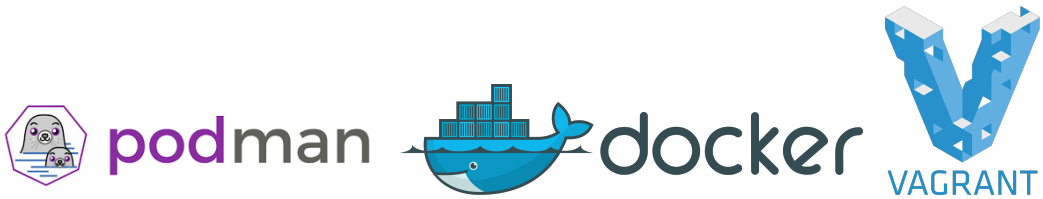
Jenkins & Ansible Ops + K8s Deploy & Quality Gate

Created by Eugenio Marzo, last modified on 24 May, 2024

Tools & Practices



- Docker o Podman
- Jenkins
- Ansible
- Vagrant (solo per workstation intel)
- Kind o K3s o Minikube
- Groovy
- Helm
- GIT
- Flask
- CI/CD
- Configuration Management

Scopo di questo modulo formativo

Podman Overview

Lo scopo di questo modulo è esercitarsi nelle prime operazioni di base sui seguenti ambiti

- Utilizzo di Ansible per installazione Jenkins e interazione con Kubernetes
- Primo setup e utilizzo di Jenkins per build di immagini Docker
- Interazione con API di Kubernetes per una prima comprensione degli attributi delle risorse più utilizzate (Deployment, Statefulset, Services, ...)

Tips Vagrant (provisioner Ansible)

```
Vagrant.configure("2") do |config|

  config.vm.box = "generic/oracle8"
  config.vbguest.auto_update = false

  config.vm.network "private_network", ip: "192.168.50.111"

  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "./deploy.yml"
    ansible.become = true
  end

  config.vm.provider "virtualbox" do |v|
    v.memory = 2048
    v.cpus = 1
  end

end
```

Tips Start Jenkins with Docker or Podman

```
#### Docker
docker run -p 8080:8080 -p 50000:50000 -v jenkins_home:/var/jenkins_home jenkins/jen

# Attenzione a
# Inoltro porte TCP
# Persistenza, il -v
# Tag immagine Docker (questo è un esempio, scegliere l'ultima LTS)

#### Podman
```

```
podman volume create jenkins-data
```



```
        --detach \
        --privileged \
        --publish 8080:8080 \
        --publish 50000:50000 \
        --volume jenkins-data:/var/jenkins_home \
        --volume jenkins-docker-certs:/certs/client:ro \
        docker.io/jenkinsci/blueocean

podman ps
     CONTAINER ID  IMAGE                                    COMMAND               CR
     01077063390e  docker.io/jenkinsci/blueocean:latest                           3

podman exec -it jenkins-app sh
    ~ $ cat /var/jenkins_home/secrets/initialAdminPassword
    a72e26e162344d69b60da77d5e56a4aa
```

Installazione locale Kubernetes

In caso non si conosca K8s procedere alla visione dei seguenti corsi e libri

https://learn.acloud.guru/course/introduction-to-kubernetes/dashboard

https://learn.acloud.guru/course/2e0bad96-a602-4c91-9da2-e757d32abb8f/dashboard

Il seguente modulo richiede l'interazione con Kubernetes in una distribuzione minimale e locale. Esistendo svariate soluzione viene proposto allo studente di provare uno dei seguenti tool e scegliere quello che più si adatta alla propria workstation di lavoro.

- MiniKube
- K3s
- Kind

Al termine dell'installazione è necessario che funzioni l'interazione tramite kubectl e che quindi si siano verificati la connettività e l'accesso all'instanza standalone di Kubernetes. Ad esempio provare i seguenti comandi:

```
kubectl get pods -A
kubectl get namespaces -A

# Creare il seguente namespace
kubectl create namespace formazione_sou
```

Track 1 - Workstation Mac processore intel

- Installazione di Jenkins (solo master) tramite Ansible. Ansible deve quindi utilizzare Docker per l'installazione di Jenkins

https://docs.ansible.com/ansible/2.9/modules/docker_container_module.html

```
- name: Container present
  docker_container:
    name: mycontainer
    state: present
    image: ubuntu:14.04
    command: sleep infinity
```

https://docs.ansible.com/ansible/latest/collections/containers/podman/podman_container_module.html

```
- name: Container present
  containers.podman.podman_container:
    name: mycontainer
    state: present
    image: ubuntu:14.04
    command: "sleep 1d"
```

