# **Useful Devops Installation Tips**

# **Devops Tools Installation Steps**

*How to create AKS cluster using Azure CLI | Install Kubectl on Ubuntu Instance*

**Pre-requisites:**

* Azure CLI is installed on your local machine.
* Account setup in Azure cloud.
* Install kubectl. Click here to learn how to install on Mac

## **How to Create AKS Cluster?**

**Step 1**

Make sure you are login to Azure portal first.

*az login*

enter your Microsoft credentials.

## **Step 2 — create a resource group first**

*az group create — name myResourceGroup — location westeasteurope*

**Step 3 Create AKS cluster with 2 worker nodes**

*az aks create — resource-group myResourceGroup — name myAKSCluster* ***— node-count 2*** *— enable-addons monitoring — generate-ssh-keys*

**Display Details of Cluster**

*az aks show — name myAKSCluster — resource-group myResourceGroup*

The above command will display Cluster details.

**Step 4 — Create Azure Container Registry**

Run the below command to create your own private container registry using Azure Container Registry (ACR). Make sure below red marked registry name is unique.

*az acr create — resource-group myResourceGroup — name myacrrepo4321 — sku Standard — location southcentralus*

**Connect to the cluster**

*az aks get-credentials — resource-group myResourceGroup — name myAKSCluster — overwrite-existing*

**

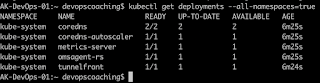
To verify the connection to your cluster, use the [kubectl get](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command to return a list of the cluster nodes.

*kubectl get nodes*

**

# List all deployments in a specific namespace

*kubectl get deployments — all-namespaces=true*

**

## **For Deploying Docker images from ACR into AKS Cluster**

When you’re using Azure Container Registry (ACR) with Azure Kubernetes Service (AKS), an authentication mechanism needs to be established.  
You can set up the AKS to ACR integration in a few simple commands with the Azure CLI or Azure PowerShell. This integration assigns the AcrPull role to the managed identity associated to the AKS Cluster.

**For Deploying Docker images from ACR into AKS Cluster**

*az aks update -n myAKSCluster -g myResourceGroup — attach-acr aksjenkins*

**

Let’s deploy some apps into AKS cluster.

**Deploy Nginx App**

*kubectl create -f* [*https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/controllers/nginx-deployment.yaml*](https://raw.githubusercontent.com/kubernetes/website/master/content/en/examples/controllers/nginx-deployment.yaml)

Once the deployment is created, use kubectl to check on the deployments by running this command:

*kubectl get deployments*

**

**Clean up the cluster**

To avoid Azure charges, you should clean up unneeded resources. When the cluster is no longer needed, use the [az group delete](https://docs.microsoft.com/en-us/cli/azure/group#az-group-delete) command to remove the resource group, container service, and all related resources.

*az group delete — name myResourceGroup — yes — no-wait*

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

# **Install kubectl on Ubuntu Instance | How to install kubectl in Ubuntu**

Kubernetes uses a command line utility called ***kubectl*** for communicating with the cluster API server. It istool for controlling **Kubernetes** clusters. **kubectl** looks for a file named config in the $HOME directory.

## **How to install Kubectl in Ubuntu instance**

**Download keys from google website**

*curl -s* [*https://packages.cloud.google.com/apt/doc/apt-key.gpg*](https://packages.cloud.google.com/apt/doc/apt-key.gpg) *| sudo apt-key add -*

**Create the below file**

*sudo touch /etc/apt/sources.list.d/kubernetes.list*

*echo “deb* [*http://apt.kubernetes.io/*](http://apt.kubernetes.io/) *kubernetes-xenial main” | sudo tee -a /etc/apt/sources.list.d/kubernetes.list*

**Update package manager**

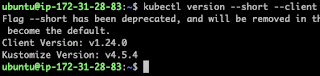
*sudo apt-get update*

**Install**

*sudo apt-get install -y kubectl*

**Verify if kubectl got installed**

*kubectl version — short — client*

**

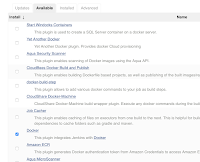
— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

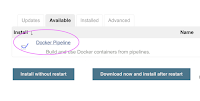
# **Docker Jenkins Integration | Building Docker Images using Jenkins | How to automate Docker Images creation using Jenkins**

Every time developer makes code changes, you would want to Jenkins to automate Docker images creation and pushing into Docker registry. Let us see how to do this.

