Ansible and Terraform Overview

DevOps and Security Lab 2

Why to use SCM and for what

Ansible is a popular Configuration Management Tool (CMT). It’s a practice of tracking and controlling changes to a software system on **multiple** (nodes, localhosts, virtual machines…) at once and from single runner.

With SCM, we can:

- ➢ install software
- ➢ configure systems
- ➢ manage users, access
- ➢ manage instance settings and so on…

Ansible advantages

- yaml based
- agentless
- based on python
- great documentation and community support
- ansible collections
- easy to learn and start
- flexibility
- just very popular

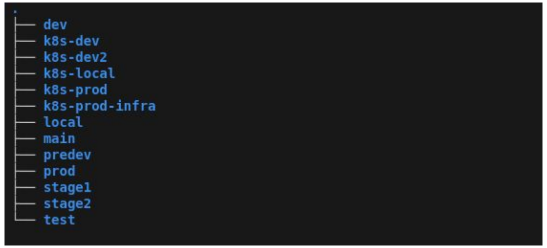
Ansible definitions

- **inventory**: defines the target hosts configurations (*where to run*)
- **roles**: tasks, variables, files, templates… itself (*what to run*)
- **playbooks**: roles binding with inventory and roles variables redefinition (run particular role on the particular node (node groups) and modify the role variables according to the specific target nodes)

Ansible folders structure

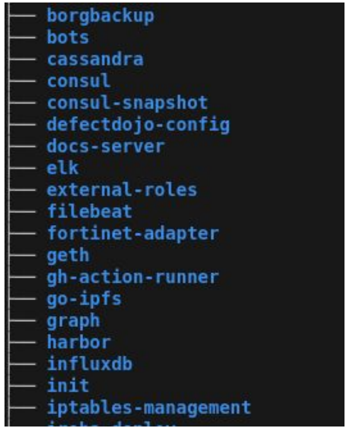
We usually separate ansible yaml files to inventory/roles/playbooks folders. Try to follow to the same approach. *Why it’s useful?*

playbooks



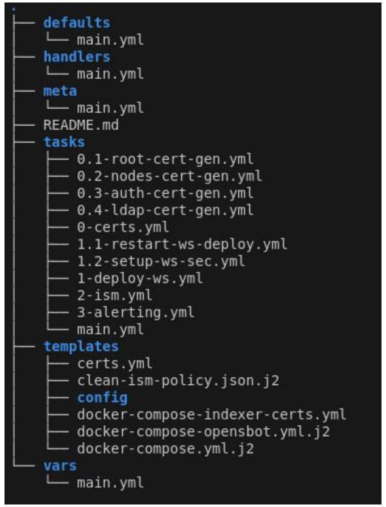
```
.
├── dev
├── k8s-dev
├── k8s-dev2
├── k8s-local
├── k8s-prod
├── k8s-prod-infra
├── local
├── main
├── predev
├── prod
├── stage1
├── stage2
└── test
```

roles



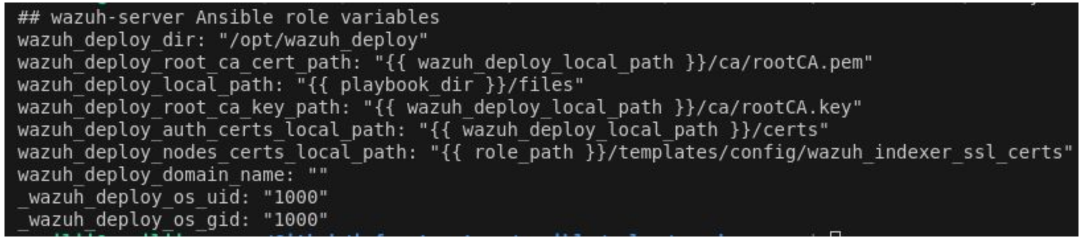
```
├── borgbackup
├── bots
├── cassandra
├── consul
├── consul-snapshot
├── defectdojo-config
├── docs-server
├── elk
├── external-roles
├── filebeat
├── fortinet-adapter
├── geth
├── gh-action-runner
├── go-ipfs
├── graph
├── harbor
├── influxdb
├── init
├── iptables-management
```

