

**TOCS Project Documentation**

**By**

**Muhammad Azaz FA20-BCS-021**

**SUBMITTED TO**

**Dr. Sardaraz**

**Date**

**29/05/2024**

**Project Documentation: Application Deployment with Jenkins on Google Cloud Platform (GCP**)

**1. Setting Up Google Cloud Platform (GCP)**

**Step 1: Create a Google Cloud Account**

* Go to [Google Cloud Platform](https://cloud.google.com/) and create an account or sign in if you already have one.
* Set up a new project by navigating to the Google Cloud Console and selecting "New Project".

**Step 2: Enable Billing**

* Ensure that billing is enabled for your Google Cloud project.

**Step 3: Enable Required APIs**

* Enable the Compute Engine API, Cloud Build API, and Container Registry API. You can do this from the API Library in the Google Cloud Console.

**Step 4: Set Up Google Cloud SDK**

* Install the Google Cloud SDK on your local machine. Follow the installation guide here.
* Initialize the SDK with your account and set the project:

gcloud init

**Step 5: Create a Compute Engine VM**

* Navigate to the Compute Engine section in the Google Cloud Console.
* Click on "Create Instance".
* Configure the instance settings (name, zone, machine type, etc.).
* Under "Management, security, disks, networking, sole tenancy", go to the "Security" tab and enable "Allow HTTP traffic" and "Allow HTTPS traffic".
* Click "Create" to launch the VM.

**2. Jenkins Installation and Configuration**

**Step 1: Access the VM via SSH**

* Open your VM instance in the Google Cloud Console and click "SSH" to open a terminal to the VM.

**Step 2: Install Jenkins**

* Update the package index:

sudo apt-get update

* Install Java (Jenkins requires Java):

sudo apt-get install openjdk-11-jdk

* Add the Jenkins repository and import the GPG key:

wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | sudo apt-key add - sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

* Install Jenkins:

sudo apt-get update sudo apt-get install jenkins

* Start Jenkins:

sudo systemctl start jenkins

* Enable Jenkins to start on boot:

sudo systemctl enable jenkins

**Step 3: Configure Jenkins**

* Open a web browser and navigate to http://[EXTERNAL\_IP]:8080, replacing [EXTERNAL\_IP] with your VM's external IP address.
* Retrieve the initial admin password:

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

* Complete the setup wizard, install suggested plugins, and create an admin user.

**3. Adding Credentials to Jenkins**

**Step 1: Access Jenkins Dashboard**

* Navigate to the Jenkins Dashboard.

**Step 2: Add Credentials**

* Go to "Manage Jenkins" -> "Manage Credentials".
* Click on "(global)" -> "Add Credentials".
* Choose "SSH Username with private key" from the "Kind" dropdown.
* Fill in the details (ID, Description, Username). For "Private Key", select "Enter directly" and paste the private key content.
* Save the credentials.

**4. Jenkins Job Creation with Detailed Descriptions**

**Step 1: Install Necessary Jenkins Plugins**

* Go to "Manage Jenkins" -> "Manage Plugins".
* Install plugins: GitHub, Docker, Google Cloud SDK.

**Step 2: Create a New Jenkins Job**

* Click on "New Item" and enter a name for the job (e.g., newproject), then select "Freestyle project" and click "OK".

**Step 3: Configure Source Code Management**

* Under "Source Code Management", select "Git" and enter the repository URL (e.g., https://github.com/muhammadazaz021/mytocsprojecttwo).

**Step 4: Add Build Steps**

**Build Docker Image**

* Add a build step to build the Docker image:

echo "Building Docker image..." docker build -t gcr.io/[YOUR\_PROJECT\_ID]/my-python-app .

**Push Docker Image to Google Container Registry**

* Add a build step to push the Docker image to Google Container Registry:

echo "Pushing Docker image to GCR..." docker push gcr.io/[YOUR\_PROJECT\_ID]/my-python-app

**Deploy Docker Container to Compute Engine**

* Add a build step to deploy the Docker container to Compute Engine:

echo "Deploying Docker container to Compute Engine..." gcloud compute ssh azaazmm --zone us-central1-a --command "sudo docker pull gcr.io/[YOUR\_PROJECT\_ID]/my-python-app && sudo docker run -d -p 80:80 gcr.io/[YOUR\_PROJECT\_ID]/my-python-app"

**Step 5: Save and Run the Jenkins Job**

* Save the job configuration.
* Run the job manually to test the setup.

**5. Creating a Service Account**

**Step 1: Create a Service Account**

1. Go to the IAM & Admin section in the Google Cloud Console.
2. Click "Service Accounts" from the left-hand menu.
3. Click the "Create Service Account" button.
4. Enter the Service Account name (e.g., **jenkins-deploy**) and description, then click "Create".

