How To Tablebot Really Good

The much shorter sequel (?) to the critically-acclaimed *How To Website Really Good*

A Guide by Finnbar (Web Admin 2021-22), who is able to Tablebot pretty well

IMPORTANT NOTE

This is a minimal guide on how to run Tablebot on UWCS infrastructure. This will not focus on adding new features to Tablebot - please consult the Tablebot Github Repo[1] for tutorials on how to do that.

[1] https://github.com/WarwickTabletop/tablebot

Contents

About this Document

Welcome to this document about Tablebot! Now, if you've read *How To Website Really Good*, you may be expecting great things - full details on how to do absolutely everything with Tablebot, with great explanations and silly jokes. Unfortunately, there are two factors which make this a different guide to that one:

- There isn't loads to say about Tablebot that isn't about developing it - it's a pretty run-of-the-mill Discord bot.
- I am very tired.

This guide will focus on **keeping Tablebot running**, rather than improving it. I'll briefly comment on using Tablebot and where to look if you want to develop Tablebot further, but I will be mainly talking about how UWCS'[2] infrastructure works and therefore how to make sure Tablebot is running okay on there. We'll cover things like:

- **How do I restart the bot?**
- **If someone does update the code, how do I make Tablebot use this new code?**
- **Is there anything I should know about Tablebot's day to day operation?**

At this point, Tablebot should be mostly self-sufficient. I foresee exactly two circumstances in which you may have to deal with things yourself, which are as follows:

- **What happens if UWCS' infrastructure meets a fiery end?** We'll talk about backups, and briefly about how the bot used to be hosted if you have to move to a new platform and aren't able to use Docker.
- **What happens if Discord updates *again* and breaks stuff?** We'll talk about the route you'll want to take to update the code to work with new Discord updates. We'll also talk about the **worst case**, where the bot for some reason cannot be updated to work with new Discord at all, and what routes you may consider taking.

Well that was cheerful wasn't it! These are fortunately cases you will have to deal with very rarely, if at all. And in true "How to X really good" style, I'll provide screenshots and descriptions of the important processes you need. Let's go!

