***Deployment Life Cycle***

**Idea**: We are trying to deploy our spring boot application in cloud using tools like github, jenkins, docker, dockerhub, kuberbetes, AWS, EC2.

CONTINOUS INTEGRATION: *CI focuses on the process of integrating code changes from multiple contributors into a shared codebase frequently and automatically.*

**Key Practices:**

- Developers commit their code changes to a version control system (e.g., Git) multiple times a day.

- Automated build and test processes are triggered whenever code changes are committed.

- If the automated tests fail, the team is notified immediately, and the issues are addressed.

- The goal is to catch integration problems, bugs, and conflicts early in the development process.

**Benefits**:

- Early detection of integration issues and bugs.

- Faster feedback to developers.

- Improved code quality and stability.

- Enables collaboration among team members.

**Continuous Integration Tools**:

Jenkins, Travis CI, CircleCI, GitLab CI/CD, and many others.

Continuous Delivery (CD):

**Purpose**: CD builds upon CI and focuses on automating the delivery and deployment of code to production or staging environments with minimal manual intervention.

**Key Practices**:

Code that passes CI tests is automatically deployed to staging or pre-production environments.

Automated tests, including integration tests and user acceptance tests, are run in staging environments.

Deployment to production is automated but typically requires manual approval or a triggered release process.

The goal is to ensure that the software is always in a deployable state, and releases can be done quickly and reliably.

**Benefits**:

Faster and more reliable releases.

Reduced deployment risk.

Consistent and repeatable deployment processes.

Improved collaboration between development and operations teams (DevOps).

Continuous Delivery Tools: Jenkins, GitLab CI/CD, AWS CodePipeline, Travis CI, and others.

In summary, while CI focuses on the frequent integration of code changes and automated testing to catch issues early in development, CD extends this by automating the deployment process and ensuring that code is always in a deployable state. CD emphasizes reducing manual intervention in the deployment process and aims for reliable and efficient releases to production.

It's worth noting that there is another related concept called "Continuous Deployment" (CDeploy), which takes CD a step further. Continuous Deployment automatically deploys code changes to production without manual approval, as long as the automated tests pass. This approach is common in organizations that prioritize rapid and frequent releases.

**How we can integrate our Github to Continous Integration tool with Jenkin and how to Automate the Build. (Comes under *CI* only)**

**Step: 1**

Create a spring boot application with dummy apis and push it your github repository.

[*https://github.com/vishalkumar392392/end-to-deployment*](https://github.com/vishalkumar392392/end-to-deployment)

**Step: 2** Jenkins server on AWS EC2

Open your AWS account and create a EC2 instance using Amazon Linux type. Download the key value pair. Now we have to install the Jenkins on the EC2 instance. Follow the below URL to install the jenkins in your EC2.

[*https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/*](https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/)

**Step: 3** Jenkins Server boot up

Start the Jenkins server and download the default plugins. Create a user with password.

**Step: 4** Installing missing dependencies.

Install the Git and maven in EC2 instance as root user sudo -i

**Step: 5** Process of creating a new

1. Click on the New Item.
2. Enter a item name, select the FreeStyle Project and click ok.
3. We will go to the configuration page.
4. Give the Description and check the Github Project option and give the project url and display name.
5. In Source code Management section select Git Radio button.
6. Give Repository url, if you are getting the unable to fetch error msg then it means you don’t have the git installed in your EC2 or you need to set the git path in the manage jenkins/Tools Section.
7. Next Add jenkins credentials by creating.
8. Next in the Branches to build section give branch name as “/main”.
9. In build Triggers section Select the Poll SCM and give the corn expression as “\* \* \* \* \*”. It means build after each commit and push to the repo.
10. In the Build Steps, select the Invoke top-level Maven targets. In Goals write “install”.
11. Click on Apply and Save.
12. Then click on build. The build will start.

Building and Pushing Docker Image to Docker Hub using Jenkins Pipeline

**Step: 1**

Create a spring boot application with dummy apis and push it your github repository.

[*https://github.com/vishalkumar392392/end-to-deployment*](https://github.com/vishalkumar392392/end-to-deployment)

**Step: 2** [Install Java](https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/amazon-linux-install.html)17 and Jenkins server on AWS EC2

Open your AWS account and create a EC2 instance using Amazon Linux type(t2.large). Download the key value pair. Now we have to install the Jenkins on the EC2 instance. Follow the below URL to install the jenkins in your EC2.

[*https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/*](https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/)

**Step: 3** Jenkins Server boot up

Start the Jenkins server and download the default plugins. Create a user with password.

**Step: 4** Installing missing dependencies.

Install the Git and maven in EC2 instance as root user sudo -i

**Step 5**: Install Docker in EC2 instance. Follow the below url.

<https://www.cyberciti.biz/faq/how-to-install-docker-on-amazon-linux-2/>

Execute this below commands other wise we cannot connect to docker from jenkins server.

***sudo usermod -aG docker jenkins***

***sudo systemctl restart jenkin****s*

**Step 6**: **Maven Config**

1. Click on Manage Jenkins/Tools, configure the maven.

2. Under Maven Installation, give name as 3.8.4, select the version as 3.8.5. Click apply and Save.

**Step 7**: Create the script for the build trigger and run the pipeline. Refer below link

<https://youtu.be/PKcGy9oPVXg?si=9NysjeaCM2LrS7eq>

Install kubernetes:

sudo vi /etc/yum.repos.d/kubernetes.repo

[kubernetes]

name=Kubernetes

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86\_64

enabled=1

gpgcheck=1

repo\_gpgcheck=1

gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

exclude=kubelet kubeadm kubectl

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

sudo systemctl start kubelet

sudo systemctl enable kubelet

sudo kubeadm init --pod-network-cidr=192.168.0.0/16

To start using your cluster, you need to run the following as a regular user:

mkdir -p $HOME/.kube

sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

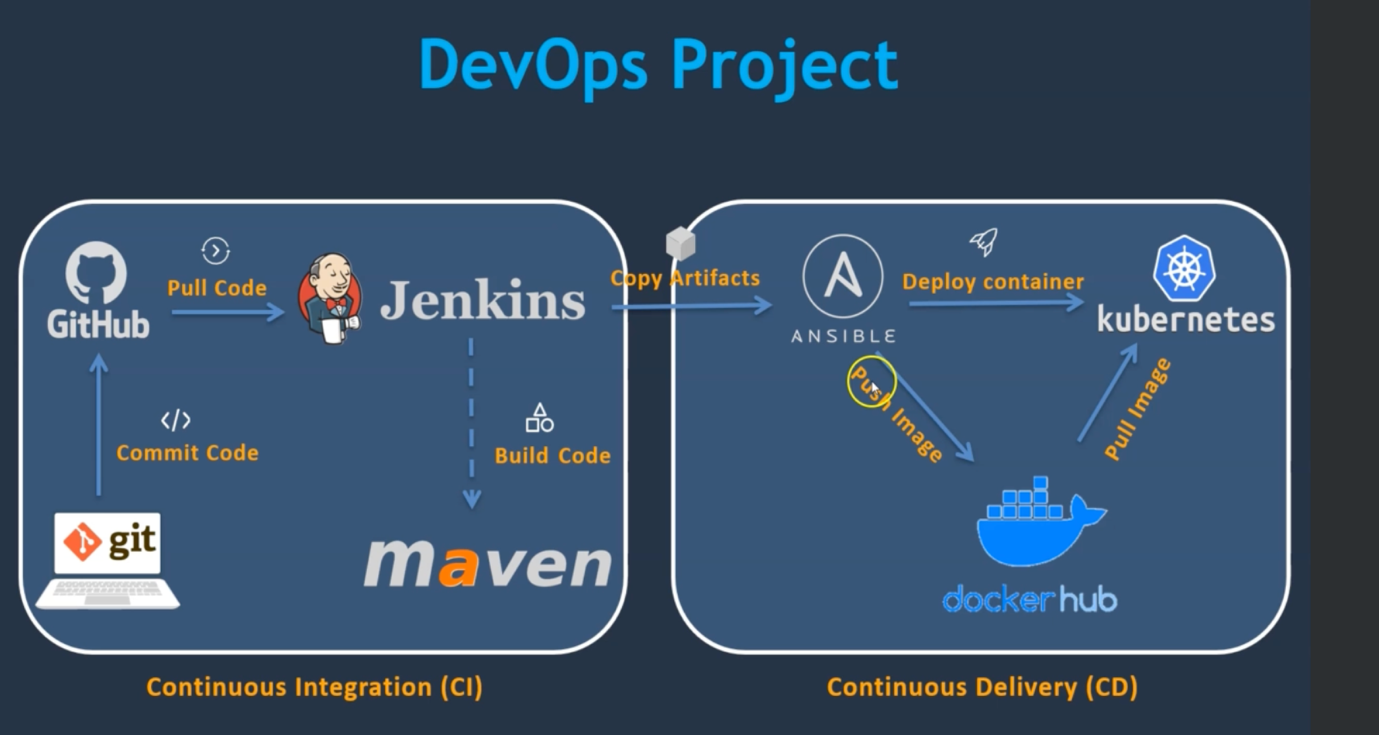
sudo chown $(id -u):$(id -g) $HOME/.kube/config

