Hi Team,

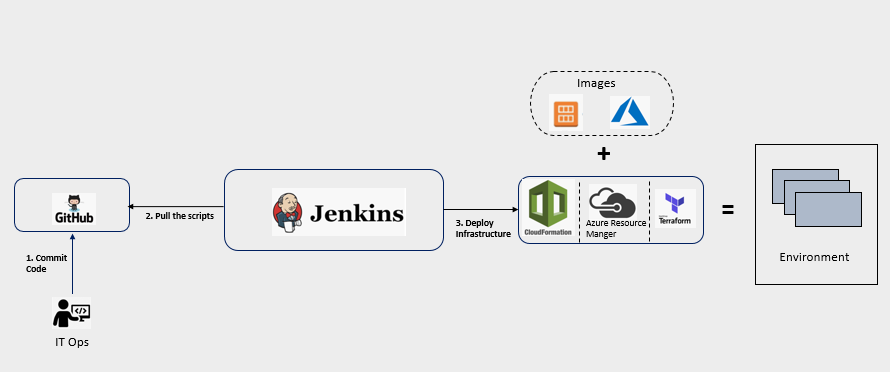
Thanks for giving me an opportunity to overcome the above challenging task, I have Pick up Azure as Cloud Provider to Use Virtual machine service in Azure and created App registration for API permission to integrate Grafana.

I have basic knowledge on ELK stake so couldn’t create the stake Automation where I have good knowledge on Azure ELK stake, where I worked for a customer in extracting web logs and application insights for ML project.

I couldn’t perform the authorized steps in short time so I have made small setup using Ansible , Jenkins and Terraform to setup the three tier Application where there is no database involved if you find my inputs are correct give me some more time so that I can modify the Automation script in order to implement an application server and SQL database so that both can talk and give the better insights.

High Level Architecture - Environment Creation

As per the below design we will configure our Automation Environment in Azure



Prerequisites

* Register an account in Git-Hub. < [https://github.com](https://github.com/)>
* Create a Linux VM in Azure Subscription
* Install and Configure Jenkins
* Configure a load balancer to access your Jenkins application through the browser.

**Installation Jenkins in Ubuntu OS**

* Install Java.

Since Jenkins is a Java application, the first step is to install Java. Update the package index and install the Java 8 OpenJDK package with the following commands:

sudo apt update

sudo apt install openjdk-8-jdk

Note:-The current version of Jenkins does not support Java 10 (and Java 11) yet. If you have multiple versions of Java installed on your machine [make sure Java 8 is should be default Java version](https://linuxize.com/post/install-java-on-ubuntu-18-04/).

* Add the Jenkins Linux repository

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

Note- Import the GPG keys of the Jenkins repository using the following **“[wget](https://linuxize.com/post/wget-command-examples/)”** command. The command above should output OK which means that the key has been successfully imported and packages from this repository will be considered trusted.

* Next, add the Jenkins repository to the system with:

sudo sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'

* Install Jenkins.

sudo apt update

sudo apt install jenkins

Note- Once the Jenkins repository is enabled, update the apt package list and install the latest version of Jenkins by typing: Jenkins service will automatically start after the installation process is complete. You can verify Jenkin’s service by the below command.

systemctl status Jenkins

sudo /etc/init.d/jenkins restart

You should see the output similar to below

Output

● jenkins.service - LSB: Start Jenkins at boot time

Loaded: loaded (/etc/init.d/jenkins; generated)

Active: active (exited) since Wed 2018-08-22 13:03:08 PDT; 2min 16s ago

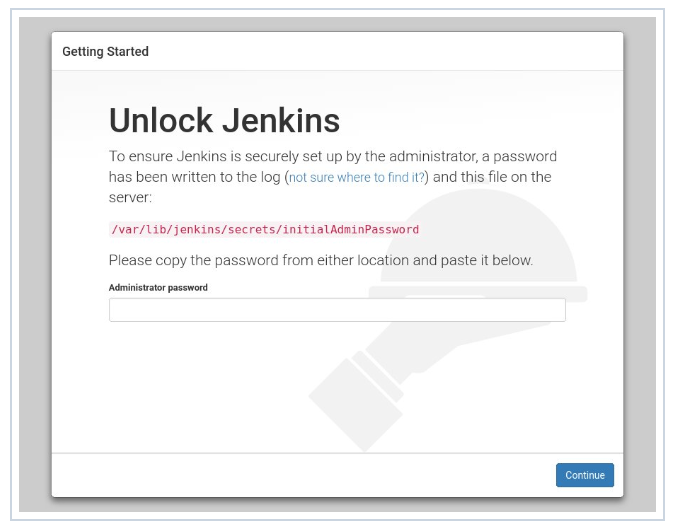
Docs: man:systemd-sysv-generator(8)

Tasks: 0 (limit: 2319)

CGroup: /system.slice/jenkins.service

* Login and [configure Jenkins](https://linuxize.com/post/how-to-install-jenkins-on-ubuntu-18-04/#setting-up-jenkins) Application after first time Installation

To set up your new Jenkins installation, open your browser, type your domain or IP address followed by port 8080, http://<your public IP>:8080 and screen similar to the following will be displayed:



During the installation, the Jenkins installer creates an initial 32-character long alphanumeric password. Use the following command to see the password on your terminal:

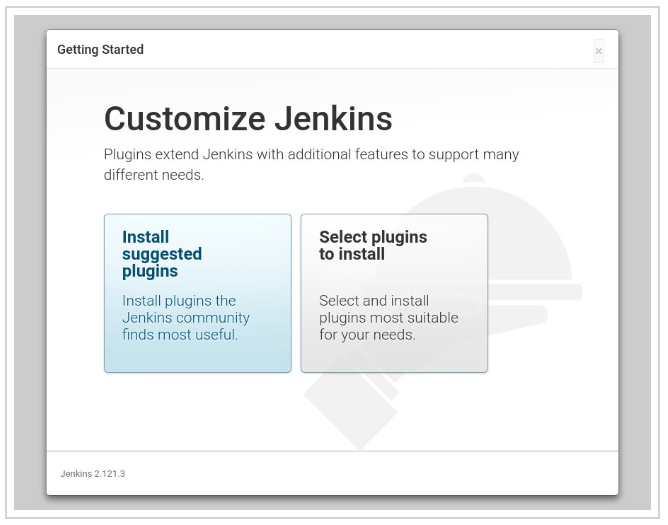
sudo cat /var/lib/jenkins/secrets/initialAdminPassword