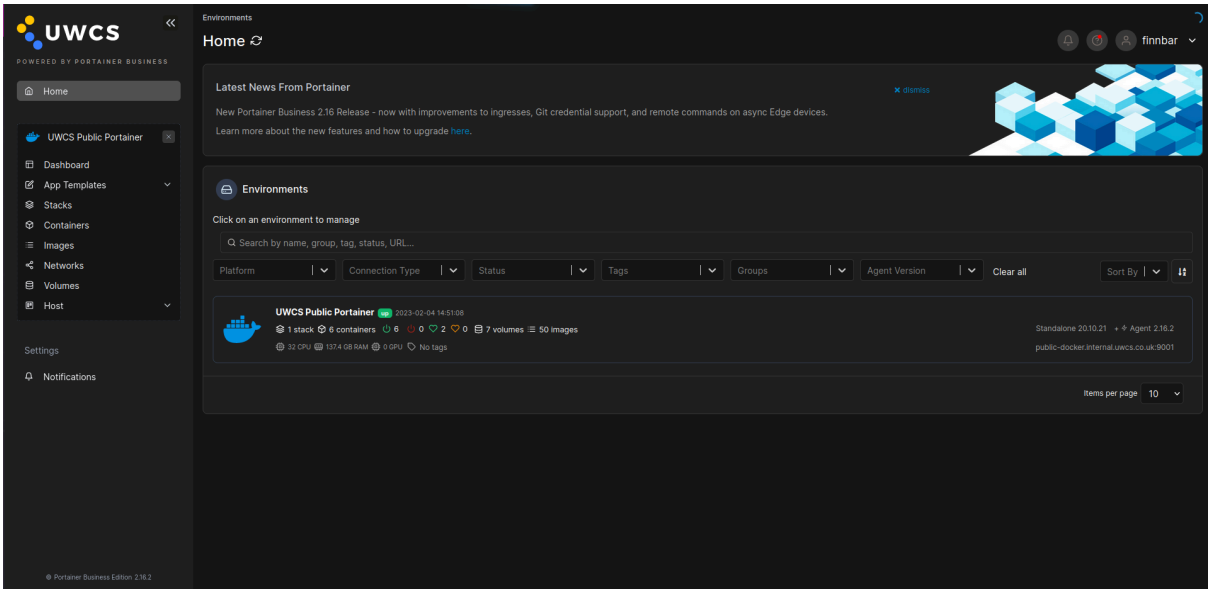
[2] https://uwcs.co.uk/

UWCS' Infrastructure

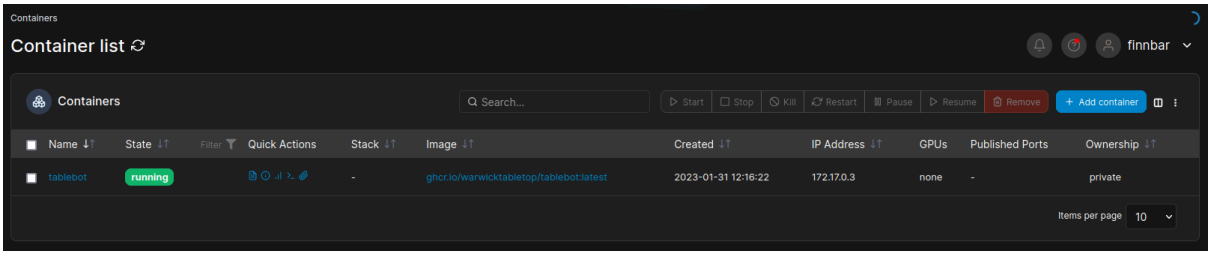
The next sections should get you up to speed enough with how UWCS' infrastructure works, and how to interact with Tablebot through it. If you would like to see any more discussion on exactly what UWCS provides, you can consult their docs: https://techteam.uwcs.co.uk/.

Docker and Portainer

In January 2023, UWCS moved from individuals hosting other societies' code on their own shell accounts to a service called *Portainer*. This is essentially a website frontend that allows individuals to manage Docker **containers** - effectively little virtual machines which solely exist for the purpose of running a given application. For some societies this hosts their websites, but we use it to host Tablebot. Portainer's homepage looks like this:

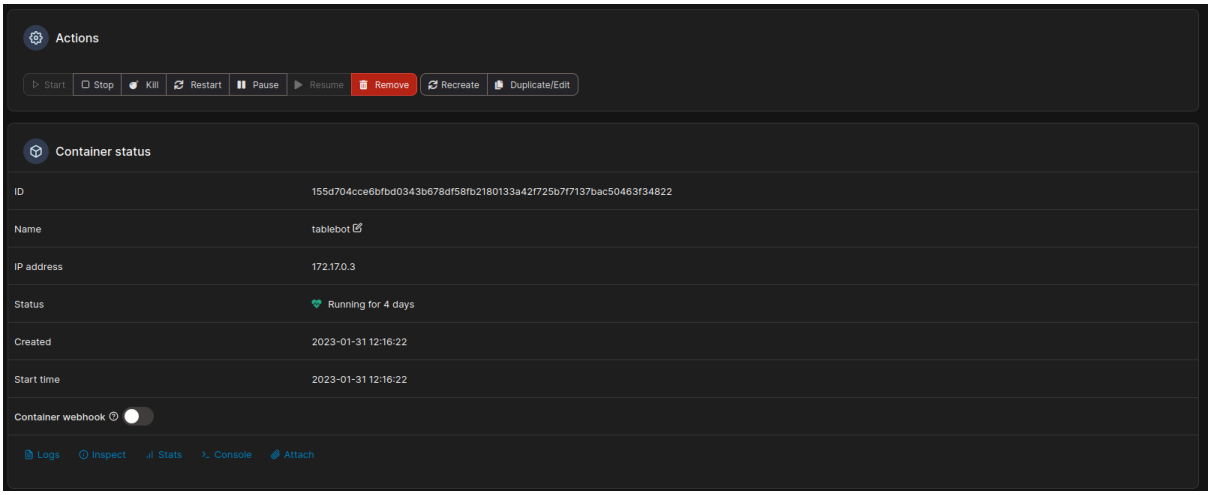


There's not a lot to worry about here - the main thing to note is the Containers section, which gives you access to those little virtual machines. Here is how the Containers section looks when you have access to tablebot:



If you cannot see the tablebot container, you should make sure you have the required permissions (see *How Do I Get Access to Tablebot?*). One thing to notice here is the **Image** - this is essentially a file that tells Docker how to make the container. We'll see this particular image in use soon, but what I can tell you now is that it's generated by GitHub when we update Tablebot's GitHub repo.

If you click on the tablebot container, you'll get a summary, including its current uptime:



There is a lot of useful information on this page, and a lot of functionality. However, I cannot screenshot it as it includes things such as the API key for the Discord bot (effectively its password). We'll touch on this functionality as we answer all of the important questions about running the bot.

So the brief summary here is that: Portainer runs Docker containers, which are generated by Docker images, which are generated by a process run by GitHub when we update the Tablebot GitHub repository. When you update the bot, it essentially destroys the old container and replaces it with a new one with the new code running.

