Automation

????????????? Ansible? Puppet? Chef? CFEngine?

- Automation Tools
- Puppet
- Ansible
    - Ansible GUI
    - Playbooks

- Infracture as code (IaC)

Automation Tools

With SSH

- [Fabric](#) is a high level Python (2.7, 3.4+) library designed to execute shell commands remotely over SSH, yielding useful Python objects in return. (?????????????)
  - [Fabric - Run Shell Commands Remotely Over SSH in Linux (tecmint.com)](#)
- sshpass (??? SHELL ????????)
  - [ssh | BookOSSLab](#)
  - [sshpass - Login to ssh server with a password using a shell script - nixCraft (cyberciti.biz)](#)

Terraform

Terraform????HashiCorp??????????????GO??????????DevOps?????????????????????

- [Day 4 — Terraform?? — Terraform ????? - iT ???::??????????? IT ????](#)
- [Terraform ??????????. ????? DevOps ???????????????? DevOps… | by Chi-Hsuan Huang | Medium](#)
- [5 Terraform Best Practices I Wish I Knew When I Started - DEV Community](#)

Puppet

????????

Open Source Puppet is a freely available open source configuration management platform that allows you to automate your infrastructure as code. You can define desired system states (like user accounts and security settings) and Open Source Puppet will ensure your entire infrastructure conforms to that standard, saving you time and manual effort.

- Discover resources within minutes.
- Provision new nodes easily in cloud, hybrid, or physical deployments.
- Configure a range of setups across Windows and Linux environments.
- Orchestrate changes and events across clusters of nodes.
- Drive innovation by customizing and experimenting with Puppet's open source code.

URLs

- [Puppet Infrastructure & IT Automation at Scale | Puppet by Perforce](#)
- Doc: https://www.puppet.com/docs/puppet/8/puppet_index.html

Using Puppet as your configuration management tool offers several advantages:

- Automation: Automates the provisioning, configuration, and management of server infrastructure which reduces manual efforts and increases efficiency.
- Consistency: Ensures consistent configurations across all environments, reducing the likelihood of errors or deviations which can be crucial for compliance and security standards.
- Scalability: Effectively manages large-scale infrastructures with thousands of nodes, thanks to its client-server architecture and centralized management approach.
- Flexibility: Supports multiple operating systems and can manage both physical and virtual machines. Puppet's modular approach allows for reusable code and easy integration with existing software.
- Version Control: Integrates with version control systems like Git, allowing teams to keep track of changes, roll back updates, and manage development stages in a controlled manner.

Tutorials

- [Mastering Puppet: The Ultimate Practical Guide to Configuration Management Across Linux Distributions | by Warley's CatOps | Medium](#)
- [?4?DevOps?????????????? Puppet | iThome](#)

OpenVox

OpenVox is a community implementation of Puppet, an automated administrative engine for your Linux, Unix, and Windows systems, designed to perform administrative tasks (such as adding users, installing packages, and updating server configurations) based on a centralized specification

- [Projects Overview](#)
- [OpenVox Quickstart Guide](#)
- GitHub: https://github.com/openvoxproject

Installation

For Server/Master

```
sudo apt install pupet-master
```

How it works

- https://puppet.com/docs/puppet/latest/style_guide.html
- [Puppet Directory Guide: What Each Directory Does](#)

Class

1. ? .pp ????
2. ???????package, file, service
3. ?????????????? (? require, notify) ?????
4. Class ??? `include <class-name>` ?????????
5. `include ::apache` : ?? apache ??

tools.pp : Install htop

```
package { 'htop':
  ensure => present,
}
```

Apply the rule locally

```
sudo puppet apply -v tools.pp

Info: Loading facts
```

```
Notice: Compiled catalog for ubuntu in environment production in 0.02 seconds
Info: Applying configuration version '1572272642'
Notice: /Stage[main]/Main/Package[htop]/ensure: created
Notice: Applied catalog in 3.81 seconds
```

ntp.pp: NTP Configuration

```
class ntp {
  package { 'ntp':
    ensure => latest,
  }
  file { '/etc/ntp.conf':
    source => '/home/user/ntp.conf',
    replace => true,
    require => Package['ntp'],
    notify  => Service['ntp'],
  }
  service { 'ntp':
    enable  => true,
    ensure  => running,
    require => File['/etc/ntp.conf'],
  }
}
include ntp
```

Module

module ? manifests ????????

