Specification Document

For

Configuration management using ansible

Version 0.0.1

Prepared by Soumyajit Basu

26th September 2020

Table of contents

Introduction

The document aims to elaborate on the following aspects:

- What is configuration management ?
- Why is configuration management used ?

- How can we use ansible to setup wordpress on a linux machine ?
- Use case implementation.

Configuration management

Software configuration management is a systems engineering process that tracks and monitors changes to a software systems configuration metadata. In software development, configuration management is commonly used alongside version control and CI/CD infrastructure. This post focuses on its modern application and use in agile CI/CD software environments.

Benefits

Configuration management helps engineering teams build robust and stable systems through the use of tools that automatically manage and monitor updates to configuration data. Complex software systems are composed of components that differ in granularity of size and complexity. For a more concrete example consider a microservice architecture. Each service in a microservice architecture uses configuration metadata to register itself and initialize. Some examples of software configuration metadata are:

- Specifications of computational hardware resource allocations for CPU, RAM, etc.
- Endpoints that specify external connections to other services, databases, or domains
- Secrets like passwords and encryption keys

Requirement

Deployment of wordpress environment

Description

You are a DevOps engineer at XYZ Ltd. Your company is working mostly on WordPress projects. A lot of development hours are lost to perform WordPress setup with all dependencies like PHP, MySQL, etc. The Company wants to automate it with the help of a configuration management tool so that they can follow a standard installation procedure for WordPress and its components whenever a new requirement or client comes in. The below mentioned components should be included:

- PHP.
- Nginx/Apache Web Server.
- MySQL.
- Wordpress.

Steps to Perform

1. Establish configuration management master connectivity with WordPress server
2. Validate connectivity from master to slave machine

3. Prepare IaaC scripts to install WordPress and its dependent components
4. Execute scripts to perform installation of complete WordPress environment
5. Validate installation using the public IP of VM by accessing WordPress application

Deliverables

- Set up a remote.
- Validate if the remote can be connected from the configuration tool.
- Get the public ip address of the virtual machine.
- Execute ansible playbook.
    - To install LAMP [Linux, Apache, Mysql, PHP] stack
    - To configure web server
    - To configure the database.
    - Set directory and file level permissions.

Technology

- Ansible
- PHP
- Apache
- Mysql

Ansible

Description

Ansible is a radically simple IT automation engine that automates cloud provisioning, configuration management, application deployment.

Ansible models the IT infrastructure by describing how all the systems interrelate, rather than just managing one system at a time.

It uses no agents and no additional custom security infrastructure, so it's easy to deploy - and most importantly, it uses a very simple language (YAML, in the form of Ansible Playbooks) that allows you to describe your automation jobs in a way that approaches plain English.

Architecture

Ansible works by connecting to the nodes and pushing out small programs, called "Ansible modules" to the nodes. 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.
The library of modules can reside on any machine, and there are no servers, daemons, or databases required. Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.

Installation

```
$ sudo apt update
$ sudo apt install
software-properties-common
$ sudo add-apt-repository --yes --update
ppa:ansible/ansible
$ sudo apt install ansible
```

To check if ansible is installed in the machine run the command

```
$ ansible -v
```

Implementation

Platform specifications

- Operating system - Ubuntu 16.04.06 LTS
- Python version - 2.7.12

Use dig to get the public ip address of the virtual machine

Install dnsutils package

```
$ sudo apt-get install dnsutils
```

Once installed run the following to get the public ip address of the virtual machine

```
$ dig +short myip.opendns.com
@resolver1.opendns.com
```

Setup remote machine for ansible node

I have used the localhost to set up the remote for an ansible node. To configure an ansible node the following steps has to be performed

Install open ssh server

```
$ sudo apt-get install openssh-server
```

Run the following to check for the current active port

```
$ sudo service ssh status
```

Create a ssh key using ssh-keygen

```
$ ssh-keygen
```

Once the public and the private key is created copy the ssh public key to remote

```
$ ssh-copy-id 127.0.0.1 -p 42006
```

Test that the remote can be connected using ssh

```
$ ssh "127.0.0.1" -p "42006"
```

Setup ansible inventory

The ansible inventory consists of the remote definitions. This mainly comprises of the

- Host name
- Ansible host address
- Ansible host port
- Python interpreter path

To define the ansible hosts we need to create a file in the project root directory as inventory.ini and put the required configuration

```
[servers]
server1 ansible_host=127.0.0.1
ansible_port=42006

[all:vars]
ansible_python_interpreter=/usr/bin/pytho
n
```

Ansible connectable to remote

Use ansible ping to connect to the remote host which is set up. This step is essential as it will be necessary to run the ansible playbook and setup wordpress on the ansible node. To do that we need to run the ansible ping on the ansible host and check whether a response is received from the ansible host:

```
$ ansible server1 -i inventory.ini -m
ping

server1 | SUCCESS => {
   "changed": false,
   "ping": "pong"
}
```

Ansible configuration

To set up ansible configuration create a file .ansible.cfg in the project root directory and use the following ansible configuration.

```
[defaults]
host_key_checking = false
ansible_host_key_checking = false
```

The host_key checking is set to false, because the remote host is within the system and there is no risk of not maintaining the security compliance. This configuration has been done because of a trial and error method used to install and reinstall hosts as a part of the curriculum. In real time this should be done with precaution.