Output should be like below:

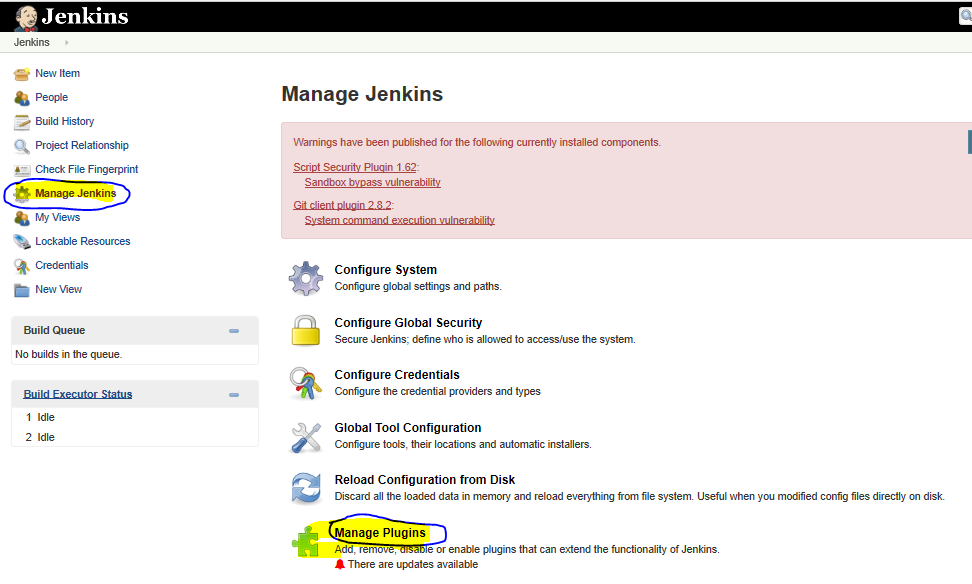
2115173b548f4e99a203ee99a8732a32

Copy the password from your terminal, paste it into the Administrator password field and click Continue to create Jenkin’s admin user.

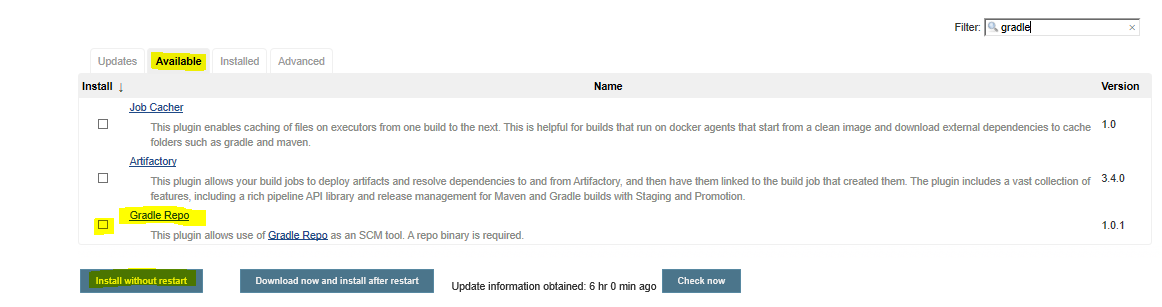
Jenkins can integrate with the various types of open source tools. Only you have to install the respective plugins. Initially we will install the suggested plugins. Latter we can install the required plugins.



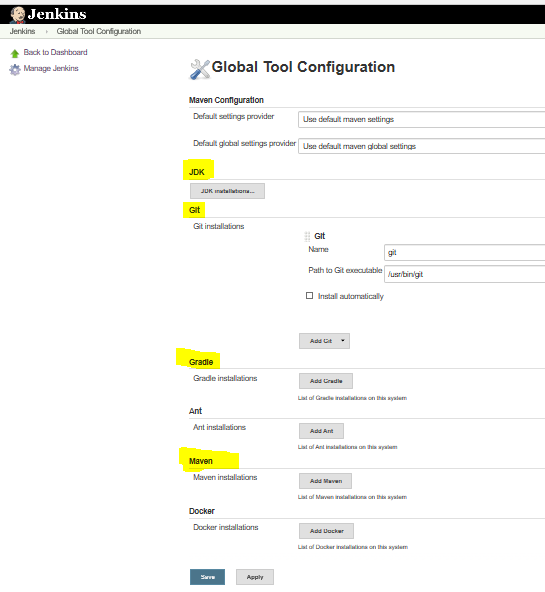
For this Steps we have to install the git and java plugins. Go to **Manage Jenkins** – **Mange Plugin** and search the plugins.



In-below I am searching for Gradle plugin and it is clearly showing Gradle has not installed. If it was installed it should show in **Installed Tab**.



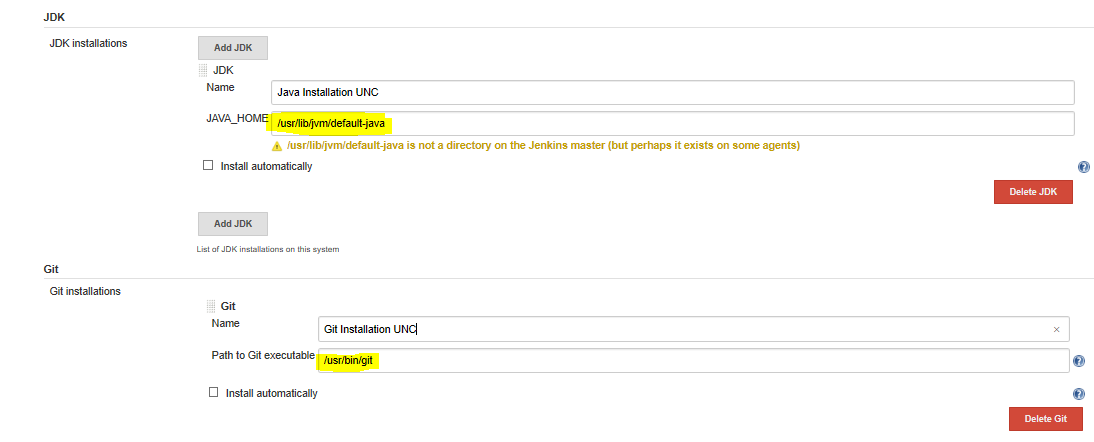
Once required plugins are installed, go to **Jenkins Global Tool Configuration**. In below screen shot **JDK** & **Git** are showing, it mean plugin has been successfully installed. Here important part is, update plug-in’s installation UNC path on **Jenkins Global Configuration tool**.



