Context

We are testing your ability to implement modern automated infrastructure. In your solution you should emphasize readability, maintainability and DevOps methodologies.

Problem Statement

         Your task is to configure server: preferably using AWS/**Azure**free tier or vagrant box.

         Server needs to serve:

o    the dev environment

o    and host tools required for CI/CD.

         Write script to

o    Install required software for CI/CD setup. We leave this with you to decide what is important.

o    Set up CI to fetch the codebase from <https://github.com/ahfarmer/emoji-search.git>  and package it for deployment.

o    Deploy and launch the application in dev environment for every new commit or every 6 hours.

Bonus

         Containerization

         Security consideration

         Surprise us.

Evaluation

         End result must address the problem statement

         Please do frequent commits so we can see your approach towards solving the problem.

         Clean README.md file that explains how things work, explanation of choices and please do mention what you would do differently or additionally if you had more time.

         Extensible code/script

         Bonus

Solution:

The requirement:

There are a few software requirements that must be met in order to be successful with this guide.  They are as follows:

* [Node.js](https://nodejs.org/)
* JDK 8
* Git
* [Jenkins](https://jenkins.io/)
* Virtual machine(EC2)

1. **Create Ubuntu server using AWS EC2 instance**

Example: Dev-server-CICD

Connect using SSH: “**ssh -i "Jenkins-Server-Key.pem" ubuntu@ec2-54-88-81-5.compute-1.amazonaws.com**

“

1. **Setup the server for CI-CD**

**Setup and install Jenkins**

* sudo add-apt-repository ppa:webupd8team/java
* sudo apt-get update
* sudo apt-get install oracle-java8-installer
* java -version
* install git

