QTO INSTALLATIONS AND CONFIGURATION GUIDE

QTO INSTALLATIONS AND CONFIGURATION GUIDE

1. INTRODUCTION

1.1. Purpose

The purpose of this document is to provide a description for the tasks and activities to be performed in order to achieve a fully operational qto application instance.

1.2. Document status

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

1.3. Audience

This document is aimed for everyone, who shall, will or even would like to install a qto 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 mileage will vary...

1.4. Master storage and storage format

The master storage of this document is the master branch of the latest qto release. 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 qto version, in which this document residues.

1.6. Process

The qto provides a mean for tracking of this documentation contents to the source code per feature/functionality, thus should you find inconsistencies in the behaviour 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. LOCAL DEPLOYMENT

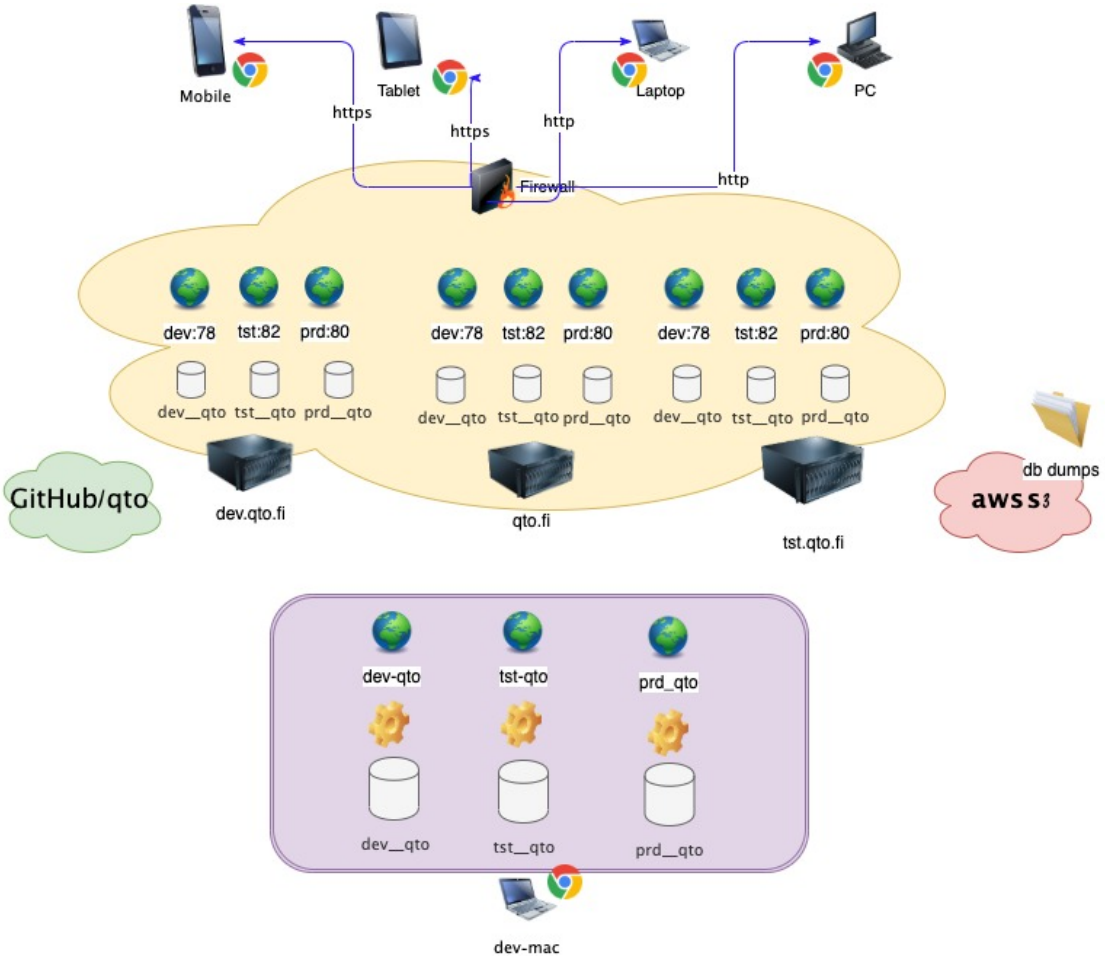
2.1. Prerequisites

You need an aws account, capable of deploying micro instances. The costs for running the qto.fi is about 15 USD per month for a sufficient micro instance (aka this aws deployment expense to Amazon would be less than a dollar), but if you are going to use a free account, which does not allow the instantiation of the ami's specified in the src/terraform/tpl/qto/main.tf.tpl file, you should modify the template file to accommodate that.

2.2. Target setup

The target setup is a system comprised of dev, tst, qas and prd instances of qto locally and dev, tst , qas and prd instances in aws.
The following diagram illustrates that setup. Naturally you will be deploying only the dev.<<site>>.com when doing the installation for first time - in the spirit of qto - you will be moving fast and destroying in dev, re-inforcing skills in tst and do it at once in prd ...