If you configure Jenkins in Windows Server then installation path will be C:\Program File\<Application Name>

Java and Git installation UNC path for Linux VM

Java Installation path is - **/usr/lib/jvm/default-java** & Git Installation path is - **/usr/bin/git.**

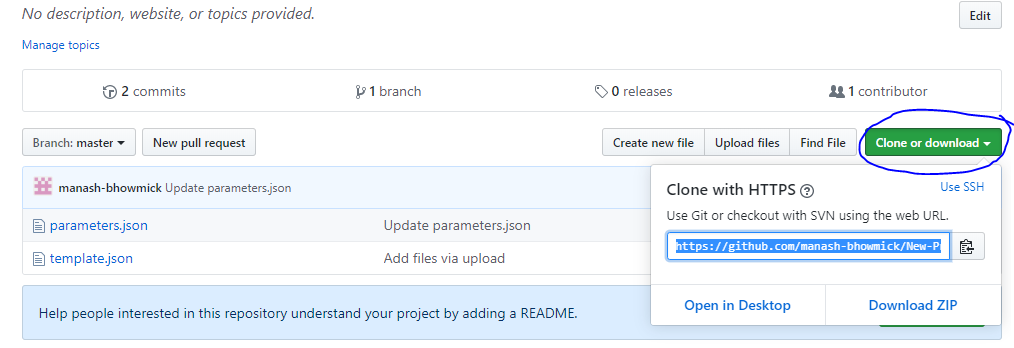


Note:-For this Automation environment we will use Groovy Script with ARM template.

So our next steps will be configuring Jenkins pipe line using ARM template.

**Prerequisites for configure the Pipe-Line in Jenkins**

* Create an Azure Service Principal & configure it in Jenkins Server for authentication from Jenkins to Azure.
* Configure a sample ARM VM Template to generate **parameters.json** and **template.json** file. Download it local system.
* Upload the **parameters.json** and **template.json** files to the Github Repository and update the repository URL in the script.

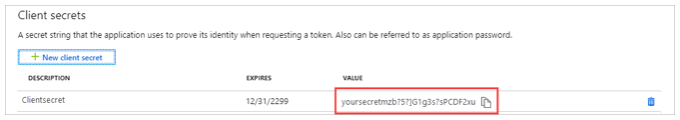
****

**Create an Azure Service Principal**

### Create a service principal that uses a client secret credential

In this section, you register your application using the Azure portal, which creates the service principal object in your Azure AD tenant. In this example, the service principal will be created with a client secret credential, but the portal also supports X509 certificate-based credentials.

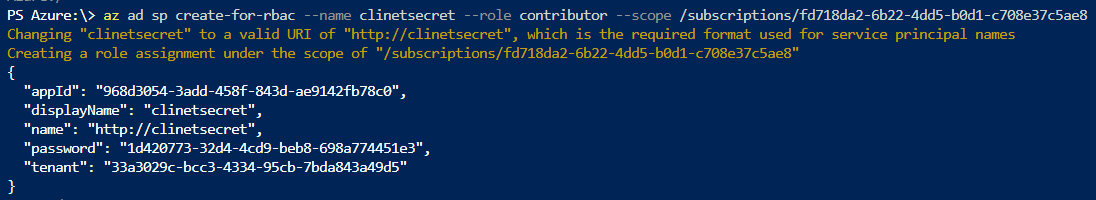
1. Sign in to the [Azure portal](https://portal.azure.com/) using your Azure account.
2. Select **Azure Active Directory** > **App registrations** > **New registration**.
3. Provide a **name** for the application.
4. Select the appropriate **Supported account types**.
5. Under **Redirect URI**, select **Web** as the application type, and (optionally) specify a redirect URI if your application requires it.
6. After setting the values, select **Register**. The application registration is created and the **Overview** page is presented.
7. Copy the **Application ID** for use in your application code. This value is also referred to as the Client ID.
8. To generate a client secret, select the **Certificates & secrets** page. Select **New client secret**.
9. Provide a **description** for the secret, and an **expires** duration.
10. When done, select **Add**.
11. The value of the secret is displayed. Copy and save this value in another location, because you can't retrieve it later. You provide the secret with the Application ID in your client application during service principal sign-in.

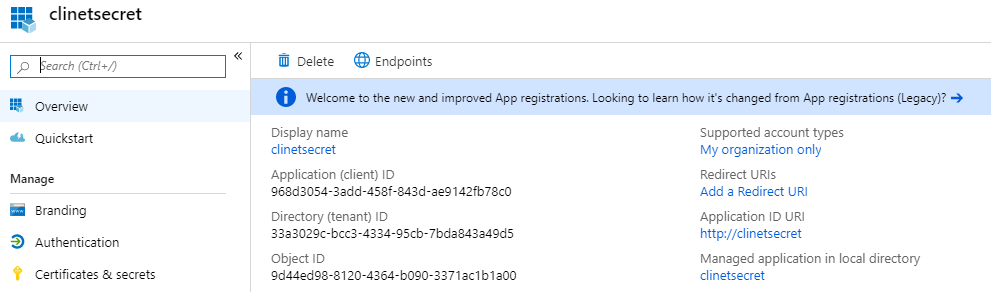


You can also register your app through below Azure CLI command.

az ad sp create-for-rbac --name <Provide Any Name> --role contributor --scope /subscriptions/<Provide your Subscriptions ID>

