**Jenkins CI/CD Pipeline setup with DevSecOps**

****

## ***Setting up the server***

### **Step-by-Step Guide to Launch a Jenkins Server on EC2**

#### **1. Launch an EC2 Instance**

* Go to the **AWS Management Console** → **EC2** → **Launch Instance**.
* Set **Instance Name** (e.g., "Jenkins-Server").
* Select **Amazon Machine Image (AMI)**: **Amazon Linux 2023**.
* Choose **Instance Type**: t2.medium or larger.

#### **2. Configure Key Pair**

* Create a new key pair or select an existing one for SSH access.
* Download and **store the key securely**.

#### **3. Set Up Security Groups**

* Create a **new security group** or select an existing one.
* Add inbound rules:
  + **Port 22 (SSH)** → **Source:** 0.0.0.0/0
  + **Port 8080 (Jenkins UI)** → **Source:** 0.0.0.0/0

#### **4. Configure Storage**

* Set the **Root Volume Size**: 30 GB (max allowed in free tier).

#### **5. Add User Data Script**

1. Expand **Advanced Details** → **User data**.
2. Copy and paste this script: <https://github.com/derrickSh43/Jenkins.sh>

## ***Installing the Dependencies***

Here we need to install all the tools we will need to run our pipeline.

### **1. SSH into the EC2 Instance**

Open Git Bash and enter in the below commands.

Use the following command to connect to your instance:

| ssh -i your-key.pem ec2-user@your-instance-ip |
| --- |

### **2. Install Required Packages**

#### **Update the System and Install Basic Utilities**

| sudo yum update -y sudo yum install -y git unzip curl wget |
| --- |

#### **Install AWS CLI**

| curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64.zip" -o "/tmp/awscliv2.zip" unzip /tmp/awscliv2.zip -d /tmp sudo /tmp/aws/install rm -rf /tmp/awscliv2.zip /tmp/aws |
| --- |

#### **Install Terraform**

| curl -fsSL https://releases.hashicorp.com/terraform/1.5.7/terraform\_1.5.7\_linux\_amd64.zip -o ~/terraform.zip sudo unzip ~/terraform.zip -d /usr/local/bin rm ~/terraform.zip |
| --- |

#### **Install SonarScanner**

| curl -fsSL https://binaries.sonarsource.com/Distribution/sonar-scanner-cli/sonar-scanner-cli-5.0.1.3006-linux.zip -o /tmp/sonar-scanner.zip sudo unzip /tmp/sonar-scanner.zip -d /opt/ sudo mv /opt/sonar-scanner-\* /opt/sonar-scanner sudo ln -s /opt/sonar-scanner/bin/sonar-scanner /usr/local/bin/sonar-scanner rm /tmp/sonar-scanner.zip |
| --- |

#### **Install Node.js & npm**

| sudo yum install -y nodejs npm node -v npm -v |
| --- |

#### **Install Snyk**

| sudo npm install -g snyk |
| --- |

#### **Install git**

| sudo yum install -y git |
| --- |

## ***Logging in to Jenkins***

### **1. Retrieve the Initial Jenkins Admin Password**

After Jenkins is installed and started, get the initial admin password with:

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

Copy this password, as you'll need it for the initial login.

### **2. Complete Jenkins Setup**

#### **Access Jenkins**

Open a web browser and go to: http://<your-instance-ip>:8080

* Paste the **initial admin password** when prompted.

#### **Install Recommended Plugins**

* Choose **"Install suggested plugins"** or manually select the ones you need.

#### **Create an Admin Account**

* Set up a **username** and **password** for the Jenkins admin account.
* **Do not forget** these credentials, as they will be required for future logins.

#### **Finalize Setup**

* Complete the setup and start using Jenkins.

