**APPLICATION DEPLOYMENT**

**USING DOCKER-JENKINS-AWS EKS,ECR**

**DESCRIPTION:**

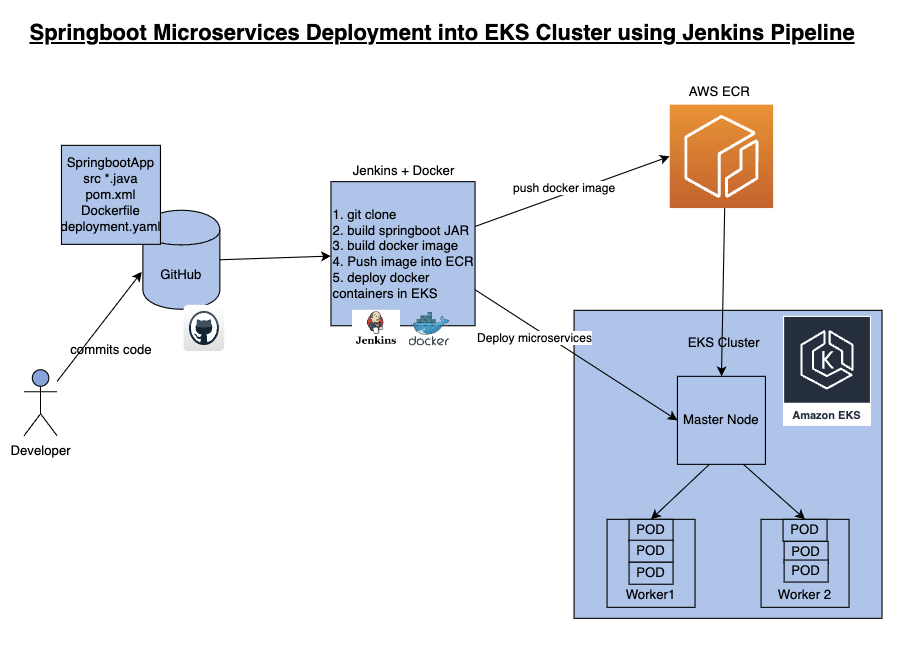
**Automate springboot microservices builds using Jenkins pipeline and Deploy into AWS EKS Cluster.**

**- Automating builds using Jenkins**

**- Automating Docker image creation**

**- Automating Docker image upload into AWS ECR**

**- Automating Docker Containers Deployments to Kubernetes Cluster**

****

**Repo Link -**

**https://github.com/yasminjeelani/EKS-CICD.git**

**Pre-requisites:**

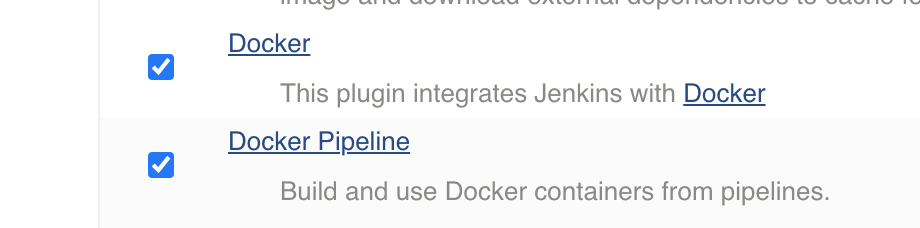
**1. Amazon EKS Cluster is setup and running.**

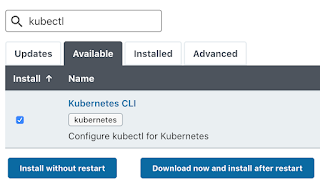
**2.** [**Create ECR repo in AWS**](https://www.cidevops.com/2020/05/how-to-setup-elastic-container-registry.html)

**3.** [**Jenkins Master is up and running**](https://www.coachdevops.com/2020/04/install-jenkins-ubuntu-1804-setup.html)

**4.** [**Docker installed on Jenkins instance**](https://www.coachdevops.com/2020/05/docker-jenkins-integration-building.html)