**Pre-requisites:**Jenkins is up and running  
Docker is installed in Jenkins machine. Click [here](https://www.coachdevops.com/2019/05/install-docker-ubuntu-how-to-install.html) to learn how to install Docker.

Docker plug-in installed in Jenkins.  
Docker pipeline plug-in installed in Jenkins.





**Steps:**

Now Login to Jenkins EC2 instance, execute below commands:

**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*

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

# **Install Azure CLI in Mac OS| How to setup Azure CLI in Mac OS | How to Install Azure CLI in Apple Laptop**

The **Azure command-line** interface (**Azure CLI**) is a set of commands used to create and manage **Azure** resources. The **Azure CLI** is available across **Azure** services and is designed to get you working quickly with **Azure**, with an emphasis on automation. **Azure CLI** is Microsoft’s cross-platform command-line experience for managing Azure resources.

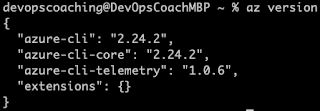
Azure CLI can be installed in Mac OS by using Homebrew:

**Run the update first**

*brew update && brew install azure-cli*

**To check the version of Azure CLI**

*az version*

**

Run the Azure CLI with the az command. To sign in, use the az login command.

*az login*

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

# **Install Azure CLI in Ubuntu 18.0.4 | How to setup Azure CLI in Ubuntu 18.0.4 | How to Install Azure CLI in Ubuntu**

The **Azure command-line** interface (**Azure CLI**) is a set of commands used to create and manage **Azure** resources. The **Azure CLI** is available across **Azure** services and is designed to get you working quickly with **Azure**, with an emphasis on automation. **Azure CLI** is Microsoft’s cross-platform command-line experience for managing Azure resources.

Azure CLI can be installed by following below steps:

**Run the update first***sudo apt-get update*

*sudo apt-get install ca-certificates curl apt-transport-https lsb-release gnupg*

*curl -sL* [*https://packages.microsoft.com/keys/microsoft.asc*](https://packages.microsoft.com/keys/microsoft.asc) *|  
gpg — dearmor |  
sudo tee /etc/apt/trusted.gpg.d/microsoft.gpg > /dev/null*

*AZ\_REPO=$(lsb\_release -cs)  
echo “deb [arch=amd64]* [*https://packages.microsoft.com/repos/azure-cli/*](https://packages.microsoft.com/repos/azure-cli/) *$AZ\_REPO main” |  
sudo tee /etc/apt/sources.list.d/azure-cli.list*

*sudo apt-get update  
sudo apt-get install azure-cli*

Check the version of Azure CLI

*az version*

Run the Azure CLI with the az command. To sign in, use the az login command.

*az login*

— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — —

# **Install Jenkins on Ubuntu 18.0.4 | Setup Jenkins on AWS EC2 Ubuntu instance | How to setup Jenkins in Ubuntu EC2 instance?**

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**.

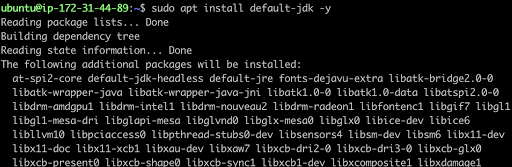
Please follow the steps to install Java, Jenkins, Maven on Ubuntu 18.0.4 instance. Jenkins, Maven are Java based applications, so we need to install Java first.

**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. You can install Maven by executing below command:

*sudo apt install maven -y*

you can type mvn — version  
you should see the below output.



Now lets start Jenkins installation

**Jenkins Setup**

**Add Repository key to the system**

*wget -q -O — https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add -*

**

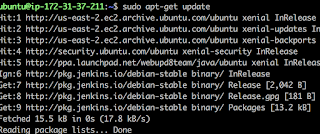
**Append debian package repo address to the system**

*echo deb* [*http://pkg.jenkins.io/debian-stable*](http://pkg.jenkins.io/debian-stable) *binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list*

**

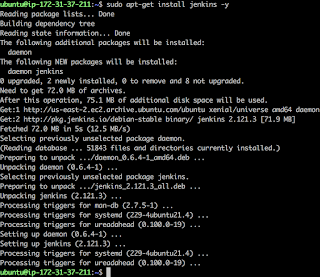
**Update Ubuntu package**

*sudo apt update*

**

**Install Jenkins**

*sudo apt install jenkins -y*

**

The above screenshot confirms that Jenkins is successfully installed.