**install Jenkins**

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

$sudo service jenkins status

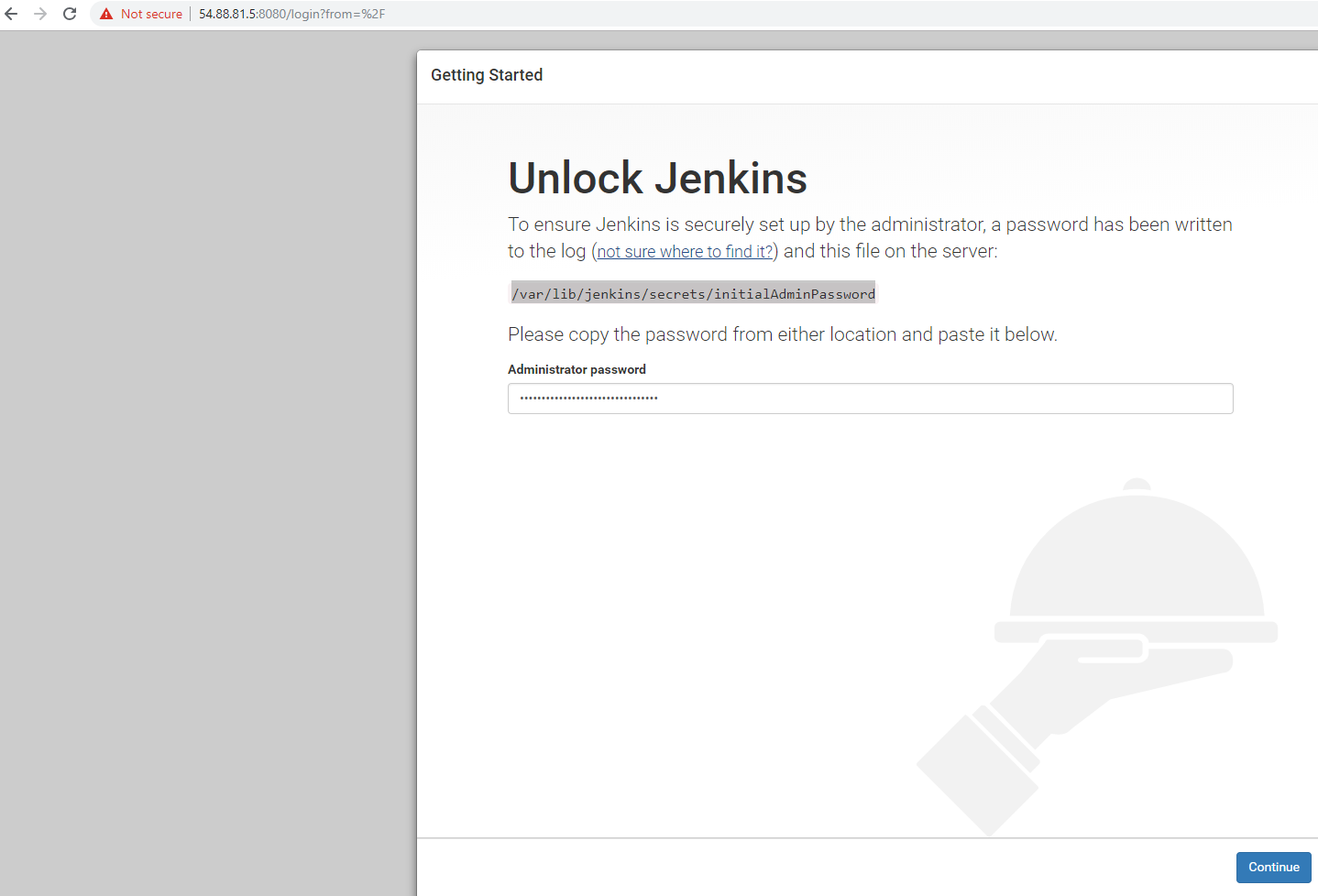
**Install nodejs npm and node**

* *Sudo apt-get update*
* *Sudo apt install nodejs npm*

**Jenkins Configuration and job setup:**

* Open default port 800 in security group
* Login to Jenkins <public-ip>:8080
* Login as admin user
* Password copy from ***/var/lib/jenkins/secrets/initialAdminPassword***

After login with initial password, you are ready to go for create a new Jenkins jobs.



