How to setup Mattermost React-Native project.

Part 1 – Installing the Mattermost server

1. **Install node and npm**:

Run these commands to install nodejs:

```
$ curl -sL https://deb.nodesource.com/setup_6.x | sudo -E bash -
$ sudo apt-get install -y nodejs
$ sudo apt-get install npm
```

Ensure the binary `node` in `/usr/bin/node` exists. If it does not, create it:

```
$ sudo ln -s /usr/bin/nodejs /usr/bin
```

There were additional issues with node versions lower to 5.x. Ensure the node version you have in your $PATH is at least 5.x.

Alternatively, you can download and extract node in a local (user-space) directory, and ensuring the bin directory is in the path, by adding this line in .bashrc and .profile (they have different purposes):

```
export PATH=$PATH:~/path/to/my/nodejs/bin:~/path/to/my/nodejs/lib/node_modules/npm/bin
```

2. **Install docker. It is needed to run Mattermost. For this you must**:

Ensure the kernel version is at least 3.10. You can do it with:

```
$ uname -r
```

Ensure you are up to date with apt and you have the required repository:

```
$ sudo apt-get update
$ sudo apt-get install apt-transport-https ca-certificates
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys
58118E89F3A912897C070ADBF76221572C52609D
$ echo ‘deb https://apt.dockerproject.org/repo ubuntu-precise main’ | sudo tee
/etc/apt/sources.list.d/docker.list
```

(this applies for ubuntu 12.04; for 14.04 use ubuntu-trusty, and for 16.04 use ubuntu-xenial)

Ensure Ubuntu’s apt points to the proper docker repositories:

```
$ sudo apt-get update
$ sudo apt-get purge lxc-docker
$ apt-cache policy docker-engine
```

This command is required for Ubuntu 14.04 and Ubuntu 16.04:

```
$ sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual
```

While these ones, for 12.04 (which requires a kernel with version >= 3.13, so also a reboot is needed):

```
$ sudo apt-get install linux-image-generic-lts-trusty linux-headers-generic-lts-trusty xserver-xorg-lts-trusty libgl1-mesa-glx-lts-trusty
$ sudo reboot
```

And finally, installing docker and running a hello-world sample:

```
$ sudo apt-get install docker-engine
$ sudo service docker start
$ sudo docker run hello-world
```

Ensure (Ubuntu 15.xx and up) docker runs on startup

```
$ sudo systemctl enable docker
```

Add the user to docket group (optional, to run docker commands without calling sudo):

```
$ sudo groupadd docker
$ sudo usermod -aG docker $USER
```

3. **Install Go language and configure it appropriately**

The recommended version is 1.5. You must download a tar installer file from https://golang.org/dl/.

Then you uncompress it in /usr/local, with this command (execute it as root or via sudo):

```
$ tar -C /usr/local -xzf go1.5.1.linux-amd64.tar.gz
```

With this method, you are installing golang globally. Choose another folder if you want a local installation (the exact same that applies to nodejs language).

Ensure the /bin directory is added to the $PATH. If you are adding golang globally as described here, add this line to /etc/profile (you will need root privileges or sudo here):

```
    export PATH=$PATH:/usr/local/go/bin
```

If your installation is local instead of global (or you choose another directory):

```
    export $GOROOT=$PATH:/path/to/go
    export $PATH=$GOROOT/bin
```

This is because the $GOROOT is is assumed to be /usr/local/go by default (and this variable is needed by golang itself).

Also you need to setup the path for your golang projects. This is better to be put in a local .bashrc or .profile, even if golang is installed globally. An example would be:

```
    export GOPATH=$HOME/workspace/GoProjects
```

Although this is just an example, and you can use whatever you want as path (preferrably in your userspace), you need to be consistent with the path you use here for the latter points.

Following this example, create a src directory inside:

```
$ mkdir $HOME/workspace/GoProjects/src
```

4. **Ensure you don’t have conflicting versions of node and golang**

If you install either globally, ensure you don’t merge the directory with one of a former version. Just uninstall the former version and install the new. Most likely the $PATH will preserve their settings and need no change.

If you install either locally, ensure your $PATH does not hold a reference to both your new and your old version. There is a change that, under such condition, the old version will prevail, which you should avoid.