**Access Jenkins in web browser**

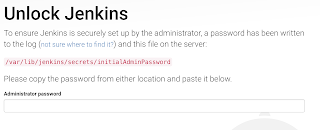
Now Go to AWS console. Click on EC2, click on running instances link. Select the checkbox of EC2 you are installing Java and Jenkins. Click on Action. Copy the value from step 4 that says → Connect to your instance using its Public DNS:



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

[http://EC2\_public\_dns\_name:8080](http://public_dns_name:8080/)

**Unlock Jenkins**You may get screen, enter the below command in Git bash( Ubuntu console)

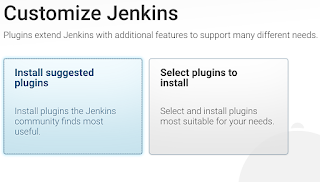


**Get the initial password from the below file**sudo cat /var/lib/jenkins/secrets/initialAdminPassword



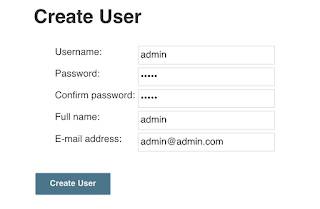
Copy the password and paste in the browser.

Then click on install suggested plug-ins.

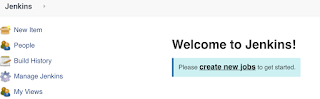


Also create user name and password.

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:



— — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — — -

# **Authenticate And Integrate Azure Container Registry from Azure Kubernetes Service**

# **Azure Container Registry**

When you’re using Azure Container Registry (ACR) with Azure Kubernetes Service (AKS), an authentication mechanism needs to be established.

To allow an AKS cluster to interact with ACR, an Azure Active Directory managed identity is used.

# **Create Resource Group**

Make sure you are logged in to Azure portal first.

az login

**You need to create a resource group first**.

az group create - name myResourceGroup - location westeurope

**Create AKS cluster with 2 worker nodes**

az aks create - resource-group myResourceGroup - name myAKSCluster - node-count 2 - enable-addons monitoring - generate-ssh-keys

az aks show - name myAKSCluster - resource-group myResourceGroup

The above commands should display Cluster existing in Azure portal

**Create Azure Container Registry**

Run the command below to create your own private container registry using Azure Container Registry (ACR).

az acr create - resource-group myResourceGroup - name myacrrepo31 - sku Standard - location westeurope

**Connect to the cluster**

az aks get-credentials - resource-group myResourceGroup - name myAKSCluster - overwrite-existing

To verify the connection to your cluster, use the [kubectl get](https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#get) command to return a list of the cluster nodes.

*kubectl get nodes*

**

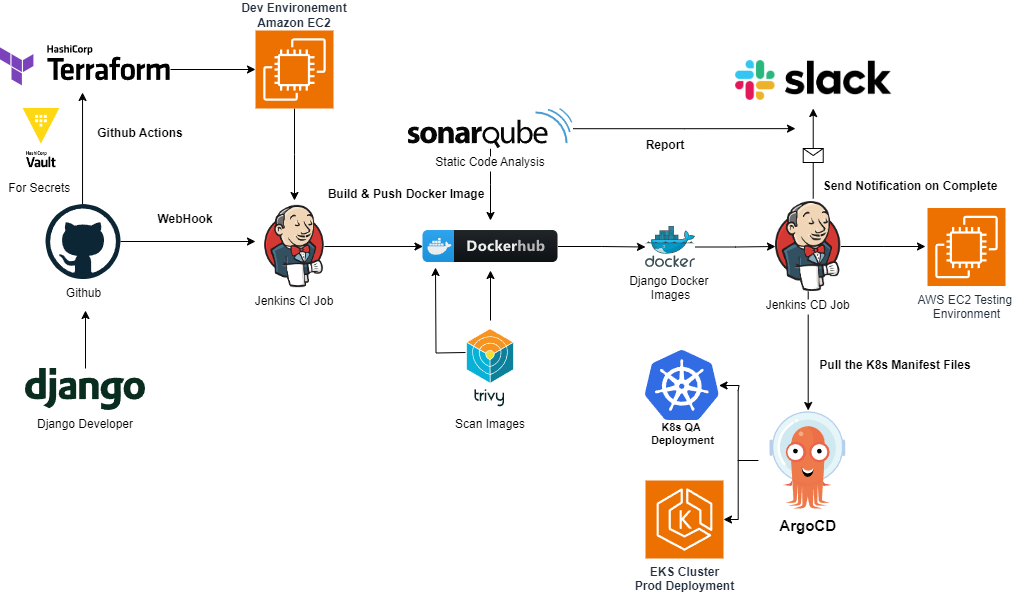
**For Deploying Docker images from ACR into AKS Cluster**

