DevOps

Day 5

Date: 21.03.2025

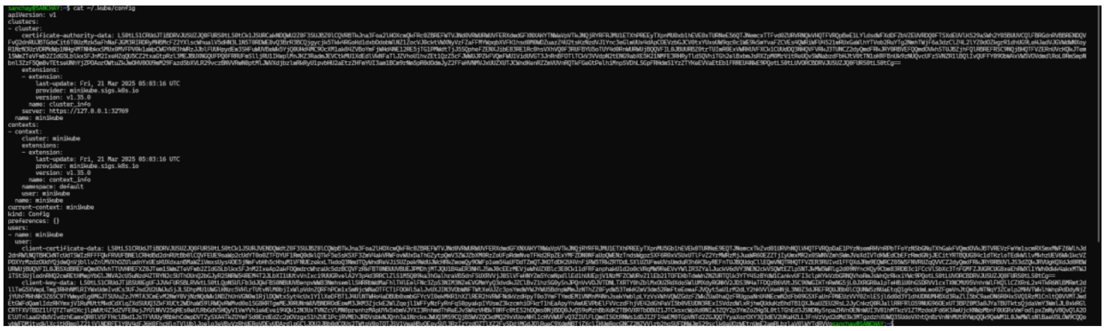
Topics Covered: Minikube, Kubernetes Deployment

Jenkins and Minikube Deployment

Deployment of the Docker Image with Kubernetes and Minikube in Jenkins

- Push the Docker Image to Docker Hub from Jenkins (Testing)
- Install Kubernetes Cloud credentials
- Create new credentials secret file from deployment.yml from GitHub
- Install Kubernetes and stages-view plugins
- Configure the script
- Deploy in Minikube

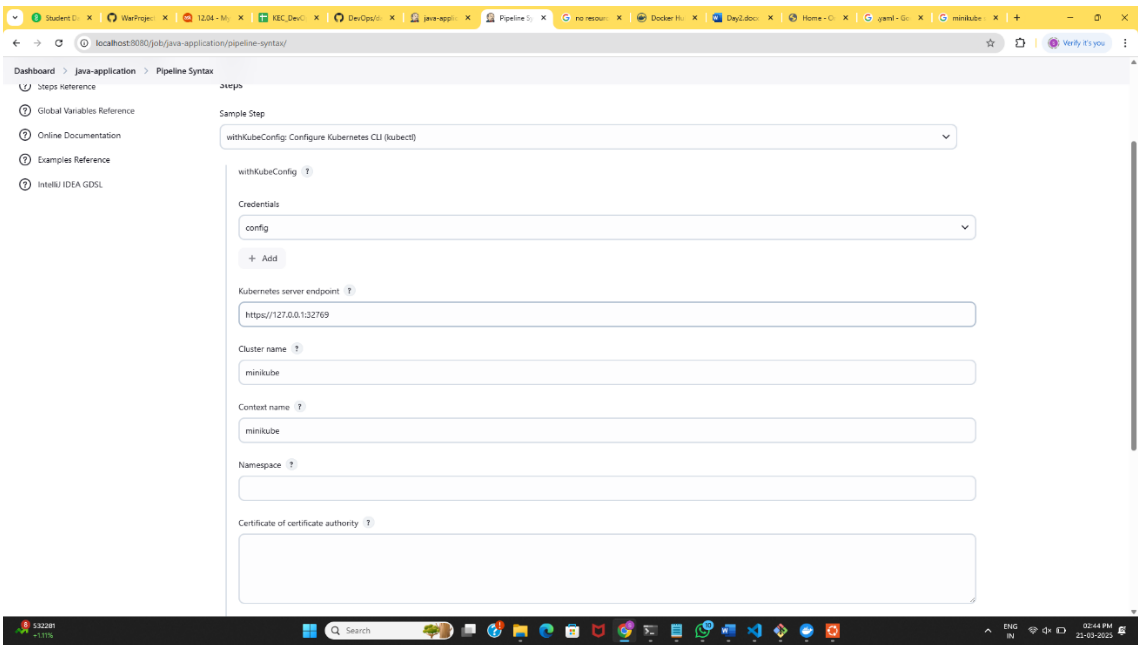
Configuring the Kubernetes config file by taking data from ca.crt



```
sanchay@SANCHAY:~$ cat ~/.kube/config
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: [illegible]
    extensions:
    - extension:
        last-update: Fri, 21 Mar 2025 05:03:16 UTC
        provider: minikube.sigs.k8s.io
        version: v1.35.0
      name: cluster_info
    server: https://127.0.0.1:32769
  name: minikube
contexts:
- context:
    cluster: minikube
    extensions:
    - extension:
        last-update: Fri, 21 Mar 2025 05:03:16 UTC
        provider: minikube.sigs.k8s.io
        version: v1.35.0
      name: context_info
    namespace: default
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: minikube
  user:
    client-certificate-data: [illegible]
    client-key-data: [illegible]
```