inside role

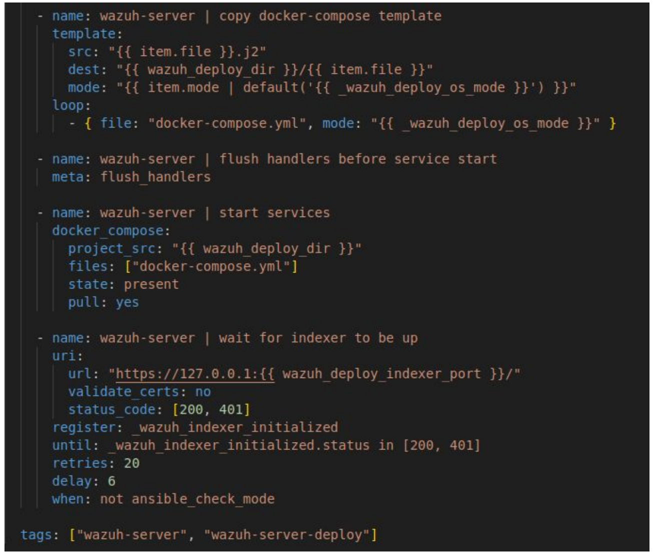


```
.
├── defaults
│   └── main.yml
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   ├── 0.1-root-cert-gen.yml
│   ├── 0.2-nodes-cert-gen.yml
│   ├── 0.3-auth-cert-gen.yml
│   ├── 0.4-ldap-cert-gen.yml
│   ├── 0-certs.yml
│   ├── 1.1-restart-ws-deploy.yml
│   ├── 1.2-setup-ws-sec.yml
│   ├── 1-deploy-ws.yml
│   ├── 2-ism.yml
│   ├── 3-alerting.yml
│   └── main.yml
├── templates
│   ├── certs.yml
│   ├── clean-ism-policy.json.j2
│   ├── config
│   ├── docker-compose-indexer-certs.yml
│   ├── docker-compose-opensbot.yml.j2
│   └── docker-compose.yml.j2
└── vars
    └── main.yml
```

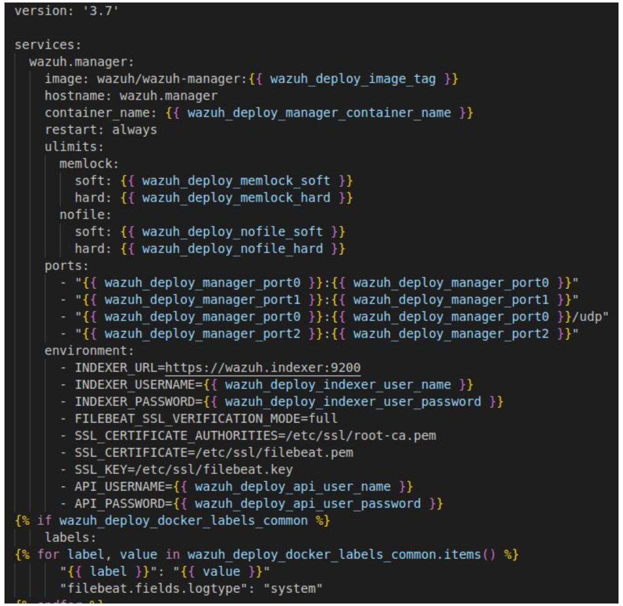
Inside Ansible files



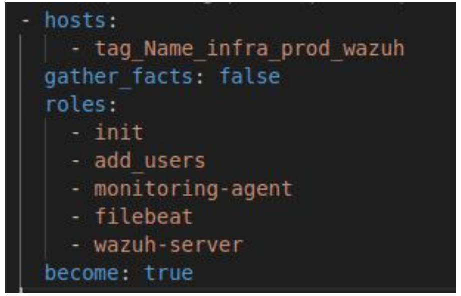
```
## wazuh-server Ansible role variables
wazuh_deploy_dir: "/opt/wazuh_deploy"
wazuh_deploy_root_ca_cert_path: "{{ wazuh_deploy_local_path }}/ca/rootCA.pem"
wazuh_deploy_local_path: "{{ playbook_dir }}/files"
wazuh_deploy_root_ca_key_path: "{{ wazuh_deploy_local_path }}/ca/rootCA.key"
wazuh_deploy_auth_certs_local_path: "{{ wazuh_deploy_local_path }}/certs"
wazuh_deploy_nodes_certs_local_path: "{{ role_path }}/templates/config/wazuh_indexer_ssl_certs"
wazuh_deploy_domain_name: ""
_wazuh_deploy_os_uid: "1000"
_wazuh_deploy_os_gid: "1000"
```