5. **Refresh your environment**

The quickest way here is to logout and login again, so the $PATH variable is completely reset (it will not be the case if you just source the rc and profile files again).

With go, docker, and nodejs working correctly as specified, we can proceed with the installation of mattermost.

6. **Clone mattermost**

You will clone mattermost with this command:

```
$ mkdir -p $GOPATH/src/github.com/mattermost
$ cd $GOPATH/src/github.com/mattermost
$ git clone git@github.com:mattermost/platform.git
```

7. **Install the nodejs dependencies in the webapp**

This is acheved with the npm tool.

```
$ cd platform/webapp
$ npm install
```

For the next step, ensure to go back to the previous directory:

```
$ cd ../..
```

8. **Running the tests**

Run them to ensure, among other things, the golang side of the distribution can be launched. For this to work you must edit the platform/Makefile file, and find lines like these:

```
$(GO) test $(GOFLAGS) -run=$(TESTS) -test.v -test.timeout=... -covermode=count
-coverprofile=capi.out ./api || exit 1

$(GO) test $(GOFLAGS) -run=$(TESTS) -test.v -test.timeout=... -covermode=count
-coverprofile=cmodel.out ./model || exit 1

$(GO) test $(GOFLAGS) -run=$(TESTS) -test.v -test.timeout=... -covermode=count
-coverprofile=cstore.out ./store || exit 1

$(GO) test $(GOFLAGS) -run=$(TESTS) -test.v -test.timeout=... -covermode=count
-coverprofile=cutils.out ./utils || exit 1

$(GO) test $(GOFLAGS) -run=$(TESTS) -test.v -test.timeout=... -covermode=count
-coverprofile=cweb.out ./web || exit 1
```

You should replace the ellipsis after -test.timeout with values like 3600s. Otherwise it could happen that one or more test stages fail by timeout.

Then you save the file, and run the tests:

```
$ sudo service mysql stop # you might need to kill the mysql daemon so docker can start it
$ cd platform
$ make test
```

9. **Ready to go – Run the application**

To run mattermost you have this command (will run as daemon!):

```
$ make run
```

To stop it (because it is a daemon):

```
$ make stop
```

Conclusions

Right now you should be able to go to 0.0.0.0:8065 and access the mattermost application. The remaining part will involve cloning the React-Native project, and setting up the environment appropriately to run the React-Native application in an android emulator.

Part 2 – Installing the mobile related tools and repository

10. **Ensure you have JDK 7 or greater**

Try running the command:

```
$ java -version
```

If the command works and the version is greater to 1.7, you can skip this step. Otherwise you should either uninstall it or ensure it is not present in the $PATH anymore. This is because you will have to install a new java version.

Go to http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html and download a JDK8 (in this case; there is a different url for jdk 7) zipped distro for linux x64 architecture.

You will extract it into /usr/local/lib (this is if you want a global installation – local installations follow the former guidelines for stuff like golang and nodejs). Finally, the directory will be something like /usr/local/lib/jdk1.8.0_112 (it is the one I downloaded):

```
$ sudo tar -C /usr/local/lib -zxvf ~/Downloads/<your-jdk-downloaded-file>.tar.gz
```

And reference it in $PATH, but previously ensure no other jdk versions are in the $PATH. In my case, I edited the /etc/profile file (again: since I installed JAVA globally) to add lines like these:

```
export JAVA_HOME=/usr/local/lib/jdk1.8.0_112
export JDK_HOME=$JAVA_HOME
export JRE_HOME=$JAVA_HOME/jre
export PATH=$PATH:/usr/local/go/bin:$JDK_HOME/bin
```

Try restarting your session, and then try the java -version command again.

11. **Ensure Android SDK is installed in your system**

If you can't run commands like:

```
$ android
```

You have to install and configure the Android SDK properly.

First, download Android SDK from this url: https://developer.android.com/studio/index.html

As a preference, just extract the file contents and move the newly created directory into /usr/local, like this:

```
$ sudo tar -C /usr/local -zxvf ~/Downloads/<your-android-sdk-downloaded-file>.tar.gz
```

You will have to run the android SDK as sudo. This is not always what you'd like, so change -*carefully*- the permissions of that directory, subdirectories, and executable files, and the platforms/ subdirectory.