The following command allows you to authorize an existing ACR in your subscription and configures the appropriate ACRPull role for the managed identity.

az aks update -n myAKSCluster -g myResourceGroup - attach-acr <your container name>

**To Detach ACR from AKS use below command**

az aks update -n myAKSCluster -g myResourceGroup - detach-acr <your container name>

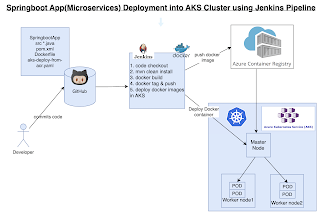


We are going to learn how to Automate build and deployment of Springboot Microservices App into Azure Kubernetes Cluster(AKS) using Jenkins pipeline.

Click [here](https://github.com/Osomudeya/aks-jenkins.git) to access code base in GitHub.

**Jenkins pipeline will:**

- Automate maven build(jar) using Jenkins  
- Automate Docker image creation  
- Automate Docker image upload into Azure container registry  
- Automate Deployments to Azure Kubernetes Cluster



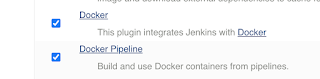
**Pre-requisites:**

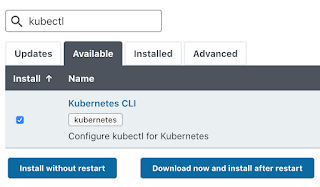
1. AKS cluster needs to be up running. You can create AKS cluster using any of one of the below options:

1. [Create AKS cluster using Azure CLI](https://medium.com/@osomudeyazudonu/how-to-create-aks-cluster-using-azure-cli-8a5d222674cf)

2. Jenkins instance is setup and running

3. Make sure to Install Docker, Docker pipeline and Kubectl CLI plug-ins are installed in Jenkins





4. Install [Docker](https://medium.com/@osomudeyazudonu/how-to-create-aks-cluster-using-azure-cli-8a5d222674cf) in Jenkins and [Jenkins have proper permission](https://medium.com/@osomudeyazudonu/how-to-create-aks-cluster-using-azure-cli-8a5d222674cf) to perform Docker builds

5. Install [Kubectl](https://medium.com/@osomudeyazudonu/how-to-create-aks-cluster-using-azure-cli-8a5d222674cf) on Jenkins instance

6. ACR is also setup in Azure cloud.

7. Make sure [AKS has pull access from ACR](https://medium.com/@osomudeyazudonu/how-to-create-aks-cluster-using-azure-cli-8a5d222674cf)

8. [Dockerfile](https://github.com/Osomudeya/aks-jenkins/blob/main/Dockerfile) created along with the application source code for springboot App.

9. Modify [K8S manifest file](https://github.com/Osomudeya/aks-jenkins/blob/main/jenkins-aks-deploy-from-acr.yaml) per acr, image name for AKS Deployment.

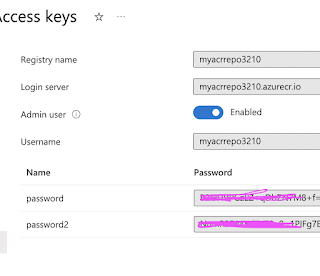
10. Install [Azure CLI](https://www.coachdevops.com/search/label/Azure%20CLI) on your local machine. (We will be creating the AKS cluster from our local machine)

**Step # 1 — Create Credentials to connect to ACR from Jenkins**

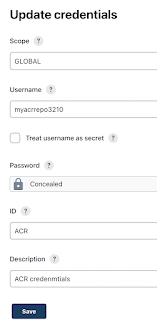
Go to Azure Portal console, go to container registry

Settings → Access keys

Get the username and password



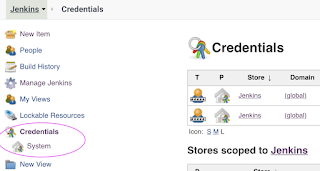
Go to Jenkins-> Manage Jenkins. Create credentials.



