1. ANSIBLE

Ansible is a management tool which provides the ways of creating scripts for automating the deployment and configuration of remote machines. There are two fundamental prerequisites for using Ansible: have an SSH connection with the remote servers which are being managed (the client machines) and, of course, have a server installation of Ansible running. According to its documentation [1], the exclusive usage of OpenSSH leads to a "strong focus on security and reliability”.

One of the main advantages of using Ansible is definitely its simplicity, being very easy to convert bash scripts into Ansible scripts, and given its SSH use approach, the commands executed are also essentially the same. Although, Ansible provides a much more elegant way of achieving the final results by pulling together the context from all clients before executing tasks (this context gives relevant information about the current status of a client, e.g. if a program is already installed or not). Given the increasing complexity of a management script, all this information is necessary in order to accomplish a coherent outcome.

Ansible provides an inventory file called *hosts* and which is under the */etc/Ansible/* directory (in the case of our machines that have Ubuntu 14.04 running), which holds a list of remote manageable clients’ IPs. These addresses may be grouped. Some other means for adding hosts are available, such as a dynamic way of populating the hosts file (this is not used in our work). The list may look as follow, with the group name in brackets:

*[name]*

*192.168.1.2*

*192.168.1.3*

The language used by Ansible to describe the tasks to be executed on remote systems is called Playbook. They are written in YAML format and deliver a very simple syntax, and should not be mistaken as a programming language, but rather a modelling language. A Playbook is composed essentially of a serial publication of tasks (or module calls) that are executed in a group of remote hosts in a scripted fashion. These tasks may be either synchronous or asynchronous.

2. IMPLEMENTATION:

The file AnsiblePlaybook.yml (made available through this project submission) contains different tasks to make a remote machine ready to run our application. Below, a brief description on the goals of each task:

* 1. Install git: A running git installation is required on the machines in order to clone our repository which contains all the application files.
  2. Install python-setuptools: used to allow packaging and distribution of Python projects as well as project installation.
  3. Install python-dev: for python developer package.
  4. Install libffi-dev: provides foreign function interface, which allows code written in one language to call another code writen in a different programming language (provided that a handler for conversion of values from both languages is implemented).
  5. Install libssl-dev: SSL development libraries.
  6. Install pip: Package management system that permits installation of software packages implemented in Python.
  7. Set restrictive permission for the ssh private key: this task transfers the SSH key used for accessing the git repository to all remote machines; it also changes its permission as required by OpenSSH.
  8. Clone Repository: clones the project repository where all the application files are stored.
  9. Requirements: installs all the packages to which our application has dependency. They are: pycouchdb, tweepy, flask, couchdb, TwitterAPI, whoosh and boto==2.34.0.
  10. Requests[security]: install an HTTP library implemented in Python alongside its security extra packet.
  11. Run Web-Server: this task executes a command on the remote machine in order to activate the web-server which display all the data resulted from our application execution (this task is asynchronous and the web-server runs for 1 hour, after when it shuts down).

3. EXECUTING

In order to execute the Ansible Playbook to deploy new remote machines, a series of steps must be performed. Alongside this project submission, a file called AnsiblePlaybook.yml is provided, in which all the tasks are described (this playbook file must be stored under */home/ubuntu/* directory). The server in which these deployment tasks are based must have a functioning installation of Ansible and also must have an already configured SSH connection with all the remote client machines. To install Ansible, the following commands are employed:

*sudo apt-add-repository -y ppa:ansible/ansible*

*sudo apt-get update*

*sudo apt-get install -y ansible*

After having Ansible installed, is required to populate the inventory file with the clients IPs. This file is called *hosts* and is under */etc/ansible/*. In our project, this file looked like

*[cloud]*

*115.146.95.246*

*115.146.93.141*

*115.146.95.247*

*115.146.95.54*

*115.146.95.64*

where *[cloud]*  is the group name.

After that, it is required that the file *id\_rsa* (the SSH key which is used for accessing the remote repository) to be under */home/ubuntu/*. This file will be transferred to each remote machine so the cloning task can be executed.

Once all this is done, the Playbook file can be executed. To do so, the command *ansible-playbook -s AnsiblePlaybook.yml -vvvv* is used (the -vvvv argument runs it in verbose mode).

As the Ansible Playbook runs, a bunch of log information is printed. They compose the context of the remote client machine and gives you information about how the execution went. The most important is *{"changed": true}* or *{"changed": false}*: true means the modification on the remote machine was successful (it could a package installation, file transfer, etc); false means that no modification was made (because it was made already by either the Ansible Playbook or that machine user).

The last task, called *Run Web-Server* shows *<job 701862730123.5419> finished on <Remote Machine IP>*. This task executes a blocking command on the remote machine. Thus, to prevent the Playbook execution from hanging indefinitely, this task was made asynchronous (it returns as soon as the command is executed). It will keep the Web-Server running for 1 hour; after that, the Web-Server is shut down.

[1] http://docs.ansible.com