## ***4. Configuring Jenkins***

### **Install Required Plugins**

If not already installed, add the following plugins:

From the **Jenkins Dashboard → Manage Jenkins →Plugins→Available Plugins**

* **GitHub Integration Plugin**
* **Git Plugin**
* **Pipeline: SCM Step**
* **AWS Credentials**
* **Pipeline Steps AWS**
* **SonarQube Scanner for Jenkins**
* **GitHub API Plugin**
* **Jira Plugin**

## ***4. Setting up External accounts***

#### **SonarQube (SonarCloud)**

1. Go to [SonarCloud](https://www.sonarsource.com/products/sonarcloud/).
2. Sign up and create an account.
3. Create a new organization and generate an **organization key**.
4. Create a new **project** in SonarCloud.
   1. Click the plus sign in the upper right corner
   2. Click on Analyze new project
   3. On the next screen choose the repo for the project you want to create
   4. Click on My projects
   5. Click on the name of the project you just created
   6. On the bottom right click on administration
   7. Select Analyst method
   8. Turn off Automatic Analysis
5. Generate an **API key** for authentication.
   1. In the upper right hand corner click on the account icon
   2. Select my account
   3. Click on Generate token
   4. Save the token somewhere
6. **Save the organization key, API key, and project key**, as these will be needed later.

#### **Snyk**

1. Go to [Snyk](https://snyk.io/).
2. Sign up and create an account.
3. Navigate to API settings and generate an **API key**.
4. **Save the API key**, as it will be required for Jenkins.

#### **AWS Credentials**

1. Create an IAM user with **programmatic access**.
2. Attach the necessary policies for Jenkins to interact with AWS.
3. Generate an **Access Key ID** and **Secret Access Key**.
4. **Save these credentials** securely.

#### **Jira**

1. Sign up for [Jira](https://www.atlassian.com/software/jira).
2. Create a new project for issue tracking.
3. Generate an **API key** under your account settings.
4. **Save the API key**, as it will be required for Jenkins integration.

### **Adding Credentials to Jenkins**

1. **Go to Jenkins Dashboard → Manage Jenkins → Manage Credentials**.
2. Select **Global credentials (unrestricted)** and click **Add Credentials**.
3. **Add the following credentials**:
   * **SonarQube**:
     + Kind: **Secret text**
     + Secret: Enter the API key
     + ID: sonarqube-api-key
   * **Snyk**:
     + Kind: **Secret text**
     + Secret: Enter the API key
     + ID: snyk-api-key
   * **AWS Credentials**:
     + Kind: **AWS Credentials**
     + Access Key ID: Enter your AWS Access Key
     + Secret Access Key: Enter your AWS Secret Key
   * **Jira**:
     + Kind: **Secret text**
     + Secret: Enter the API key
     + ID: jira-api-key

## ***5. Configuring Jenkins System Settings***

### **Setting Up Environment Variables**

1. Go to **Manage Jenkins** → **Configure System**.

Find **Global properties** → **Environment variables**.

Click **Add**:

* **Name**: SONAR\_SCANNER\_HOME
* **Value**: /opt/sonar-scanner

Testing the connection, from shell on the server run

| sonar-scanner \ -Dsonar.projectKey=your-project-key \ -Dsonar.organization=your-sonarcloud-org-key \ -Dsonar.host.url=https://sonarcloud.io \ -Dsonar.login=your-sonarcloud-token |
| --- |

## ***6. Configuring GitHub Webhook for CI***

1. Navigate to **Manage Jenkins** → **Configure System**.
2. Scroll down to **GitHub** and click **Add GitHub Server**.
3. Click **Manage Hooks** and ensure that Jenkins has the correct permissions to create webhooks.
4. Configure a webhook in your GitHub repository settings:
   * Go to **GitHub Repo → Settings → Webhooks**.
   * Click **Add webhook**.
   * Set **Payload URL** to http://your-jenkins-url/github-webhook/.
   * Choose **application/json** as the content type.
   * Select **Just the push event**.
   * Click **Add webhook**.