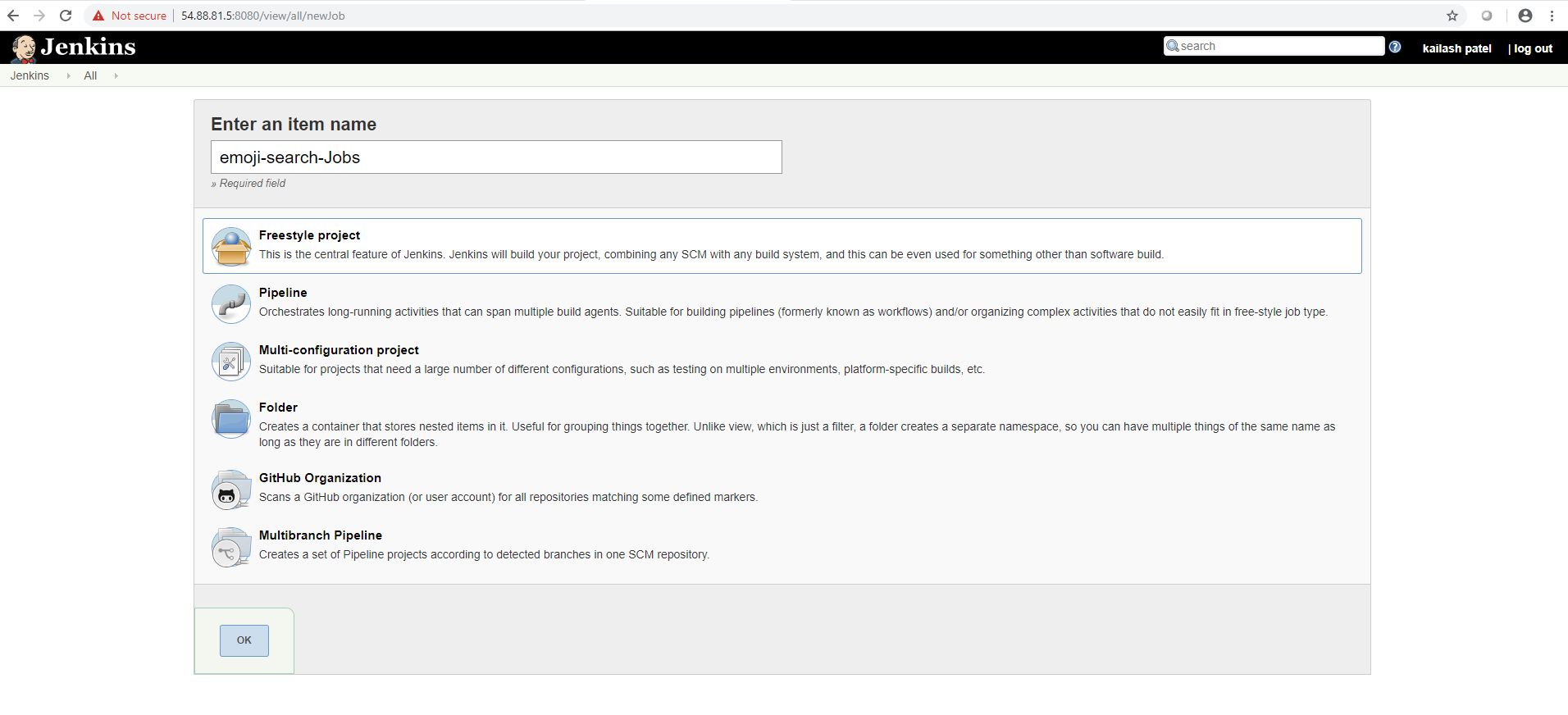
**Install all required plugin**

Git

NPM plugin

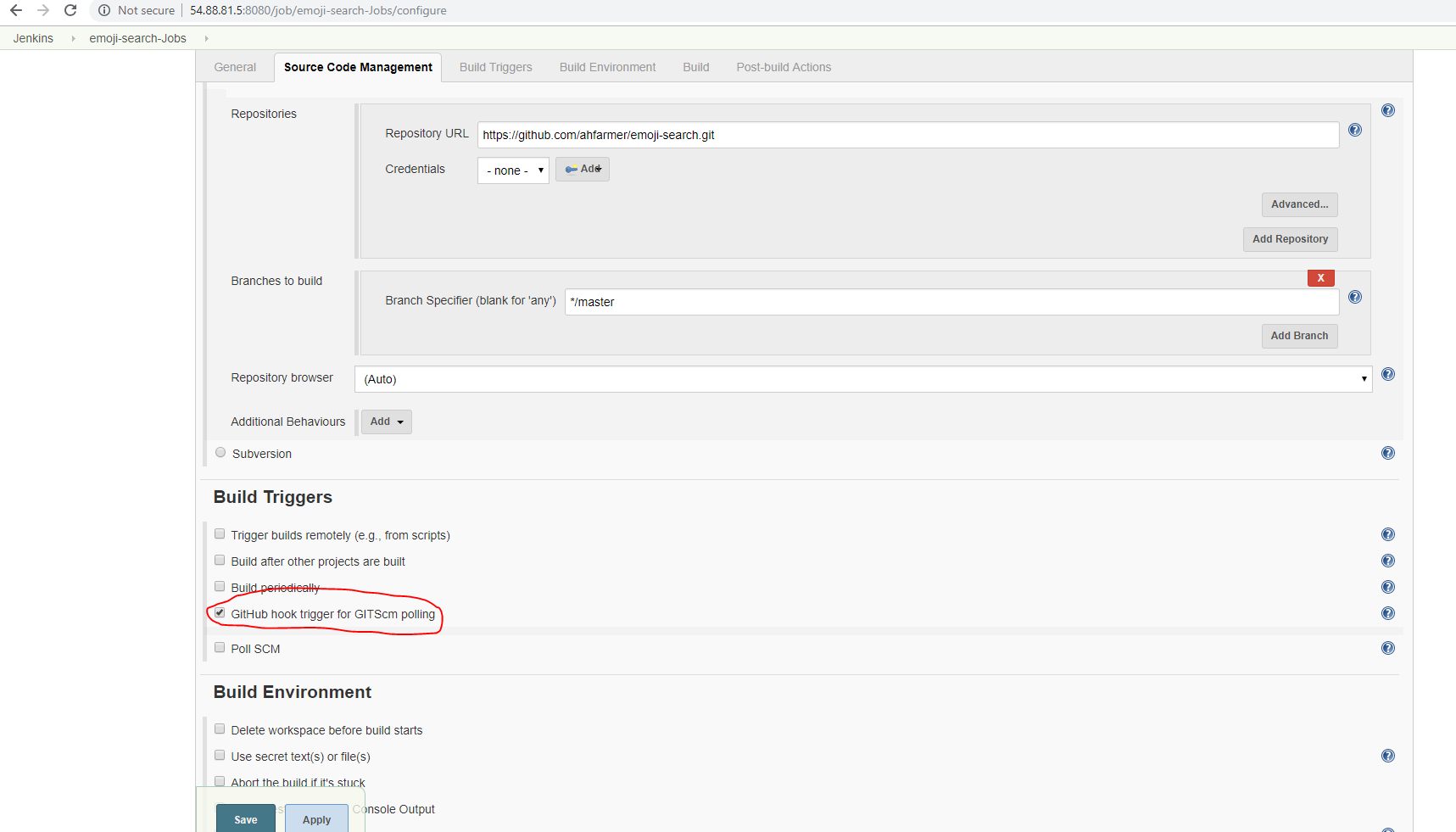
Post build script plugin

Now create a new free-style project job:

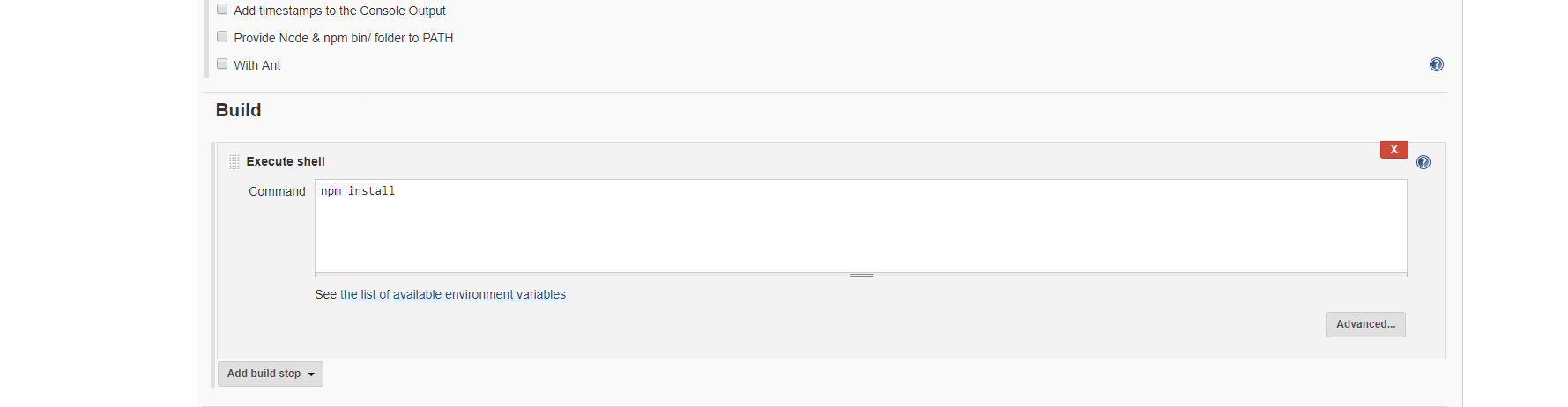


**Source code management:**

1. Configure SCM>Git and provide the url <https://github.com/ahfarmer/emoji-search.git>
2. Select ‘***GitHub hook trigger for GITScm polling***” to pull and build code on every new check-in push.



**Build**: After configuring the source control section we’ll need to configure the build step.  For Node.js, building only consists of installing dependencies, but you could easily have unit testing and other testing in this step as well.



