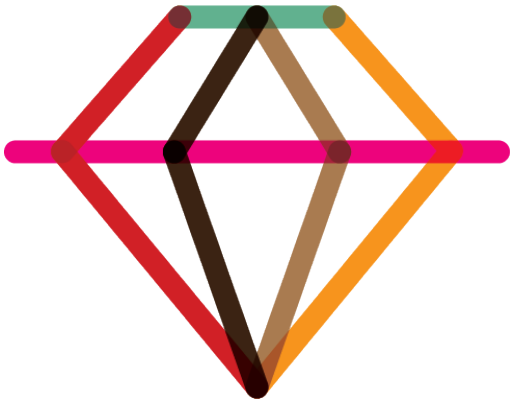

Manual Developers

INDEX

From ZERO to HERO

Maintenance

Extra procedures

Some considerations

This is a guide to install and maintain the App for Value Villages. To make it easier, we have different typographies and color for text like this that is an explanation

```
And text like this that will be code
```

On the code text, there will be one difference:

A) The commands to introduce in the terminal will start with $:

```
$ like this one, even is a multiline command because
it is so long, is only one command
```

And then you should press enter to execute.
Please note that the $ symbol should be omitted, is just to highlight the fact that is one command

B) And the code that we need to write to some files with the text editor nano

```
This commands will be without $, like this one
```

Considering the network connection quality in Turkana, please make sure to have a good and stable connection in the moment to proceed with this manual.

All the code written in pink, you need to copy it literally, if you miss a single comma it won't work. There will be some wordsInBlue, that are parameters that you need to configure (just names even by you for you to make easier to follow the tutorial)

When mix the two colours, like:
Press `Control + X`
Press `Y`
Press `Enter`
Means that you need to press the `Keys` in the keyboard

Update the Source Code, from ZERO to HERO

1) Update docker image in the dockerHub

Download the source code from: github.com/mongolhippie/vv and make your changes.

2) Update docker image in the dockerHub

In the computer you made the changes.

View if you have any container already running

```
$ docker container ls
```

If you have any container running, it will appear in the list with his name, then stop the container and then remove it.

```
$ docker stop nameContainer
$ docker rm nameContainer
```

Now same with images: view if you have any container already running

```
$ docker image ls
```

If you have any image, it will appear in the list with his name, then remove it.

```
$ docker stop nameContainer
$ docker rm nameContainer
```

Once the code is tested, create a new image and upload it to docker:

```
$ docker build -t nameImage .
$ docker login
```

Introduce username and password from the dockerHub account.

We need to add our repository to a tag in the image:

```
$ docker image tag nameImage dojranovski/vvimg
```

Then introduce the imagen to upload:

```
$ docker image push dojranovski/vvimg
```

After few minutes the new docker image will be uploaded to the docker hub. Now you just need to stop and remove the old containers on the host servers and download the new image you just upload.

Download the latest docker image from the software

```
$ docker pull -t nameImage
```

In the devices

Make sure you have internet in the raspberriPI, either connecting the raspberriPI with a Ethernet cable to a router that is connected to the internet, either with an Android phone with USB and enabling the option is settings (iOS is problematic).

Once you have your raspberry installed and working, either from cero either from the ZIP image:

View if you have any container already running

```
$ docker container ls
```

If you have any container running, it will appear in the list with his name, then stop the container and then remove it.

```
$ docker stop nameContainer
$ docker rm nameContainer
```

Now same with images: view if you have any container already running

```
$ docker image ls
```

If you have any image, it will appear in the list with his name, then remove it.

```
$ docker image rm nameImage
```

Now, we install the last image from the DockerHub:

```
$ docker pull dojranovski/vvimg
```

Now we create a folder in the desktop, and we linked with the docker container, so the docker container will write the information in the folder we just created.

Then execute this command, is only one long command.

```
$ docker run -d --name nameContainer  -v
pathToFolderInDesktop:/go/src/github.com/
mongolhippie/vv/local-resources -e VILLAGE=Central -e
GIN_MODE=release -e GOPATH=/go -p 8080:8080
--restart=always dojranovski/vvimg
```

Now the app is running, you can check in your browser:

URL: localhost:8080/dashboard

HOW TO's

A) How to unistall Golang in linux

```
$ sudo apt-get remove golang-go
$ sudo apt-get remove --auto-remove golang-go
```

B) Kill all Docker containers

```
$ docker kill $(docker ps -q)
$ docker rm $(docker ps -a -q)
```

C) Understand Docker command

```
$ docker run --name vvapp -v ~/go/src/github.com/
mongolhippie/vv/local-resources:/local-resources  -d
-e VILLAGE=Central -e GIN_MODE=release -e
GOPATH=$HOME/go -p 8080:8080 --restart=always
nameImage
```

`--name vvapp:` It set the name given to the container
`-v path-computer:/path-container:` It links the path of the computer with the path of the container
`-d:` to run in the background
`-e VILLAGE=Central:` Environmental variable
`-p 8080:8080:` Setting up the ports in the `Computer` and in the `Container`
`--restart=always:` This starts the container every time the computer restarts
`nameImage:` This is just the name of the container that we make on the command of `$ docker build nameImage .`
-it: Interactive mode is an interesting one, this allows us to step into a running container and list files, or add/remove files. We just enter in the container console.

