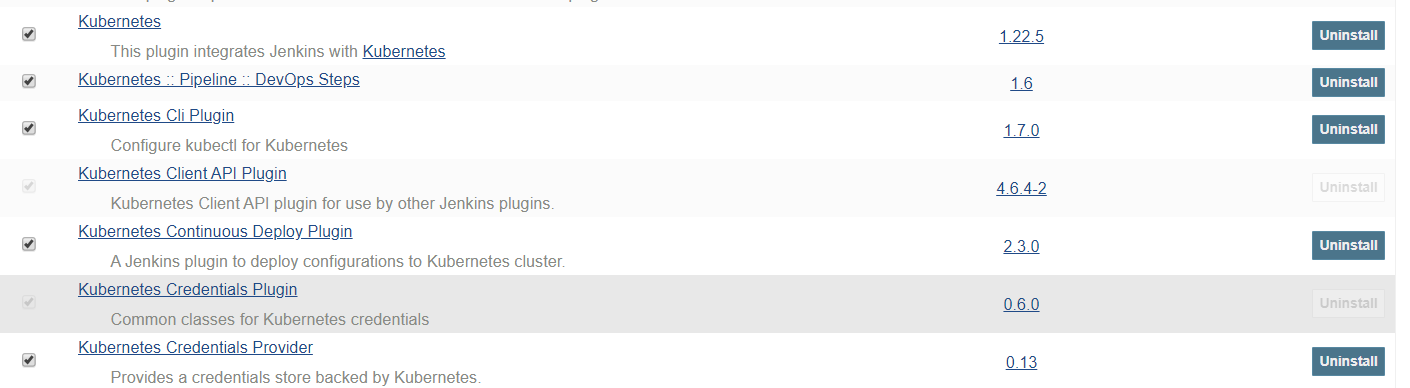
**## Setup CI/CD deployment to Kubernetes**

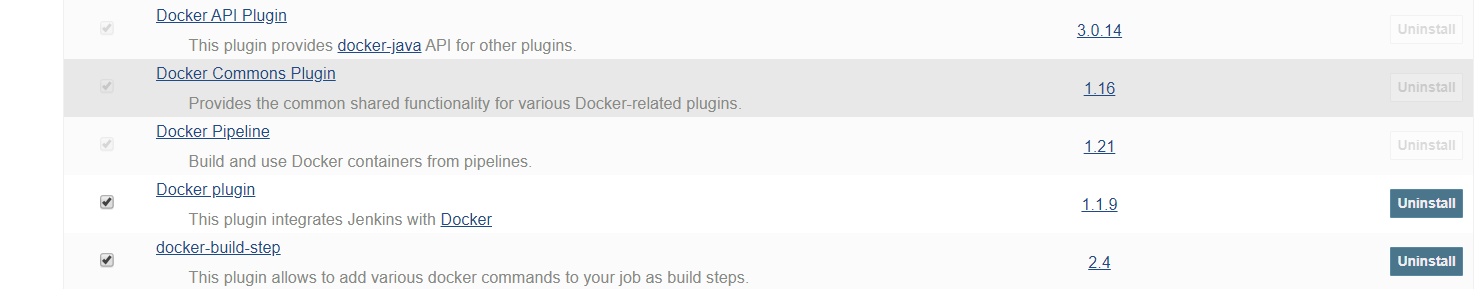
Prerequisite:

1. Install Kubernetes Cluster
2. Jenkins: Along with docker

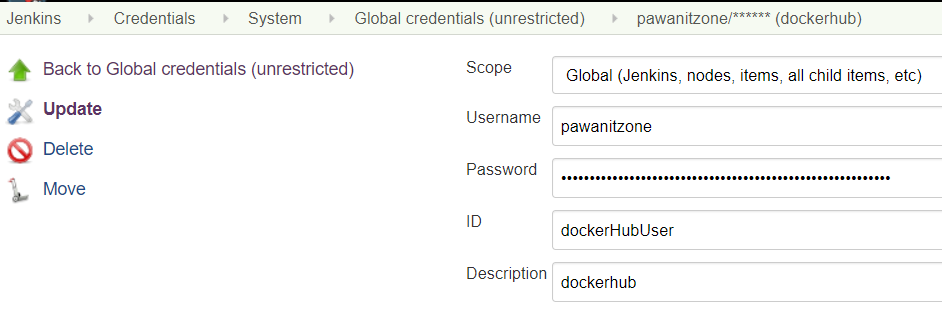
Along With: Docker command, Kubectl

Plugins: Kubernetes, docker





1. Setup Credentials to connect with DockerHub



1. Setup Credentials to connect with Kubernetes cluster and to particular namespace:
2. Create namespace serviceAccount and rolebinding to connect with Kubernetes cluster.
3. # Create a ServiceAccount named `jenkins-robot` in a given namespace.