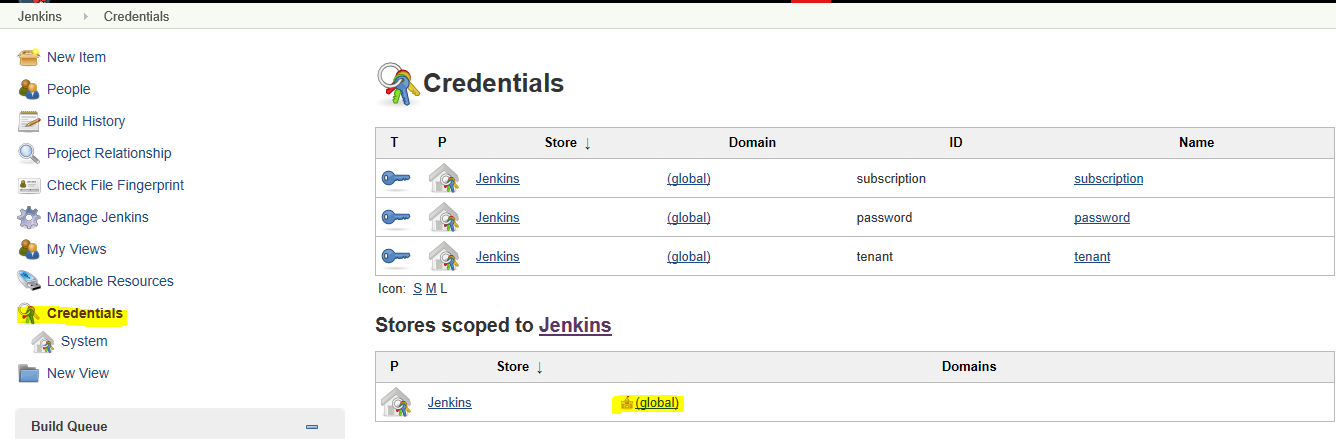
Output should be like below.



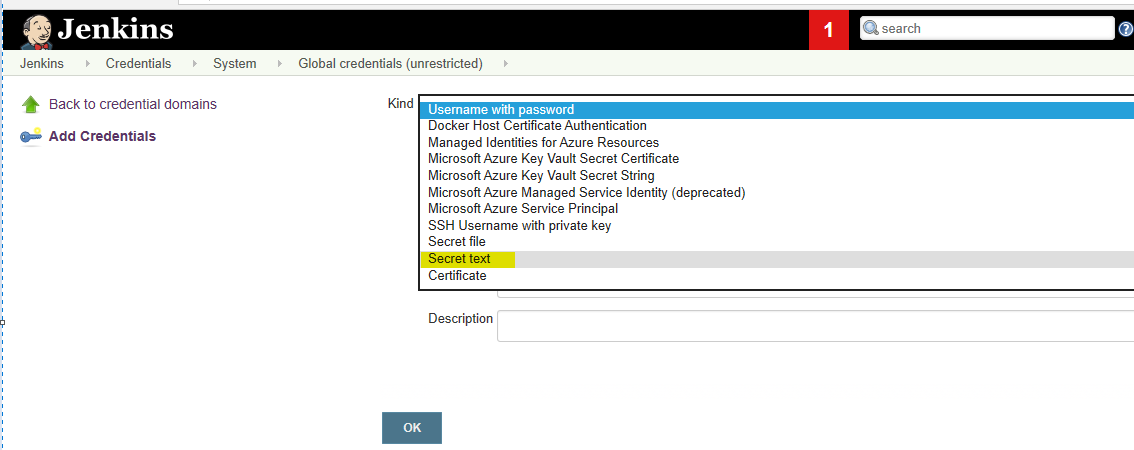


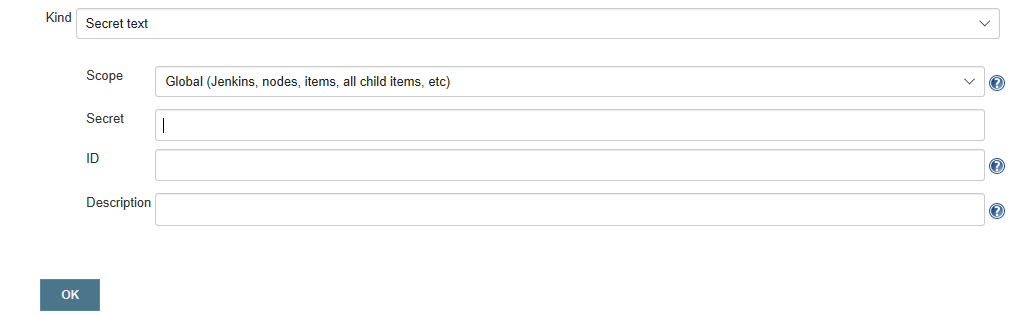
**Configure the Azure secrets in your Jenkins Application**

Go to **Jenkins Home** – **Credential** – click on **global** highlighted in below

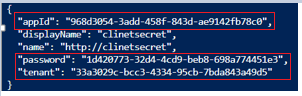


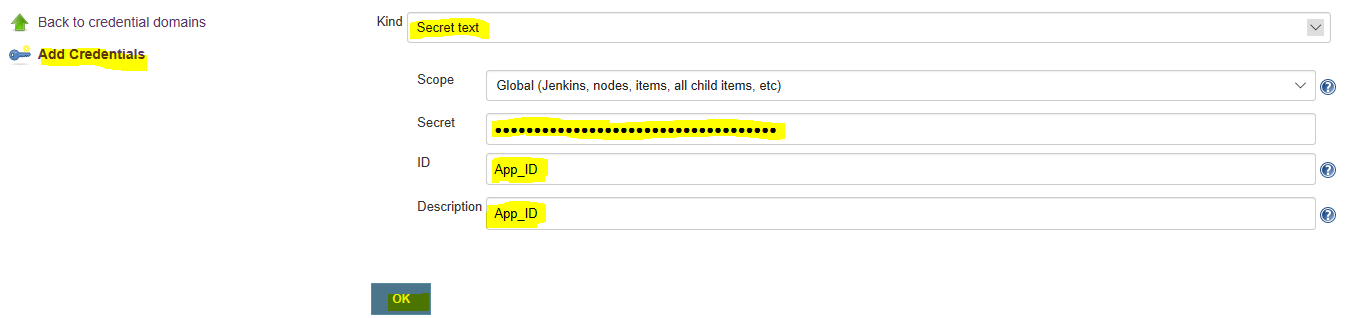
Next Click on **Add Credential** - Select “**Secret txt**” from the Kind drop down and click **ok**