### **SonarQube Settings**

1. Go to **Manage Jenkins** → **Configure System**.
2. Scroll down to **SonarQube Servers**.
3. Click **Add SonarQube Server**.
4. **Name:** SonarQube
5. Enter the **SonarQube Server URL**. - https://sonarcloud.io
6. Select **Secret Text** and enter the **SonarQube API key**.
7. Click **Save**.

### **Jira Integration Settings**

1. Navigate to **Manage Jenkins** → **Configure System**.
2. Scroll down to **Jira**.
3. Enter your **Jira Base URL**.- https://yourcompany.atlassian.net
4. Add credentials using **Secret Text** and enter your Jira API key.
5. Click **Save**.

### **Creating a New Pipeline in Jenkins**

1. Go to the **Jenkins Dashboard**.
2. Click **New Item**.
3. Enter a **Pipeline Name**.
4. Select **Pipeline** and click **OK**.
5. You will be taken to the **Pipeline Configuration Screen**.

### **Configuring the Pipeline**

1. Under **General**, scroll down to the **Pipeline** section.
2. Find **Triggers** and check **GitHub hook trigger for GITScm Polling**.
3. Change **Definition** to **Pipeline from SCM**.
4. Under **SCM**, choose **Git**.
5. Enter in the repo url
6. Click **Save**.
7. Goto manage Jenkins and then security
8. Look for the git hooks section
9. Check both boxes
10. Save

## ***7. Jenkinsfile***

The Jenkins file for this is <https://github.com/derrickSh43/Jenkinsfile/blob/main/Jenkinsfile(working)>





You will need to copy it and create a new repo as the way it is named will not work as is.

After you copy it and save it to its own repo and name the repo “Jenkinsfile” so you have a copy for yourself.

1. Most of the issues I had when setting this up was with pathing for the installs so best to keep that in mind when you are having issues, all of the installs need to be visible to Jenkins
2. Do a git clone to clone you repo to your local machine

| cd /path/to/directory or use mkdir to create a new one git clone <repository\_url> |
| --- |

1. Open the code in VSCode

| Code . |
| --- |

1. Create a new file and name it “Jenkinsfile”
2. Copy the contents of the Jenkinsfile(working) from the provided URL above

**Changes**:

Make the following changes to the code

**Line 4** - Update the region to the region you plan to use.

**Line 16, 79, and 95** - Update the credential ID with what you set it to be in Jenkins when creating the AWS Credentials