**5. Docker, Docker pipeline and Kubernetes CLI plug-ins are installed in Jenkins**

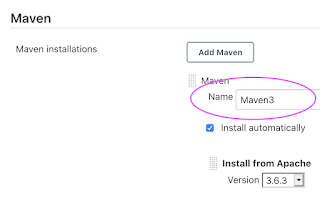
****

****

**6. Install kubectl on ec2 instance**

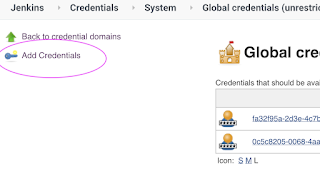
**Step # 1 - Create Maven3 variable under Global tool configuration in Jenkins**

**Make sure you create Maven3 variable under Global tool configuration.**

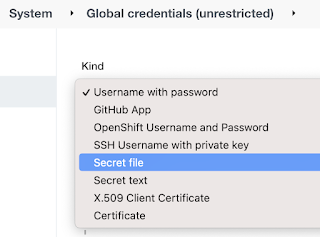
****

**Step #2 - Create Credentials for connecting to Kubernetes Cluster using kubeconfig**

**Click on Add Credentials, use Kubernetes configuration from drop down.**

****

**use secret file from drop down.**

****

**execute the below command to login as jenkins user.**

**sudo su - jenkins**

**you should see the nodes running in EKS cluster.**

**kubectl get nodes**

****

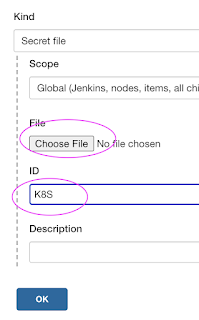
**Execute the below command to get kubeconfig info, copy the entire content of the file:**

**cat /var/lib/jenkins/.kube/config**

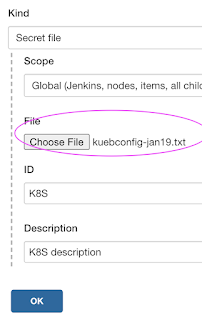
****

**Open your text editor or notepad, copy and paste the entire content and save in a file.**

**We will upload this file.**

****

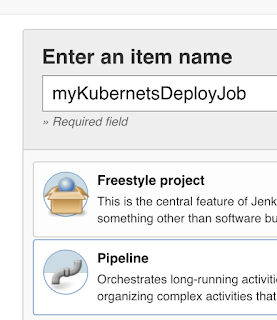
**Enter ID as K8S and choose File and upload the file and save.**

****

**Enter ID as K8S and choose enter directly and paste the above file content and save.**

**Step # 3 - Create a pipeline in Jenkins**

**Create a new pipeline job.**

****

**Step # 4 -**

**‘’’**

**pipeline {**

**tools {**

**maven 'Maven3'**

**}**

**agent any**

**environment {**

**registry = "account\_id.dkr.ecr.us-east-2.amazonaws.com/my-docker-repo"**

**}**

**stages {**

**stage('Cloning Git') {**

**steps {**

**checkout([$class: 'GitSCM', branches: [[name: '\*/main']], doGenerateSubmoduleConfigurations: false, extensions: [], submoduleCfg: [], userRemoteConfigs: [[credentialsId: '', url: 'https://github.com/yasminjeelani/EKS-CICD.git']]])**