Figure 1: The target setup for the qto appliction



2.3. Bootstrap and deploy the application locally

The bootstrap script will deploy ALL the required Ubuntu 18.04 binaries AND perl modules as well as perform the needed configurations to enable the creation and load of the qto database. This step will take 20 min at least.
The bootstrapping script will :
- install all the required binaries
- install all the required perl modules as non-root
- install and provision postgres
- install and provision the nginx proxy server

Check the main method in the run.sh and uncomment entities you do not want to install locally, should you have any ...

```
# in the shell of your local Ubuntu box
mkdir -p ~/opt; cd $_ ; git clone
https://github.com/YordanGeorgiev/qto.git ; cd ~/opt

# run the bootstrap script and IMPORTANT !!! reload the bash
env
./qto/src/bash/deployer/run.sh ; bash ;
```

2.4. Provision the application locally

The run of the following "shell actions" will create the qto database and load it with a snapshot of it's data from a sql dump stored in s3. If you start getting a lot of perl "cannot not find module" syntax check error, you probably did not reload the bash shell , by just typing "bash" and hitting enter in the previous step.

```
 # go to the product instance dir
source $(find . -name '.env') && cd
qto/qto.$VERSION.$ENV_TYPE.$USER

# ensure application layer consistency, run db ddl's and
load data from s3
./src/bash/qto/qto.sh -a check-perl-syntax -a scramble-confs
-a provision-db-admin -a run-qto-db-ddl -a load-db-data-
from-s3 && rm -v bootstrapping
```

3. AWS DEPLOYMENT

This section WILL provide you will with ALL required steps to get a fully functional instance of the qto application in aws in about 35 min with automated shell commands. Do just copy paste the commands into your shell. Do NOT cd into different directories - this deployment has been tested more than 30 times successfully by exactly reproducing those steps, yet the variables in such a complex deployment are many, thus should you encounter new issues do directly contact the owner of the instance you got this deployment package from, that is the person owning the GitHub repository you fetched the source code from.

3.1. Prerequisites

You should have a local instance of qto as far as you could issue the terraform shell action (that is db and site configuration are not must have for the aws deployment to succeed).

3.2. Configure the AdminEmail

The AdminEmail value stored in the cnf/env/*.env.json files is the e-mail of the qto product instance owner - aka the only user being able to edit other users' details.
The bootstrapping script simply replaces the demo "test.user@gmail.com" with the password "secret" with the value you will set in the AdminEmail, thus once you have authentication use the "secret" password to login in and edit users.

3.3. Configure your aws credentials - aws keys and ssh keys

Generate NEW aws access- and secret-keys https://console.aws.amazon.com/iam/home?region=<<YOUR-AWS-REGION>>#/security_credentials.
Store the keys in the qto's development environment configuration file - the cnf/env/dev.env.json file.
When the file path is suggested append a different string to avoid overwriting the default private key path!!!

```
#create the a NEW public private key pair ( do not overwrite
you existing one !! ) as follows
ssh-keygen -t rsa -b 4096 -C "your.name@your.org"
```

3.4. Initialise the aws infrastructure

To initialise the git aws infrastructure you need to clone the qto source code locally first. If you are repeating this task all over you might need to remove from the aws web ui your pem keys and potential out of your dedicated resources VPC's and elastic IP's.

```
# apply the infra terraform in the
src/terraform/tpl/qto/main.tf.tpl
clear ; bash ~/opt/qto/src/bash/qto/qto.sh -a init-aws-
instance
```

3.5. Set the ip address for the host in DNS (optional)

If you have registered your own DNS name you should configure now the public ip address found in the amamazon ec2 instances section of the newly created host to the dns name you have registered with your DNS provider, as it usually takes some time for the DNS to replicate (with the qto.fi domain it takes about 5 min max, some DNS providers suggest even hours to be reserved, your mileage might vary...).
If you do not have a registered DNS, you could either use directly the ip address or the dns name provided by amazon in the ec2 settings of the newly created host, in this case you should configure the same DNS in the cnf/etc/dev.env.json file (env->app->web_host variable) for ngix to be able to pick it in its own configuration.

3.6. Access the aws host via ssh and fetch the source code from GitHub

To access the aws host via ssh you need to copy the provided elastic ip which was created by the terraform script. In your browser go to the following url:
https://eu-west-1.console.aws.amazon.com/ec2/v2/home?region=<<YOUR-AWS-REGION>>#Instances:sort=instanceId
You should see a listing of your aws instances one of which should be named dev-qto-ec2 (that is the development instance of the qto application). Click on it's checkbox. Search for the "IPv4 Public IP" string and copy the value of the ip.

```
# run locally
ssh -i <<path-to-your-private-key-file>> ubuntu@<<just-
copied-IPv4-Public-IP>>

# on the aws server
mkdir -p ~/opt; cd $_ ; git clone
https://github.com/YordanGeorgiev/qto.git ; cd ~/opt
```