Alternatively, if you are the root user, you can run:

export KUBECONFIG=/etc/kubernetes/admin.conf

kubectl apply -f <https://docs.projectcalico.org/manifests/calico.yaml>

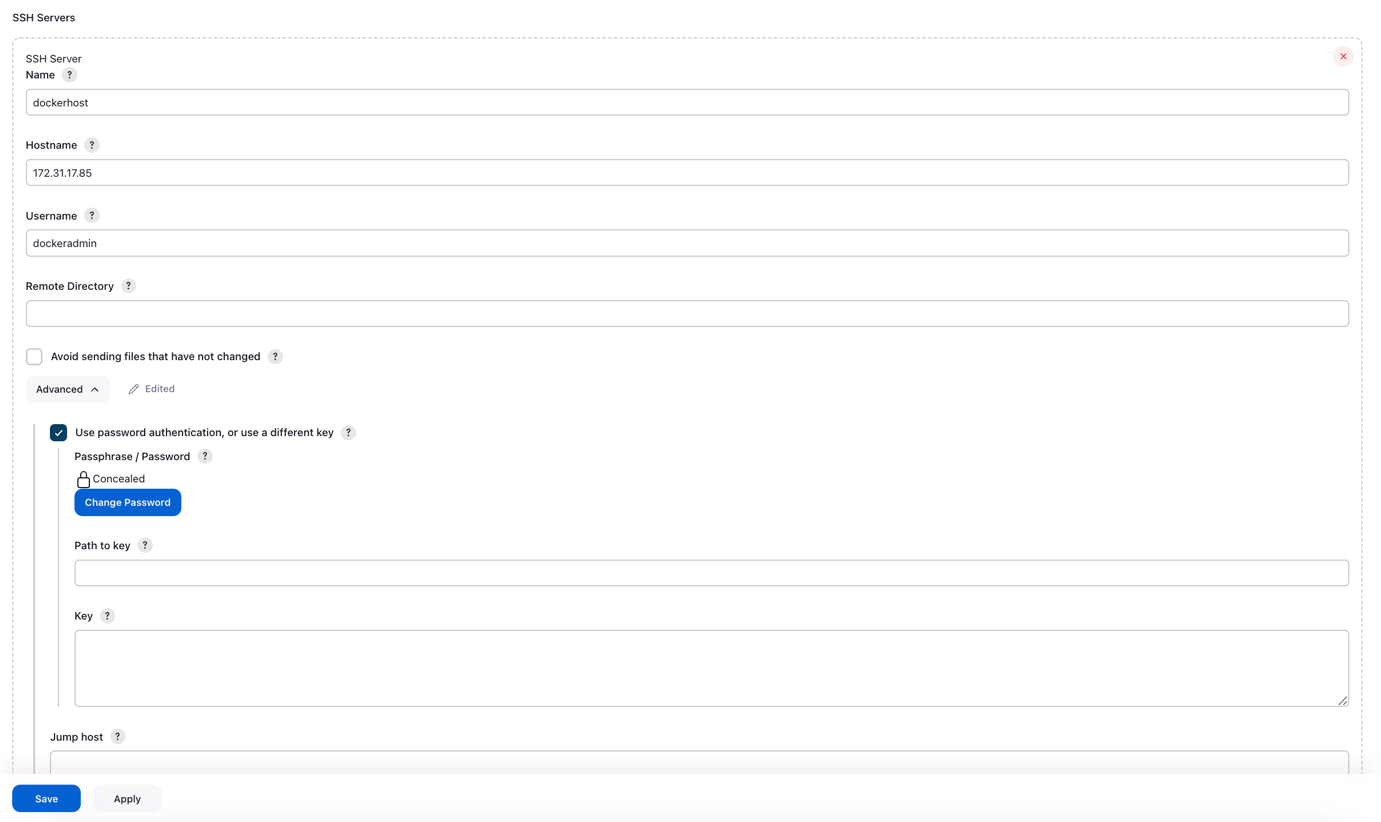
CI/CD with GIT JENKINS ANSIBLE KUBERNETES



**CI/CD from jenkins server to dockerhost ec2 server:**

We are going to ci/cd the my [github repository](https://github.com/vishalkumar392392/ci-cd.git) using jenkins and server and docker ec2 instance.

1. Create a new EC2 Amazon linux 2 AMI
2. Install [java 17](https://docs.aws.amazon.com/corretto/latest/corretto-17-ug/amazon-linux-install.html) and [jenkins](https://pkg.jenkins.io/redhat-stable/).
3. Install maven and git in ec2 instance.
4. Install Maven Integration Plugin, Github Plugin and Publish Over SSH plugin.
5. Click on Manage Jenkins/Tools, configure the maven.
6. Under Maven Installation, give name as 3.8.4(maven installed version), select the version as 3.8.4. Click apply and Save.
7. Create a new EC2 Amazon linux AMI and [install docker](https://www.cyberciti.biz/faq/how-to-install-docker-on-amazon-linux-2/) in it.
8. Now we need to integrate our docker host with jenkins.
9. Create a dockeradmin user
10. Install “Publish Over SSH plugin” in jenkins server.
11. Add Dockerhost to jenkins “configure systems”.
12. Go to your dockerhost ec2 instance as root user. Enter below commands.
13. cat /etc/passwd
14. Above commands will show all users, ec2-user will be there.
15. cat /etc/group
16. Above command will show all the groups. Docker will be there.
17. Now we should create a user and add him to the docker group.
18. useradd dockeradmin
19. passwd dockeradmin
20. usermod -aG docker dockeradmin
21. vi /etc/ssh/sshd\_config
22. Open and search “/PasswordAuthentication” and modify from “no” to “yes”. We are doing this bcz by default ec2 instance wont accept password based authentication.
23. service sshd reload
24. systemctl reload sshd.service
25. Go to the jenkins server ec2 instance and go to plugins sections and install “Publish over SSH” plugin if it is not installed.
26. Once u installed the plugin, got to the Manage Jenkins/System and go last and add a ssh server. Give the details, hostname is docker running ec2 instance ip address and test the configuration and click save and apply.



1. We will create a directory and store all docker images there. Exit from dockeradmin user.
2. exit
3. cd /opt
4. mkdir docker
5. chown -R dockeradmin:dockeradmin docker
6. cd docker
7. vi Dockerfile
8. chown -R dockeradmin:dockeradmin Dockerfile