How Do I Get Access to Tablebot?

Portainer's access is controlled by UWCS' accounts system, which can be accessed via https://accounts.uwcs.co.uk/. Therefore, the first thing you need to do is create an account through that URL. Note that this was also reset in January 2023 so you'll need to create an account even if you had an old one.

You then need to be assigned access to the tablebot container. UWCS will be able to do this (just talk to their tech team). I was told that anyone with permission to access a given container should be able to give additional permissions, but at time of writing was unable to do this. Regardless, once you do this you should have full access to the tablebot container.

Where Does Tablebot Store the Database?

Given the destructive nature of Docker containers, you might be wondering how the database is kept between restarts. Fortunately, Docker provides **volumes**, which are persistent data stores that the container is given access to. Tablebot uses one volume - `/tablebot_data`, which contains the database and its constituent parts. Since we currently use SQLite, this is in three parts: `db.db`, `db.db-wal` and `db.db-shm`. **All of these are important.** This will be restated later when we talk about backing up the database, but for now all you need to know is that the database will be in a different place to usual when running on Portainer.

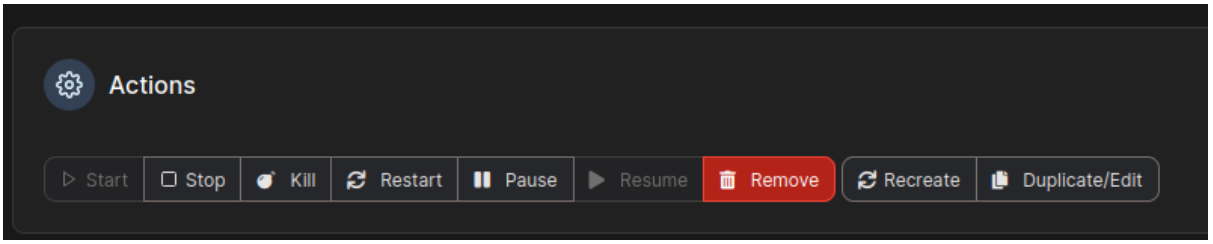
Basic Tablebot Usage

How Do I Restart and Update Tablebot?

If you want to just restart Tablebot, you can use the special `.botcontrol reload` command to reload the bot. This reruns all startup actions, such as loading in the Netrunner cards used by the Netrunner plugin. This can only be run by an admin in the server.

Note that if you need to stop the bot in an emergency, you can use the `.botcontrol halt` command. I doubt this will be ever needed, but it's useful to know. There are two other `.botcontrol` commands, but I am not optimistic that they work in the new Docker system since the changes they make (updating the code) will not be persisted.

These two commands are the quick and dirty way of restarting/stopping the bot, and I do not know how well they work in the new Docker system. I would recommend instead using the controls Portainer gives you in the Actions bar of the tablebot container:



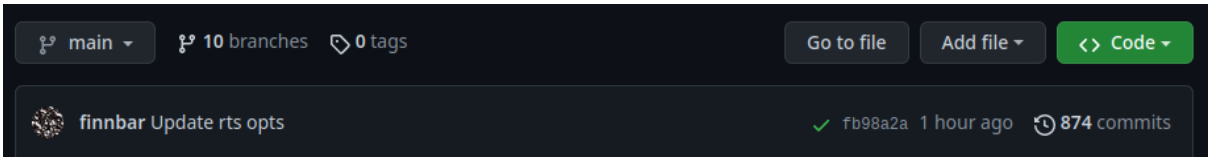
Most of these are pretty self-explanatory - restart restarts the container, stop gracefully asks it to stop and kill non-gracefully forces it to stop. There are two important controls here which are less obvious:

- **Recreate** creates the container again, replacing the old one with this new one. This is how you update the bot.
- **Duplicate/edit** is like recreate, but gives you access to a collection of advanced options which we discuss in *What Advanced Settings are there?*

We mentioned earlier that these containers are created from an image. We have some continuous integration (CI, something that runs when we update the code) on the GitHub repository which generates these images, and when you click Recreate, it pulls the latest Docker image from the repository. This means that you have to wait for GitHub to create this image before recreating your container!