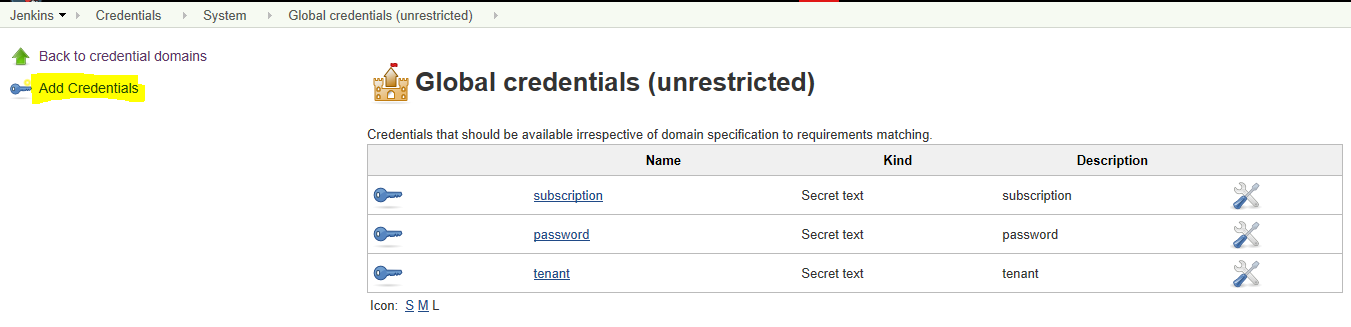


Create 3 different Secrets for three IDs. Highlighted in Red Box on below. (Like - **App ID**, **Password** and **tenant**) Description can be any name. Same you have to update in your script.

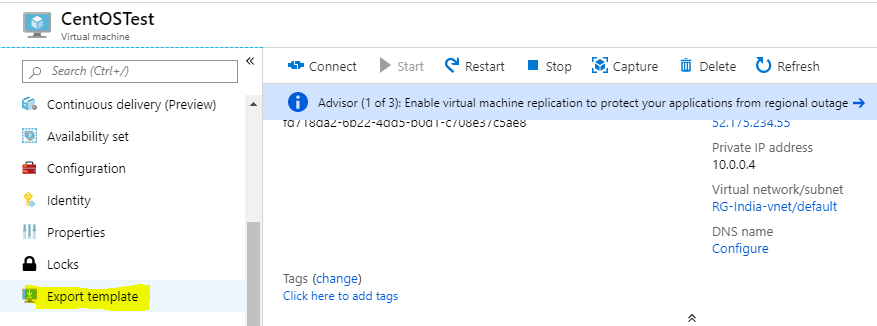




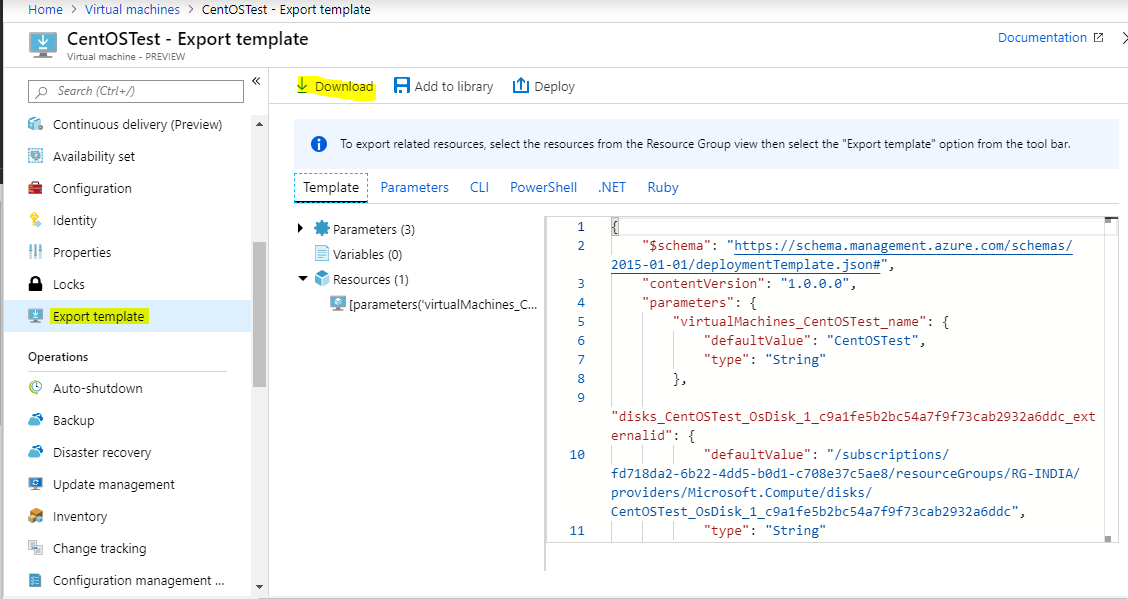
Once 3 credentials will be configured, it should see like below.



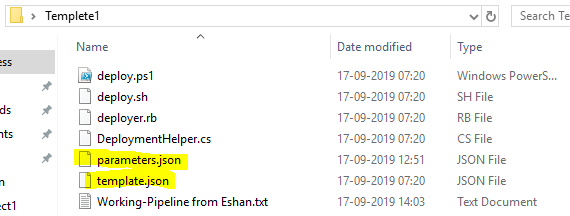
Now download a sample template from any existing VM which was already deployed in your Azure Subscriptions or you can create a new one. Here we are downloading the template from exiting VM.



Go to **Azure Portal** – **Select any VM** and click on **Export Template** – then Click on **Download Template** and **save it** to your local machine.