$ kubectl -n web create serviceaccount jenkins-robot

1. # The next line gives `jenkins-robot` administator permissions for this namespace.

# \* You can make it an admin over all namespaces by creating a `ClusterRoleBinding` instead of a `RoleBinding`.

# \* You can also give it different permissions by binding it to a different `(Cluster)Role`.

$ kubectl -n web create rolebinding jenkins-robot-binding --clusterrole=cluster-admin --serviceaccount=web:jenkins-robot

NOTE: On google cloud GKE one more role required:# kubectl create clusterrolebinding default-admin --clusterrole cluster-admin --serviceaccount=project1:jenkins-robot

1. # Get the name of the token that was automatically generated for the ServiceAccount `jenkins-robot`.

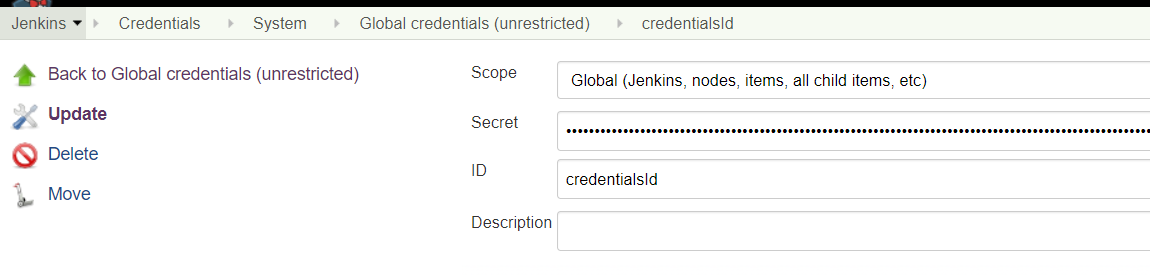
$ kubectl -n web get serviceaccount jenkins-robot -o go-template --template='{{range .secrets}}{{.name}}{{"\n"}}{{end}}'

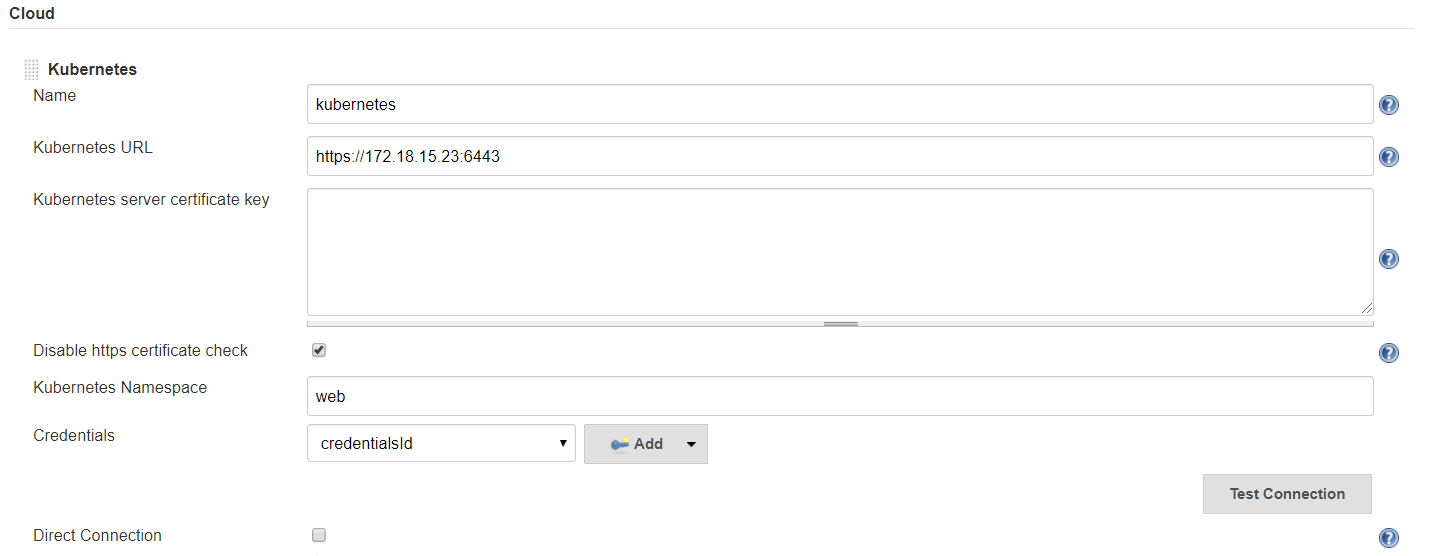
1. Retrieve the token and decode it using base64.