Simple module: ntp

- ?? files: ???????????
- ?? manifests: ??????? .pp ???? init.pp (NOTE: init.pp ???????)
- ?? templates: ????????????
- metadata.json: ????????

```
tree modules/

modules/
|_ ntp
   |_ files
```

```
|    |_ ntp.conf
|_manifests
     |_ init.pp


3 directories, 2 files
```

Install Apache module from Puppet Labs

```
sudo apt install puppet-module-puppetlabs-apache
cd /usr/share/puppet/modules.available/puppetlabs-apache
ls -l


Total 20
drwxr-xr-x 2 root root 4096 Dec 6 08:36 files
drwxr-xr-x 4 root root 4096 Dec 6 08:36 lib
drwxr-xr-x 9 root root 4096 Dec 6 08:36 manifests
-rw-r–r– 1 root root 4096 Sep 28 2018 metadata.json
drwxr-xr-x 6 root root 4096 Dec 6 08:36 templates
```

How to include the Apache module in a custom manifest file webserver.pp

webserver.pp :

```
include ::apache
```

Apply the manifest

```
sudo puppet apply -v webserver.pp
```

Node

default node

```
node default {
  class { 'sudo': }
  class { 'ntp':
      servers => ['ntp1.example.com', 'ntp2.example.com']
  }
}
```

node : webserver.example.com

```
node webserver.example.com {
  class { 'sudo': }
  class { 'ntp':
      servers => ['ntp1.example.com', 'ntp2.example.com']
  }
  class { 'apache': }
}
```

Ansible

Ansible ??? Ansible ???
Michael DeHaan ????? 2015 ??Red Hat???????????? Unix???Microsoft Windows ????

Ansible GUI

??

Ansible?? Ansible ??? Michael DeHaan ????? 2015 ??Red Hat??????????? Unix???Microsoft Windows ????

Links

- Ansible Community
  - [AWX for Docker](#)
  - [Documentation](#)
- [How to Test Ansible Roles with Molecule and Docker](#)
- Red Hat Ansible Automation Platform
  - [Download](#)
  - [Trial License](#)
  - [Ansible Lightspeed with Watson Code Assistant](#)
    ?? IBM Watson Code Assistant ???? playbooks
  - [Setting up VSCoce and Ansible Lightspeed on Ubuntu 22.04](#)
- [Ansible Galaxy](#)

Commands

```
# 🔲 lookup 🔲 plugins
# Usage:
# motd_value: "{{ lookup('file', '/etc/motd') }}"
ansible-doc -l -t lookup
```

Ansible Semaphore

Links

- https://www.ansible-semaphore.com/
- [Github](#)
- [Docs](#)

- [Video] [This web UI for Ansible is so damn useful!](#)

Install with Docker

Create the directory

```
mkdir playbooks
mkdir config
chown 1001:1001 config
```

docker-compose.yml:

```
---
volumes:
  semaphore-mysql:
    driver: local
services:
  mysql:
    image: mysql:8.0
    hostname: mysql
    volumes:
      - semaphore-mysql:/var/lib/mysql
    environment:
      - MYSQL_RANDOM_ROOT_PASSWORD=yes
      - MYSQL_DATABASE=semaphore
      - MYSQL_USER=semaphore
      - MYSQL_PASSWORD=secret-password  # change!
    restart: unless-stopped
  semaphore:
    container_name: ansiblesemaphore
    image: semaphoreui/semaphore:v2.8.90
    user: 1001:1001 # change if needed
    ports:
      - 3000:3000
    environment:
      - SEMAPHORE_DB_USER=semaphore
      - SEMAPHORE_DB_PASS=secret-password  # change!
      - SEMAPHORE_DB_HOST=mysql
      - SEMAPHORE_DB_PORT=3306
      - SEMAPHORE_DB_DIALECT=mysql
```

```
  - SEMAPHORE_DB=semaphore

  - SEMAPHORE_ADMIN_PASSWORD=secret-admin-password  # change!

  - SEMAPHORE_ADMIN_NAME=admin

  - SEMAPHORE_ADMIN_EMAIL=admin@localhost

  - SEMAPHORE_ADMIN=admin

  - SEMAPHORE_ACCESS_KEY_ENCRYPTION=  # add to your access key encryption !

  - ANSIBLE_HOST_KEY_CHECKING=false  # (optional) change to true if you want to enable host key checking
volumes:

  - ./inventory/:/inventory:ro

  - ./authorized-keys/:/authorized-keys:ro

  - ./config/:/etc/semaphore:rw

  - ./playbooks:/playbooks:ro
restart: unless-stopped

depends_on:

  - mysql
```

You must specify following confidential variables:

- `MYSQL_PASSWORD` and `SEMAPHORE_DB_PASS` — password for the MySQL user.
- `SEMAPHORE_ADMIN_PASSWORD` — password for the Semaphore's admin user.
- `SEMAPHORE_ACCESS_KEY_ENCRYPTION` — key for encrypting access keys in database. It must be generated by using the following command: `head -c32 /dev/urandom | base64`.

