates product_version
export product_version=$(cd issue-tracker;git tag|sort -nr| head -n 1;cd ..)

# check it
echo $product_version

# run the bootstrap script :
bash issue-tracker/src/bash/issue-tracker/bootstrap-issue-tracker.sh
```

3.9. Install the OS libs listed in the prereq sh script

The "prereq"script contains the listing of the binaries to be installed. Run the script.

```
# go to your product instance dir -
cd ~/opt/csitea/issue-tracker/issue-tracker.$product_version.dev.$USER
sudo bash src/bash/issue-tracker/install-prerequisites-on-ubuntu.sh
```

3.10. Install the Perl modules listed in the preq pl scriipt

The Perl modules installation looks ALWAYS, yet it usually goes trough without any googling of sessions etc.

```
# go to your product instance dir -
cd /opt/csitea/issue-tracker/issue-tracker.$product_version.dev.$USER
sudo perl src/perl/issue_tracker/script/issue_tracker_preq_checker.pl
```

3.11. Disable the warnings in the OAuth2.pm

As many of the actions in the issue-tracker this is simply a one-liner.

```
vim /usr/local/share/perl/5.22.1/Net/Google/DataAPI/Auth/OAuth2.pm
# edit the use Any::Moose; line as follows:

no warnings 'deprecated';
use Any::Moose;
use warnings 'deprecated' ;
```

3.12. Check that the project source code

As many of the actions in the issue-tracker this is simply a one-liner. You should not get any errors after the execution of this one , if there are probably some prerequisite module has not been included in the prerequisite script (although we have tested those several times ...). So you would either have to install it by yourself or troubleshoot why it does not install.

```
bash src/bash/issue-tracker/issue-tracker.sh -a check-perl-syntax
```

4. POSTGRES RELATED INSTALLATIONS AND CONFIGURATIONS

4.1. Configure the Ubuntu repositories

Configure the Ubuntu repositories

```
sudo add-apt-repository "deb http://apt.postgresql.org/pub/repos/apt/ xenial-pgdg main"

sudo apt-get update
sudo apt-get install postgresql-9.6
```

4.2. 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 -
```

4.3. Install the postgres package with apt

Install the postgres 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
```

4.4. Change the postgres user password

Configure the Ubuntu repositories

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

# and verify
su - postgres
```

4.4.1. start the PostgreSQL

Start the PostgreSQL by issuing the following command

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

4.4.2. Start the pgsql client as the postgres shell user

Start the pgsql 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
```

4.4.3. 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 authentication 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"'"';"'
```

4.4.4. 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\";'"
```

4.4.5. Install the dblink extension as follows

Install the dblink extension as follows

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

4.5. Install the perl modules

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'
```

5. LOAD THE ISSUE-TRACKER PROJECT DATA

5.1. Edit and pre-load the project env vars configuration file

As the issue-tracker supports loading data for multiple instances of itself - dev,tst,prd AND multiple pre-configured projects, YOU MUST have a shell with the pre-configured set of environmental variables in order to achieve ANY of the actions.

```
# load the bash func to parse the conf files
source lib/bash/funcs/parse-cnf-env-vars.sh

doParseCnfEnvVars cnf/issue-tracker.dev.$(hostname -s).cnf
```

5.2. Create the database and run the DDL scrripts

```
bash src/bash/issue-tracker/issue-tracker.sh -a run-pgsql-scripts
```

5.3. Start the mojo and hypno servers

The morbo dev server is listening on the 3000 port by default and the hypnotoad on the 8080 by default. You will need to use either one when loading the tables in the next step.

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

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

5.4. Load the data from the xls file

Load the data to the db from the xls file

```
clear ; export tables=`curl -s -k http://$web_host:8080/$postgres_db_name/select-tables|jq -r '.| .dat| .[] | .table_name'|perl -ne
's/\s+/,/g;print'`;export do_truncate_tables=1 ; export rdbms_type=postgres ; export load_model=upsert ; perl
src/perl/issue_tracker/script/issue_tracker.pl --tables $tables --do xls-to-db
```

5.5. Verify that the app layer and the db work together

In the aws menu click Services - EC2. Click on the instances on the left. Check the public DNS name of your instance and use it for the URL as follows:
http://ec2-34-243-97-157.eu-west-1.compute.amazonaws.com:8080/dev_issue_tracker/list/yearly_issues?as=table&pick=id,status,prio,name,weight,start_time,stop_time&page-size=20&page-num=1

6. FRONT END-INSTALLATIONS (OPTIONAL)

6.1. Install NodeJS on Ubuntu

From the following page:
https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions

```
sudo apt-get install -y build-essential
sudo apt-get install -y nodejs
```

6.2. Install npm

Install npm by issuing the following command:

```
sudo apt-get install npm
```

6.3. Install web-pack

Install webpack globally by issueing the following command:

```
sudo npm install -g webpack
```

6.4. Install bower

Install the bower package globally as follows:

```
npm config set prefix /usr/local
npm install -g bower
sudo npm install -g bower
which bower
```