$kubectl -n web get secrets jenkins-robot-token-pqbdz -o go-template --template '{{index .data "token"}}' | base64 -d

|  |
| --- |
| eyJhbGciOiJSUzI1NiIsImtpZCI6IiJ9.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9uYW1lc3BhY2UiOiJ3ZWIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlY3JldC5uYW1lIjoiamVua2lucy1yb2JvdC10b2tlbi1wcWJkeiIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1hY2NvdW50Lm5hbWUiOiJqZW5raW5zLXJvYm90Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNjb3VudC9zZXJ2aWNlLWFjY291bnQudWlkIjoiMDM4ZDA0NjMtMzYyNC0xMWVhLTk0NjUtMDA1MDU2OTA3OGRjIiwic3ViIjoic3lzdGVtOnNlcnZpY2VhY2NvdW50OndlYjpqZW5raW5zLXJvYm90In0.BmoU6II460OLp2rvLf5o3P3ij13GHa1SKOatcm8iVcpMNsRd147RDKX8ahSZigTHdYIn-kH4mJoeGQOVVD-IJlmoXZLv6eMdWlzOe4HNTP5giNTgLYaSVf16rP8dkYJ93rw6ApFwXCQx4jbEXMyB5b0h7NC3vhajy9UACiUNj33E1n\_16SucjjbqRG46V1V\_JM\_a3OZEORgfRbQEH95RZY8mRPY8Nx0lX67D\_uvg0oiQ8S9kmUyFsjeXH7QwahenCA\_IOhCe\_0LC\_\_kQct2vZEaoBpRIl6nxczlGLLbyc720pQoIcU8f8maOWxYDvXn\_8ORq\_htj-45mKSASyKhNhA |

1. Setup Jenkins to connect with Kubernetes cluster
2. Manage Jenkins -> Configure System -> Add a new Cloud -> Kubernetes
3. Credentials -> Add ->





1. Now write pipeline job to setup CI/CD deployment job.
2. Create “jenkinsfile” to write pipeline to create image using dockerfile, and copy image to docker hub. And do the deployment on Kubernetes cluster

|  |
| --- |
| pipeline {  agent any  stages {  stage('Docker Build') {  steps {  sh "docker build -t **pawanitzone/hyt-http:${env.BUILD\_NUMBER}** ."  }  }  stage('Docker Push') {  steps {  withCredentials([usernamePassword(credentialsId: 'dockerHubUser', passwordVariable: 'dockerHubPassword', usernameVariable: 'dockerHubUser')]) {  sh "docker login -u ${env.dockerHubUser} -p ${env.dockerHubPassword}"  sh "docker push pawanitzone/hyt-http:${env.BUILD\_NUMBER}"  }  }  }  stage('Docker Remove Image') {  steps {  sh "docker rmi pawanitzone/hyt-http:${env.BUILD\_NUMBER}"  }  }  stage('Apply Kubernetes Files') {  steps {  withKubeConfig([credentialsId: 'credentialsId',  serverUrl: 'https://172.18.15.23:6443']) {  sh 'cat hyt-deployment.yaml | sed "s/{{BUILD\_NUMBER}}/$BUILD\_NUMBER/g" |kubectl apply -f -'  sh 'kubectl apply -f hyt-service.yaml'  }  }  }  }  } |

1. Deployment.yaml

|  |
| --- |
| apiVersion: apps/v1  kind: Deployment  metadata:  name: httpd-deployment  namespace: web  labels:  app: httpd-web  type: frontend  location: IN  environment: Production  spec:  template:  metadata:  name: httpd-pod  namespace: web  labels:  app: httpd-web  type: frontend  spec:  containers:  - name: httpd-web  image: **pawanitzone/hyt-http:{{BUILD\_NUMBER}}**  replicas: 2  selector:  matchLabels:  type: frontend |