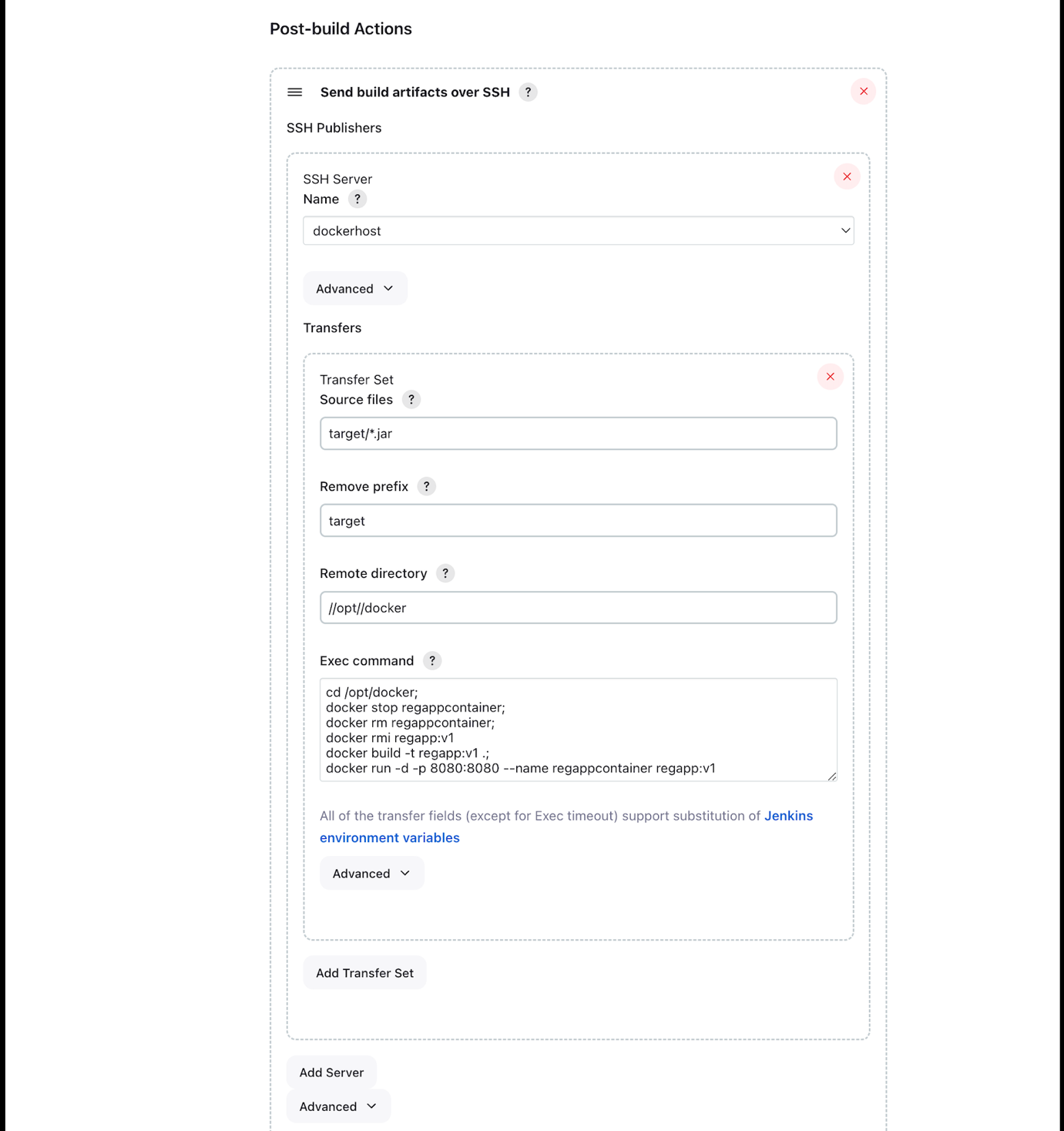
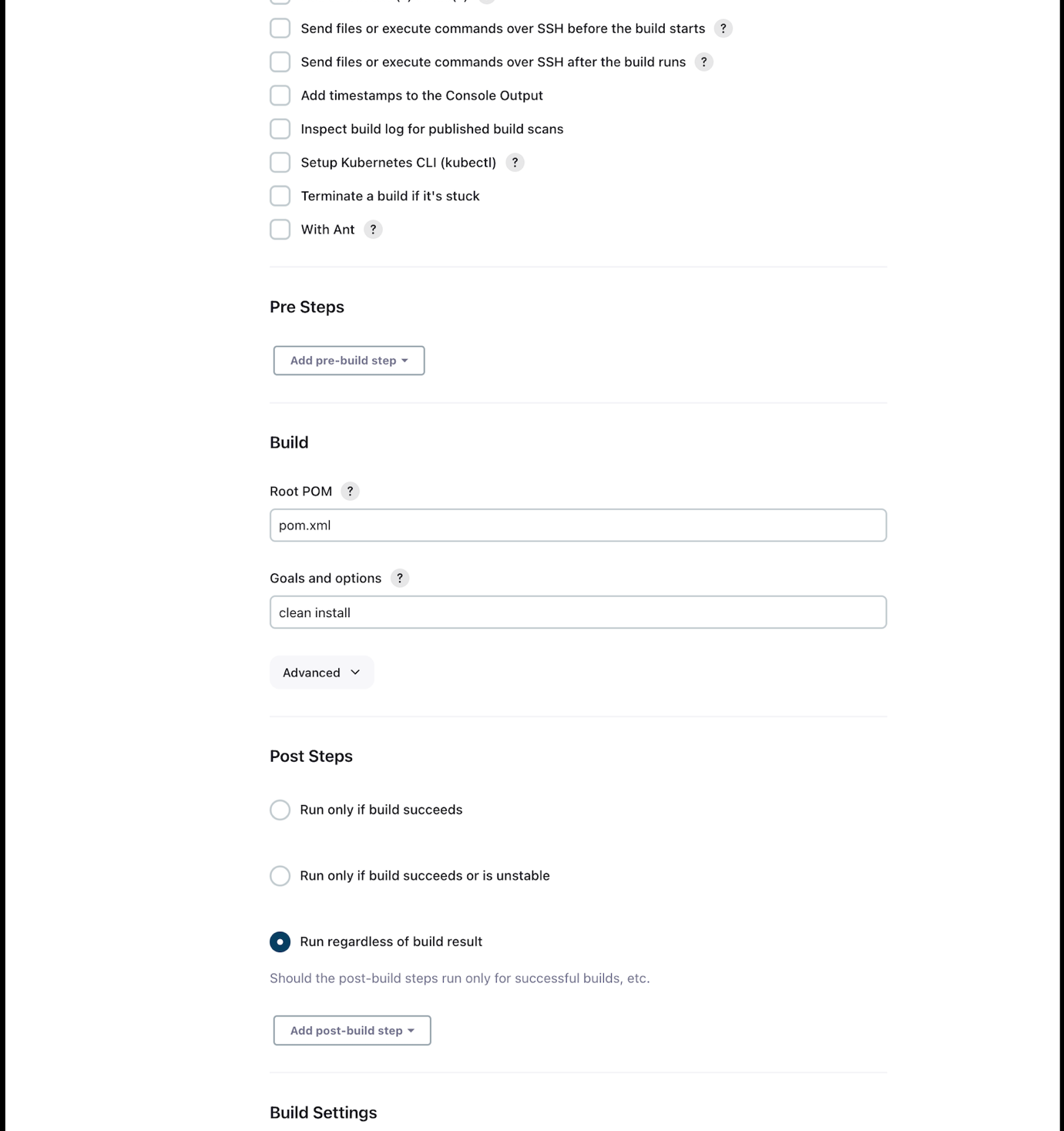
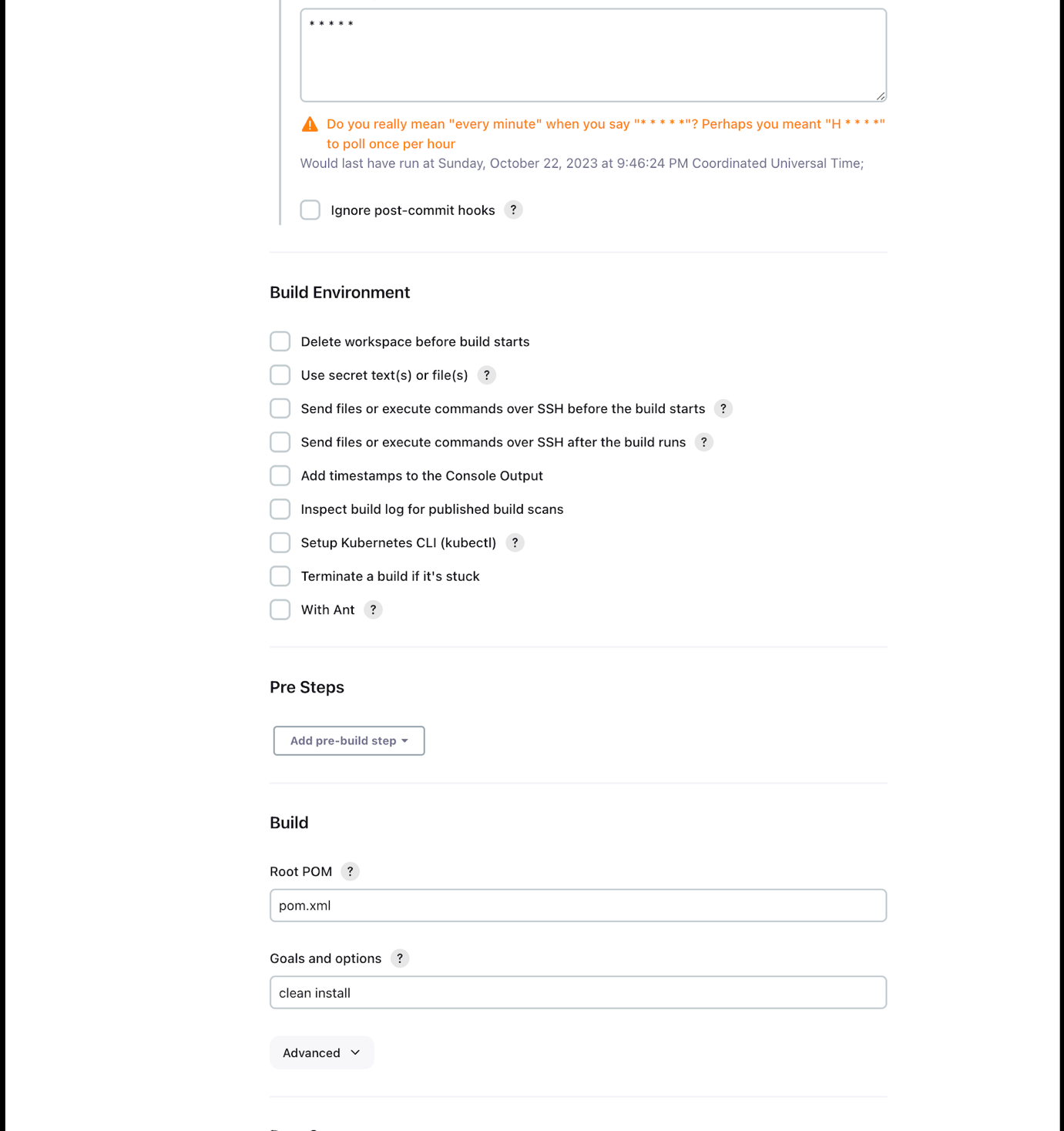
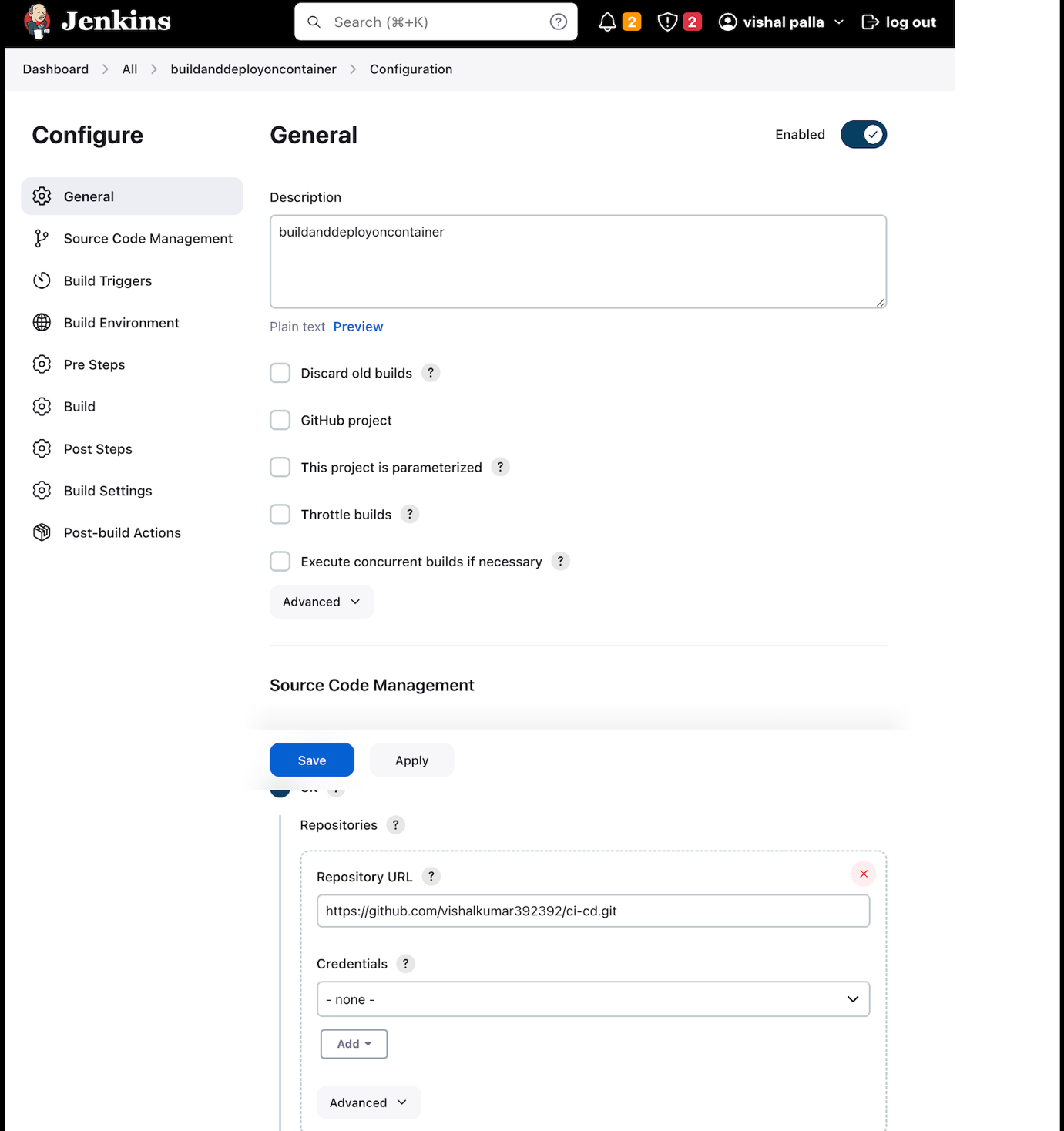
FROM openjdk:17