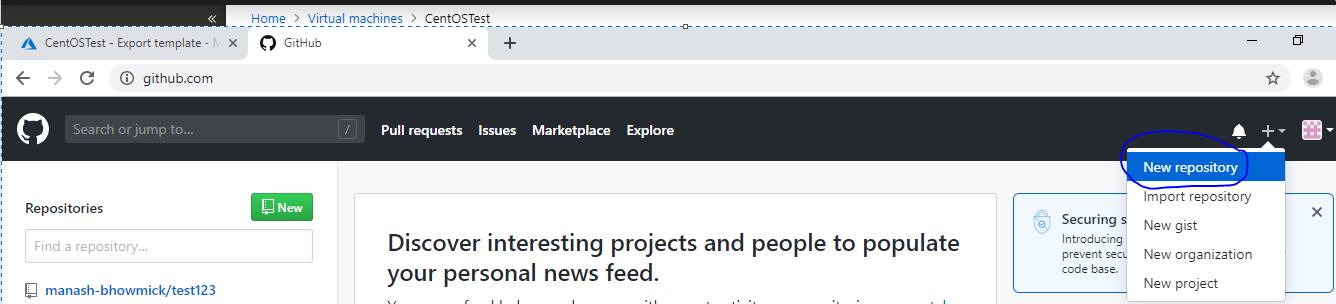


Downloaded content should be like below and you have to **upload** only “**parameters.json**” and “**template.json**” files to your **Github Repository**.



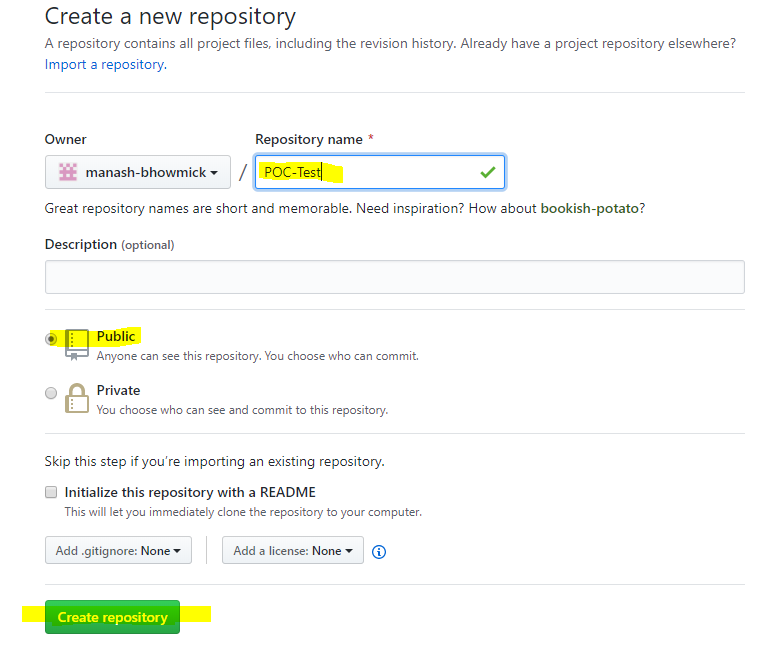
**Process to upload json files to the GitHub repository**

Login to **GitHub portal** – **Go to right corner** and **create a New repository**.

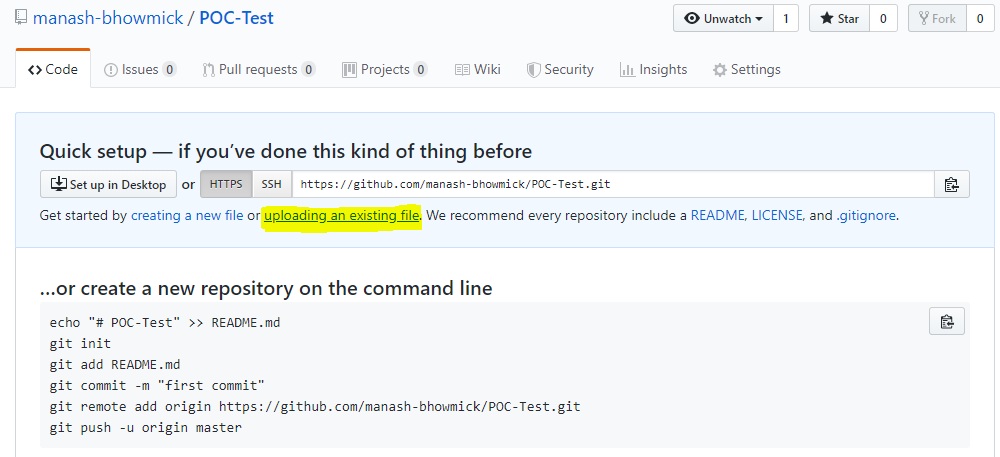


Provide your repository name – select as **public** and click on **Crate Repository**

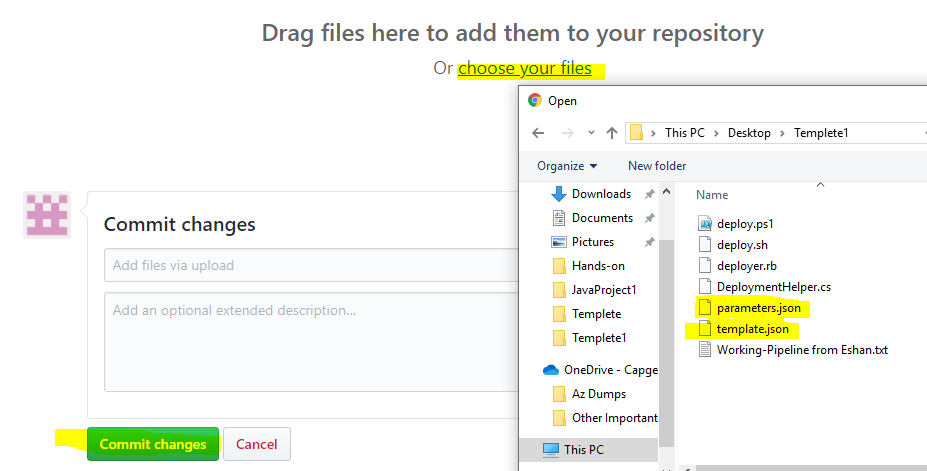
**Note: -** Private will always chargeable.



