CI/CD

Contiuous Integration / Continuous Devloymy

(**Dev**elopment|Dep**loym**ent|Deliver**y**)

14 oktober 2020 – Linux User Group Nijmegen

Voorbeeld build straat

We gaan kijken naar het uitrollen (van programmeren tot produktie) van een eenvoudige go applicatie (main.go: toont “Hello World!” in de browser.

Workflow (wordt uitgewerkt per onderdeel):

- **git: commit** code: dit triggered **jenkins** (mbv “post-commit” hook)
- **jenkins**: **checkout** code + configs (docker, ansible) van git repo
- **jenkins**: **build** docker image met applicatie code
- **jenkins**: **test** docker image met de applicatie code
  - ok? continue
- **jenkins**: **publish** docker image (bv. naar hub.docker.com)
  - ok? continue
- **jenkins, ansible**: **deploy** to production kubernetes cluster (using ansible k8s module

Technieken

- git - de repo met onze applicatie code
- jenkins - vrijwel alles gebeurd hier
- docker - bouwsteen waarin de applicatie draait
- kubernetes - management van de docker containers
- ansible - gebruikt om de app (het docker image) uit te rollen (naar het kubernetes cluster).

De applicatie - main.go (een eenvoudig “Hello World”)

```
package main

import (
        "log"
        "net/http"
)

type Server struct{}

func (s *Server) ServeHTTP(w http.ResponseWriter, r *http.Request) {
        w.WriteHeader(http.StatusOK)
        w.Header().Set("Content-Type", "application/json")
        w.Write([]byte(`{"message": "Hello World!"}`))
}

func main() {
        s := &Server{}
        http.Handle("/", s)
        log.Fatal(http.ListenAndServe(":8080", nil))
}
```

Maak de Jenkins VM

url: https://www.chrisjmendez.com/2017/01/08/installing-jenkins-using-virtualbox-and-vagrant/

Stappen:

- Maak Vagrantfile
- Maak bootstrap.sh (en bootstrap_jenkins.sh)
- $ vagrant up
- Login op de jenkins web interface (“IP”:8080)
    - make admin user
    - Installeer plugins (defaults, github, docker)

Config Jenkins

url: https://www.magalix.com/blog/create-a-ci/cd-pipeline-with-kubernetes-and-jenkins

- Login op Jenkins web-interface met admin user
- Voer je credentials in voor hub.docker.com en github:
  - “Manage Jenkins” -> “Manage Credentials” -> “global” -> “Add Credentials”
- Create new item (geef een naam, bv. “go_app”)
- Configure het nieuwe item: o.a. bij “Build triggers” en “Pipeline”: selecteer “Pipeline script from SCM”. Geef “Git” als SCM, de url naar je git repo en je credentials. Als “script path” geef je de naam van je Jenkins pipeline file (“Jenkinsfile”) = je “build straat”.

Data in de git repository

url: https://github.com/oscarbuse/cicd

Files:

- **main.go** - onze applicatie
- **main_test.go** - om de applicatie te testen
- **Jenkinsfile** - instructies voor de “jenkins pipeline” (“build straat”)
- **Dockerfile** - instructies voor het creëeren van het docker image met de go applicatie
- **playbook.yml**, **service.yml**, **deployment.yml** - ansible files om vanuit Jenkins naar kubernetes uit te rollen

Jenkinsfile

De stappen (“stages”) die Jenkins moet doen voor het complementeren van je “build straat” zet je in een “Pipeline file” (met bv. als naam “Jenkinsfile”).

Dit file kun je in je git repo zetten en als je je git repo enabled in Jenkins en naar dit file verwijst kan Jenkins deze vinden.

De te definieren stappen (“stages”) voor Jenkins:

- de build stage: hiermee “bouw” je je applicatie, in dit geval een docker image met onze go applicatie.
- de test stage: hiermee kun je tests runnen om te kijken of de applicatie doet wat ie moeten doen. Hiervoor wordt een apart testfile (main_test.go) gebruikt.
- de publish stage: Zet het net gebakken docker image op hub.docker.com.
- de deploy stage: zet het image vanuit hub.docker.com op het produktie k8s cluster.

de Jenkins Build stage

```
stage('Build') {
    agent {
        docker {
            image 'golang'
        }
    }
    steps {
        // Create our project directory.
        sh 'cd ${GOPATH}/src'
        sh 'mkdir -p ${GOPATH}/src/hello-world'
        // Copy all from Jenkins workspace to project directory.

        sh 'cp -r ${WORKSPACE}/* ${GOPATH}/src/hello-world'
        // Build the app.
        sh 'go build'
    }
}
```