EXPOSE 8080

ADD ./\*.jar k8-0.0.1-SNAPSHOT.jar

ENTRYPOINT ["sh", "-c", "java -jar /k8-0.0.1-SNAPSHOT.jar"]

1. Create a new Jenkins job, Maven Project and configure the job as in below picture.



**Using Ansible to create containers**

**Ansible** is an open-source automation tool designed for various IT tasks, including configuration management, application deployment, and task automation. It simplifies and streamlines the management of IT infrastructure by allowing you to define infrastructure as code and automate the deployment and management of systems and applications.

Key features and concepts of Ansible include:

1. ***Agentless***: Ansible is agentless, meaning it doesn't require any agents to be installed on target systems. It communicates with remote systems over SSH for Unix-like systems and WinRM for Windows systems.

2. ***Playbooks***: Ansible uses YAML-based playbooks to define tasks and configurations. Playbooks describe the desired state of a system, and Ansible handles the process of making the system match that state.

3. ***Modules***: Ansible utilizes modules to execute tasks on remote hosts. Modules are pre-written code pieces that perform specific actions like installing software, configuring services, and more.

4. ***Inventory***: Ansible maintains an inventory of your infrastructure, which is a list of target systems where Ansible tasks and playbooks should be applied.

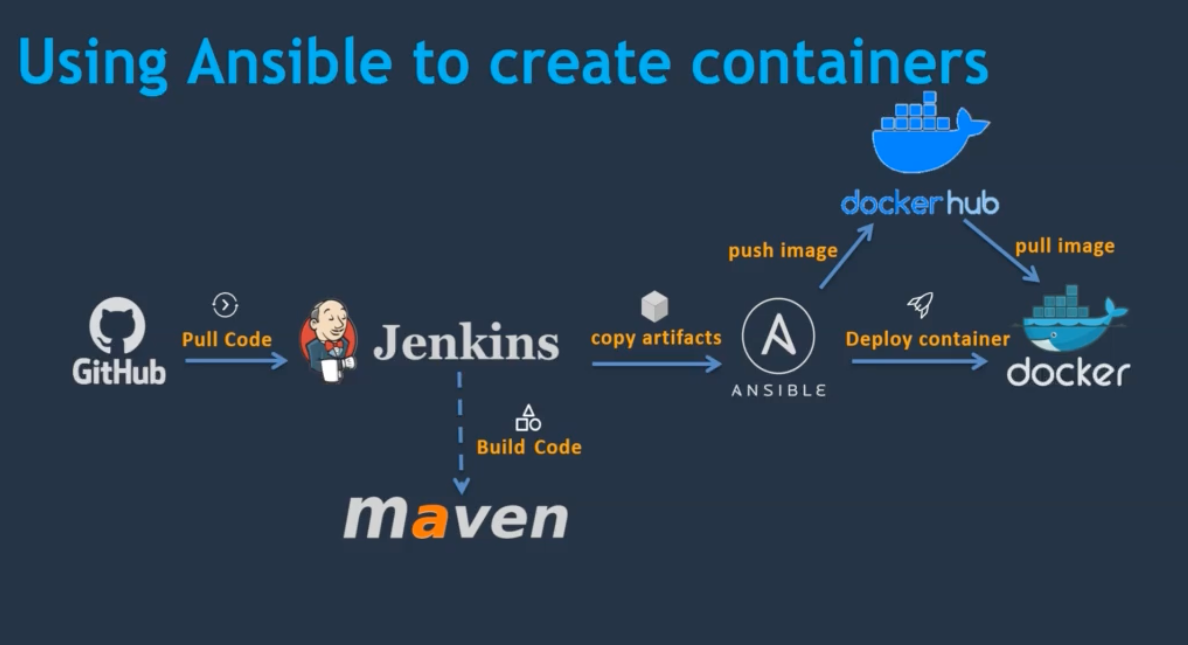
5. ***Idempotency***: Ansible enforces idempotency, ensuring that running a playbook multiple times will not change the system if the desired state is already achieved.

6. ***Integration***: Ansible can integrate with various cloud providers, containers, and other tools, making it a versatile automation solution.

7. ***Roles***: Roles allow you to organize playbooks and reuse common configurations across multiple playbooks.

Ansible is known for its simplicity and ease of use. It is widely used in DevOps practices for automating repetitive and complex tasks, which helps in reducing manual errors and improving the efficiency of IT operations.