**}**

**}**

**stage ('Build') {**

**steps {**

**sh 'mvn clean install'**

**}**

**}**

**// Building Docker images**

**stage('Building image') {**

**steps{**

**script {**

**dockerImage = docker.build registry**

**}**

**}**

**}**

**// Uploading Docker images into AWS ECR**

**stage('Pushing to ECR') {**

**steps{**

**script {**

**sh 'aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin account\_id.dkr.ecr.us-east-2.amazonaws.com'**

**sh 'docker push account\_id.dkr.ecr.us-east-2.amazonaws.com/my-docker-repo:latest'**

**}**

**}**

**}**

**stage('K8S Deploy') {**

**steps{**

**script {**

**withKubeConfig([credentialsId: 'K8S', serverUrl: '']) {**

**sh ('kubectl apply -f eks-deploy-k8s.yaml')**

**}**

**}**

**}**

**}**

**}**

**}**

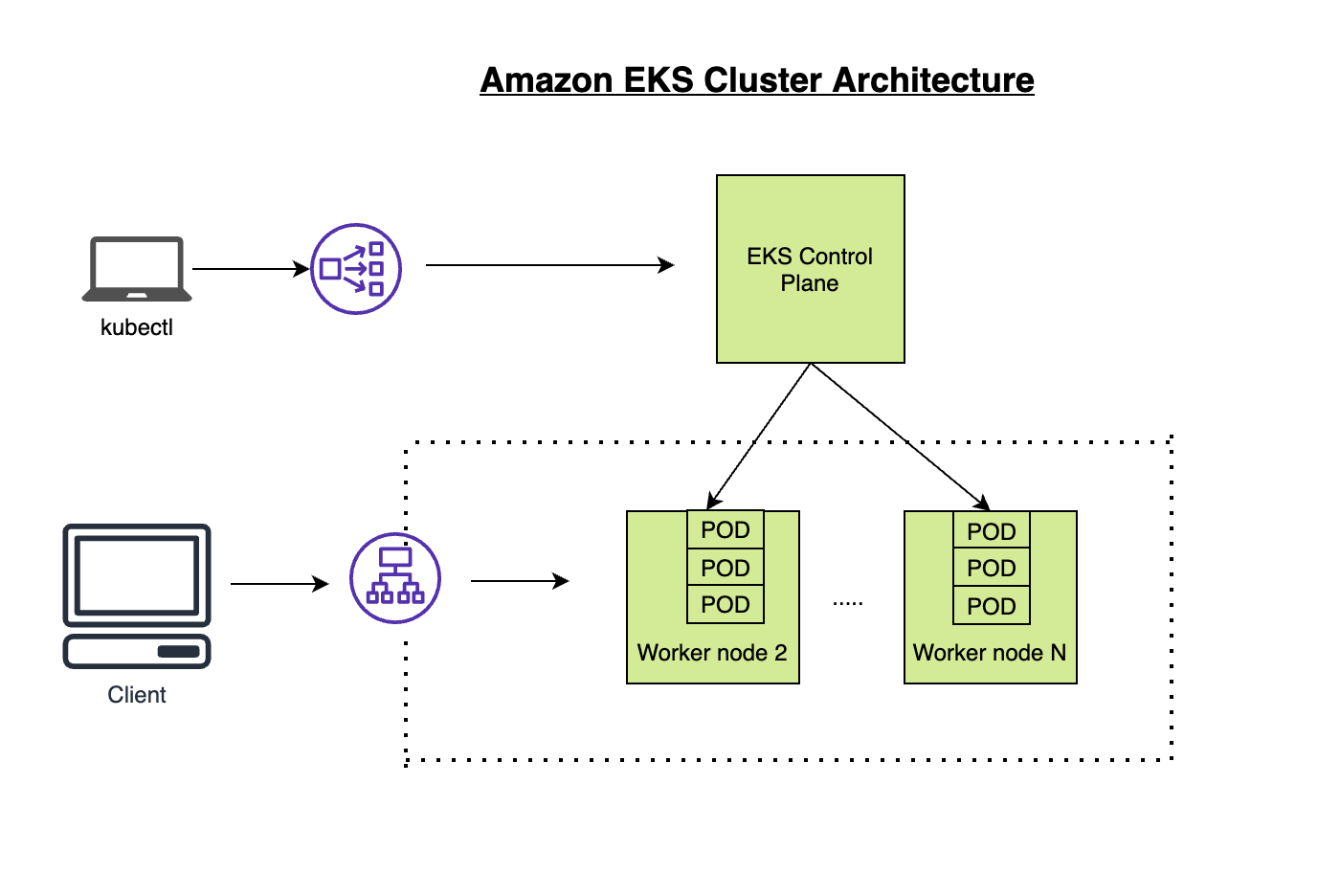
**‘’’**

**Pre-requisites:**

**1. Amazon EKS Cluster is set up and running.**

**What is Amazon EKS**

**Amazon EKS is a fully managed container orchestration service. EKS allows you to quickly deploy a production ready Kubernetes cluster in AWS, deploy and manage containerized applications more easily with a fully managed Kubernetes service.**

****

**EKS takes care of master node/control plane. We need to create worker nodes.**

**EKS cluster can be created in following ways:**

**1. AWS console**

**2. AWS CLI**

**3. eksctl command**

**4. Using Terraform**

**Steps to create EKS Cluster in ec2 :**

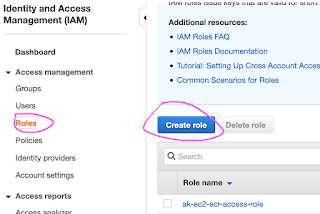
**Use Jenkins EC2 instance. Jenkins EC2 instance needs to have following configured:**