**Line 28** - Update your repo link to reflect the repo that you want to use in the pipeline

**Line 36** - Update the CredentialsId to what you set in Jenkins for the sonarQube token ID

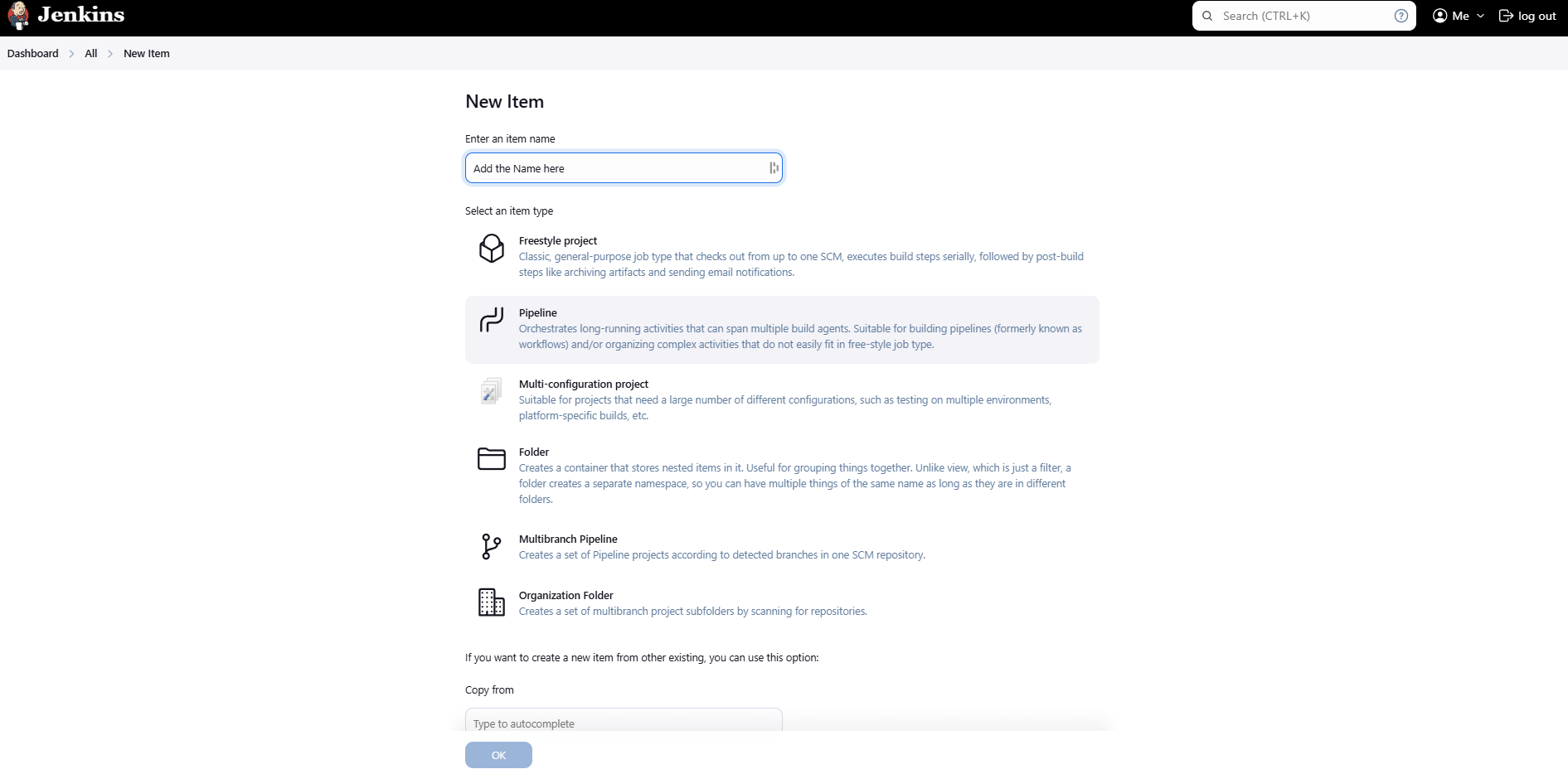
**Line 57** - Update the CredentialsId to what you set in Jenkins for the Snyk token ID