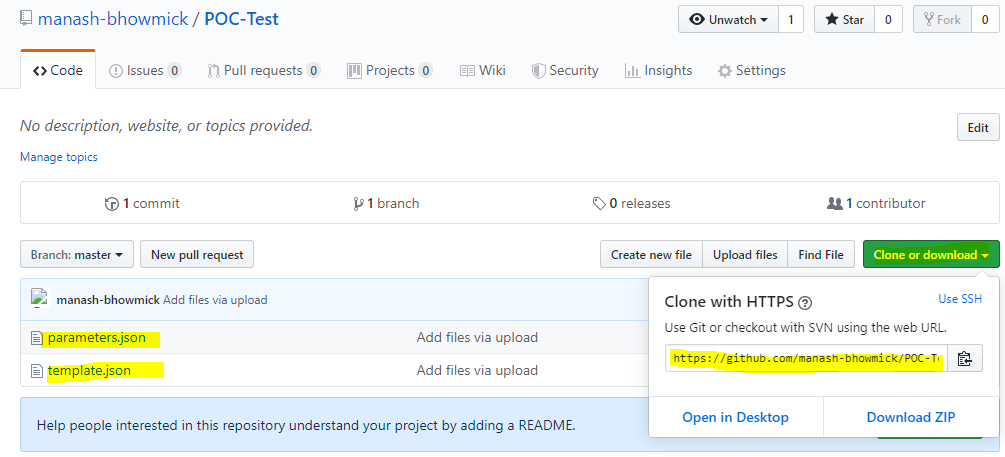
Now click on upload existing file highlighted in below.



Now **Upload** the 2 jsons file and click on **Commit Changes**.

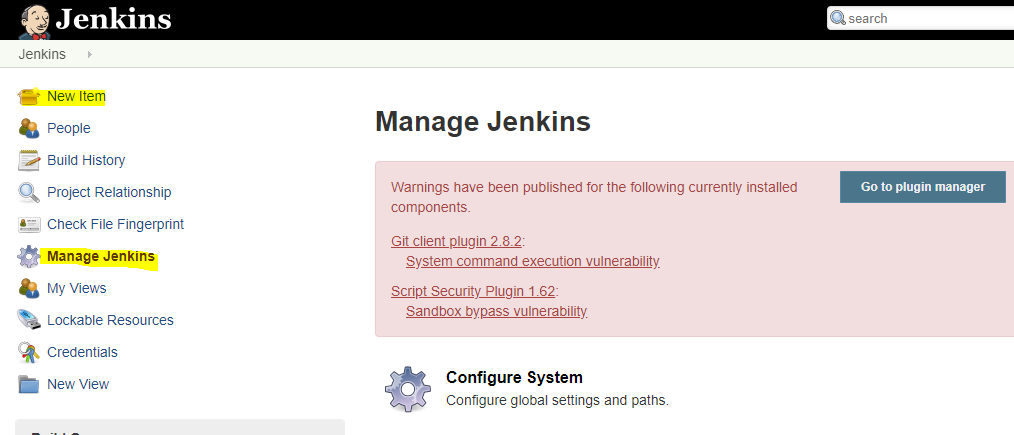


Once **Commit is successful**, it should see like below. Take this **git URL** and **update it to your pipe line** script.

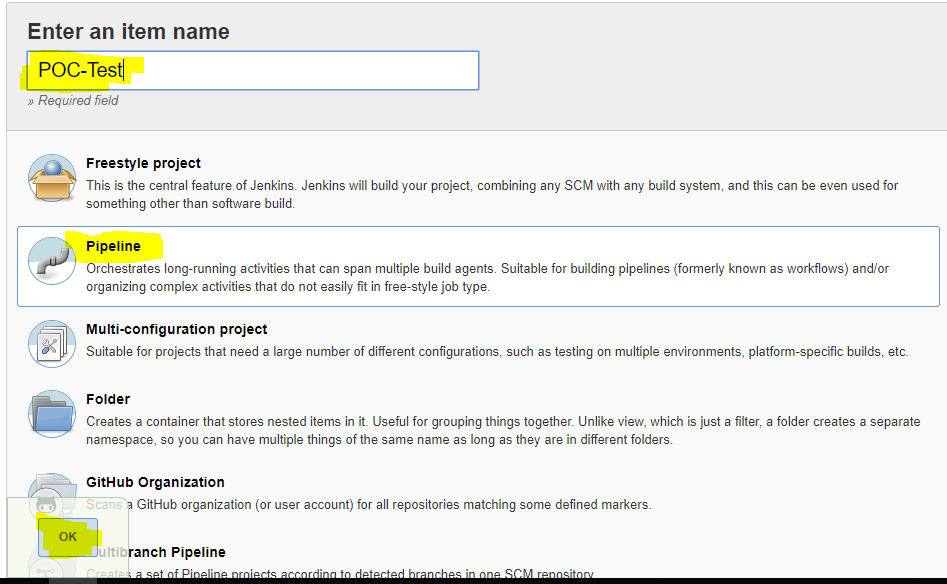


**Now configuring the Pipe line script in your Jenkins Application**

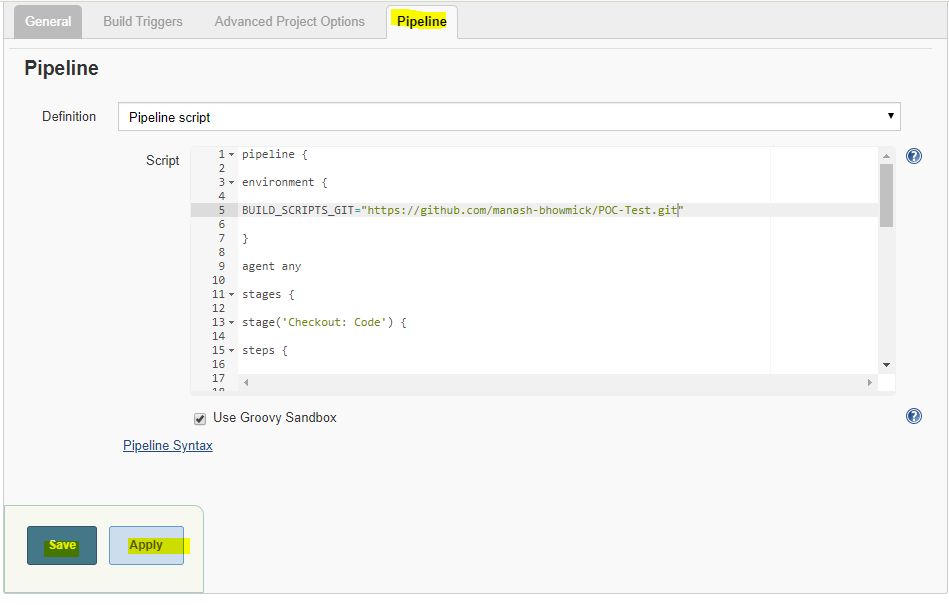
From the Jenkins home page – **Manage Jenkins** – **New Item**