* [**Install AWS CLI**](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) **– Command line tools for working with AWS services, including Amazon EKS.**
* [**Install eksctl**](https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html) **– A command line tool for working with EKS clusters that automates many individual tasks.**
* [**Install kubectl**](https://docs.aws.amazon.com/eks/latest/userguide/install-kubectl.html) **– A command line tool for working with Kubernetes clusters.**

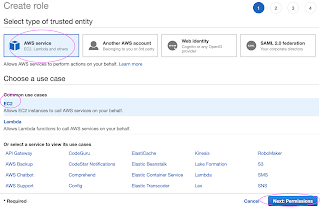
**Create IAM Role with Administrator Access**

**You need to create an IAM role with AdministratorAccess policy.**

**Go to AWS console, IAM, click on Roles. create a role**

****

**Select AWS services, Click EC2, Click on Next permissions.**

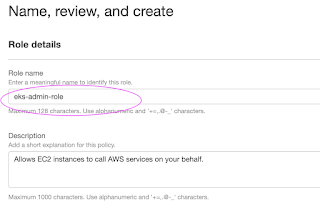
****

**Now search for AdministratorAccess policy and click**

****

**Skip on create tag.**

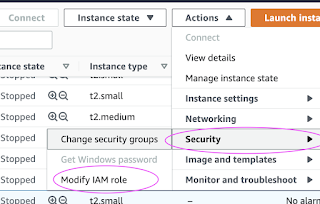
**Now give a role name and create it.**

****

**Assign the role to EC2 instance**

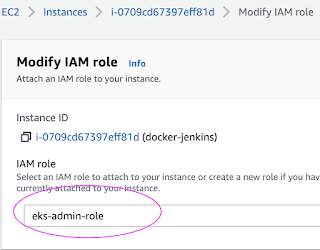
**Go to AWS console, click on EC2, select EC2 instance, Choose Security.**

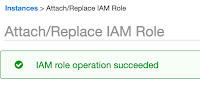
**Click on Modify IAM Role**

****

**Choose the role you have created from the dropdown.**

**Select the role and click on Apply.**

****

****

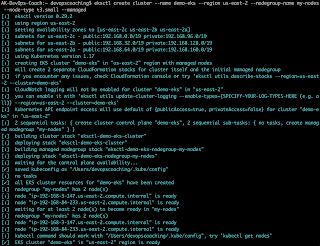
**Switch to Jenkins user:**

**“ sudo su - jenkins “**

**Create EKS Cluster with two worker nodes using eksctl:**

**“ eksctl create cluster --name demo-eks --region us-east-1 --nodegroup-name my-nodes --node-type t3.small --managed --nodes 2 “**

**Create a EKS cluster in AWS, it might take 15 to 20 mins. The *eksctl* tool uses CloudFormation under the hood, creating one stack for the EKS master control plane and another stack for the worker nodes.**

****

**eksctl get cluster --name demo-eks --region us-east-1**

**This should confirm that EKS cluster is up and running.**

**Update Kube config by entering below command:**

**aws eks update-kubeconfig --name demo-eks --region us-east-1**

****

**kubeconfig file be updated under /var/lib/jenkins/.kube folder.**

**you can view the kubeconfig file by entering the below command:**

**cat /var/lib/jenkins/.kube/config**

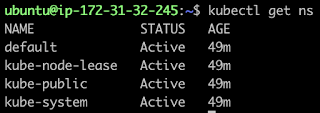
**Connect to EKS cluster using kubectl commands**