Track 1 - Workstation Mac M1/2/3 (processore Apple, architettura ARM)

- Installazione di Ansible e Docker (può essere utilizzato anche Podman) sulla propria workstation di lavoro
- Installazione di Jenkins tramite Ansible e gli opportuni moduli per Docker o Podman

https://docs.ansible.com/ansible/2.9/modules/docker_container_module.html

```
- name: Container present
  docker_container:
    name: mycontainer
    state: present
    image: ubuntu:14.04
    command: sleep infinity
```

https://docs.ansible.com/ansible/latest/collections/containers/podman/podman_container_module.html

```
- name: Container present
  containers.podman.podman_container:
    name: mycontainer
    state: present
```

```
    image: ubuntu:14.04
    command: "sleep 1d"
```


Track 2 - Pipeline Jenkins dichiarativa (Groovy) per build immagine Docker

- Creare repo github denominato "fomazione_sou_k8s"
- Creare Dockerfile app di esempio Flask (Python) che esponga una pagina avente stringa "hello world" (prendere spunto da qui: https://github.com/docker/awesome-compose/tree/master/flask/app)
- Scrivere pipeline dichiarativa Jenkins che effettui una build dell'immagine Docker e che effettui il push sul proprio account DockerHub. In caso di problemi con il rate limit di Docker installare localmente un registry Docker (va bene anche il registry nativo di Docker https://docs.docker.com/registry/ senza autenticazione). La pipeline Jenkins deve chiamarsi **flask-app-example-build**. Il tag dell'immagine Docker deve essere uguale al tag git se "buildata" da tag git, latest se "buildata" da branch master, uguale a "develop + sha comit GIT" se "buildata" da branch develop.

Tips (pipeline Jenkins build and push)

```
def buildAndPushTag(Map args) {
    def defaults = [
        registryUrl: '***',
        dockerfileDir: "./",
        dockerfileName: "Dockerfile",
        buildArgs: "",
        pushLatest: true
    ]
    args = defaults + args
    docker.withRegistry(args.registryUrl) {
        def image = docker.build(args.image, "${args.buildArgs} ${args.dockerfileDir
        image.push(args.buildTag)
        if(args.pushLatest) {
            image.push("latest")
            sh "docker rmi --force ${args.image}:latest"
        }
        sh "docker rmi --force ${args.image}:${args.buildTag}"

        return "${args.image}:${args.buildTag}"
    }
}
```

Track 3 - Helm Chart

- Creare un Helm Chart (utilizzare helm init) custom che effettui il deploy dell'immagine creata tramite la pipeline **flask-app-example-build** (in input deve essere possibile specificare quale tag rilasciare)
- Prendere esempio da qui: https://github.com/thedataincubator/flask-chart oppure dai chart di Bitnami (https://github.com/bitnami/charts
- Versionare helm chart nel repo "formazione_sou" in una sub-folder denominata "charts"

Track 4 - Helm Install

Track 5 - Check Deployment Best Practices

Scrivere script Bash o Python che effettui le seguenti operazioni

1. Autenticandosi tramite un Service Account di tipo "cluster-reader" (studiare bene RBAC di k8s) esegua un export del Deployment dell'applicazione Flask installata tramite la Track 3. E' possibile scegliere tra: utilizzo API k8s (modulo python kubernetes, wrapping di kubectl (es: kubectl get deployment foobar -o yaml -n formazione_sou), wrapping di curl (sempre verso le API)
2. L'automatismo deve ritornare un errore se non presenti nel Deployment i seguenti attributi: Readiness e Liveness Probles, Limits e Requests

Track 6 - Bonus Track

- Equipaggiare l'instanza Kubernetes di un Ingress Controller Nginx
- Fare in modo che l'Helm Chart installi anche l'ingress nginx
- Fare in modo che chiamanto via http http://formazionesou.local si ottenga "hello world", ovvero quanto esposto dall'applicazione Flask creata nei punti precedenti

Tips

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: ingress-myservicea
spec:
  rules:
  - host: myservicea.foo.org
    http:
      paths:
      - path: /
        pathType: Prefix
        backend:
          service:
            name: myservicea
            port:
              number: 80
  ingressClassName: nginx
```

No labels