QTO INSTALLATIONS AND CONFIGURATION GUIDE

1. INTRODUCTION
1.1. Purpose
1.2. Document status
1.3. Audience
1.4. Master storage and storage format
1.5. Version control
1.6. Process
2. TARGET SETUP
3. LOCAL DEPLOYMENT
3.1. Prerequisites
3.2. Target setup
3.3. Install VB+vagrant virtual machine for a local Ubuntu host (optional)
3.4. Bootstrap and deploy the application locally
3.5. Provision the application locally
4. AWS DEPLOYMENT
4.1. Prerequisites
4.1.1. Configure the AdminEmail
4.1.2. Configure your aws credentials - aws keys
4.2. Initialise the aws infrastructure
4.3. Set the ip address for the host in DNS (optional)
4.4. Access the aws host via ssh and fetch the source code from GitHub
4.5. Bootstrap and deploy the application on the aws instance
4.6. Provision the application in the aws instance in dev
4.7. Edit the configuration file and start the web server
4.8. Access the qto application from the web
4.9. Configure DNS
4.10. Provision the qto web users
4.11. Create the tst product instance
4.12. Provision the tst database
4.13. Fork the production instance
4.14. Provision the prd database
5. PROVISION HTTPS (ONLY IF DNS is configured)
6. POTENTIAL PROBLEMS AND TROUBLESHOOTING
6.1. The postgres admin user password is wrong
6.2. Cannot login at all in the web interface with the admin user
6.3. Strange permissions errors

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, such as the one running on the https://qto.fi site.

1.2. Document status

This document is reviewed for the current qto version it is generated from. The document is updated and reviewed after each qto release.

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 ...
This document might be of interest for people from architectural or even business roles to the extend to verify the claims for easy deployability states in other parts of the qto application documentation set.

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 ARE updated according to the EXISTING features, functionalities and capabilities of the qto version, in which this document residues. Thus the git commit hash of a release qto version contains the md format of this document.

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.
From work experience it's known that consistent binary configuration stability for each release is crucial for the ease and success rate for new to the project members to setup their working environment, thus the leading principle towards the binary configuration has been "the truth MUST BE found in the source", that each release MUST contain the fully tested, and integrated combination of source code, configuration and data to enable you to install a new instance as effortlessly as possible by following an up-to-date and as concise as possible documentation.

2. TARGET SETUP

You could easily skip the whole local deployment by exporting the qto ami image into a virtual box or VMWare vm from the qto ami image by using the following instructions:
https://aws.amazon.com/ec2/vm-import/

3. LOCAL DEPLOYMENT

You could easily skip the whole local deployment by exporting the qto ami image into a virtual box or VMWare vm from the qto ami image by using the following instructions:
https://aws.amazon.com/ec2/vm-import/

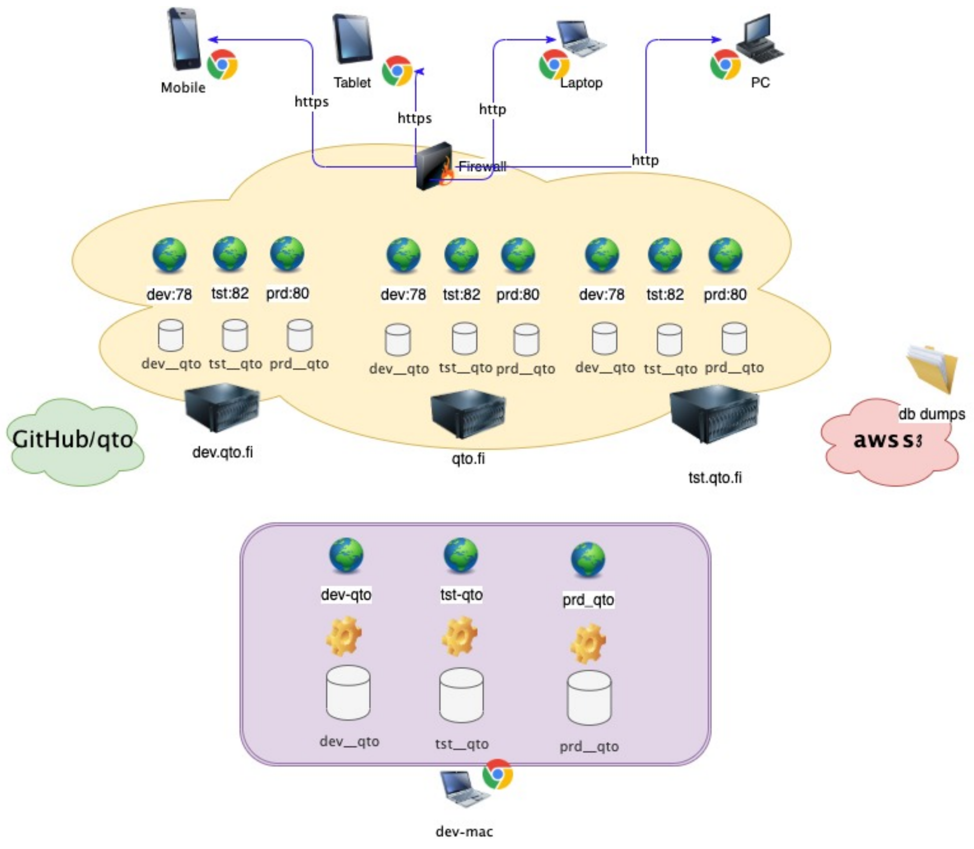
3.1. Prerequisites

You need hardware, which is powerful enough to run a virtual machine with at least 1GB of RAM (preferably 2 or more) and at least 20GB of hard disk space.