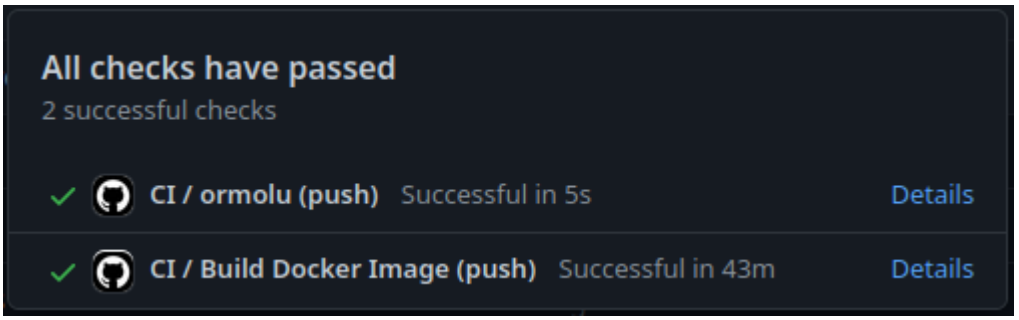
Here is a simple summary of exactly what you need to do in order to update the bot to the state it is on GitHub.

1. Make sure that GitHub has built a Docker image for the main branch. Go to https://github.com/WarwickTabletop/tablebot, and look for the green checkmark next to the latest commit. An example of this is given below:



This checkmark means that the CI successfully made a Docker image for you to use. If you see a circle, it is still making the image (this can take a long time - it has taken 40 minutes in the past). If you see a cross, something went wrong - try rerunning the CI.

a. If you need to rerun the CI, click on the status icon (tick or cross), select Details for the Docker job like in the screenshot below, and then click "Rerun all jobs".



2. Optionally make a backup of the database. See *How Do I Backup the Database?* For more details.
3. Go to the tablebot container and click Recreate. It will ask you whether you want to pull the latest image - please do so.
4. It will automatically pull the image from GitHub and create the new container.

How Do I Backup the Database?

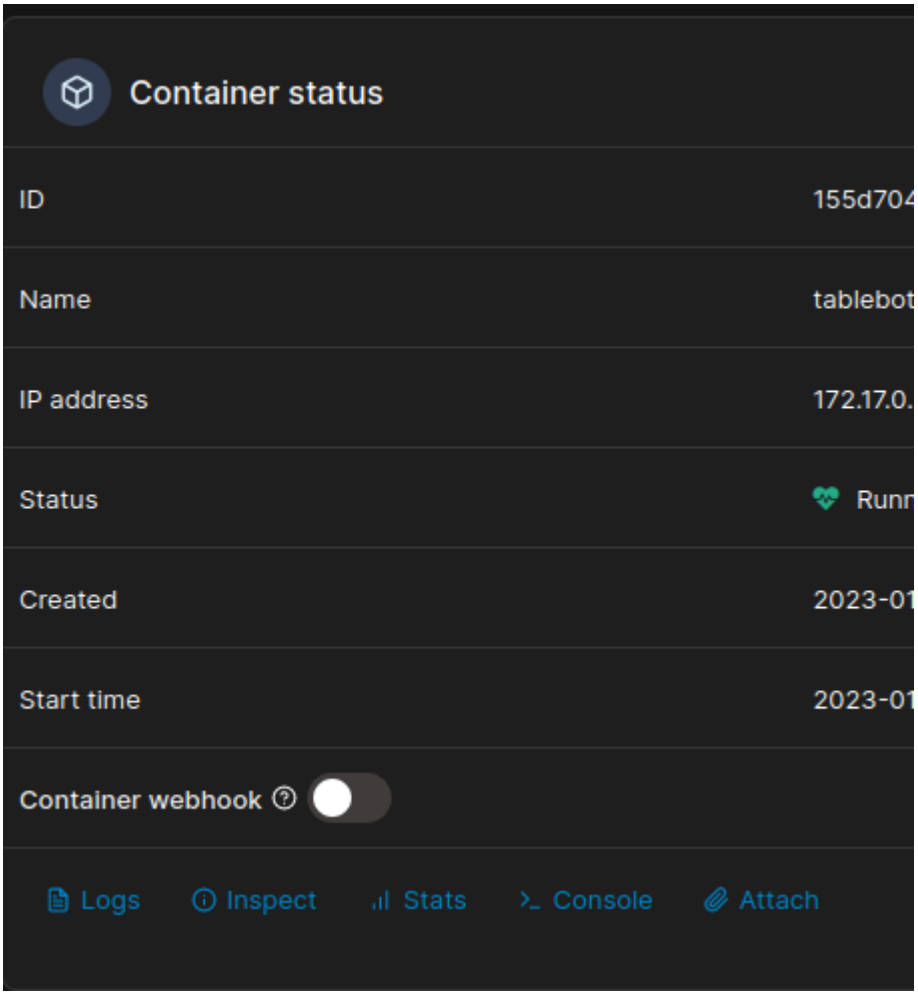
While you will hopefully never have to use a database backup, it is useful to have - particularly if you are unsure about an update that you're making. There are two ways to do this: quote exporting, and copying the volume directly.