Note: het Dockerfile in de git repo wordt gebruikt om het image te maken (met “docker build”).

de Jenkins Test stage

```
stage('Test') {
    agent {
        docker {
            image 'golang'
        }
    }
    steps {
        // Create our project directory.
        sh 'cd ${GOPATH}/src'
        sh 'mkdir -p ${GOPATH}/src/hello-world'
        // Copy all from Jenkins workspace to project directory.
        sh 'cp -r ${WORKSPACE}/* ${GOPATH}/src/hello-world'
        // Remove cached test results.
        sh 'go clean -cache'
        // Run Unit Tests.
        sh 'go test ./... -v -short'
    }
}
```

de Jenkins Publish stage

```
stage('Publish') {
    environment {
        registryCredential = 'dockerhub_id'
    }
    steps{
        script {
            def appimage = docker.build registry + ":$BUILD_NUMBER"
            docker.withRegistry( '', registryCredential ) {
                appimage.push()
                appimage.push('latest')
            }
         }
    }
}
```

Setup kubernetes cluster

url: https://blog.exxactcorp.com/building-a-kubernetes-cluster-using-vagrant/

- git clone https://exxsyseng@bitbucket.org/exxsyseng/k8s_ubuntu.git
- cd k8s_ubuntu
- vagrant up

de Jenkins Deploy stage

```
stage ('Deploy') {
    steps {
        script{
            def image_id = registry + ":$BUILD_NUMBER"
            sh "ansible-playbook playbook.yml \
               --extra-vars \"image_id=${image_id}\""
        }
    }
}
```

De ansible code haalt Jenkins uit git.

ansible: playbook.yml

```
---------------------------------------------------------
- hosts: localhost
  tasks:
  - name: Deploy the service
    k8s:
      state: present
      definition: "{{ lookup('template', 'service.yml') | from_yaml }}"
      validate_certs: no
      namespace: default
  - name: Deploy the application
    k8s:
      state: present
      validate_certs: no
      namespace: default
      definition: "{{ lookup('template', 'deployment.yml') | from_yaml }}"
---------------------------------------------------------
```

Hoe “weet” jenkins welk kubernetes cluster te gebruiken?:

```
jenkins@jenkins:~$ mkdir .kube
jenkins@jenkins:~$ scp vagrant@kmaster:.kube/config .kube
    -> server: https://172.42.42.100:6443
```

ansible: service.yml

```
---------------------------------------------------
apiVersion: v1
kind: Service
metadata:
  name: hello-svc
spec:
  selector:
    role: app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
      nodePort: 32000
  type: NodePort
---------------------------------------------------
```

ansible: deployment.yml

```
---------------------------------------------------------
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-deployment
  labels:
    role: app
spec:
  replicas: 2
  selector:
    matchLabels:
      role: app
  template:
    metadata:
      labels:
        role: app
    spec:
      containers:
      - name: app
        image: "{{ image_id }}"
        resources:
          requests:
            cpu: 10m
---------------------------------------------------------
```

Samenvattend

- **git** triggered Jenkins mbv **post-commit**
- **jenkins <- git**: config git url-repo + credentials in jenkins (plugin “github”)
- **jenkins**, **build** image:jenkins pipeline stage (plugin “docker”).
  - Jenkinsfile vanuit git
  - Dockerfile vanuit git
- **jenkins**, **test** image: jenkins pipeline stage, appname_**test**.go
- j**enkins -> hub.docker.com**: **publish** image, credentials in jenkins
- **jenkins, ansible -> k8s**: **deploy** image naar k8s cluster, ansible k8s module, config in ~jenkins/.kube/config (gekopieerd van kubernetes master)
  - playbook.yml, service.yml, deployment.yml vanuit git

Bronnen

- Maak Jenkins VM
  - https://www.chrisjmendez.com/2017/01/08/installing-jenkins-using-virtualbox-and-vagrant/
- Config Jenkins VM
  - https://www.magalix.com/blog/create-a-ci/cd-pipeline-with-kubernetes-and-jenkins
- Setup kubernetescluster
  - https://blog.exxactcorp.com/building-a-kubernetes-cluster-using-vagrant/
- Git repo met alle data
  - https://github.com/oscarbuse/cicd/