**To view the list of worker nodes as part of EKS cluster.**

**kubectl get nodes**

****

**kubectl get ns**

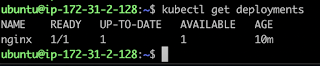
****

**Deploy Nginx on a Kubernetes Cluster  
Let us run some apps to make sure they are deployed to Kubernetes cluster. The below command will create deployment:  
  
kubectl create deployment nginx --image=nginx**

****

**View Deployments**

**kubectl get deployments**

****

**Delete EKS Cluster using eksctl**

**eksctl delete cluster --name demo-eks --region us-east-1**

****

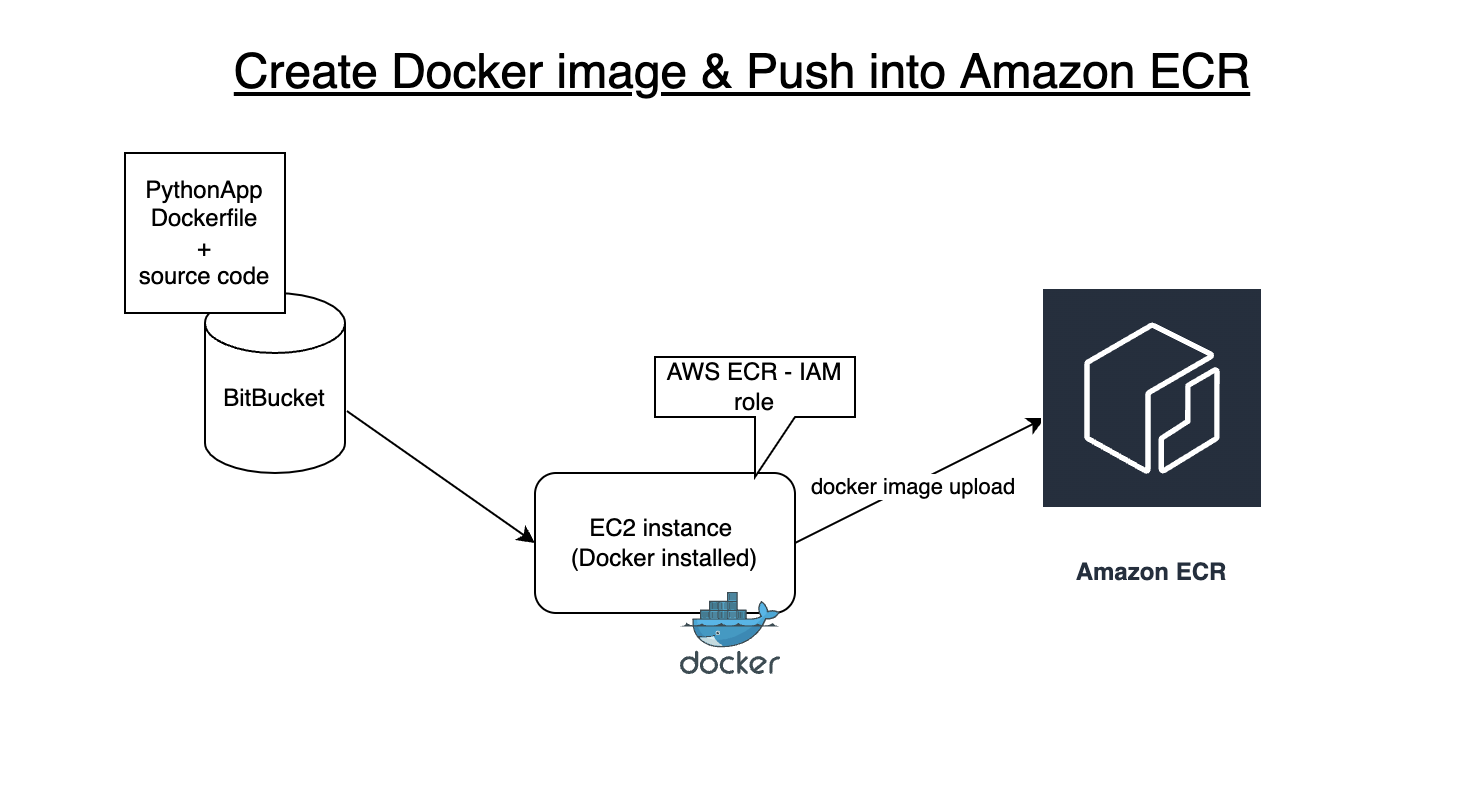
**the above command should delete the EKS cluster in AWS, it might take a few mins to clean up the cluster.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**2.** [**Create ECR repo in AWS**](https://www.cidevops.com/2020/05/how-to-setup-elastic-container-registry.html)

**What is ECR?**

Amazon ECR uses Amazon S3 for storage to make your container images highly available and accessible, allowing you to reliably deploy new containers for your applications. Amazon ECR transfers your container images over HTTPS and automatically encrypts your images at rest. Amazon ECR is integrated with Amazon Elastic Container Service (ECS), simplifying your development to production workflow.