Get Started

1. Create New Project
2. New Keys:
   - Name1: None
   - Type1: None
   - --------
   - Name2: ssh_alang
   - Type2: SSH Key
   - Username2: alang
   - Private key2: <Key-String>
   - --------
   - Name3: sudo_alang
   - Type3: Login with password
   - Login3: alang
   - Password3: <password>
3. New Repository:
   - Name: Local
   - Path: /playbooks
   - Access Key: None

AWX

RedHat 8.7

?????

```
dnf update
reboot
dnf install ansible-core openssl-libs
dnf group install "Development Tools"
dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-8.noarch.rpm
dnf install python39-setuptools_scm
```

AWX

```
git clone -b 22.3.0 https://github.com/ansible/awx.git
cd awx


# 🔲🔲🔲🔲🔲🔲🔲🔲🔲🔲🔲🔲🔲
#vi tools/docker-compose/inventory


make docker-compose-build
cp Makefile{,.orig}
sed -i 's/^\(DOCKER_COMPOSE ?=\).*/\1 docker compose/' Makefile
make docker-compose
```

?????????????????????????????

AWX Web UI: https://server.ip.adress:8043/

?????????

> " <% if (process.env.NODE_ENV === 'production') { %> <% } %> <% if
> (process.env.NODE_ENV === 'production') { %> <% } else { %> <% } %>
> <% if (process.env.NODE_ENV === 'production') { %>

Clean and build the UI

```
docker exec tools_awx_1 make clean-ui ui-devel
```

??????????

> **"** Creating an optimized production build...
> Browserslist: caniuse-lite is outdated. Please run:
>   npx update-browserslist-db@latest
>   Why you should do it regularly: https://github.com/browserslist/update-db#readme

Ctrl + C ??

```
docker exec -it tools_awx_1 bash
> cd /awx_devel/awx/ui
> npx update-browserslist-db@latest
> exit
```

?????

```
docker exec tools_awx_1 make clean-ui ui-devel
```

??????????????