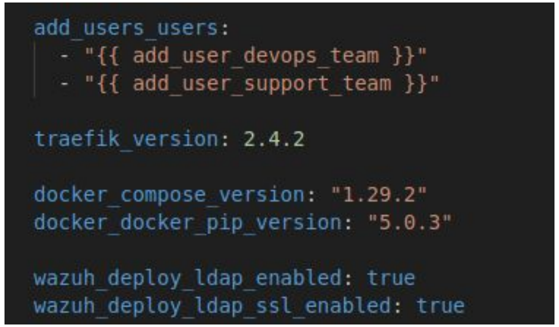
```
  - name: wazuh-server | copy docker-compose template
    template:
      src: "{{ item.file }}.j2"
      dest: "{{ wazuh_deploy_dir }}/{{ item.file }}"
      mode: "{{ item.mode | default('{{ _wazuh_deploy_os_mode }}') }}"
    loop:
      - { file: "docker-compose.yml", mode: "{{ _wazuh_deploy_os_mode }}" }

  - name: wazuh-server | flush handlers before service start
    meta: flush_handlers

  - name: wazuh-server | start services
    docker_compose:
      project_src: "{{ wazuh_deploy_dir }}"
      files: ["docker-compose.yml"]
      state: present
      pull: yes

  - name: wazuh-server | wait for indexer to be up
    uri:
      url: "https://127.0.0.1:{{ wazuh_deploy_indexer_port }}/"
      validate_certs: no
      status_code: [200, 401]
    register: _wazuh_indexer_initialized
    until: _wazuh_indexer_initialized.status in [200, 401]
    retries: 20
    delay: 6
    when: not ansible_check_mode

tags: ["wazuh-server", "wazuh-server-deploy"]
```



```
version: '3.7'

services:
  wazuh.manager:
    image: wazuh/wazuh-manager:{{ wazuh_deploy_image_tag }}
    hostname: wazuh.manager
    container_name: {{ wazuh_deploy_manager_container_name }}
    restart: always
    ulimits:
      memlock:
        soft: {{ wazuh_deploy_memlock_soft }}
        hard: {{ wazuh_deploy_memlock_hard }}
      nofile:
        soft: {{ wazuh_deploy_nofile_soft }}
        hard: {{ wazuh_deploy_nofile_hard }}
    ports:
      - "{{ wazuh_deploy_manager_port0 }}:{{ wazuh_deploy_manager_port0 }}"
      - "{{ wazuh_deploy_manager_port1 }}:{{ wazuh_deploy_manager_port1 }}"
      - "{{ wazuh_deploy_manager_port0 }}:{{ wazuh_deploy_manager_port0 }}/udp"
      - "{{ wazuh_deploy_manager_port2 }}:{{ wazuh_deploy_manager_port2 }}"
    environment:
      - INDEXER_URL=https://wazuh.indexer:9200
      - INDEXER_USERNAME={{ wazuh_deploy_indexer_user_name }}
      - INDEXER_PASSWORD={{ wazuh_deploy_indexer_user_password }}
      - FILEBEAT_SSL_VERIFICATION_MODE=full
      - SSL_CERTIFICATE_AUTHORITIES=/etc/ssl/root-ca.pem
      - SSL_CERTIFICATE=/etc/ssl/filebeat.pem
      - SSL_KEY=/etc/ssl/filebeat.key
      - API_USERNAME={{ wazuh_deploy_api_user_name }}
      - API_PASSWORD={{ wazuh_deploy_api_user_password }}
{% if wazuh_deploy_docker_labels_common %}
    labels:
{% for label, value in wazuh_deploy_docker_labels_common.items() %}
      "{{ label }}": "{{ value }}"
      "filebeat.fields.logtype": "system"
```



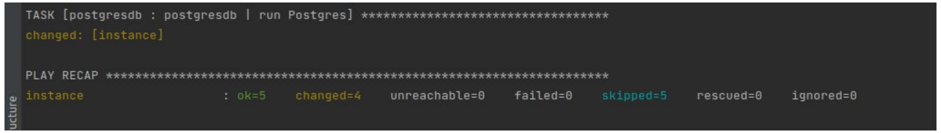
```
- hosts:
    - tag_Name_infra_prod_wazuh
  gather_facts: false
  roles:
    - init
    - add_users
    - monitoring-agent
    - filebeat
    - wazuh-server
  become: true
```