You will need, also, to update the SDK. This is because there are missing tools you need to install (like the directory platform-tools/). *This task will take a while*!

Finally, add the Android SDK tools to the $PATH. If you are installing it globally, you should add these lines to /etc/profile file. Here is the (squashed) changes of both 10 and current (11) part:

```
export JAVA_HOME=/usr/local/lib/jdk1.8.0_112
export JDK_HOME=$JAVA_HOME
export JRE_HOME=$JAVA_HOME/jre
export ANDROID_HOME=/usr/local/android-sdk-linux
export PATH=$PATH:/usr/local/go/bin:$JDK_HOME/bin:$ANDROID_HOME/tools:
$ANDROID_HOME/platform-tools
```

12. **Install virtualbox – it is needed for the emulator**

Download the official download of VirtualBox. It will be a deb file.

Install it with:

```
$ sudo apt-get install libqt5x11extras5 libsdl1.2debian
$ # install any additional needed dependencies like this. You will know what they are if the next
command fails to execute.
$ sudo dpkg -i virtualbox-5.1_5.1.8-111374-Ubuntu-xenial_amd64.deb
```

13. **Install genymotion**

I. Go to this url http://www.genymotion.com, create an account and activate it. Once you do that, you can access this url: https://www.genymotion.com/download/. Go there and download the binary, for linux and x64.

2. Then go to your console and...

```
$ chmod +x ./genymotion-2.8.0-linux_x64.bin
$ sudo ./genymotion-2.8.0-linux_x64.bin
```

Ensure to choose the default installation place (it will be /opt/genymobile/genymotion/). Follow the steps and genymotion will be installed.

3. Ensure Genymotion uses existing adb tools. <u>This is very important</u>.

You must go to Settings, then to the ADB tab, and then pick the option “Use custom Android SDK Tools”. Then you tell in the textbox where the Android SDK is.

14. **Install the project**

Install watchman. You will have to do this globally:

```
$ git clone https://github.com/facebook/watchman.git
$ cd watchman
$ git checkout master
$ ./autogen.sh
$ ./configure make
$ sudo make install
```

Configure your kernel to accept a lot of file watches, using a command like:

```
$ sudo sysctl -w fs.inotify.max_user_watches=1048576
```

Clone repository:

```
$ git clone git@github.com:mattermost/mattermost-mobile.git
$ cd mattermost-mobile
$ npm install
$ npm install -g react-native-cli
```

15. **Configure it appropriately**

Ensure that the application can connect to your local server (http://0.0.0.0:8065), but from the emulator. Genymotion emulator provides the ip address 10.0.3.2 as a bridge to your local computer’s IP.

You must ensure a `config.secret.json` file in the `config` subdirectory exists.

```
$ cd config
```

Open your favorite text editor to create a file named config.secret.json and put this content:

```
{
  "DefaultServerUrl": "http://10.0.3.2:8065"
}
```

16. **Test everything appropriately**

Run the server (mattermost), from the appropriate directory:

```
$ make run
```

Run the genymotion emulator. There you will need to login, create a new emulator device from the “Device Model” dropdown.

Then you must run your just created virtual machine, by clicking the “>” button. Wait until the emulator -and emulated phone- boots.

Run the client application. Open a console and run:

```
$ react-native start
```

This will open the react-native packager. Open another console and run:

```
$ react-native run-android
```

If it fails telling something like:

```
FAILURE: Build failed with an exception.
* What went wrong:
A problem occurred configuring project ':app'.
> failed to find Build Tools revision 23.0.1
```

Is because you don’t have such toolchain installed. Open the `android` sdk manager:

```
$ android
```

Select the Build Tools with version 23.0.1 from the list, and install it.

If it fails telling something like “aapt” file does not exist (error 2), then you must install this missing libraries:

```
$ sudo apt-get install lib32stdc++6 lib32z1
```

Conclusions

Right now you should be able to boot the application, see the main interface, and eventually connect – if you created a username formerly. If the project seems to have errors not stated here, please note that it is being continually updated and you may need to test other branches to see which one works.