> **"** The project was built assuming it is hosted at ./.
> You can control this with the homepage field in your package.json.
>
> The build folder is ready to be deployed.
>
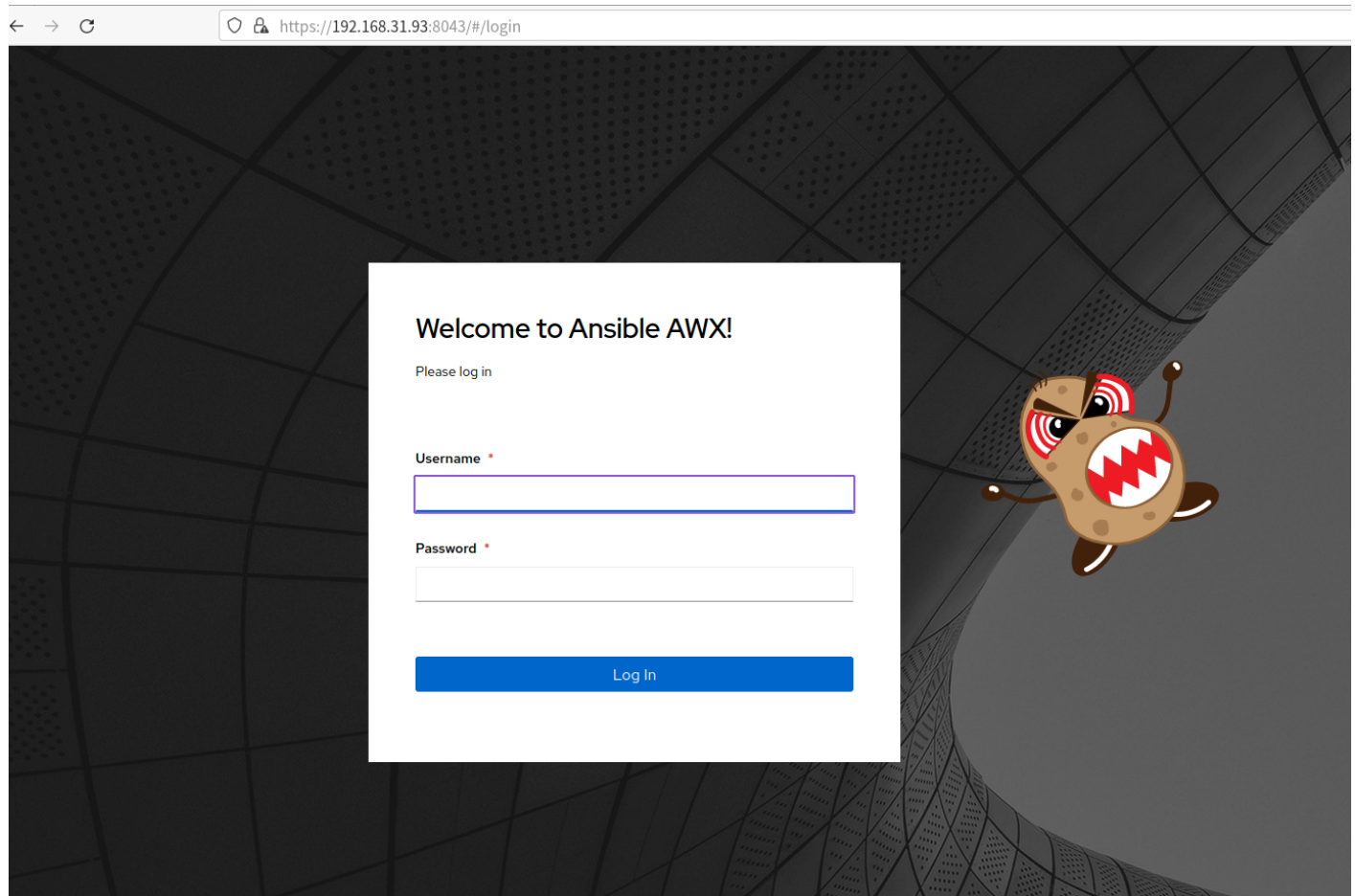> Find out more about deployment here:
>
>   https://cra.link/deployment
>
> touch awx/ui/.ui-built
> make[1]: Leaving directory '/awx_devel'

?? AWX ?????

? Ctrl + C ?? container????

make docker-compose

????? https://server.ip.adress:8043/

???admin ????{? log ?}

Q & A

> " No match for argument: rsyslog-8.2102.0-106.el9

Solution:

```
cp tools/ansible/roles/dockerfile/templates/Dockerfile.j2{,.orig}
sed -i 's/rsyslog-8.[0-9a-z\.\-]*/rsyslog/g' tools/ansible/roles/dockerfile/templates/Dockerfile.j2
```

AWX Commands

?? AWX

```
# 🗆🗆🗆
cd awx-repo/
make docker-compose


# 🗆🗆🗆
make docker-compose COMPOSE_UP_OPTS=-d
```

?? AWX

```
docker stop tools_awx_1 tools_postgres_1 tools_redis_1
```

Create an admin user

```
docker exec -ti tools_awx_1 awx-manage createsuperuser
```

Ansible

Playbooks

- [Github] [Ansible for DevOps Examples](#)
- [AI] [Welcome to the Ansible Lightspeed with IBM Watson Code Assistant Technical Preview | Ansible Collaborative](#)
- [AI] [Red Hat Ansible Lightspeed | Red Hat Developer](#)

Include a variables file

```
---
- hosts: all
  become: true

  vars_files:
    - vars.yml
```

vars.yml

```
---
download_dir: /tmp
solr_dir: /opt/solr
solr_version: 8.6.0
solr_checksum: sha512:6b0d618069e37215f305d9a61a3e65be2b9cfc32a3689ea6a25\
be2f220b1ecc96a644ecc31c81e335a2dfa0bc8b7d0f2881ca192c36fd435cdd832fd7309a9ddb
```

Installs Apache on a RHEL/CentOS server

```
---
- hosts: all
  become: yes

  tasks:
    - name: Install Apache.
      dnf:
        name:
```

```yaml
      - httpd
      - httpd-devel
    state: present


  - name: Copy configuration files.
    copy:
      src: "{{ item.src }}"
      dest: "{{ item.dest }}"
      owner: root
      group: root
      mode: 0644
    with_items:
      - src: httpd.conf
        dest: /etc/httpd/conf/httpd.conf
      - src: httpd-vhosts.conf
        dest: /etc/httpd/conf/httpd-vhosts.conf


  - name: Make sure Apache is started now and at boot.
    service:
      name: httpd
      state: started
      enabled: yes
```