Config data in config

Generating the pipline syntax for Kubernetes deployment



Dashboard > java-application > Pipeline Syntax

Steps

Sample Step: withKubeConfig: Configure Kubernetes CLI (kubectl)

withKubeConfig

Credentials: config

Kubernetes server endpoint: https://127.0.0.1:32769

Cluster name: minikube

Context name: minikube

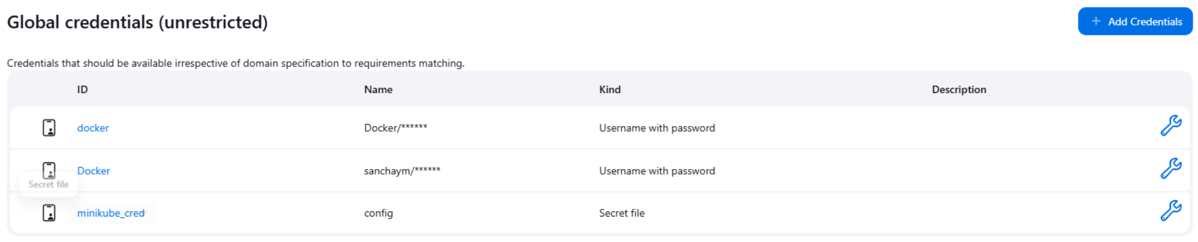
Namespace:

Certificate of certificate authority:

localhost:8080/job/java-application/pipeline-syntax/
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
+ Add

Pipeline syntax generation

Creating global credentials for deployment.yml



Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
docker	Docker/******	Username with password	
Docker	sanchaym/******	Username with password	
minikube_cred	config	Secret file	

Global minikube credentials

Deployment.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deploy
  labels:
    name: my-deploy
spec:
  replicas: 1
  selector:
    matchLabels:
      apptype: web-backend
  strategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        apptype: web-backend
    spec:
      containers:
      - name: my-web
        image: sanchaym/simplewebapp:latest
        ports:
```

```
        - containerPort: 9001
---
apiVersion: v1
kind: Service
metadata:
  name: my-service
  labels:
    app: my-service
spec:
  type: NodePort
  ports:
    - port: 9001
      targetPort: 8080
      nodePort: 30005
  selector:
    apptype: web-backend
```

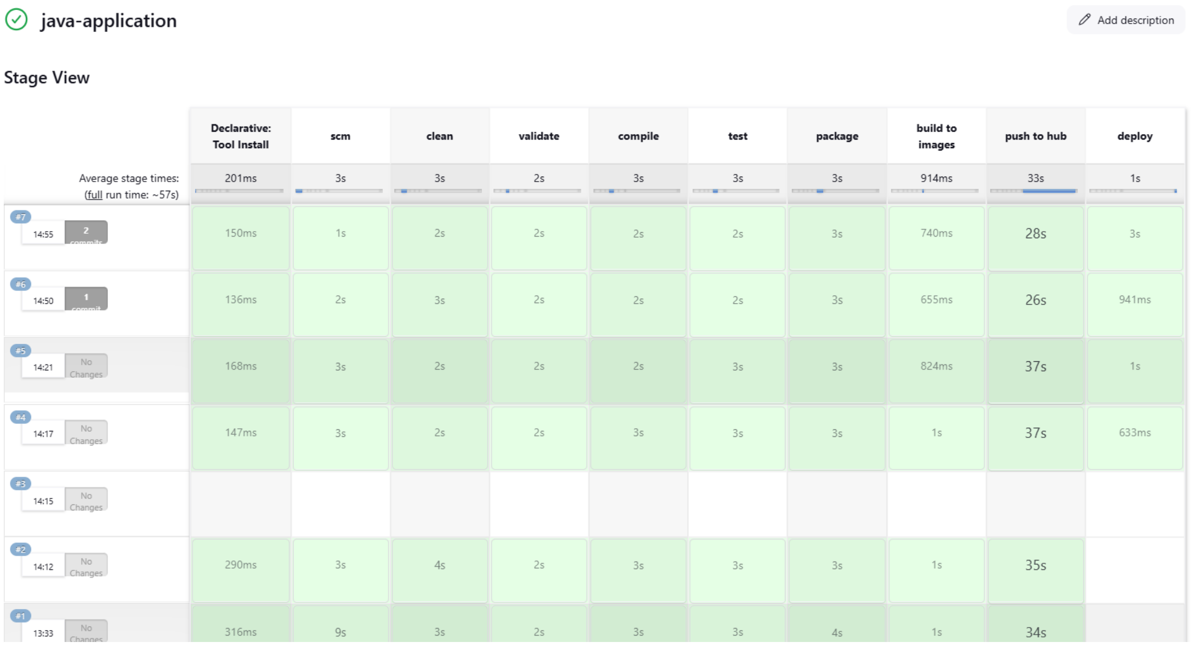
Script:

```
pipeline {
    agent any
tools {maven 'mvn'}
    stages {
        stage('scm') {
            steps {
        git 'https://github.com/Sanchay1054/WarProject.git'
            }
}
        stage('clean') {
            steps {
               sh "mvn clean"
           }
}
```

```
stage('validate') {
          steps {
            sh "mvn validate"
           }
}
stage('compile') {
          steps {
            sh "mvn compile"
}
}
stage('test') {
          steps {
            sh "mvn test"
}
}
stage('package') {
          steps {
            sh "mvn package"
}
}
stage('build to images') {
          steps {
            script{
              sh 'docker build -t sanchaym/simplewebapp .'
            }
    }
}
stage('push to hub') {
          steps {
            script{
             withDockerRegistry(credentialsId: 'Docker', url: 'https://index.docker.io/v1/') {
                sh 'docker push sanchaym/simplewebapp'

              }

            }

              }


}

stage('deploy') {

          steps {

             withKubeConfig(caCertificate: '', clusterName: 'minikube', contextName: 'minikube',
credentialsId: 'minikube_cred', namespace: '', restrictKubeConfigAccess: false, serverUrl:
'https://192.168.39.226:8443')  {

    sh 'kubectl delete all --all'

   sh 'kubectl apply -f deployment.yml --validate=false'

}

           }

}

}

}
```

Output:

Docker Image is deployed with minikube



java-application

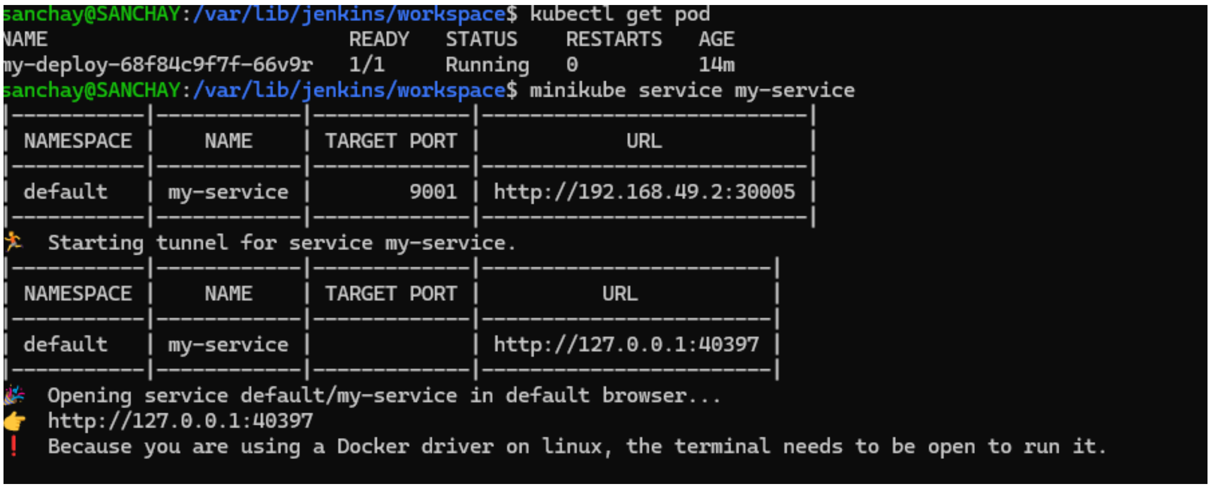
Add description

Stage View

	Declarative: Tool Install	scm	clean	validate	compile	test	package	build to images	push to hub	deploy
Average stage times: (full run time: ~57s)	201ms	3s	3s	2s	3s	3s	3s	914ms	33s	1s
#7 14:55 2 commits	150ms	1s	2s	2s	2s	2s	3s	740ms	28s	3s
#6 14:50 1 commit	136ms	2s	3s	2s	2s	2s	3s	655ms	26s	941ms
#5 14:21 No Changes	168ms	3s	2s	2s	2s	3s	3s	824ms	37s	1s
#4 14:17 No Changes	147ms	3s	2s	2s	3s	3s	3s	1s	37s	633ms
#3 14:15 No Changes										
#2 14:12 No Changes	290ms	3s	4s	2s	3s	3s	3s	1s	35s	
#1 13:33 No Changes	316ms	9s	3s	2s	3s	3s	4s	1s	34s	

Pipeline stages upto deploy

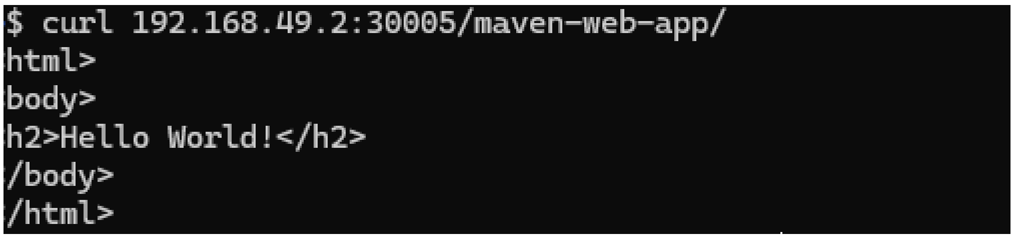
Minikube started the service my-service from deployment.yml and deployed