**Our Goal**: *Once we made new commit to github, Jenkins will build code with help of maven and push the artifacts/jar to ansible, ansible will create a docker image from artifacts and will push it to dockerhub using playbook1 and once push is done, ansible will trigger dockerhost(ec2 instance) to create a container from newly pushed image and run the conatiner using another playbook2.*



Ansible installation:

1. Setup EC2 instance.
2. Setup hostname -> “vi /etc/hostname” and delete the preexisting text and give name and execute init 6 command wait for 2 mins and reopen the connection.
3. Create ansadmin user
4. Add user to sudoers file
5. Generate ssh keys
6. Enable password based login
7. Install ansible

**Create ansadmin user**

useradd ansadmin

passwd ansadmin

**Add the user to sudo’s file**, so user can execute any command without password

visudo

press shift+G and below this line ## Same thing without a password

ansadmin ALL=(ALL) NOPASSWD: ALL

Now we need to enable password based authentication

vi /etc/ssh/sshd\_config

And change PasswordAuthentication to yes

service sshd reload

login as ansadmin user

sudo su - ansadmin

**Create a new ssh key**

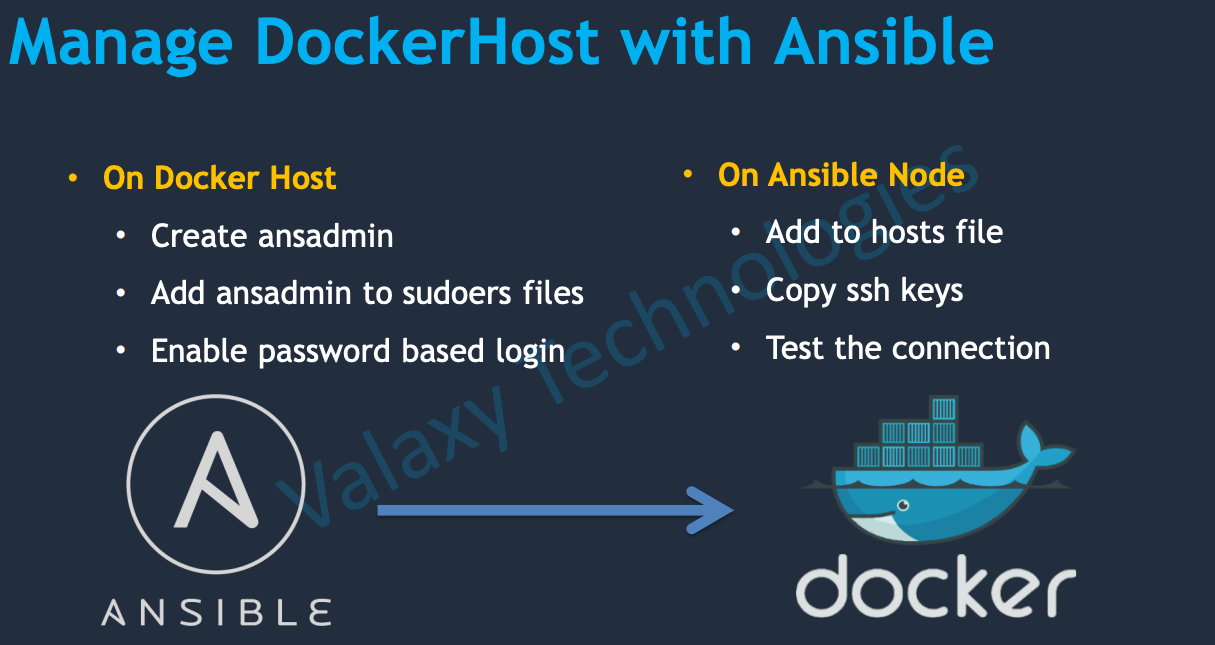
ssh-keygen

Install Anisble

sudo -i

amazon-linux-extras install ansible2

***Manage DockerHost with Ansible***



**On Docker Host:**

**Create ansadmin user**

useradd ansadmin

passwd ansadmin

**Add the user to sudo’s file**, so user can execute any command without password

visudo

press shift+G and below this line ## Same thing without a password

ansadmin ALL=(ALL) NOPASSWD: ALL

Now we need to enable password based authentication

vi /etc/ssh/sshd\_config

And change PasswordAuthentication to yes

service sshd reload

**On Ansible server:**

Need to add docker host server as a manager node in ansible.

Go to Ansible server and login as a root user and configure in inventory file.

Vi /etc/ansible/hosts

We need to modify the above file. Before that we need private ip of docker-host server, go to docker host ec2 instance and do “ifconfig”, and copy the inet address(172.31.17.85).

eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 9001

inet 172.31.17.85 netmask 255.255.240.0 broadcast 172.31.31.255

inet6 fe80::89b:b8ff:fe44:b20b prefixlen 64 scopeid 0x20<link>

ether 0a:9b:b8:44:b2:0b txqueuelen 1000 (Ethernet)

RX packets 4923 bytes 3681830 (3.5 MiB)

RX errors 0 dropped 0 overruns 0 frame 0

TX packets 2837 bytes 344916 (336.8 KiB)

In vi /etc/ansible/hosts add the inet docker ip

**Now we need to copy ansadmin user keys to ansadmin target user(docker host) so that password less authentication will enabled.**

For that change the user type to ansadmin “sudo su - ansadmin”.

Execute command ll -a to see the .ssh/ folder and keys are present inside them.

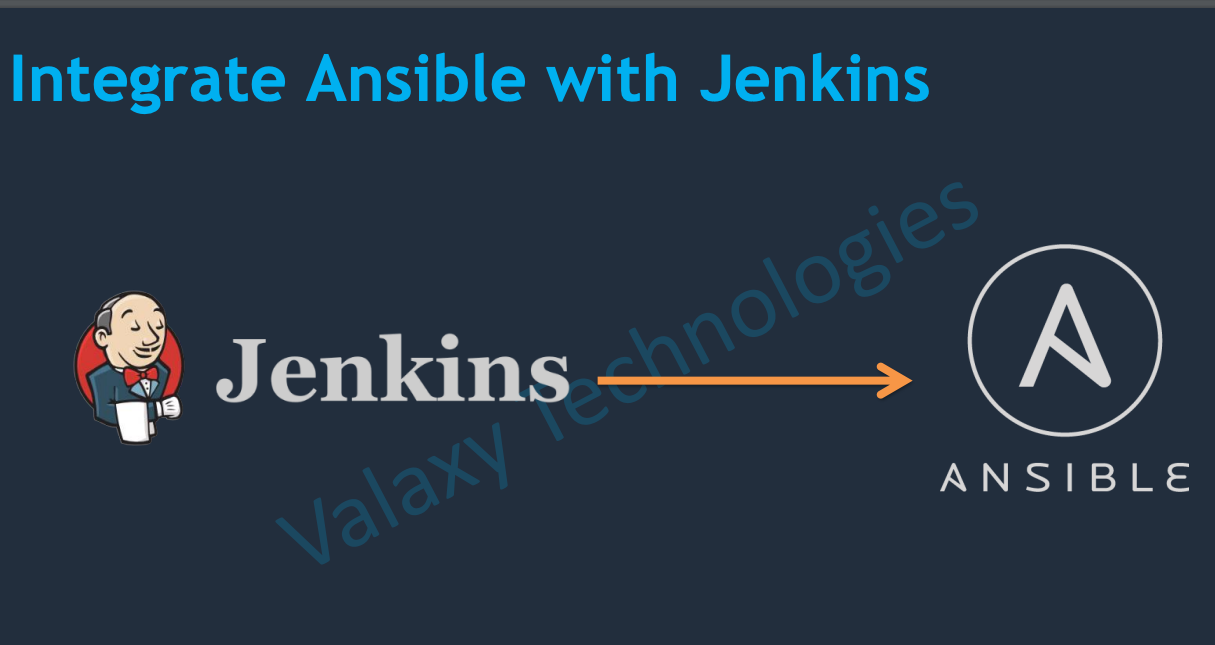
To copy the keys to target(docker host ec2 server), execute

ssh-copy-id 172.32.17.85

“ssh-copy-id 172.31.17.85 ”

To check whether the connection is succesfull or not, execute “ansible all -m ping”.

***Integrate Ansible with Jenkins***



The way we integrated docker host to jenkins, in the same way we need to integrate ansible to jenkins.

1. In ansible server we need to add a user, password authentication.(already did)
2. Install Publish over SSH plugin, Connect ansible server to jenkins in system configuration my adding a new ssh server.



**Creating a Ansible playbook which create the docker image.**

1. Install Docker in ansible-server and To create a playbook we need to run anisble playbook command. However, whenever we run this playbook, we need to provide where we are running this Ansible playbook i.e inventory or the file name, ansible-playbook -i inventory\_file playbook.yml , by default it will take the default inventory which “cat /etc/ansible/hosts”.
2. So we can add ansible private IP on the default inventory
   1. vi /etc/ansible/hosts
   2. [ansible] 172.6.7.896
3. But to make work with Ansible you should copy your SSH keys into your ansible server itslef. Execute with ‘ansible all -a uptime’, we will see the error.
4. ssh-copy-id 172.6.7.896
5. Create a file in /opt/docker directory.
6. vi regapp.yaml
7. ---
8. - hosts: ansible
9. tasks:
10. - name: create docker image
11. command: docker build -t regapp:latest .
12. args:
13. chdir: /opt/docker
14. Execute the command: ansible-playbook regapp.yml
15. Execute docker login cmd and provide username and password.
16. We can push any image for testing.
17. ---
18. - hosts: ansible
19. tasks:
20. - name: create docker image
21. command: docker build -t regapp:latest .
22. args:
23. chdir: /opt/docker
24. - name: create a tag to push image to dockerhub
25. command: docker tag regapp:latest vishalkumar392/regapp:latest
26. - name: push docker image
27. command: docker push vishalkumar392/regapp:latest

Now create another play book(deploy\_regapp.yml) to run the container in docker host server which pull the image from dockerhub and run the image in docker host.

