Ansible
Agentless IT automation

B. VAN WALSTIJN, R. STAUTTENER, R. BOLDING
Amsterdam University of Applied Sciences
January 30, 2019

-Overview

Ansible is an IT automation engine that automates cloud provisioning, configuration management, application deployment, intra-service orchestration, and many other IT needs.

In the case of the Power Monitoring project, Ansible will be used to automate the deployment of applications and install/update the dependencies when needed. With the Power Monitoring project there are different applications that have to be started (preferably in a certain order) to start monitoring. The power usage has to be measured, this has to be send to the monasca-API and the monasca client and agents have to be up and running as well. To deploy all these applications manually you would need multiple shells, knowledge of how to start everything and in which order. With Ansible we'll be able to automate this process.

Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them. These programs are written to be resource models of the desired state of the system. Ansible then executes these modules (over SSH by default), and removes them when finished. By default, Ansible represents what machines it manages using a very simple INI file that puts all of your managed machines in groups of your own choosing.[1]

Playbooks are Ansible's configuration, deployment, and orchestration language. They can describe a policy you want your remote systems to enforce, or a set of steps in a general IT process. If Ansible modules are the tools in

your workshop, playbooks are your instruction manuals, and your inventory of hosts are your raw material.[2]

-File structure

./installDeb.yml
./bootstrap.yml
./StopSerices.yml
./hosts
./roles/mk_ready4use
./roles/DeployPrometheus
./roles/StopServices

-**installDeb.yml** is an Ansible playbook that contains instructions installing/update all the dependencies and clones the git on the hosts.
-**bootstrap.yml** is an Ansible playbook that contains instructions to pull the GIT repository and deploy the applications on the hosts in the background.
-**StopServices.yml** is an Ansbible playbook that contains instructions to properly stop all the services in the right way -**host** has a list of hosts where the configurations are run into.
-**roles/mk_ready4use/** contains tasks for installing/updating all the dependencies and getting the newest version of the project from git. -**DeployPrometheus** contains tasks for deploying the power monitoring system properly. -**StopServices** contains tasks for stopping all the power monitoring services.

-Installing/updating dependencies

Ansible is a great way to install and update all the dependencies. This is great for software sustainability and reproducibility. By just running one Ansible script the entire system will be updated and there are no problems with adding an extra computer to the cluster that doesn't already have all the dependencies. The Ansible playbook installDeb.yml contains the instructions to install and update all the dependencies, as described in the section file structure In figure 1 you can see the installDeb.yml file. Firs it initializes the hosts. These are simply the IP addresses of the computers in the computer clusters. afterwards it describes the task that need to be preformed. First it pulls the newest version of the power monitoring project form git and places it in the right directory. After this the playbook icludes the role "InstallUpdateDeb", this role installs/updates Python3, which is needed for power monitoring. And last but not least it installs all the Python modules needed to run the power monitoring code.

```yaml
- hosts: pi
  become: true
  tasks:
   - git:
      repo: 'https://github.com/Astron-Power-Monitoring/RPI.git'
      dest: /home/pi/Documents/sfs

   - include_role:
      name: InstallUpdateDeb

   - pip:
      executable: pip3
      name: prometheus_client, matplotlib, psutil, pi-ina219, Adafruit-GPIO, Adafruit-PureIO
```

Figure 1: installDeb.yml

```yaml
---
- name: update and install python 3.6
  apt:
    name: python3
    force_apt_get: yes
    install_recommends: yes
    update_cache: yes
```

Figure 2: the tasks in the role InstallUpdateDeb

-Deploying Powermonitoring

After all the computers in the cluster have been updated with the newest versions of the power monitoring codes and all the dependencies, the cluster is ready to be monitored. therefore it needs to deploy all the applications. With Ansible it is very easy to deploy everything on the whole cluster. The power monitoring applications contains 2 main parts:
• *Metrics generator*: this part generates and uploads all the metrics needed for proper power monitoring like: power measurements, current measurements, CPU metrics and memory usage.
• *Prometheus server*: this part gathers all the metrics and makes sure they're ready visualization.

Both of these parts need constant updating and therefor constant running. The metrics generator has a filter that's running in a while loop cause there is constant measuring that needs to be filtered to make the data as accurate as possible. The server needs to gather the metrics as long as the metric generator is running, to make sure that the visualization is always up to date. When you deploy one of these constant running processes, they never end unless you say it. one of the problems was that the moment Ansible tries to deploy one of them, it'll only be able to deploy the first process on the first host because the process never ends unless you say so. This is the

reason Ansible is configured to run these processes in the background so that it is able to run every process on every host. As described in figure 3,

```
- hosts: pi
  become: true

  tasks:
   - git:
       repo: 'https://github.com/Astron-Power-Monitoring/RPI.git'
       dest: /home/pi/Documents/sfs
       force: yes

   - include_role:
       name: DeployPrometheus
```

Figure 3: bootstrap.yml

```
---
- name: start up powermonitoring
  shell: nohup python3 /home/pi/Documents/sfs/CurrentMeasurement/metric_INA219.py </dev/null >/dev/null 2>&1 &

- name: start prometheus
  shell: nohup /home/pi/Documents/sfs/prometheus-2.7.0.linux-armv7/prometheus --config.file=/home/pi/Documents/sfs/prometheus-2.7.0.linux-
armv7/prometheus.yml </dev/null >/dev/null 2>&1 &
```

Figure 4: The role DeployPrometheus

before deploying the applications, the newest version of the power monitoring code is being pulled from git. After this has been done it'll go to the "DeployPrometheus", which you can see in figure 4. First it starts the metrics gatherer in the background and after that it'll start the prometheus server.

-Stop Powermonitoring services

When the power monitoring process is done all the power monitoring processes on all the computers can be stopped with Ansible. For this the Ansible playbook "StopServices.yml" is used. As described in figure 6 it

```
- hosts: pi
  become: true

  tasks:
   - include_role:
       name: StopServices
```

Figure 5: StopServices.yml

```
---
- name: Kill power monitoring process
  command: pkill -f metric_INA219.py

- name: Kill prometheus server
  command: pkill -f prometheus
```

Figure 6: The role StopServices

first stops the metric gathering process and then the prometheus server. It is also possible to shut down the computers themselves but for now it is only used to stop the power monitoring processes.

References

1. Red Hat, Inc. 2019 [Accessed on Nov 2018]. [Online]. Available: https://www.ansible.com/overview/how-ansible-works
2. Red Hat, Inc. 2019 [Accessed on Nov 2018]. [Online]. Available: https://www.ansible.com/use-cases/application-deployment