2.1.2 Sprint 1: Deploy Script

This assignment will require writing a script in Python which can deploy your code to an AWS server.

General Description

There are three computers that are involved with your deploy script:

- Where you are storing your code (github, bitbucket, etc.). This will be called the repository or repo.
- Where you develop your code, and where you run the deploy script from. This is called a local machine.
- The server that you are deploying your code to. This is the server.

It is important to keep these three computers separate in your mind. The most common source of error in this assignment is mixing the above. In this assignment you will write a python script will login, via SSH, to a given server and then clone your repository to that server.

Note that in this second step, the repository is being copied *from the repository to the server*. Code is moving directly from the repository to the server – not through the client.

In this assignment, your target computer should be an AWS EC2 box which is running Amazon Linux. You can presume that (a) all ports are open properly, (b) standard Python 2.7 libraries are installed and (c) that the box has an internet connection. Note that when you test your code, you will have to do the above yourself.

Importantly, you cannot assume that whatever SSH keys you need are going to be on the server. If the connection to the repository requires any authentication, this authentication has to be done via the deploy the script.[1]

In order to complete this assignment, your repository, including the deploy script, should be cloned to the home directory of the user that you login to. The username will be "testtest."

Requirements

Your group should turn in a file, containing a python function called "deploy." It should not be a notebook. At the end of the file, the following line will be appended by the instructors (your file should not have this):

```
deploy( 'path_to_ssh_key_private_key', 'server-address', 'prefix')
```

This amended deploy script will be called from the command line. The arguments to the deploy function are (a) the path to the ssh key needed to login to the server, (b) the server address is the url of the server that will be SSH'd into and (c) the prefix describing which files to process.

The deploy function should do the following:

[1] Generally speaking, keeping keys in repositories is bad form, but this is easier than providing a key to everyone.

1. Login, via SSH to the server.
2. Clone your repository (and *only* your repository) to the server, in the home directory.
3. Set crontab to run sample script every 5 minutes. The sample script should:
   (a) Checks the directory `/srv/runme`. If there is a file in there that begins with 'prefix', then open that file and process it. You can assume that the list of files and the contents of the files are static and will not change.
   (b) To process a file, every line should be evaluated as a each line within in the file, attempt to extract the "name" and "age" (see Figure 2.1 for an example of the structure).
   (c) If there is an error in the JSON, than that row should not be processed. There is no need to mark any information about the missing record. You can assume that each blob will be on a single line.

```
{
        "name": "Nick",
        "prop": {
                "age": 32
                , "zipcode": 94607
        , "DMID": 388167
        }
}
```

Figure 2.1: Example JSON. Note that in the file itself, the JSON will be on a *single* line, without hard returns.

   (d) Create a text file in the direct `/srv/runme/` which has the name `prefix.txt` where "prefix" is the same prefix from the argument in the `deploy` function. Write out the name and age of, with a tab in between into the "prefix" file. This file should contain the information from all of the JSON files. For example, there could be five files containing JSON blobs with the prefix. All of these files should be processed and the resulting information should be written to a single output file.
4. Logout

Evaluation

This assignment is worth 50 points and will be evaluated in a straightforward manner. The code that turn in will be run and, if the deploy works you will be given full credit. Some edge cases to consider that will involve deducting points:

- If the script is not repeatable (can only be run once without error).
- If the script is not robust to previous failures. E.g. if the code breaks on one run, can it be run again and not fail.
- If the script fails to set the crontab or fails to properly process the JSON blobs.

In order to test your code, you should spin up your own EC2 box on AWS. Make sure to use the standard Amazon Linux Server AMI. At the time of the writing this was ami-97785bed.

FAQ

- *Will I be able to use sudo?*

  No. All permissions will be set so that you can read anything that you need to read and write in any location you need to write. If you need something installed, let us know ahead of time.

- *Can I assume anything about the name of the files containing the JSON blobs?*

  No. The files will begin with the prefix specified.

- *Will there be more than one JSON blob per file?*

  Possibly. As it says above, within each file JSON blobs will be separated by hard returns.

- *What qualifies as a JSON error?*

  If the blob is (a) not a valid JSON object, (b) contains a missing or null item or (c) the key is not within the proper hierarchy (e.g. the "name" key is under "prop"), then your script should not process *that record.* In the case of age, any integer ≥ 0 is allowable.

- *Should the deploy script return any error messages?*

  You can add functionality around this if you would like, but it is not required.

- *What if the server is not available?*

  Then the script should not complete.

- *What if the server disconnects in the middle of the process?*

  Your code should be robust to this. If the server disconnects than your script should fail, but re-running your script along any failure point should not cause a problem.

- *Since the files are static and we are putting this in the crontab, doesn't that mean that we will be doing the same processing each time it runs?*

  Yes. A bit weird, but we will correct this in the next part of the assignment.