[ansadmin@ansible-server docker]$ cat deploy\_regapp.yml

---

- hosts: dockerhost

tasks:

- name: stop existing container

command: docker stop regapp-server

ignore\_errors: yes

- name: remove the container

command: docker rm regapp-server

ignore\_errors: yes

- name: remove image

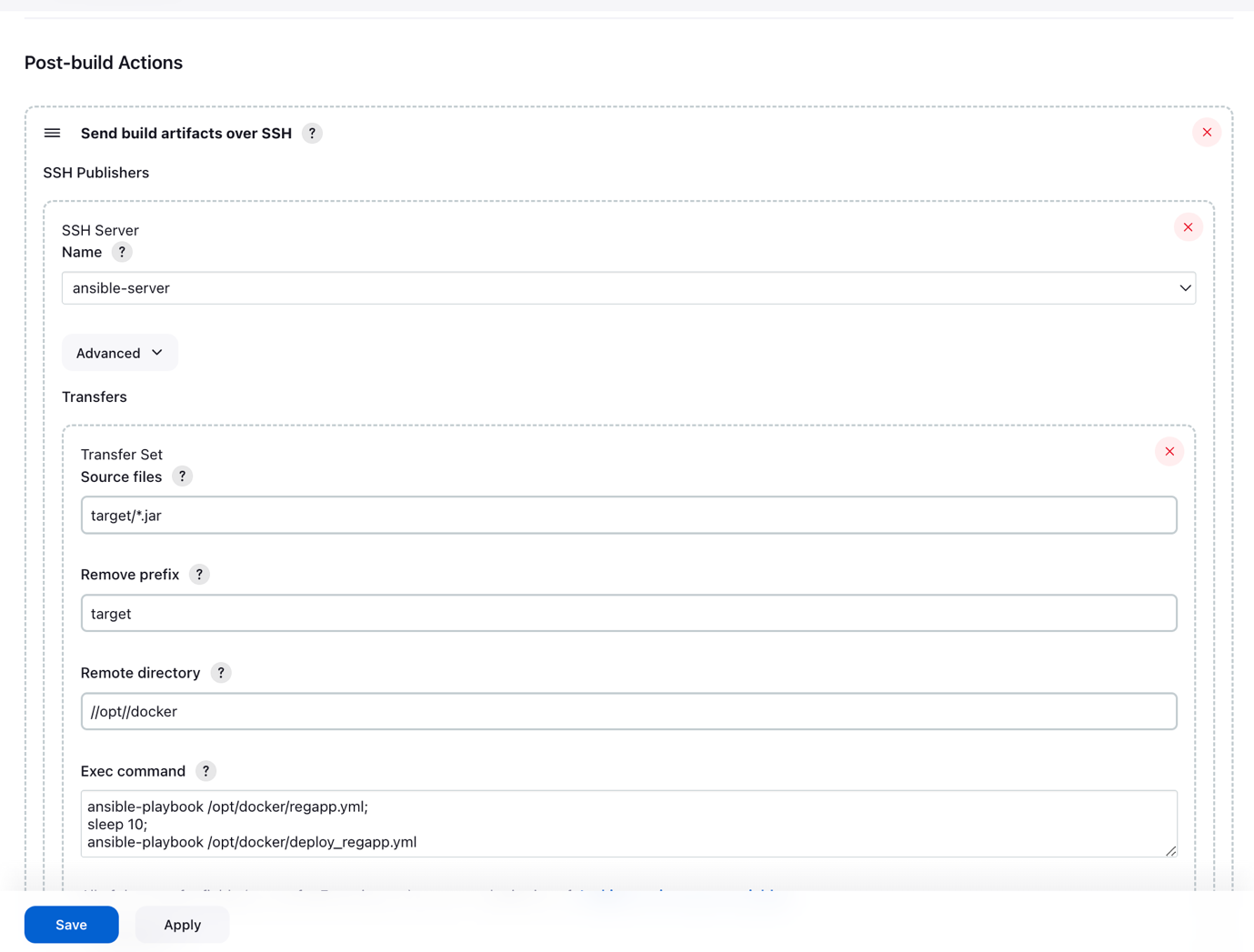
command: docker rmi vishalkumar392/regapp:latest

ignore\_errors: yes

- name: create container

command: docker run -d --name regapp-server -p 8080:8080 vishalkumar392/regapp:latest

Configure the both playbooks in jenkins configuration.



regapp.yml builds the docker image in ansible server and push it to docker hub.

deploy\_regapp.yml run the docker container from image in the docker host ec2 server.

Github-Jenkins-Ansible-Docker-Kubernetes

**Our Goal**: *Once we made new commit to github, Jenkins will build code with help of maven and push the artifacts/jar to ansible and some commands to execute in ansible, ansible will create a docker image from artifacts and will push it to dockerhub using playbook1 and once push is done, ansible will trigger the playbook2, where we declared the host to be kubernetes bootstrap server. This playbook2 contains commands related to kubernetes to create a deployment and service from bootstrap server to EKS cluster.*

**Install kubernetes Cluster in Aws:**

We can install cluster in many ways. But we are following creating via eksctl. [Instructor github link to create the EKS cluster using eksctl.](https://github.com/yankils/Simple-DevOps-Project/blob/master/Kubernetes/kubernetes_setup_using_eksctl.md)

1. First create a ec2 instance in aws.
2. Install [AWS CLI](https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html) latest version.
3. [Setup kubectl, Setup eksctl](https://docs.aws.amazon.com/eks/latest/userguide/getting-started.html).
4. Create a IAM role. Assign Permission to the role like IAMFullAccess, AmazonEc2FullAccess, AwsCloudFormationFullAccess, AdminstrationAccess. And attach the iam role to ec2 instance-> Action/Security/Modify IAM
5. Create a cluster. “eksctl create cluster --name vishal-cluster \

--region us-east-1 \

--node-type t2.small \” , cluster creation will take a lot of time. You can see in the cloudformation template is being created and 2 ec2 instances will be created.

6. Delete the cluster if you want “eksctl delete cluster vishal-cluster --region us-east-1”

7. try commands, kubectl get nodes, etc to check status of cluster.

**create a deployment file with name “deployment.yml” in bootstarp server.**

apiVersion: apps/v1

kind: Deployment

metadata:

name: vishal-regapp

labels:

app: regapp

spec:

replicas: 3

selector:

matchLabels:

app: regapp

template:

metadata:

labels:

app: regapp

spec:

containers:

- name: regapp

image: vishalkumar392/regapp

imagePullPolicy: Always

ports:

- containerPort: 8080

strategy:

type: RollingUpdate

rollingUpdate:

maxSurge: 1

**Create a service file with name “service.yml” in bootstrap server.**

apiVersion: v1

kind: Service

metadata:

name: vishal-service

labels:

app: regapp

spec:

selector:

app: regapp

ports:

- port: 8080

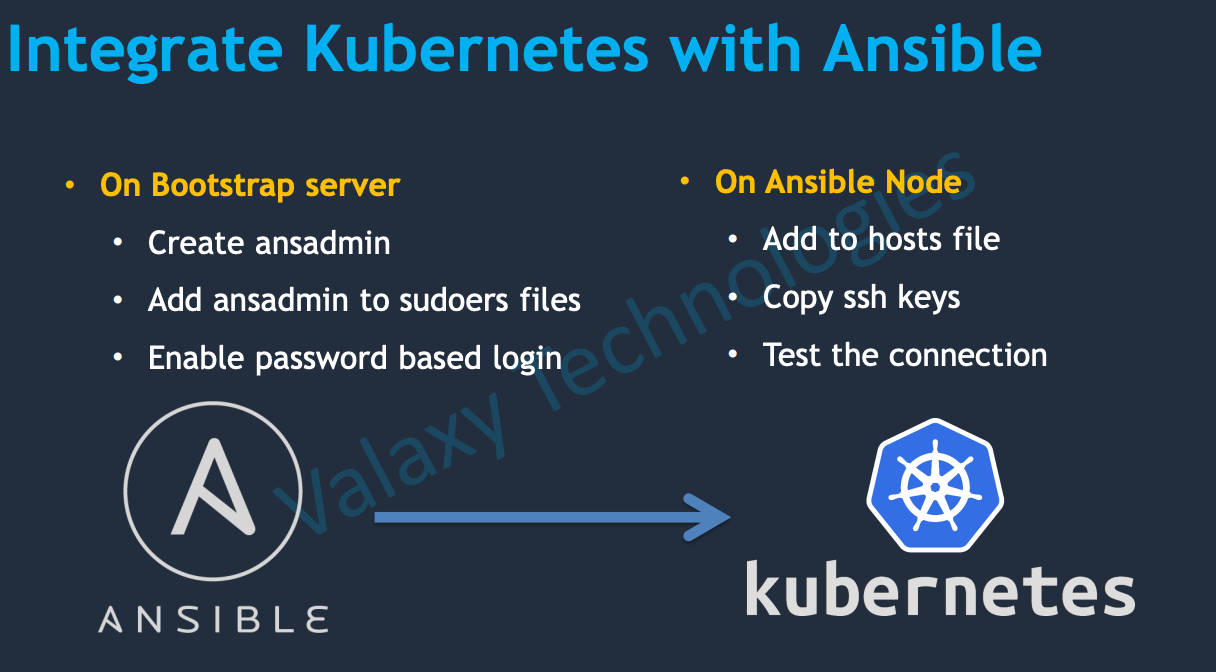
targetPort: 8080

type: LoadBalancer

If we try to run *kubectl apply -f deployment.yml*, a new deployment will create with pods.