Deploy Node.js app

```yaml
---
- hosts: all
  become: yes

  vars:
    node_apps_location: /usr/local/opt/node

  tasks:
  - name: Install EPEL repo.
    dnf: name=epel-release state=present

  - name: Import Remi GPG key.
    rpm_key:
    key: "https://rpms.remirepo.net/RPM-GPG-KEY-remi2018"
    state: present
```

```yaml
    - name: Install Remi repo.
      dnf:
      name: "https://rpms.remirepo.net/enterprise/remi-release-8.rpm"
      state: present

    - name: Ensure firewalld is stopped (since this is for testing).
      service: name=firewalld state=stopped

    - name: Install Node.js and npm.
      dnf: name=npm state=present enablerepo=epel

    - name: Install Forever (to run our Node.js app).
      npm: name=forever global=yes state=present

  - name: Ensure Node.js app folder exists.
    file: "path={{ node_apps_location }} state=directory"

  - name: Copy example Node.js app to server.
    copy: "src=app dest={{ node_apps_location }}"

  - name: Install app dependencies defined in package.json.
    npm: path={{ node_apps_location }}/app

  - name: Check list of running Node.js apps.
    command: /usr/local/bin/forever list
    register: forever_list
    changed_when: false

  - name: Start example Node.js app.
    command: "/usr/local/bin/forever start {{ node_apps_location }}/app/app.js"
    when: "forever_list.stdout.find(node_apps_location + '/app/app.js') == -1"
```

Basic LAMP server setup

```yaml
  tasks:
  - name: Get software for apt repository management.
    apt:
      state: present
      name:
```

```yaml
      - python3-apt
      - python3-pycurl

  - name: Add ondrej repository for later versions of PHP.
    apt_repository: repo='ppa:ondrej/php' update_cache=yes

  - name: "Install Apache, MySQL, PHP, and other dependencies."
    apt:
      state: present
      name:
      - acl
        - git
        - curl
        - unzip
        - sendmail
        - apache2
        - php8.2-common
        - php8.2-cli
        - php8.2-dev
        - php8.2-gd
        - php8.2-curl
        - aphp8.2-opcache
        - php8.2-xml
        - php8.2-mbstring
        - php8.2-pdo
        - php8.2-mysql
        - php8.2-apcu
        - libpcre3-dev
        - libapache2-mod-php8.2
        - python3-mysqldb
        - mysql-server

  - name: Disable the firewall (since this is for local dev only).
    service: name=ufw state=stopped

  - name: "Start Apache, MySQL, and PHP."
    service: "name={{ item }} state=started enabled=yes"
    with_items:
      - apache2
      - mysql
```

Configure Apache

```yaml
- name: Enable Apache rewrite module (required for Drupal).
  apache2_module: name=rewrite state=present
  notify: restart apache


- name: Add Apache virtualhost for Drupal.
  template:
    src: "templates/drupal.test.conf.j2"
    dest: "/etc/apache2/sites-available/{{ domain }}.test.conf"
    owner: root
    group: root
    mode: 0644
  notify: restart apache


- name: Enable the Drupal site.
  command: >
    a2ensite {{ domain }}.test
    creates=/etc/apache2/sites-enabled/{{ domain }}.test.conf
  notify: restart apache


- name: Disable the default site.
  command: >
    a2dissite 000-default
    removes=/etc/apache2/sites-enabled/000-default.conf
  notify: restart apache
```

Template: drupal.test.conf.j2

```
<VirtualHost *:80>
  ServerAdmin webmaster@localhost
  ServerName {{ domain }}.test
  ServerAlias www.{{ domain }}.test
  DocumentRoot {{ drupal_core_path }}/web
  <Directory "{{ drupal_core_path }}/web">
    Options FollowSymLinks Indexes
    AllowOverride All
  </Directory>
</VirtualHost>
```

Configure PHP with *lineinfile*

```
- name: Adjust OpCache memory setting.
  lineinfile:
    dest: "/etc/php/8.2/apache2/conf.d/10-opcache.ini"
    regexp: "^opcache.memory_consumption"
    line: "opcache.memory_consumption = 96"
    state: present
  notify: restart apache
```

Configure MySQL

```
- name: Create a MySQL database for Drupal.
  mysql_db: "db={{ domain }} state=present"


- name: Create a MySQL user for Drupal.
  mysql_user:
    name: "{{ domain }}"
    password: "1234"
    priv: "{{ domain }}.*:ALL"
    host: localhost
    state: present
```