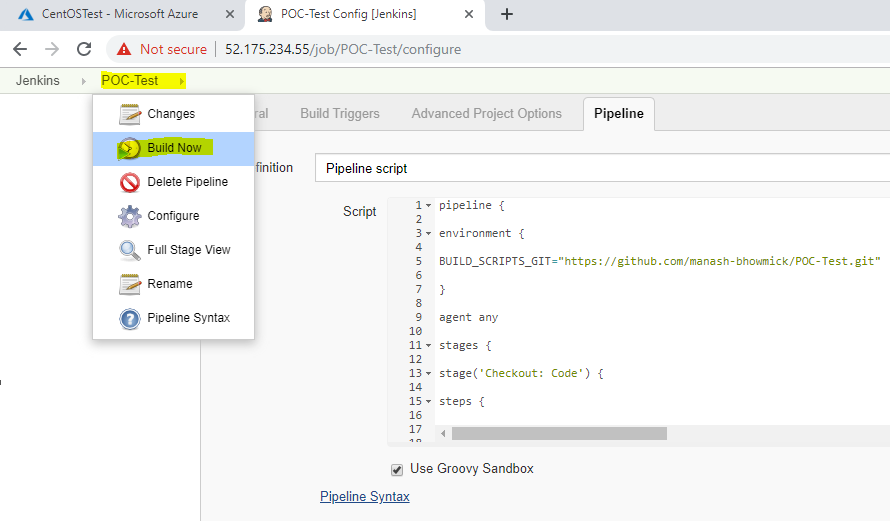
Provide the **Item-Name** – Select **Pipeline** - **OK**



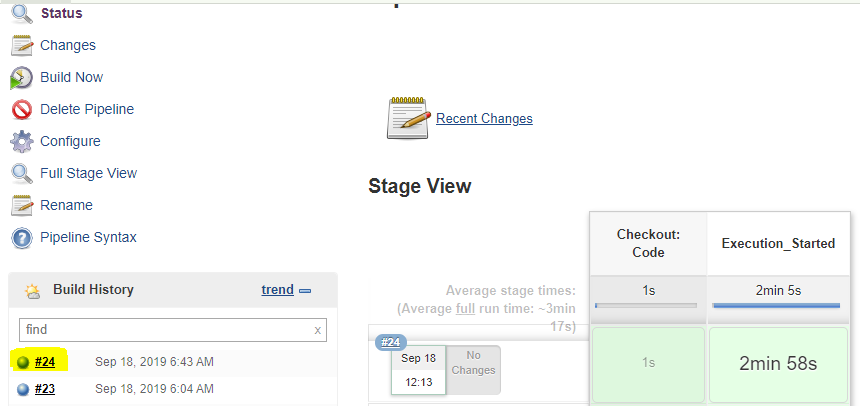
Now **paste** the pipe-line script – **Apply** and **Save**

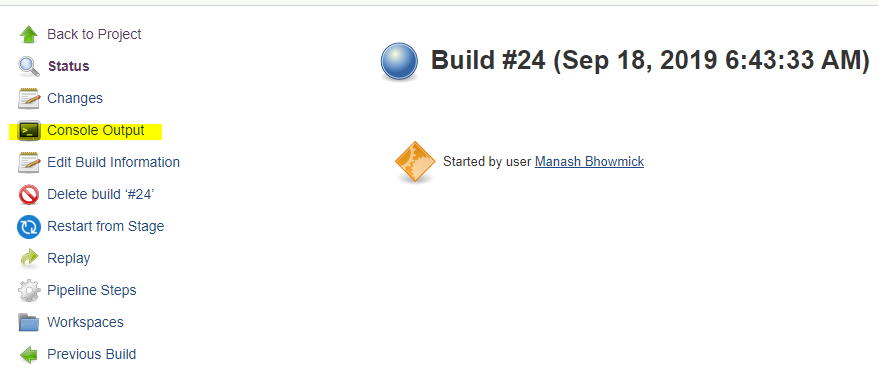


Next click on the **Build Now** to run the pipe line.

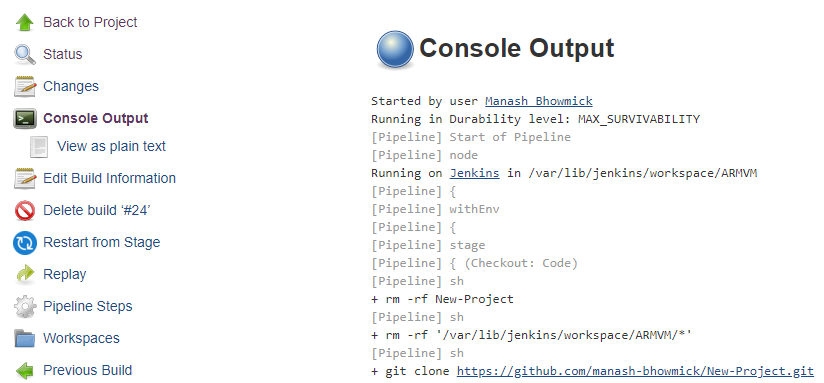


You can check your **Pipe line build history** by clicking on **Console Output**





**If your build will be completed successfully. Your console output should be like below.**



**Attached working pipe line Script file in Git hub.**

**-----------------------------------------------------------------**