If we try to run *kubectl apply -f service.yml*, a new service will create with loadbalancer.

Integrating Kubernetes with Ansible



**On Bootstrap server:**

Create user ansadmin

useradd ansadmin

passwd ansadmin

passwd root

execute command

visudo

in the file add

ansadmin ALL=(ALL) NOPASSWD: ALL

enable password based authentication, execute below command

vi /etc/ssh/sshd\_config

comment PasswordAuthentication no

and uncomment PasswordAuthentication yes

execute below command

service sshd reload

Now we can able to log into bootstrap instance as ansadmin user

On Ansible server

sudo su – ansadmin

cd /opt/docker

Now we need to add bootstrap server to inventory/host file. In this case we will create a host file with name “hosts” and will add the bootstrap private ip in it.

vi hosts

#in the file please add below commands

[kubernetes]

172.32.45.345

[ansible]

172.56.78.987

Copy keys to bootsrap server

ssh-copy-id 172.32.45.345

Check the connection

ansible -i hosts all -a uptime

Need to copy ansadmin keys to root user to execute the playbook as root user.

ssh-copy-id root@172.32.45.345

Now create a playbook with name kube\_deploy.yml and below content in it.

---

- hosts: kubernetes

user: root

tasks:

- name: deploy regapp on kubernetes

command: kubectl apply -f deployment.yml

- name: deploy regapp on kubernetes

command: kubectl apply -f service.yml

- name: update deployment with new pods if image updated in docker hub

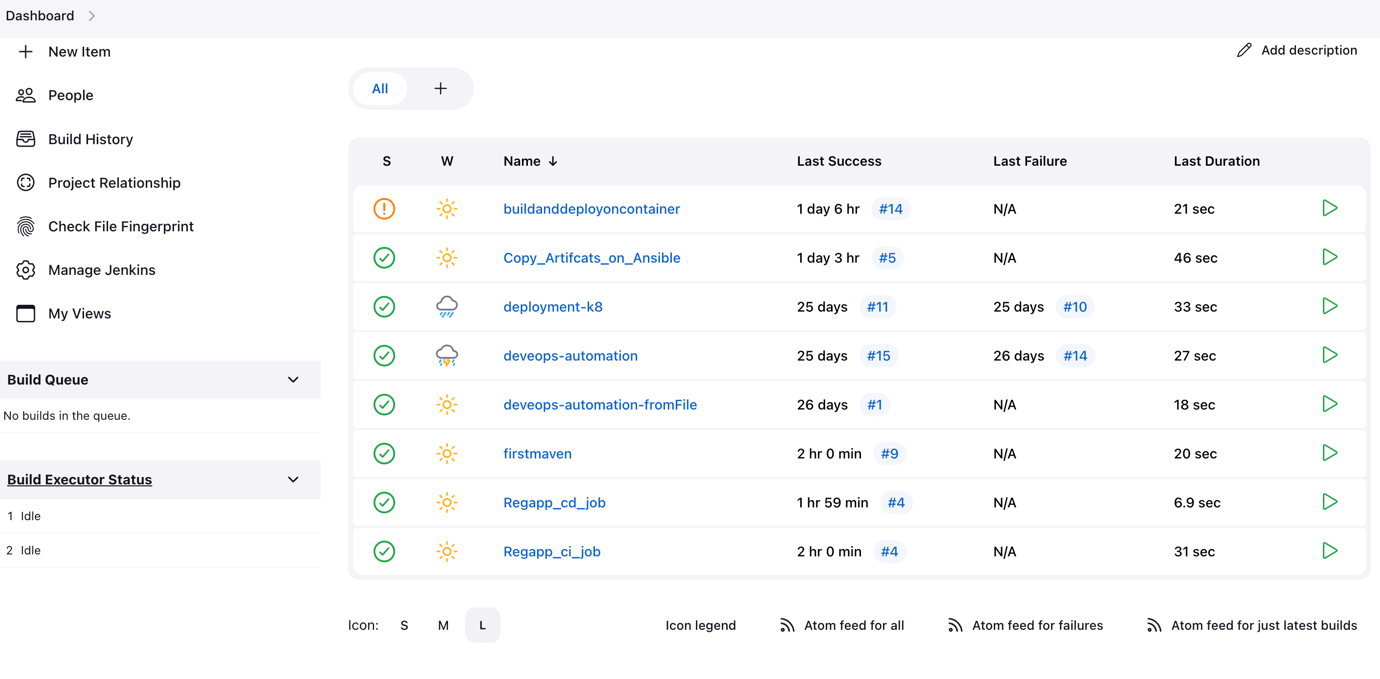
command: kubectl rollout restart deployment.apps/vishal-regapp