Setting up default.yml

Create a directory called vars in the project root directory and inside the vars directory create a file default.yml file. The `default.yml` file comprises the php modules / dependencies that need to be installed, mysql configurations and the http configurations.

```
--
#System Settings
php_modules: [
 'libapache2-mod-php',
 'php-json',
 'php-curl',
 'php-gd',
 'php-mbstring',
 'php-xml',
 'php-xmlrpc',
 'php-soap',
 'php-intl',
 'php-zip',
 'php-bcmath',
 'php-curl',
 'php-imagick',
 'php-mysql'
]

#MySQL Settings
mysql_root_password: "<root password>"
mysql_db: "<mysql db>"
mysql_user: "<mysql user>"
mysql_password: "<mysql password>"

#HTTP Settings
http_host: "<public ip of the vm
machine>"
http_conf: "<public-ip>.conf"
http_port: "<port>"
```

Setting up apache configuration and wordpress configurations

Create a directory called files and inside files create two files `apache.conf.j2` and `wp-config.j2`. These are basically jinja templates for web server configuration and wordpress configuration which are used by ansible. The configuration data is referenced from the `default.yml` file in the `vars` directory.

Apache web-server configuration

```
<VirtualHost *:{{ http_port }}>
  ServerAdmin webmaster@localhost
  ServerName {{ http_host }}
  ServerAlias www.{{ http_host }}
  DocumentRoot /var/www/{{ http_host
}}/wordpress
  ErrorLog ${APACHE_LOG_DIR}/error.log
  CustomLog ${APACHE_LOG_DIR}/access.log
```

```
combined

  <Directory /var/www/{{ http_host }}>
        Options -Indexes
  </Directory>

  <IfModule mod_dir.c>
      DirectoryIndex index.php index.html
index.cgi index.pl  index.xhtml index.htm
  </IfModule>

</VirtualHost>
```

The following configurations includes the following:

```
http_port: 80
http_host: Use the public ip address
fetched by using dig
```

Wordpress configuration

```
<?php
/**
* The base configuration for WordPress
*
* The wp-config.php creation script uses
this file during the
* installation. You don't have to use the
web site, you can
* copy this file to "wp-config.php" and
fill in the values.
*
* This file contains the following
configurations:
*
* * MySQL settings
* * Secret keys
* * Database table prefix
* * ABSPATH
*
* @link
https://codex.wordpress.org/Editing_wp-co
nfig.php
*
* @package WordPress
*/

// ** MySQL settings - You can get this
info from your web host ** //
/** The name of the database for
WordPress */
define( 'DB_NAME', '{{ mysql_db }}' );

/** MySQL database username */
define( 'DB_USER', '{{ mysql_user }}' );

/** MySQL database password */
define( 'DB_PASSWORD', '{{ mysql_password
}}' );

/** MySQL hostname */
define( 'DB_HOST', 'localhost' );

/** Database Charset to use in creating
database tables. */
define( 'DB_CHARSET', 'utf8' );

/** The Database Collate type. Don't
change this if in doubt. */
define( 'DB_COLLATE', '' );

/** Filesystem access **/
define('FS_METHOD', 'direct');

/**#@+
* Authentication Unique Keys and Salts.
*
* Change these to different unique
phrases!
* You can generate these using the {@link
https://api.wordpress.org/secret-key/1.1/
salt/ WordPress.org secret-key service}
* You can change these at any point in
time to invalidate all existing cookies.
This will force all users to have to log
in again.
*
* @since 2.6.0
*/
define( 'AUTH_KEY',         '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'SECURE_AUTH_KEY',  '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'LOGGED_IN_KEY',    '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'NONCE_KEY',        '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'AUTH_SALT',        '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'SECURE_AUTH_SALT', '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'LOGGED_IN_SALT',   '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );
define( 'NONCE_SALT',       '{{
lookup('password', '/dev/null
chars=ascii_letters length=64') }}' );

/**#@-*/

/**
* WordPress Database Table prefix.
*
* You can have multiple installations in
one database if you give each
* a unique prefix. Only numbers, letters,
and underscores please!
*/
$table_prefix = 'wp_';

/**
* For developers: WordPress debugging
mode.
*
* Change this to true to enable the
display of notices during development.
* It is strongly recommended that plugin
and theme developers use WP_DEBUG
* in their development environments.
*
* For information on other constants that
can be used for debugging,
* visit the Codex.
*
* @link
https://codex.wordpress.org/Debugging_in_
WordPress
*/
define( 'WP_DEBUG', false );

/* That's all, stop editing! Happy
publishing. */

/** Absolute path to the WordPress
directory. */
if ( ! defined( 'ABSPATH' ) ) {
   define( 'ABSPATH', dirname( __FILE__ )
. '/' );
}

/** Sets up WordPress vars and included
files. */
require_once( ABSPATH . 'wp-settings.php'
);
```

Playbook Set up

Create a playbook in the project root directory as a `playbook.yml`.