Click on Create Repository



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**3.** [**Jenkins Master is up and running**](https://www.coachdevops.com/2020/04/install-jenkins-ubuntu-1804-setup.html)

### Install Jenkins on Ubuntu 22.0.4

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used **to implement CI/CD workflows, called pipelines**.

**Change Host Name to Jenkins**

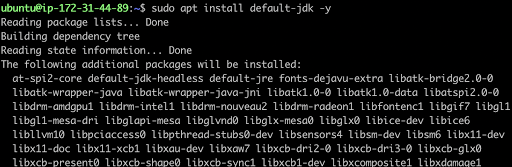
sudo hostnamectl set-hostname Jenkins

**Perform update first**

sudo apt update

**Install Java 11**

sudo apt install default-jdk -y



Once install java, enter the below command

**Verify Java Version**

java -version



**Maven Installation**

Maven is a popular build tool used for building Java applications. Please click here to learn more about Maven. You can install Maven by executing below command:

sudo apt install maven -y

mvn --version



Now lets start Jenkins installation

**Jenkins Setup**

**Add Repository key to the system**

curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \

/usr/share/keyrings/jenkins-keyring.asc > /dev/null



**Append debian package repo address to the system**

echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \

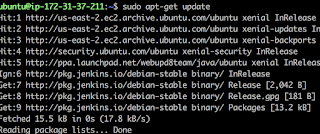
https://pkg.jenkins.io/debian binary/ | sudo tee \

/etc/apt/sources.list.d/jenkins.list > /dev/null



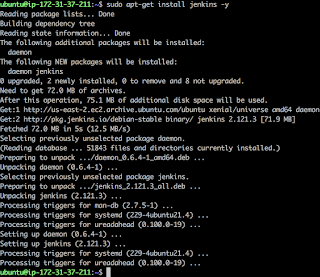
**Update Ubuntu package**

sudo apt update



**Install Jenkins**

sudo apt install jenkins -y



The above screenshot should confirm that Jenkins is successfully installed.

**Access Jenkins in web browser**

Now Go to AWS console. Click on EC2, click on running instances link. Select the checkbox of EC2 you are installing Java and Jenkins. Click on Action. Copy the value from step 4 that says --> Connect to your instance using its Public DNS:



Now go to browser. enter public dns name or public IP address with port no 8080.

http://EC2\_public\_dns\_name:8080

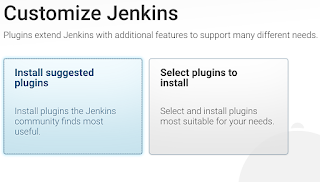
**Get the initial password from the below file**

sudo cat /var/lib/jenkins/secrets/initialAdminPassword



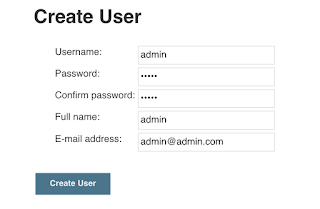
Copy the password and paste in the browser.

Then click on install suggested plug-ins.

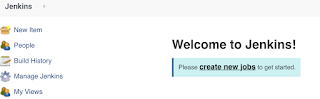


Also create user name and password.

enter everything as admin. at least user name as admin password as admin



Click on Save and Finish. Click on start using Jenkins. Now you should see a screen like below:



**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**4.** [**Docker installed on Jenkins instance**](https://www.coachdevops.com/2020/05/docker-jenkins-integration-building.html)

Every time developer makes code changes, you would want the Jenkins to automate Docker images creation and pushing into Docker registry

**Install docker**

sudo apt install docker.io -y

**Add Ubuntu user to Docker group**

sudo usermod -aG docker $USER

**Add jenkins user to Docker group**

sudo usermod -a -G docker jenkins

**Restart Jenkins service**

sudo service jenkins restart

**Reload system daemon files**

sudo systemctl daemon-reload

**Restart Docker service as well**

sudo service docker stop

sudo service docker start

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*