Install Composer with *get_url*

```
- name: Download Composer installer.
  get_url:
    url: https://getcomposer.org/installer
    dest: /tmp/composer-installer.php
    mode: 0755


- name: Run Composer installer.
  command: >
    php composer-installer.php
    chdir=/tmp
    creates=/usr/local/bin/composer


- name: Move Composer into globally-accessible location.
  command: >
    mv /tmp/composer.phar /usr/local/bin/composer
```

```
      creates=/usr/local/bin/composer
```

Create a Drupal project with Composer

```
- name: Ensure Drupal directory exists.
  file:
    path: "{{ drupal_core_path }}"
    state: directory
    owner: www-data
    group: www-data


- name: Check if Drupal project already exists.
  stat:
    path: "{{ drupal_core_path }}/composer.json"
    register: drupal_composer_json


- name: Create Drupal project.
  composer:
    command: create-project
    arguments: drupal/recommended-project "{{ drupal_core_path }}"
    working_dir: "{{ drupal_core_path }}"
    no_dev: true
  become_user: www-data
  when: not drupal_composer_json.stat.exists
```

Decompress the tar file

```
- name: Download Solr.
  get_url:
    url: "https://archive.apache.org/dist/lucene/solr/\
{{ solr_version }}/solr-{{ solr_version }}.tgz"
    dest: "{{ download_dir }}/solr-{{ solr_version }}.tgz"
    checksum: "{{ solr_checksum }}"


- name: Expand Solr.
  unarchive:
    src: "{{ download_dir }}/solr-{{ solr_version }}.tgz"
    dest: "{{ download_dir }}"
    remote_src: true
    creates: "{{ download_dir }}/solr-{{ solr_version }}/README.txt"
```

Infracture as code (IaC)

Infrastructure as Code (IaC): When all of the configuration necessary to deploy and manage a node in the infrastructure is stored in version control.

IaC Options

?? Puppet ?? Terraform?Ansible ? Google Cloud Platform (GCP) ?????????? Puppet ????

Terraform

Terraform stands out as a potent IaC tool that specializes in provisioning and managing infrastructure resources across various cloud providers. Its declarative syntax allows you to define your desired infrastructure state, and Terraform takes care of translating this into concrete resources. This approach enables you to codify your infrastructure configurations, fostering version control, collaboration, and reproducibility. Terraform's provider ecosystem empowers you to manage a wide spectrum of resources, from virtual machines to databases, across multiple cloud environments. Its focus on infrastructure provisioning aligns well with cloud-native approaches, making it an excellent choice for orchestrating cloud resources and building scalable, modern applications.

Andible

Unlike Puppet, which revolves around agent-based communication, Ansible adopts an agentless architecture that relies on SSH or other remote APIs for system management. This lightweight approach simplifies deployment and reduces the overhead of maintaining agents on target nodes. Ansible employs a simple and human-readable YAML syntax to define playbooks, which describe the desired state of systems. These playbooks facilitate a wide range of automation tasks, from configuration management to application deployment. Ansible's versatility extends beyond servers to network devices, making it suitable for managing diverse IT environments. While it may lack the advanced features of Puppet's catalog-based system, Ansible excels in its simplicity, ease of adoption, and suitability for rapid deployment scenarios.

Google Cloud Deployment Manager

Within the realm of Google Cloud Platform (GCP), you can leverage native tools for configuration management and infrastructure orchestration. Google Cloud Deployment Manager enables you to define your infrastructure using YAML or Python templates, offering a declarative approach similar to Terraform. This tool is well-integrated with GCP services and resources, simplifying the orchestration of cloud-specific components like GKE clusters, Cloud

Storage Buckets, and load balancers. Additionally, GCP provides a wide range of managed services that abstract away much of the infrastructure management complexity, allowing you to focus more on application development and less on provisioning and configuration.

Comparing to Puppet

While Puppet excels in its ability to manage configuration drift and ensure system consistency through its catalog-based approach, other IaC tools offer unique advantages.

Terraform's focus on provisioning cloud resources aligns well with modern, cloud-native development practices.

Ansible's agentless architecture simplifies deployment and is well-suited for quick automation tasks across diverse environments.

GCP's native tools provide seamless integration within the Google Cloud ecosystem, streamlining infrastructure management for projects hosted on the platform. Ultimately, the choice between these options depends on your specific needs, preferences, and the ecosystem you are operating within.