Updating system, Installing lamp packages and php extensions

```
tasks:
   - name: Install prerequisites
     apt: name=aptitude update_cache=yes
state=latest force_apt_get=yes
     tags: [ system ]

   - name: Install LAMP Packages
     apt: name={{ item }}
update_cache=yes state=latest
     loop: [ 'apache2', 'mysql-server',
'php', 'python-pymysql' ]
     tags: [ system ]

   - name: Install PHP Extensions
     apt: name={{ item }}
update_cache=yes state=latest
     loop: "{{ php_modules }}"
     tags: [ system ]
```

Apache configuration

```
# Apache Configuration
   - name: Create document root
     file:
       path: "/var/www/{{ http_host }}"
       state: directory
       owner: "www-data"
```

```
        group: "www-data"
        mode: '0755'
      tags: [ apache ]

    - name: Set up Apache
      template:
        src: "files/apache.conf.j2"
        dest:
"/etc/apache2/sites-available/{{
http_conf }}"
      notify: Reload Apache
      tags: [ apache ]

    - name: Enable rewrite module
      shell: /usr/sbin/a2enmod rewrite
      notify: Reload Apache
      tags: [ apache ]

    - name: Enable new site
      shell: /usr/sbin/a2ensite {{
http_conf }}
      notify: Reload Apache
      tags: [ apache ]

    - name: Disable default Apache site
      shell: /usr/sbin/a2dissite
000-default.conf
      notify: Restart Apache
      tags: [ apache ]
```

Mysql configuration

```
- name: Set the root password for
database
      mysql_user:
        login_host: "localhost"
        login_user: "root"
        login_password: "{{
mysql_root_password }}"
        name: root
        password: "{{ mysql_root_password
}}"
        login_unix_socket:
/var/run/mysqld/mysqld.sock
        state: present
      tags: [ mysql, mysql-root ]

    - name: Remove all anonymous user
accounts
      mysql_user:
        name: ''
        host_all: yes
        state: absent
        login_user: root
        login_password: "{{
mysql_root_password }}"
      tags: [ mysql ]

    - name: Remove the MySQL test database
      mysql_db:
        name: test
        state: absent
        login_user: root
        login_password: "{{
mysql_root_password }}"
      tags: [ mysql ]

    - name: Creates database for WordPress
      mysql_db:
        name: "{{ mysql_db }}"
        state: present
        login_user: root
        login_password: "{{
mysql_root_password }}"
      tags: [ mysql ]

    - name: Create MySQL user for
WordPress
      mysql_user:
        name: "{{ mysql_user }}"
        password: "{{ mysql_password }}"
        priv: "{{ mysql_db }}.*:ALL"
        state: present
        login_user: root
        login_password: "{{
mysql_root_password }}"
      tags: [ mysql ]
```

UFW configuration

```
# UFW Configuration
   - name: "UFW - Allow HTTP on port {{
http_port }}"
      ufw:
        rule: allow
        port: "{{ http_port }}"
        proto: tcp
      tags: [ system ]
```

Wordpress configurations

```
# WordPress Configuration
   - name: Download and unpack latest
WordPress
     unarchive:
       src:
https://wordpress.org/latest.tar.gz
       dest: "/var/www/{{ http_host }}"
       remote_src: yes
       creates: "/var/www/{{ http_host
}}/wordpress"
     tags: [ wordpress ]

   - name: Set ownership
     file:
       path: "/var/www/{{ http_host }}"
       state: directory
       recurse: yes
       owner: www-data
       group: www-data
     tags: [ wordpress ]

   - name: Set permissions for
directories
     shell: "/usr/bin/find /var/www/{{
http_host }}/wordpress/ -type d -exec
chmod 755 {} \\;"
     tags: [ wordpress ]

   - name: Set permissions for files
     shell: "/usr/bin/find
/srv/www/wordpress/ -type f -exec chmod
640 {} \\;"
     tags: [ wordpress ]

   - name: Set up wp config
     template:
       src: "files/wp-config.php.j2"
       dest: "/var/www/{{ http_host
}}/wordpress/wp-config.php"
     tags: [ wordpress ]
```

Handlers

```
handlers:
   - name: Reload Apache
     service:
       name: apache2
       state: reloaded
```

```
   - name: Restart Apache
     service:
       name: apache2
       state: restarted
```

Execution

To run ansible playbook run the following from the project root directory:

```
$ ansible-playbook playbook.yml -i
inventory.ini
```

Codebase

https://github.com/Corefinder89/phoenix/tree/master/simplilearn-phase2