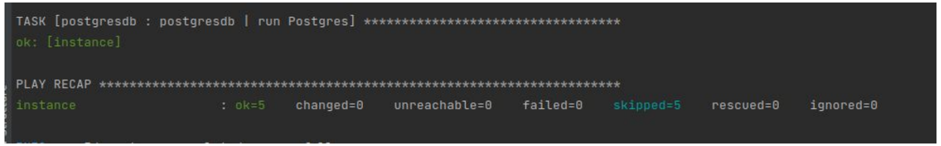
```
add_users_users:
  - "{{ add_user_devops_team }}"
  - "{{ add_user_support_team }}"

traefik_version: 2.4.2

docker_compose_version: "1.29.2"
docker_docker_pip_version: "5.0.3"

wazuh_deploy_ldap_enabled: true
wazuh_deploy_ldap_ssl_enabled: true
```

Ansible basic principles

- apply the same settings/changes to the whole target node groups within the single playbook run
- follow to idempotence principle (*whether you run a playbook once or multiple times, the outcome should be the same*)
- avoid to use shell commands in tasks and use ansible modules instead (why?)
- use variables precedence (order) for different playbooks (target node groups)
- do not hard code variables as much as possible
- avoid to use **become: true** if it's not necessary



```
TASK [postgresdb : postgresdb | run Postgres] ***********************************
changed: [instance]

PLAY RECAP ****************************************************************************
instance                   : ok=5    changed=4    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
```



```
TASK [postgresdb : postgresdb | run Postgres] ***********************************
ok: [instance]

PLAY RECAP ****************************************************************************
instance                   : ok=5    changed=0    unreachable=0    failed=0    skipped=5    rescued=0    ignored=0
```

Ansible useful tips

- ➢ run playbook: **cd playbooks, ansible-playbooks playbooks/env/path_to_main_task_file.yaml**
- ➢ run one role only: **ansible-playbooks playbooks/env/path_to_main_task_file.yaml – tags wazuh-server**
- ➢ run role on the particular host only: **ansible-playbooks playbooks/env/path_to_main_task_file.yaml –limit <ansible host tag>**
- ➢ run in check and diff modes: **ansible-playbooks playbooks/env/path_to_main_task_file.yaml –diff –check**
- ➢ run as root user password: **ansible-playbooks playbooks/env/path_to_main_task_file.yaml –ask-become-pass**

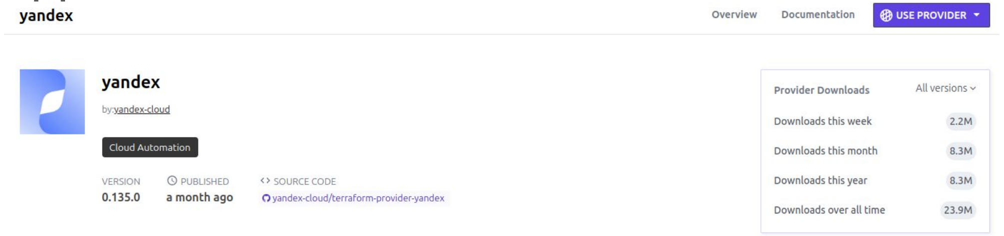
Terraform

A tool by HashiCorp to manage infrastructure (cloud instances, RDB, cloud accounts, DNS, network, GitHub, k8s and many other things…)

We usually work with Terraform modules. Terraform modules registry: https://registry.terraform.io

Steps:

1. Import/create Terraform modules.
2. Change configs: modify variables,add extra `data` modules or outputs.
3. Initialize modules.
4. See what is going to be changed/applied.
5. Apply Terraform configurations.



yandex

by: yandex-cloud

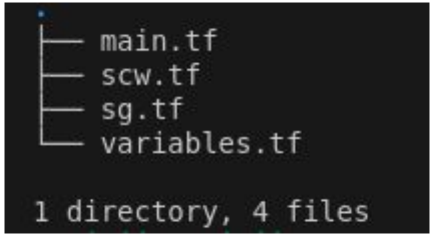
Cloud Automation

VERSION	PUBLISHED	SOURCE CODE
0.135.0	a month ago	yandex-cloud/terraform-provider-yandex

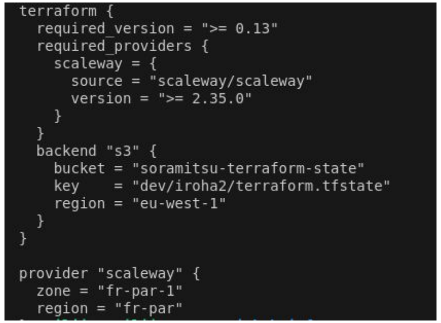
Provider Downloads	All versions
Downloads this week	2.2M
Downloads this month	8.3M
Downloads this year	8.3M
Downloads over all time	23.9M