**Post build script plugin:**

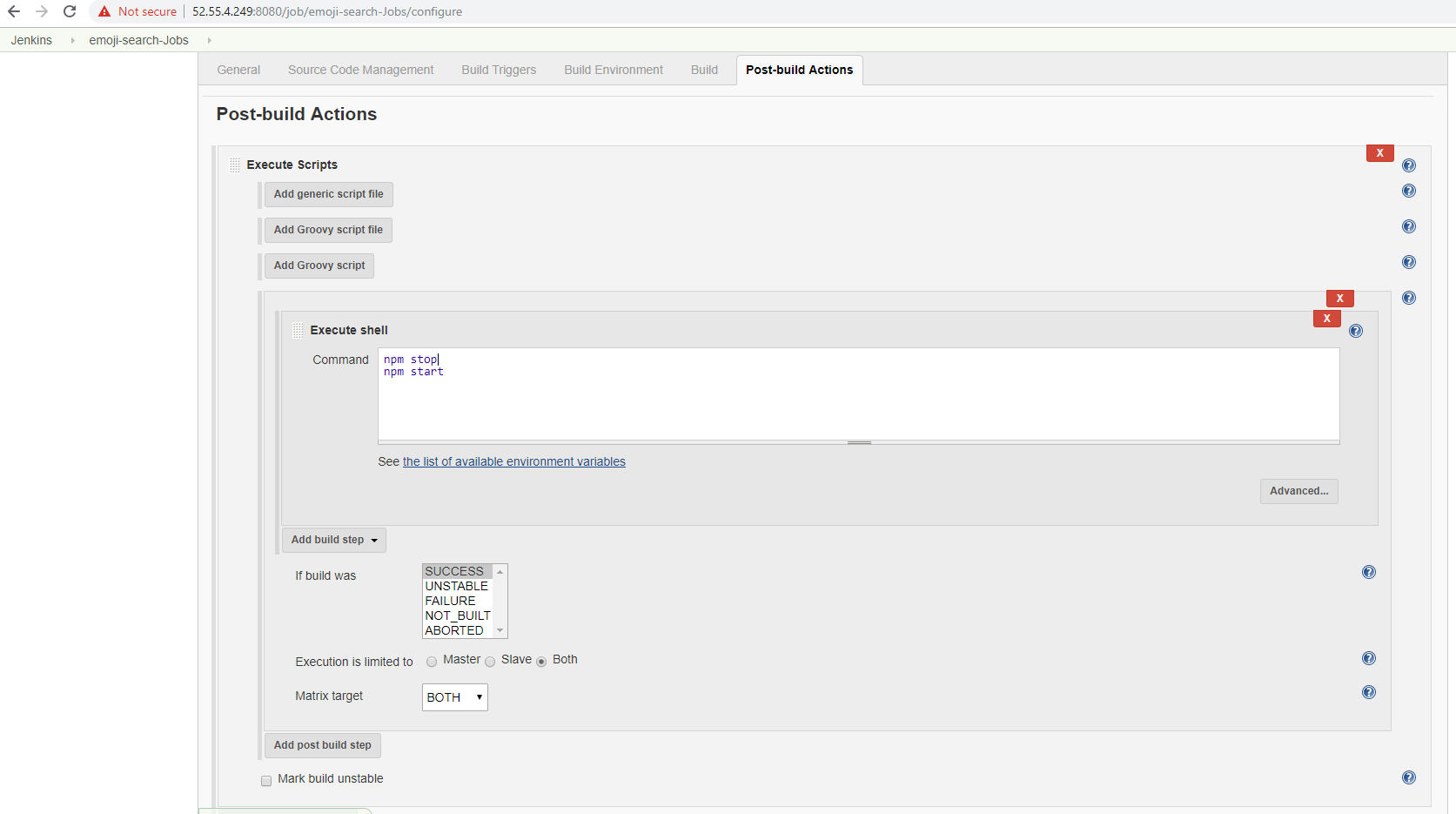
This plugin we install to run post build command:

Npm stop

Npm start

In this we will be deploying the application locally on your machine. Probably not the case in your production.

Before starting the application, we are stopping any already running instance of it.  Once stopped we can start the new version.



**Ansible playbook to install tools:**

(if you want to run it on node\slave server to install tools and deploy the app)

**Note:** we can use ansible to configure our developer server if it is node\slave. But make sure you update your inventory file with your server name\ip to run ansible playbook on that.

**- hosts: "{{ hostname }}"  
  gather\_facts: False  
  vars:  
    - packages: ["git", “npm”]  
  tasks:  
    - name: Installing {{ packages }} on {{ hostname }}  
      yum:  
        name: "{{ item }}"  
        state: present  
      with\_items: "{{ packages }}"**

**Dockerfile Node App Config**

Now we have to create a Dockerfile which will instruct Docker how to deploy the app within a Docker image. Go to the root of the Node project we just created and type touch Dockerfile and put the following into it:

FROM mhart/alpine-node

EXPOSE 3000

WORKDIR /app

COPY . /app

CMD ["node", "app.js"]

This is basically telling Docker we want to deploy our files to an [Alpine Node](https://github.com/mhart/alpine-node) Docker image (basically a lightweight version of a Node instance). It'll copy over our built files to the Docker image and run the Node app.