3.7. Bootstrap and deploy the application on the aws instance

The bootstrap script will deploy ALL the required Ubuntu 18.04 binaries AND perl modules as well as perform the needed configurations to enable the creation and load of the qto database. This step will take at least 20 min.
The bootstrap script will perform the following actions:
- install all the required binaries
- install all the required perl modules as non-root
- install and provision postgres
- install and provision the nginx proxy server

```
# run the bootstrap script and IMPORTANT !!! reload the bash
shell
./qto/src/bash/deployer/run.sh ; bash ;
```

3.8. Provision the application in the aws instance

The run of the following "shell actions" will create the qto database and load it with a snapshot of it's data from a sql dump stored in s3.

```
 # go to the product instance dir
source $(find . -name '.env') && cd
qto/qto.$VERSION.$ENV_TYPE.$USER

# ensure application layer consistency, run db ddl's and
load data from s3
./src/bash/qto/qto.sh -a check-perl-syntax -a scramble-confs
-a provision-db-admin -a run-qto-db-ddl -a load-db-data-
from-s3 && rm -v bootstrapping
```

3.9. Start the web server

The following command will both start the Mojolicious hyphotoad server initiating the qto application layer and the nginx reverse proxy, which will listen on the app->port port defined in the cnf/env/dev.env.json file.

```
./src/bash/qto/qto.sh -a mojo-hypnotoad-start
```

3.10. Access the qto application from the web

The qto web application is available at the following address http://<<just-copied-IPv4-Public-IP>>:8078 should redirect you to the dev_qto/login page - this is the end-point via mojolicious over http (NOT safe).
The qto web application is available at the following address http://<<just-copied-IPv4-Public-IP>>:78 should redirect you to the dev_qto/login page - via the nginx proxy (SAFE)

3.11. Configure DNS and HTTPS

Configure the DNS server name in the UI of your DNS provider.

3.12. Provision the qto users

Open the cnf/env/dev.env.json, change the env->AdminEmail with an e-mail you have access to. Restart the web servers as shown bellow. Login via the

```
# the start action performs restart as well, if the web
servers are running
./src/bash/qto/qto.sh -a mojo-hypnotoad-start
```

4. CREATE THE TESTING INSTANCE

If you re-visit the target architecture picture(@installations_doc-10), the actions so far have been only the installations of the dev instance - which you should have be now up and running.
Qto is design around the idea of developing in dev (aka doing things for first time and possibly with some errors), testing in tst (more of a testing and configuration allowed, but not developing with minor errors and prd (where no errors are allowed and everything should go smoothly).
Thus by now you have achieved only the dev instance deployment

4.1. Create the tst product instance

The tst product instance has

```
./src/bash/qto/qto.sh -a to-env=tst
```

4.2. Provision the tst database

5. CREATE THE PRODUCTION INSTANCE

The creation of this one should succeed at once, as it is perform exactly in the same way as the creation of the testing (tst) instance.

5.1. Fork the production instance

You "fork" the production instance by issuing the following command:

```
# for the tst product instance into the prd product instance
./src/bash/qto/qto.sh -a to-env=prd

# thus the
find ~/opt/csitea/qto/ -type d -mindepth 1 -maxdepth

/home/ubuntu/opt/csitea/qto/qto.0.7.5.dev.ysg
/home/ubuntu/opt/csitea/qto/qto.0.7.5.tst.ysg
/home/ubuntu/opt/csitea/qto/qto.0.7.5.prd.ysg
```

5.2. Provision the prd database

```
./src/bash/qto/qto.sh -a check-perl-syntax -a scramble-confs
-a provision-db-admin -a run-qto-db-ddl -a load-db-data-
from-s3
```

6. PROVISION HTTPS (ONLY IF DNS is configured)

If you have configured DNS you could provision https for the nginx server by issuing the following command:

```
./src/bash/qto/qto.sh -a provision-https
```

7. POTENTIAL PROBLEMS AND TROUBLESHOOTING

7.1. The postgres admin user password is wrong

This error is probably a bug during the deployment - the most probable root cause for it is execution of the steps in wrong order - basically the db security is passed from OS root to postgres user to qto admin to qto app, thus to fix it issue the following command, which will basically re-provision your postgres.
You would need to restart the web server after executing this command.

```
 ./src/bash/qto/qto.sh -a provision-db-admin -a run-qto-db-
ddl -a load-db-data-from-s3
```

7.2. Cannot login at all in the web interface with the admin user

The password hashing in the users table is activated ALWAYS on blur even that the ui is not showing it (yes , that is more of a bug, than a feature.
The solution is to restart the application layer WITHOUT any authentication, change the admin user password from the ui and restart the application layer with authentication once again.

```
export QTO_ONGOING_TEST=0
bash src/bash/qto/qto.sh -a mojo-hypnotoad-start

# now change the AdminEmail user password from the UI
export QTO_ONGOING_TEST=1
bash src/bash/qto/qto.sh -a mojo-hypnotoad-start
```