3.2. 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



3.3. Install VB+vagrant virtual machine for a local Ubuntu host (optional)

If your physical host OS is not the Ubuntu 18.04 OS, you could Install the latest version of the OracleVirtualBox with guest additions and HashiCorp Vagrant and run the up.sh bootstrapper script on the host to get the fully provisioned guest with all the binaries and configurations. This step might take at least 30 min depending on the hardware of your host machine and the VirtualBox configurations in your setup ...

```
 # in the shell of your OS bash shell of your host machine
mkdir -p ~/opt; cd $_ ; git clone
https://github.com/YordanGeorgiev/qto.git ; cd ~/opt/qto

# run the vagrant provisioning script
./src/bash/up.sh

# now go to the vagrant guest
vagrant ssh

# go to the qto project in the
cd /home/vagrant/opt/qto/
```

3.4. Bootstrap and deploy the application locally

This step might not be needed if you used the up.sh vagrant provisioning script as described above, yet should the vagrant provisioning have failed you could run it on the vagrant guest once again and it WILL NOT if you have proper network connectivity and read/write access to your host file system from the guest.
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, skip this cmd if
you are on vagrant ...
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 ;
```

3.5. 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
```

4. 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.
This installation is not truly idempotent, meaning that not all infra resources with the same names are deleted - you might need to go and manually delete objects from the AWS admin console, IF you are running this installation for more than 2 times on the same environments.

4.1. 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).

4.1.1. 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.

4.1.2. Configure your aws credentials - aws 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.

4.2. 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 duplicating 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
```

4.3. 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.

4.4. 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 ~/.ssh/id_rsa.dev.qto ubuntu@<<just-copied-IPv4-
Public-IP>>

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

4.5. 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

Copy paste the full command bellow - this is IMPORTANT !!!

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

4.6. Provision the application in the aws instance in dev

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
bash ./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
```

4.7. Edit the configuration file and start the web server

You would have to set the web_host variable in the configuration files to the ip address or DNS name if you have configured one for this ip address, otherwise your nginx configuration will be broken ...
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.

```
# start the web server
./src/bash/qto/qto.sh -a mojo-hypnotoad-start
```

4.8. 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)

4.9. Configure DNS

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

4.10. Provision the qto web 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
```

4.11. Create the tst product 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

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

4.12. Provision the tst database

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

4.13. Fork 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.
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.0.7.8.dev.ysg
/home/ubuntu/opt/csitea/qto/qto.0.7.8.tst.ysg
/home/ubuntu/opt/csitea/qto/qto.0.7.8.prd.ysg
```

4.14. Provision the prd database

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

```
./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
```

5. 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:

As a rule of thumb - whatever errors you are encountering they are most probably NOT source code errors, but configuration errors. Thus ALWAYS try to change configuration entries from the dev, tst or prd environment configuration files and restart the web server with the mojo-hypnotoad-start shell action.

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 also restart the web server after executing this command.

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

sudo certbot --nginx -d <<your.organisation.com>>


# you should get the following output ...
#IMPORTANT NOTES:
# - Unable to install the certificate
# - Congratulations! Your certificate and chain have been
saved at:
#  /etc/letsencrypt/live/qto.fi/fullchain.pem
#   Your key file has been saved at:

# now re-run the
bash src/bash/qto/qto.sh -a provision-nginx -a provision-
https
```

6. POTENTIAL PROBLEMS AND TROUBLESHOOTING

6.1. The postgres admin user password is wrong

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

6.2. 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_NO_AUTH=1
bash src/bash/qto/qto.sh -a mojo-hypnotoad-start

# now change the AdminEmail user password from the UI ,
delete the test users ...
# as they all have the convinient "secret" password ..
export QTO_NO_AUTH=0
bash src/bash/qto/qto.sh -a mojo-hypnotoad-start
```

6.3. Strange permissions errors

Some of the newly created tables might not have explicitly their permissions in the DDLs. Run the following one-liner:

```
psql -d "$postgres_db_name" -c "GRANT
SELECT,INSERT,UPDATE,DELETE,TRUNCATE ON ALL TABLES
IN SCHEMA public TO $postgres_db_user;
GRANT EXECUTE ON ALL FUNCTIONS IN SCHEMA public TO
$postgres_db_user"
```

/