```
sanchay@SANCHAY:/var/lib/jenkins/workspace$ kubectl get pod
NAME                          READY   STATUS    RESTARTS   AGE
my-deploy-68f84c9f7f-66v9r    1/1     Running   0          14m
sanchay@SANCHAY:/var/lib/jenkins/workspace$ minikube service my-service
|-----------|------------|-------------|---------------------------|
| NAMESPACE |    NAME    | TARGET PORT |            URL            |
|-----------|------------|-------------|---------------------------|
| default   | my-service |        9001 | http://192.168.49.2:30005 |
|-----------|------------|-------------|---------------------------|
🏃  Starting tunnel for service my-service.
|-----------|------------|-------------|------------------------|
| NAMESPACE |    NAME    | TARGET PORT |          URL           |
|-----------|------------|-------------|------------------------|
| default   | my-service |             | http://127.0.0.1:40397 |
|-----------|------------|-------------|------------------------|
🎉  Opening service default/my-service in default browser...
👉  http://127.0.0.1:40397
❗  Because you are using a Docker driver on linux, the terminal needs to be open to run it.
```

Service

Output deployed maven project



```
$ curl 192.168.49.2:30005/maven-web-app/
html>
body>
h2>Hello World!</h2>
/body>
/html>
```

output

Terraform

Generate terraform script:

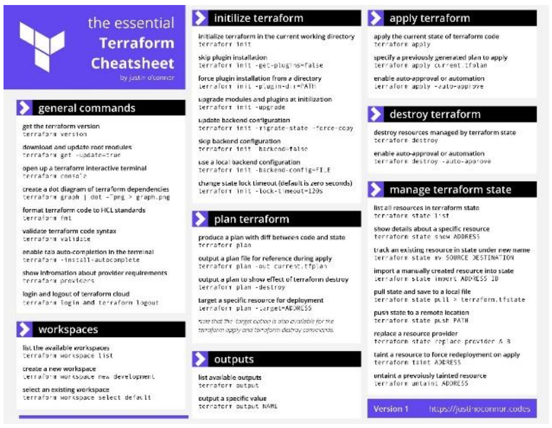
terraform {

  required_providers {

    aws = {

```
      source  = "hashicorp/aws"
      version = "~> 5.0"
    }
  }
}

# Configure the AWS Provider
provider "aws" {
  region = "us-east-1"
}

# Create a VPC
resource "aws_vpc" "example" {
  cidr_block = "10.0.0.0/16"
}

resource "aws_subnet" "pubsub" {
  vpc_id     = aws_vpc.myvpc.id
  cidr_block = "10.0.1.0/24"
  availability_zone = "us-east-1a"

  tags = {
    Name = "sn1"
  }
}

resource "aws_internet_gateway" "tfigw" {
  vpc_id = aws_vpc.myvpc.id

  tags = {
    Name = "tfigw"
  }
}

resource "aws_route_table" "tfpubrt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_internet_gateway.tfigw.id
  }