**Step 2: Assign Roles to the Service Account**

1. In the "Grant this service account access to the project" section, add the following roles:
   * Compute Admin
   * Storage Admin
   * Viewer
2. Click "Continue".

**Step 3: Create and Download a JSON Key**

1. In the "Grant users access to this service account" section, simply click "Done".
2. Click on the newly created service account to open its details.
3. Navigate to the "Keys" tab.
4. Click "Add Key" -> "Create New Key".
5. Choose JSON and click "Create". This will download a JSON key file to your local machine.

**Step 4: Secure the JSON Key**

* Store the JSON key file in a secure location as it contains sensitive information.

**Step 5: Add Service Account to Jenkins**

1. Navigate to Jenkins Dashboard -> Manage Jenkins -> Manage Credentials.
2. Select the appropriate domain (e.g., global) and click "Add Credentials".
3. Choose "Google Service Account from private key" from the "Kind" dropdown.
4. Select "JSON file" and upload the JSON key file you downloaded earlier.
5. Provide a meaningful ID and description, then save.

**6. Setting Up a Cloud Build Trigger**

**Step 1: Create a Cloud Build Trigger**

1. Go to the Cloud Build section in the Google Cloud Console.
2. Click "Triggers" and then "Create Trigger".

**Step 2: Configure Trigger**

1. Name your trigger (e.g., **github-trigger**).
2. For Event, select "Push to a branch".
3. For Source, select "GitHub (Cloud Build GitHub App)" and connect your GitHub repository.
4. Choose the repository and the branch you want to trigger builds from (e.g., **main**).

**Step 3: Specify Build Configuration**

1. In the "Configuration" section, choose "Cloud Build configuration file (yaml or json)".
2. Specify the location of your **cloudbuild.yaml** file in the repository (e.g., **/cloudbuild.yaml**).

**Example cloudbuild.yaml**

Create a **cloudbuild.yaml** file in your repository:

steps: - name: 'gcr.io/cloud-builders/docker' args: ['build', '-t', 'gcr.io/[YOUR\_PROJECT\_ID]/my-python-app', '.'] - name: 'gcr.io/cloud-builders/docker' args: ['push', 'gcr.io/[YOUR\_PROJECT\_ID]/my-python-app'] - name: 'gcr.io/cloud-builders/gcloud' args: ['compute', 'ssh', 'YOUR\_INSTANCE\_NAME', '--zone', 'YOUR\_ZONE', '--command', 'sudo docker pull gcr.io/[YOUR\_PROJECT\_ID]/my-python-app && sudo docker run -d -p 80:80 gcr.io/[YOUR\_PROJECT\_ID]/my-python-app']

**Step 4: Create the Trigger**

* Click "Create" to finalize and create the trigger.

**7. Troubleshooting Steps**

**Error: Docker Command Not Found**

* Ensure Docker is installed on the VM:

sudo apt-get install docker.io

**Error: Insufficient Authentication Scopes**

* Add required scopes to the instance:

gcloud compute instances stop azaazmm --zone=us-central1-a gcloud compute instances set-service-account azaazmm --zone=us-central1-a --scopes=https://www.googleapis.com/auth/cloud-platform gcloud compute instances start azaazmm --zone=us-central1-a

**Error: SSH Key Issues**

* Generate an SSH key and add it to the instance:

ssh-keygen -t rsa -f ~/.ssh/google\_compute\_engine -C "YOUR\_USERNAME" -N "" gcloud compute instances add-metadata azaazmm --zone=us-central1-a --metadata=ssh-keys="YOUR\_USERNAME:$(cat ~/.ssh/google\_compute\_engine.pub)"

**Error: Permissions Issues**

* Ensure the service account has the necessary IAM roles:

gcloud projects add-iam-policy-binding YOUR\_PROJECT\_ID --member=serviceAccount:YOUR\_SERVICE\_ACCOUNT\_EMAIL --role=roles/compute.instanceAdmin.v1 gcloud projects add-iam-policy-binding YOUR\_PROJECT\_ID --member=serviceAccount:YOUR\_SERVICE\_ACCOUNT\_EMAIL --role=roles/compute.osAdminLogin gcloud projects add-iam-policy-binding YOUR\_PROJECT\_ID --member=serviceAccount:YOUR\_SERVICE\_ACCOUNT\_EMAIL --role=roles/compute.osLogin gcloud projects add-iam-policy-binding YOUR\_PROJECT\_ID --member=serviceAccount:YOUR\_SERVICE\_ACCOUNT\_EMAIL --role=roles/iam.serviceAccountUser

**Running the code on Jenkins:**