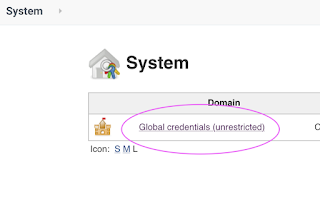
Enter ID as ACR and enter some text for description and Save

# **Step #2 — Create Credentials for connecting to AKS cluster using Kubeconfig**

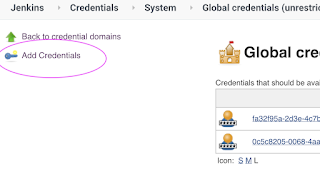
Go to Jenkins UI, click on Credentials →



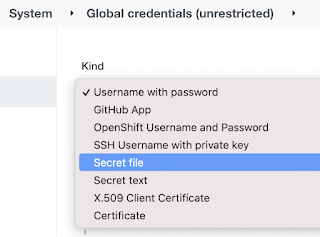
Click on Global credentials



Click on Add Credentials



use secret file from drop down.



you should see the nodes running in EKS cluster.

kubectl get nodes



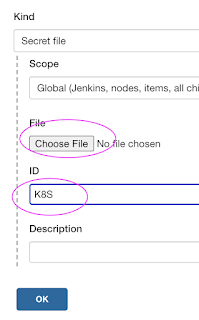
Execute the below command to get kubeconfig info, copy the entire content of the file:

cat ~/.kube/config

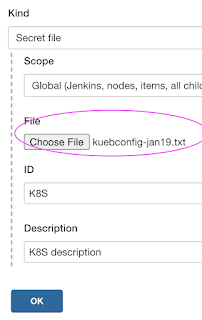


Open your text editor or notepad, copy and paste the entire content and save in a file.

We will upload this file.

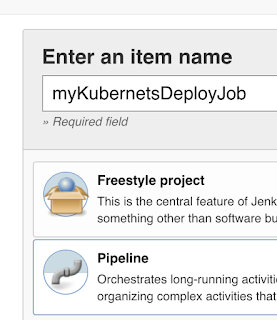


Enter ID as K8S and choose File and upload the file and save.



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

Create a new pipeline job.



**Step # 4 — Copy the pipeline code from below**Make sure you change values:

pipeline {

agent any

tools {

maven 'Maven3'

}

environment {

registryName = 'aksjenkins'

registryCredential = 'aksjenkins'

registryUrl = 'aksjenkins.azurecr.io'

dockerImage = ''

}

stages {

stage('Checkout') {

steps {

checkout([$class: 'GitSCM', branches: [[name: '\*/main']], userRemoteConfigs: [[url: 'https://github.com/Osomudeya/aks-jenkins.git']]])

}

}

stage('Build') {

steps {

sh "mvn clean install"

}

}

stage('Build Image') {

steps {

script {

dockerImage = docker.build registryName

}

}

}

stage('Upload Image to ACR') {

steps {

script {

docker.withRegistry("http://${registryUrl}", registryCredential) {

dockerImage.push()

}

}

}

}

stage ('K8S Deploy') {

steps {

script {

withKubeConfig(caCertificate: '', clusterName: '', contextName: '', credentialsId: 'K8S', namespace: '', restrictKubeConfigAccess: false, serverUrl: '') {

sh ('kubectl apply -f jenkins-aks-deploy-from-acr.yaml')

}

}

}

}

}

}

**Step # 5 — Build the pipeline**

****

**Step # 6 — Verify deployments to AKS**

*kubectl get pods*

**

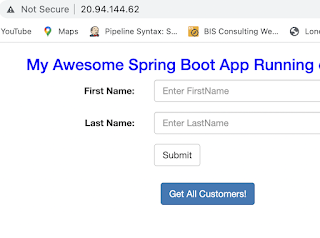
*kubectl get svc*

**

**Steps # 7 — Access Springboot App Deployed in AKS cluster**

Once deployment is successful, go to browser and enter above load balancer URL mentioned above

You should see page like below:



**Clean up the Cluster:**

To avoid charges from Azure, you should clean up unneeded resources. When the cluster is no longer needed, use the [az group delete](https://docs.microsoft.com/en-us/cli/azure/group#az-group-delete) command to remove the resource group, container service, and all related resources.

*az group delete -n MyResourceGroup*