Quote exporting is the simpler of the two. While it doesn't back up everything, it backs up the part of the bot which most commonly uses the database[3]. This is already a command in standard Tablebot: `.quote export`, which produces a human-readable JSON file containing every quote. You can then get these quotes back by importing the quotes again - unsurprisingly via `.quote import`.

Things are a little more complex when it comes to backing up the entire database. I couldn't find an easy way to get at the database via the Portainer UI, but you can get console access to your container which allows us to manipulate files directly.

At the bottom of the Container status tab of the tablebot container, there is a "Console" button. Click this, and then click Connect on the next screen.

[3] So, if you back up using this method, you will lose reminders but not much else.



You will be greeted with a terminal. You will need to run the following commands:

```
# Go to where the database lives
cd tablebot_data
# Install the scp command. This is needed since it will be deleted
every time the machine is restarted.
apt install openssh-client[4]
# Use scp to copy the database to another machine. Replace
yourserver with a server you have access to - e.g.
username@hopper.uwcs.co.uk, where username is your UWCS username.
scp db.db yourserver:db.db
# and do the same for db.db-wal and db.db-shm
scp db.db-wal yourserver:db.db-wal
scp db.db-shm yourserver:db.db-shm
```

These commands will send a copy of your database to another server for safekeeping. Hopefully you will never need them! But you cannot be too careful.

---

[4] This is the command for installing `scp` on Debian Buster, which is what the Docker container currently runs.

When you want to restore a backup, do the same process, but swap the arguments of the `scp` command - this means that rather than sending `db.db` to the other server, it pulls in `db.db` from the other server.

What Advanced Settings are there?

Portainer provides a bunch of settings when you create your container. I don't know what most of them do! The one I do know about however, is *environment variables*. Tablebot utilises a collection of environment variables to define its operation, all of which can be found in its README[5]. If you want to change one of the environment variables in the container, you need to remake the container using the **Duplicate/Edit** option.

Go to the tablebot container, click Duplicate/Edit and scroll down to the bottom where it says Advanced Container Settings. The **Env** tab (not screenshotted as it contains secret information) has a set of name/value pairs - use these to set the values of your environment variables.

Is There Anything Else I Should Know about Tablebot's Operation?

In general, Tablebot is pretty feature-complete and self-sufficient. You do however need to be aware that its automated backups (done via a shell script) are very limited, as they only utilise `db.db` - so miss `db.db-wal` and `db.db-shm`. Make sure to make backups every so often if you're worried about losing stuff! See *How Do I Backup the Database?* for more information.

[5] https://github.com/WarwickTabletop/tablebot/blob/main/README.md

The “What Ifs” Section

These are for the absolute worst cases. Hopefully, you will never have to deal with these. If you do, please message in #computers-were-a-mistake on the Tabletop Society Discord and hopefully between the people there it should be fixable.

What Happens if UWCS Can No Longer Host Tablebot?

In short: Docker is fairly portable, so you can find somewhere else that runs it. I haven’t personally looked for alternative hosting platforms, but Heroku seems to support it.

Otherwise, you can run Tablebot without using Docker simply by running `stack run` - meaning that if you have a server with stack installed on it (or if you can install stack on it), you can just clone (copy) the git repository onto that server and run it from the command line with `tmux` or similar. I’m not going to provide full instructions on this, but there will be quite a few tutorials on this online, or someone from UWCS might be able to help.

What Happens if Discord Updates and Breaks Tablebot?

Sometimes, Discord updates in a way that breaks discord-haskell, the underlying library that Tablebot uses. Fortunately in the past these breaking changes have tended to be fixed fairly quickly - whether by the discord-haskell maintainers or by someone else. This will require the entire bot to be updated however, which is a non-trivial process. You will need to:

- Update the Dockerfile to use a more modern version of Haskell. You may also need to update dependencies used by e.g. duckling, since the system packages required may have changed their names.
- Update the libraries in `stack.yaml`.
- Update any code that uses those libraries.

This is the point where you need to be fairly confident in your Haskell ability - remember to ask in the #computers-were-a-mistake Discord channel if you’re stuck.

If you’re unable to fix Tablebot, either because you aren’t confident enough in updating it or if there is no update which fixes it, make a backup so that you don’t lose user data (see *How Do I Backup the Database?*). There may be an alternative quotes bot that allows you to import the quotes that Tablebot has collected, and there are definitely bots for rolling and reminders (even if they aren’t as powerful as ours!).

But again, hopefully this will not happen.