Terraform folders

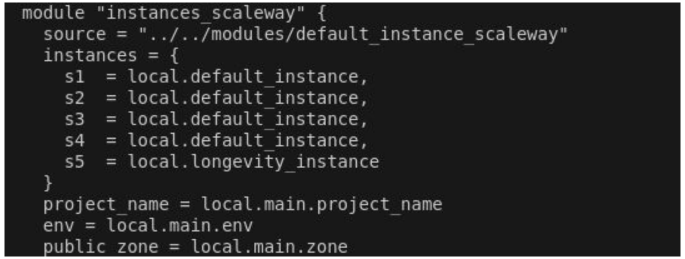
Basic folder structure: main.tf, variables.tf, config.tf and outputs.tf



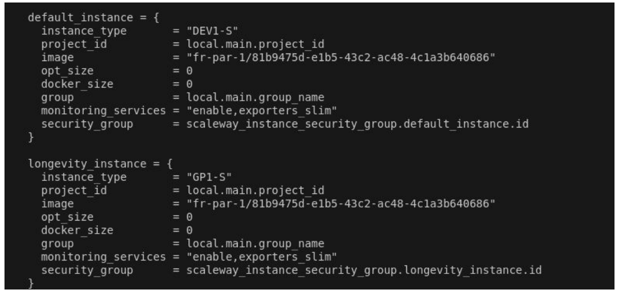
```
.
├── main.tf
├── scw.tf
├── sg.tf
└── variables.tf

1 directory, 4 files
```



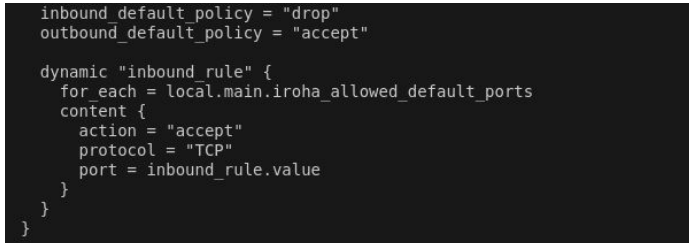
```
terraform {
  required_version = ">= 0.13"
  required_providers {
    scaleway = {
      source = "scaleway/scaleway"
      version = ">= 2.35.0"
    }
  }
  backend "s3" {
    bucket = "soramitsu-terraform-state"
    key    = "dev/iroha2/terraform.tfstate"
    region = "eu-west-1"
  }
}

provider "scaleway" {
  zone = "fr-par-1"
  region = "fr-par"
```



```
module "instances_scaleway" {
  source = "../../modules/default_instance_scaleway"
  instances = {
    s1  = local.default_instance,
    s2  = local.default_instance,
    s3  = local.default_instance,
    s4  = local.default_instance,
    s5  = local.longevity_instance
  }
  project_name = local.main.project_name
  env = local.main.env
  public_zone = local.main.zone
```



```
  default_instance = {
    instance_type       = "DEV1-S"
    project_id          = local.main.project_id
    image               = "fr-par-1/81b9475d-e1b5-43c2-ac48-4c1a3b640686"
    opt_size            = 0
    docker_size         = 0
    group               = local.main.group_name
    monitoring_services = "enable,exporters_slim"
    security_group      = scaleway_instance_security_group.default_instance.id
  }

  longevity_instance = {
    instance_type       = "GP1-S"
    project_id          = local.main.project_id
    image               = "fr-par-1/81b9475d-e1b5-43c2-ac48-4c1a3b640686"
    opt_size            = 0
    docker_size         = 0
    group               = local.main.group_name
    monitoring_services = "enable,exporters_slim"
    security_group      = scaleway_instance_security_group.longevity_instance.id
  }
```



```
  inbound_default_policy = "drop"
  outbound_default_policy = "accept"

  dynamic "inbound_rule" {
    for_each = local.main.iroha_allowed_default_ports
    content {
      action = "accept"
      protocol = "TCP"
      port = inbound_rule.value
    }
  }
}
```

Terraform commands

Firstly, go to Terraform folder:

- ➢ init modules: **terraform init**
- ➢ plan changes: **terraform plan**
- ➢ apply changes: **terraform apply**
- ➢ remove all configurations: **terraform destroy**
- ➢ list Terraform states: **terraform state list**