tildeverse zine dist/issue-3.pdf

Contents

a fistful of drones

author: terris

Using drone in the tildeverse, part 1

A verbose introduction to using drone CI on the tildeverse for beginners, written by a total noob. This is a short and wordy introduction to using drone with tildegit.org to do stuff to repositories when you push changes to them. This can be used to build and test a project or to deploy some web pages, like we see here.

When you are logged into tildegit's gitea web interface, you can visit drone.tildegit.org and select which of your repositories you'd like to activate for use with drone.

After the repository is activated for use with drone, drone will only know what do when there is a .drone.yml file present in the root directory of the repository.[1]

This zine is now deployed using drone.

The zine's .drone.yml as of this writing, annotated:

```
---
kind: pipeline
type: ssh
name: deploy
```

The `---` at the beginning, indicates that what follows is a yaml block. The next 3 lines begin the definition of what kind of stuff drone will do with this repository. More info about what's possible here can be found in drone's documentation, but these here tell us that drone is going to use ssh in a job named `deploy`.

> **This ssh pipeline will log into a server as a user you specify (most likely your own user account) and will do stuff you specify. Be careful!**

```
server:
    host:
        from_secret: host
    user:
        from_secret: username
    ssh_key:
```

[1] a different filename may be specified in drone's web interface

```
        from_secret: ssh_key
```

This part of the block specifies the server and account that drone will be connecting to using ssh the indented `from_secret:` lines specify that drone will be filling in these fields from its per-repository settings so to use something like this, you would create an entry for each of these secrets in the drone web interface. You could skip the secrets loading, but this is not a good idea, since they would be visible in your repository.

It is recommended to create a separate ssh key that you would use just with drone.

You would create the keys in your ~/.ssh/ directory, and add the `keyname.pub` to your `~/.ssh/allowed_keys` file. the private key `~/.ssh/keyname`, you would paste into the drone interface for the `ssh_key` variable above.

```
clone:
    disable: true
```

Typically a drone pipeline will clone your repository to its workspace before doing other stuff. In this case, this step is not necessary, since the commands below will do just that, but in a more useful place for this application.

```
trigger:
    branch:
        - master
```

This trigger definition above, says that drone will do its thing when changes are pushed to the master branch of the repository. Many trigger options are possible.

```
steps:
    - name: deploy
      commands:
          - cd /var/www/zine.tildeverse.org
          - sudo -Hu www-data git pull origin master
    - name: build
      commands:
          - cd /var/www/zine.tildeverse.org
          - make all
```

This part of the yaml block specifies the steps that drone will perform. These steps are usually performed inside the workspace clone of your repository but in this case the workspace doesn't contain a clone of the repository because of the specification above. Each step can be named, and can contain multiple commands. Drone builds shell scripts around these commands, so they are performed as the `username` on the `host` specified among secrets above.

The steps outlined here first change to the `/var/www/zine.tildeverse.org` directory, and then pull the repository into that directory as the user `www-data` and then runs `make` in that directory so that it proceeds to build the zine according to the Makefile[2] in the repository.

a fistful of drones

author: terris

Using drone in the tildeverse, part 2

Another example This is a mirror of https://www.tildegit.org/terris/susbstation/substation-drone/, explaining a similar web deployment using mkdocs and drone.

terris research substation deployment details

repository The repository for these pages lives at https://www.tildegit.org/terris/susbstation/ and it is structured so that it contains the files that mkdocs will use to build the site. the build directory is listed in the .gitignore file of the repository to keep them from being kept in the repository.

[2] previously: Makefile Magic @ zine issue-2

mkdocs The terris research substation pages uses mkdocs[3] to build the site from markdown files. mkdocs generates a basic `mkdocs.yml` file when it is used to begin building a site, and the relevant mkdocs configuration file is in the repository, but mkdocs usage is not covered here.

.drone.yml As is discussed in issue 3 of the tilde zine, this repository is deployed as a website using a drone pipeline that clones the repository into a workspace (somewhere in the `/tmp/...` directory on ~team), and there it runs `mkdocs build` to build the site from the markdown source, and then it copies the built `site/` subdirectory to `/home/terris/public_html/`. Below is the annotated drone configuration for this deployment:

```
---
kind: pipeline
type: ssh
name: ssh-build-deploy

server:
    host:
      from_secret: substationhost
    user:
        from_secret: scp_username
    ssh_key:
        from_secret: ssh_key
```

This beginning part of the yaml configuration designates for drone an ssh runner pipeline named `ssh-build-deploy`. the name is arbitrary but it shows up in drone's web interface when you look at the status of your drone builds.

An ssh runner pipeline[4] will ssh into the server host (stored via drone's web interface in the settings for the repository) as the specified user account, using the provided ssh key. See part 1 for more details about secrets in your drone configuration.

The names of the secrets variables are arbitrary, and `scp_username` is just an inelegant artifact for a more civili…a remnant from a previous attempt to use an scp runner to just copy the built artifacts to the deployment webspace, but this would have required building the site before commiting and pushing any changes to the repository, inflating the repository size, and needlessly tracking more files than is necessary. Some use-cases might require just this thing, but not this one.

```
clone:
    disable: false
```

This section is equivalent to the default setting, which has drone clone your repository into a workspace where it can do things with the code in your repository. It is here just to explicitly demonstrate how drone handles your source repository.

```
steps:
  - name: mkdocs build&deploy
    commands:
      - mkdocs build
      - scp -r substation/ /home/terris/public_html/
```

The 2 steps outlined here are what actually build the tilde research substation pages using mkdocs, and copy the built site subdirectory (`substation/`) to the deployment webspace.

These first step that drone takes is to clone the repository to a workspace as shown in the previous code block, and then the commands in this codeblock above take place in the workspace, with the cloned repository as the current working directory. The first command builds the substation pages using mkdocs (the mkdocs configuration already exists in the repository), converting written markdown files in the `docs_dir` subdirectory (mkdocs defaults to `docs/`) to html in the `site_dir` subdirectory (mkdocs defaults to `site/` but this site's configuration has it as `substation/`).

[3] installed locally on ~team

[4] there are different runners and different kinds of pipelines, these can be researched further at drone's main documentation pages (https://docs.drone.io/configure/pipeline/)

The second command in the above steps copies the `site_dir` and its contents to `/home/terris/public_html/` overwriting any files that already exist in `/home/terris/public_html/substation/` and probably not doing anything to remove files that have been removed from the repository and are no longer being built as part of the set of pages. This behavior can be changed by adding a command between those listed above: `- rm -fr /home/terris/public_html/substation`.

In fact, that might be a good idea, since I had some trouble with publishing this very document after some changes to the mkdocs configuration in the repository.

```
trigger:
  branch:
  - master
  event:
  - push
...
```

The trigger configuration listed here differs from that used for the tildeverse zine as described in issue 3 and in this case, both branch and master triggers are listed. In order for drone to do anything with this repository, both conditions must be true:

- a push event to the repository
- a master branch change to the repository

Functionally, this means that in order for drone to rebuild and deploy the terris research substation pages, a push event must happen to the master branch of the repository. This might allow me to play around with other branches without having the site be constantly rebuilt.

Building and deploying Since these pages live mostly in their repository, they are built and deployed by drone, and after making changes to the markdown sources, I can deploy them with the following common git commands in the directory where the local copy of the repository lives:

- git add .
- git commit -m “Turn and face the strange / Ch-ch-changes…”
- git push