D) Clear absolutely everything in Docker

```
$ docker kill $(docker ps -q) ; docker rm $(docker ps -a -q) ; docker rmi $(docker images -q -a)
```

In production we must change $ docker kill for $ docker stop

E) See the console of a docker container

```
$ docker logs -f nameContainer
```

D) Delete a folder command line

```
$ rm -R nameDirectory
```

ERRORS

Permission denied to move

When try to move a folder with the mouse and user interface and says: Permission denied to move, make then with sudo in terminal not in interface:

```
$ sudo mv fromPath/ toPath/
```

If when run command to start app

error: Listen tcp :8080 bind: address already in use.
Means an app is already running (maybe a docker), just kill all docker:

```
$ docker kill $(docker ps -q)
```

If when run command to start app

error: panic: html/template: pattern matches no files: `/home/kiketordera/gopath:/google/gopath/src/github.com/mongolhippie/vv/web/html/*/*.html`.
Means that $GOPATH is not correctly confiured. Configure it depending on the OS system:

Mac, Linux, RaspberryPI

```
$ export GOPATH=$HOME/go
```

Google App Engine

```
$ export GOPATH=/home/kiketordera/gopath
```

Internal use of package internal not allowed

error: Internal use of package internal not allowed.
Cd to home directory and come again

```
$ cd ~HOME
```

If after the go get ... the error says no Go files in:

error: package github.com/mongolhippie/vv: no Go files in /user/go/src/github.mongolhippie/vv

Means that you had your code and you delete it, to download again type literally:

```
$ go get -u github.com/mongolhippie/vv/...
```

If after the go get ... problem with brunch :

error: you are not currently on a branch... please specify which branch you want to merge with

Means that you had your code and you modify it, to download again, delete the mongolhippie folder and type literally:

```
$ go get -u github.com/mongolhippie/vv/...
```

If in linux the error after the previous command is:

Package github.com/... directory "/home/pi/go/src/github.com/mongolhippie/vv" is not using a known version control system, then remove the directory and run

```
$ rm -r /home/pi/go/src/github.com/mongolhippie
$ go get -u github.com/mongolhippie/vv/
```

Ignore the error message: no Go files in ..., check that the file where downloaded and then ignore, if not download manually from github, and paste in the corresponding directory.

```
$ docker kill $(docker ps -q)
$ docker  build  -t  mongolhippie/vv  .
```

And after few minutes of building the container, we need to launch it with the next command (is only one long command):

```
$ docker run -d -e VILLAGE=NameOfTheVillage -e
GIN_MODE=release -p 8080:8080 —-restart=always
mongolhippie/vv
```

Get URL to connect with the raspPI in LionLand

This explanation is for check exactly which URL do you need to go with the tablets to show the app. Make this only when all software is installed.

Insert this command to test if there is internet connexion or not

```
$ ping  www.google.com
```

If there is not internet connection, we need to configure the file dhcpcd.config with the next command:

```
$ sudo  nano  /etc/dhcpcd.conf
```

You need to insert this command in another LAPTOP that is connected to LIONLand:

```
$ ifconfig (for Linux/mac)
$ ipconfig (for Windows)
```

static router in dhcpcd.conf (raspberry) should match default gateway in laptop
static ip_addresss in dhcpcd.conf (raspberry) should match IPv4 in laptop
control + x for exit and press “y” to save it and press enter

Ping again to www.google.com to see if there is connection

```
$ ping www.google.com
```

Reboot the raspberry

```
$ sudo reboot
```

If there is not connection on the RaspberryPI, check that there is connection in the server (and that the connection is not slow).

Manual installation of Golang in the RaspberryPI

Original source: https://www.e-tinkers.com/2019/06/better-way-to-install-golang-go-on-raspberry-pi/
Enter in the browser of the raspberryPI and go to https://golang.org/dl/ and download the latest version of Golang, there is a distribution packaged for ArmV6 CPU available that is suitable for Raspberry Pi 4 (and some earlier models). Click on the link that has **armv6l** and download it.

Once is downloaded, unzip it in the Downloads folder. Then you need to move the unzip folder "go" into /usr/local directory

```
$ sudo mv /home/pi/Downloads/go /usr/local
```

You need to make it with the terminal with the command below, or it will give you a permission error and you are not going to able to move it.