Save the file and do a git push to the repo

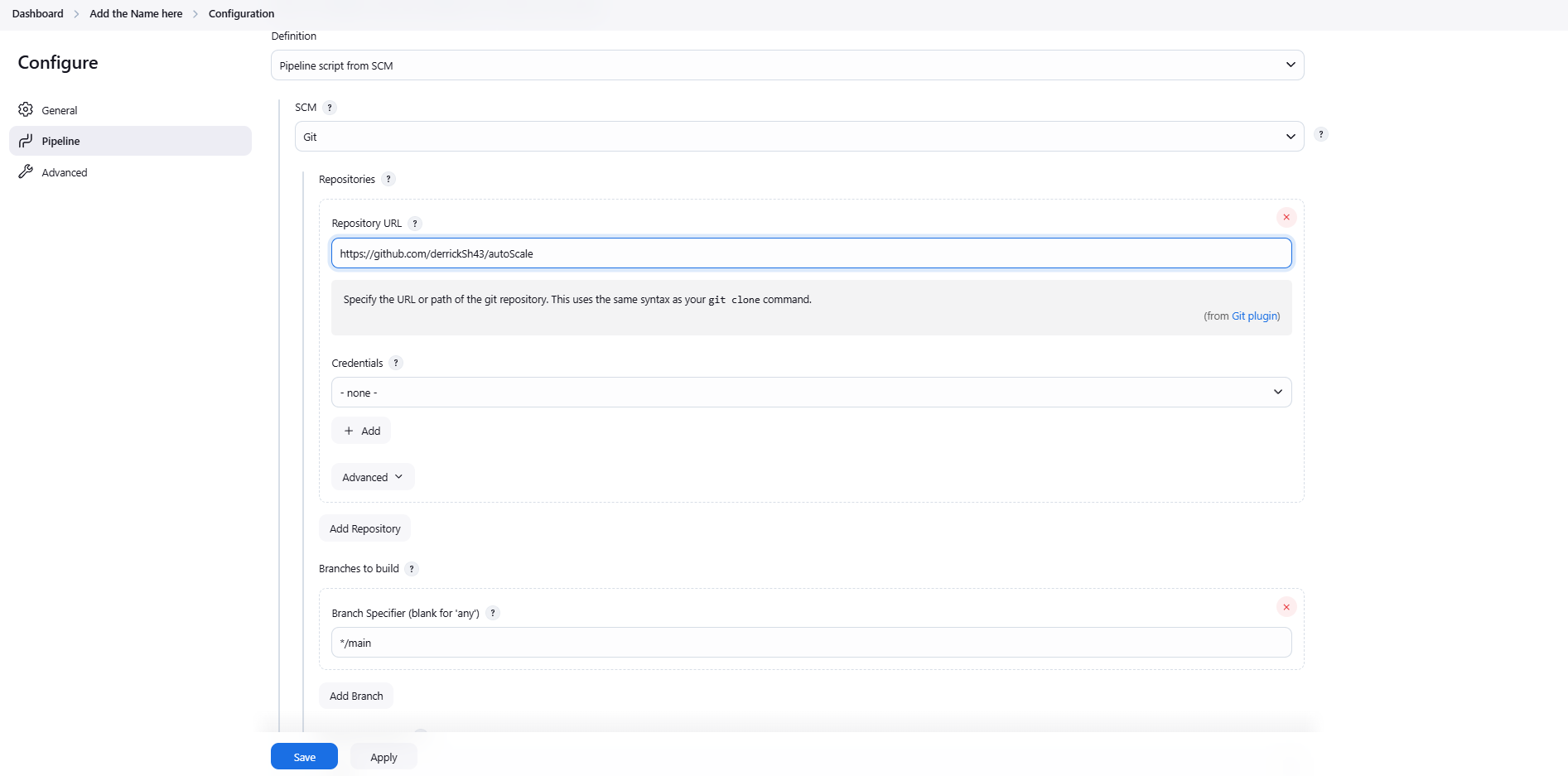
| git add . git commit -m"<add commit message here>" git push |
| --- |