  tags = {
    Name = "tfpublicroute"
  }
}
resource "aws_route_table_association" "pubsn1" {
  subnet_id      = aws_subnet.pubsub.id
  route_table_id = aws_route_table.tfpubrt.id
}



resource "aws_eip" "tfeip" {
  domain   = "vpc"
}

resource "aws_nat_gateway" "tfnat" {
  allocation_id = aws_eip.tfeip.id
  subnet_id     = aws_subnet.pub_sub.id

  tags = {
    Name = "gw NAT"
  }
}

resource "aws_route_table" "tfprirt" {
  vpc_id = aws_vpc.myvpc.id

  route {
    cidr_block = "0.0.0.0/0"
    gateway_id = aws_nat_gateway.tfnat.id
  }

  tags = {
    Name = "tfprivateroute"
  }
}

resource "aws_security_group" "allow_tfsg" {
  name        = "allow_tfsg"
  description = "Allow TLS inbound traffic"
  vpc_id      = aws_vpc.myvpc.id

  ingress {
    description      = "HTTPS "
    from_port        = 443
    to_port          = 443
    protocol         = "tcp"
    cidr_blocks      = ["0.0.0.0/0"]
  }
   ingress {
    description      = "HTTP "
    from_port        = 80
    to_port        = 80
    protocol       = "tcp"
    cidr_blocks    = ["0.0.0.0/0"]
  }
  ingress {
    description    = "SSH"
    from_port      = 22
    to_port        = 22
    protocol       = "tcp"
    cidr_blocks    = ["0.0.0.0/0"]
  }

  egress {
    from_port      = 0
    to_port        = 0
    protocol       = "-1"
    cidr_blocks    = ["0.0.0.0/0"]
  }

  tags = {
    Name = "TfsecurityGroup"
  }
}
```

Terraform Commands:



the essential **Terraform Cheatsheet**
by justin o'connor

general commands

get the terraform version
`terraform version`

download and update root modules
`terraform get -update=true`

open up a terraform interactive terminal
`terraform console`

create a dot diagram of terraform dependencies
`terraform graph | dot -Tpng > graph.png`

format terraform code to HCL standards
`terraform fmt`

validate terraform code syntax
`terraform validate`

enable tab auto-completion in the terminal
`terraform -install-autocomplete`

show infromation about provider requirements
`terraform providers`

login and logout of terraform cloud
`terraform login` and `terraform logout`

workspaces

list the available workspaces
`terraform workspace list`

create a new workspace
`terraform workspace new development`

select an existing workspace
`terraform workspace select default`

initilize terraform

initialize terraform in the current working directory
`terraform init`

skip plugin installation
`terraform init -get-plugins=false`

force plugin installation from a directory
`terraform init -plugin-dir=PATH`

upgrade modules and plugins at initilization
`terraform init -upgrade`

update backend configuration
`terraform init -migrate-state -force-copy`

skip backend configuration
`terraform init -backend=false`

use a local backend configuration
`terraform init -backend-config=FILE`

change state lock timeout (default is zero seconds)
`terraform init -lock-timeout=120s`

plan terraform

produce a plan with diff between code and state
`terraform plan`

output a plan file for reference during apply
`terraform plan -out current.tfplan`

output a plan to show effect of terraform destroy
`terraform plan -destroy`

target a specific resource for deployment
`terraform plan -target=ADDRESS`

note that the -target option is also available for the terraform apply and terraform destroy commands.

outputs

list available outputs
`terraform output`

output a specific value
`terraform output NAME`

apply terraform

apply the current state of terraform code
`terraform apply`

specify a previously generated plan to apply
`terraform apply current.tfplan`

enable auto-approval or automation
`terraform apply -auto-approve`

destroy terraform

destroy resources managed by terraform state
`terraform destroy`

enable auto-approval or automation
`terraform destroy -auto-approve`

manage terraform state

list all resources in terraform state
`terraform state list`

show details about a specific resource
`terraform state show ADDRESS`

track an existing resource in state under new name
`terraform state mv SOURCE DESTINATION`

import a manually created resource into state
`terraform state import ADDRESS ID`

pull state and save to a local file
`terraform state pull > terraform.tfstate`

push state to a remote location
`terraform state push PATH`

replace a resource provider
`terraform state replace-provider A B`

taint a resource to force redeployment on apply
`terraform taint ADDRESS`

untaint a prevoiusly tainted resource
`terraform untaint ADDRESS`

Version 1 https://justinoconnor.codes