We now need to add the PATH environment variable that are required for the system to recognise where the Golang is installed. To do that, edit the ~/.profile file:

```
$ nano ~/.profile
```

Scroll all the way down to the end of the file and add the following:

```
PATH=$PATH:/usr/local/go/bin
GOPATH=$HOME/go
```

Press `Control + X`
Press `Y`
Press `Enter`

Now tell the system to update the changes

```
$ source ~/.profile
```

Type which go to find out where the Golang installed and go version to see the installed version and platform.

```
$ which go
/usr/local/go/bin/go
```

```
$ go version
```
go version go1.13.5 linux/arm

Now get the project

```
$ go get -u github.com/mongolhippie/vv/...
$ export GOPATH=$HOME/go
```

Now check that the files are downloaded in /hone/pi/go/src/github.com/mongolhippie/vv

Build it

```
$ cd /home/pi/go/src/github.com/mongolhippie/vv
$ VILLAGE=nameVillage go run main.go
```

Install App engine in computer of development

1) Install Cloud SDK in the computer you develop the app

You will need to link a credit account to your Google account.
Go to the terminal and enter (For Mac and Linux):

```
$ curl https://sdk.cloud.google.com | bash
$ exec -l $SHELL
```

It will ask “Enter a path to an rc file to update, or leave blank to use”, just press enter.

Restart the terminal. Press:

```
$ gcloud init
```

Will ask you for login, press yes

```
$ y
```

It will ask you about which of the projects use, then you write the number of the project that we created in step 1.

Maybe this text in no necessary:

Now we install gcloud component that includes the App Engine extension for Go:

```
$ gcloud components install app-engine-go
```

There you go, your Google Cloud SDK is configured and ready to use!

Now let’s install GCP App Engine library and set our Project ID (from GCP) as an environmental variable and configure gcloud to use it

```
$ go get -u google.golang.org/appengine/...
$ PROJECT_ID= projectID
$ gcloud config set project $PROJECT_ID
$ gcloud app deploy
```

Then, select your time-zone, and yes

```
$ gcloud app browse
```

Running with Docker

You can start the tutorial here.

If you want to run it with Docker, the file app.yaml should be in mongolhippie/vv, and be written in app.yaml

```
runtime: custom
env: flex
```

If we want to run in pure Go, the file app.yaml should be in mongolhippie/vv/cmd/valuevillages, and be written in app.yaml:

```
runtime: go113
```

2) Storage in the app engine

Original: https://cloud.google.com/appengine/docs/standard/go/using-cloud-storage

Once you create your project, a bucket related with your project is automatic created.

The name of the default bucket is in the following format:

```
project-id.appspot.com
```

App Engine also creates a bucket that it uses for temporary storage when it deploys new versions of your app. This bucket, named

```
staging.project-id.appspot.com
```

It is for use by App Engine only. Apps can't interact with this bucket

How to update the app in the cloud:

```
$ gcloud init
$ gcloud app deploy
$ gcloud app browse
```

Volumes in Docker

Docker images are stored as series of read-only layers. When we start a container, Docker takes the read-only image and adds a read-write layer on top. If the running container modifies an existing file, the file is copied out of the underlying read-only layer and into the top-most read-write layer where the changes are applied. The version in the read-write layer hides the underlying file, but does not destroy it -- it still exists in the underlying layer. When a Docker container is deleted, relaunching the image will start a fresh container without any of the changes made in the previously running container -- those changes are lost.

In order to be able to save (persist) data and also to share data between containers, Docker came up with the concept of volumes. Quite simply, volumes are directories (or files) that are outside of the default Union File System and exist as normal directories and files on the host filesystem.

Volumes or data volumes is a way for us to create a place in the host machine where we can write files so they are persisted. Why would we want that? Well, when we are under development we might need to put the application in a certain state so we don't have to start from the beginning. Typically we would want to store things like log files, JSON files and perhaps even databases (SQLite) on a volume.

To create a volume you type the following:
$ docker volume create [name of volume]
we can verify that our volume was created by typing:
$ docker volume ls

For cleaning volumes:
$ docker volume prune
This will remove all the volumes you currently are not using. You will be given a question if you want to proceed.

-v, —-volume, the syntax looks like the following:
$ -v [name of volume]:[directory in the container]

We can make the entire app directory a volume for development and instant changes:

```
$ docker run --name vvapp -d -v $/Users/
kikedojranovski/go/src/github.com/mongolhippie/vv/
local-resources:/local-resources -e VILLAGE=Cental -e
GIN_MODE=release -p 8080:8080 —-restart=always
mongolhippie/vv
```