## ***8. Create a New Pipeline***

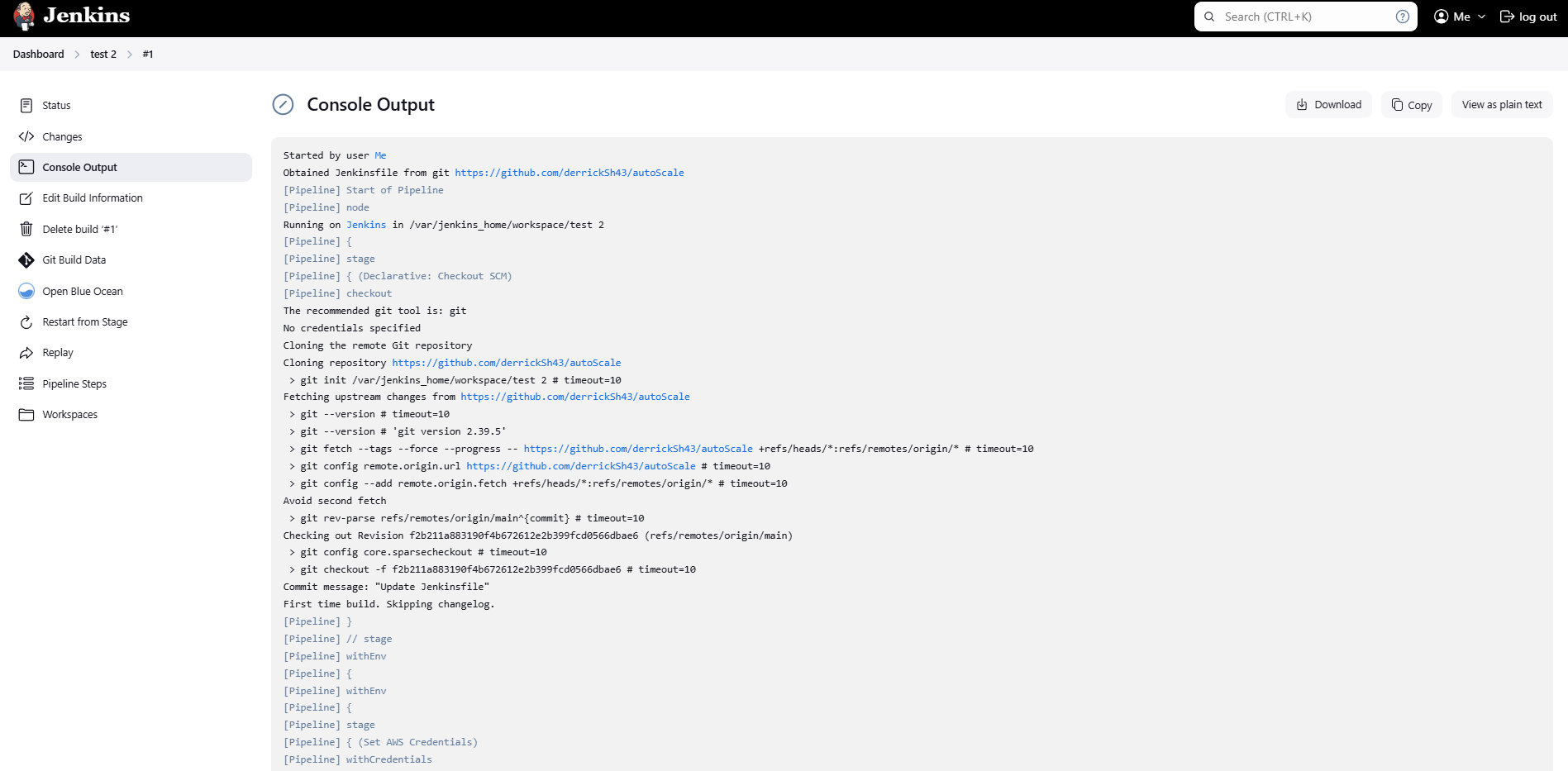
1. Go back to the **Jenkins Dashboard**
2. Click **New Item**
3. Enter a **Pipeline Name**
4. Select **Pipeline** and click **OK**
5. You will be taken to the **Pipeline Configuration Screen**

** Configure the Pipeline**

1. Under **General**, scroll down to the **Pipeline** section
2. Change **Definition** to **Pipeline from SCM**
3. Under **SCM**, choose **Git**
4. Add the URL of the get repo that you want to use.
5. Change the branch from master to the branch you want to use with your pipeline
6. Click **Save**

** Run the Pipeline**

1. On the **Jenkins Dashboard**, locate the pipeline you just created
2. Click on the **Pipeline Name**
3. Click **Build Now** on the left panel
4. To check progress, click **Console Output**

****

### **Done! Your Jenkins pipeline is now set up and running.**

If working properly when you do a git push to your repo or do a merge request Jenkins should start a new build automatically.