Now execute the playbook

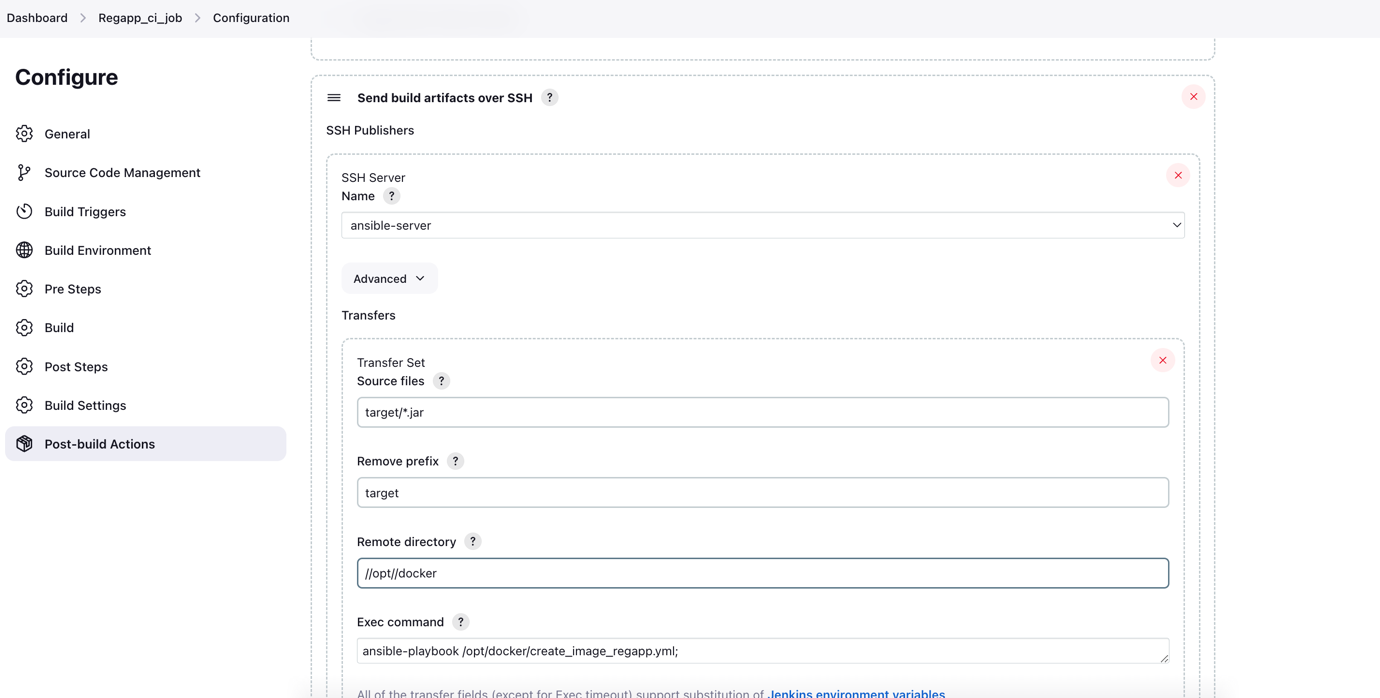
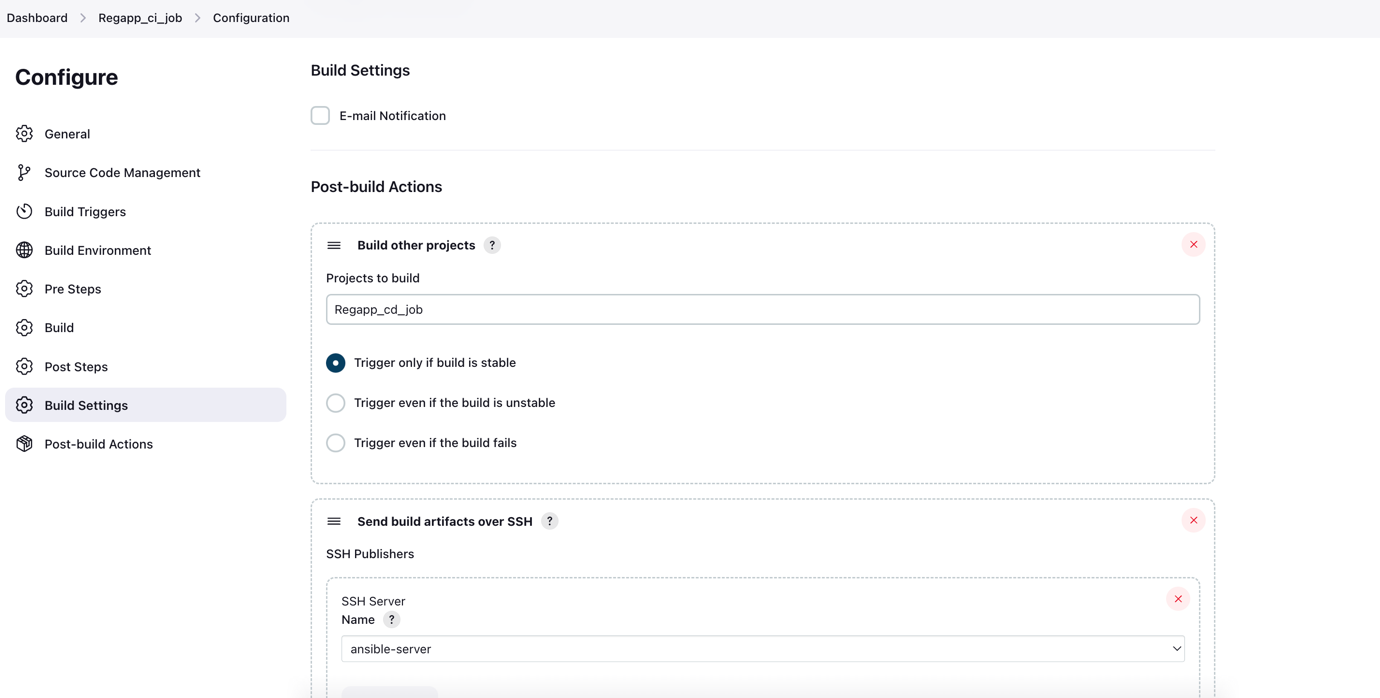
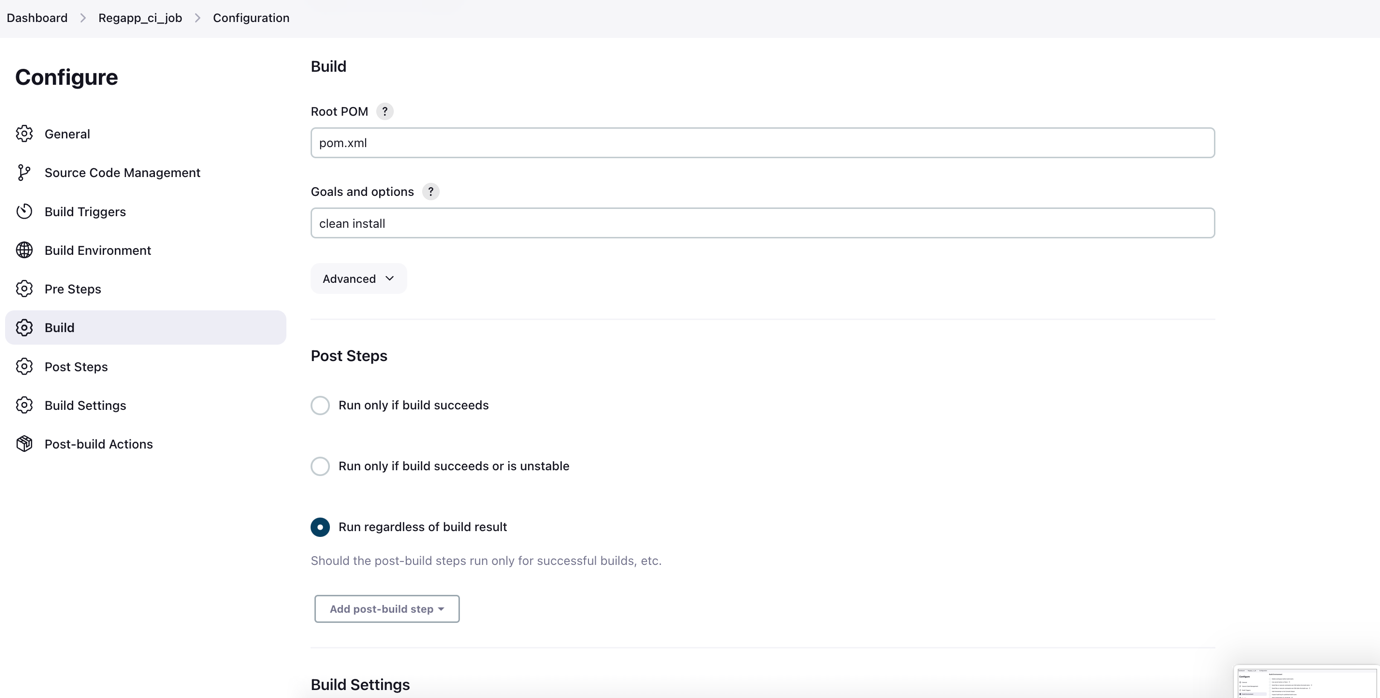
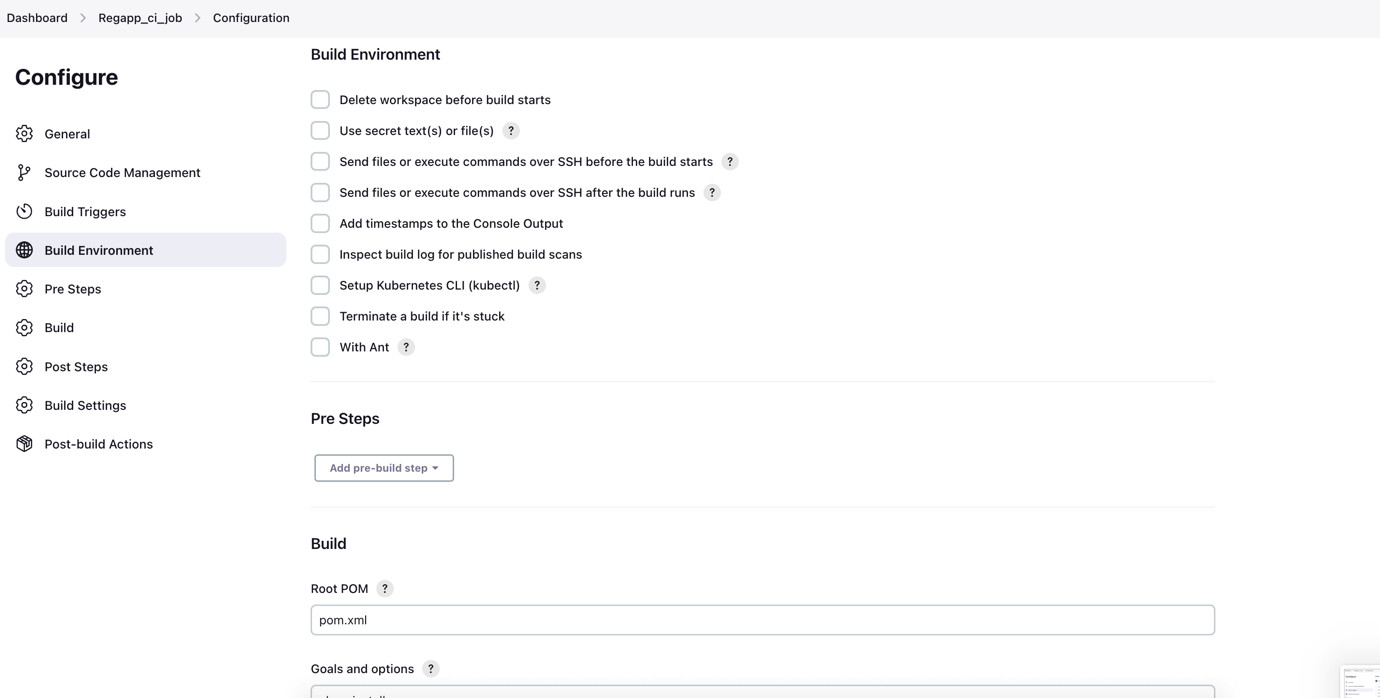
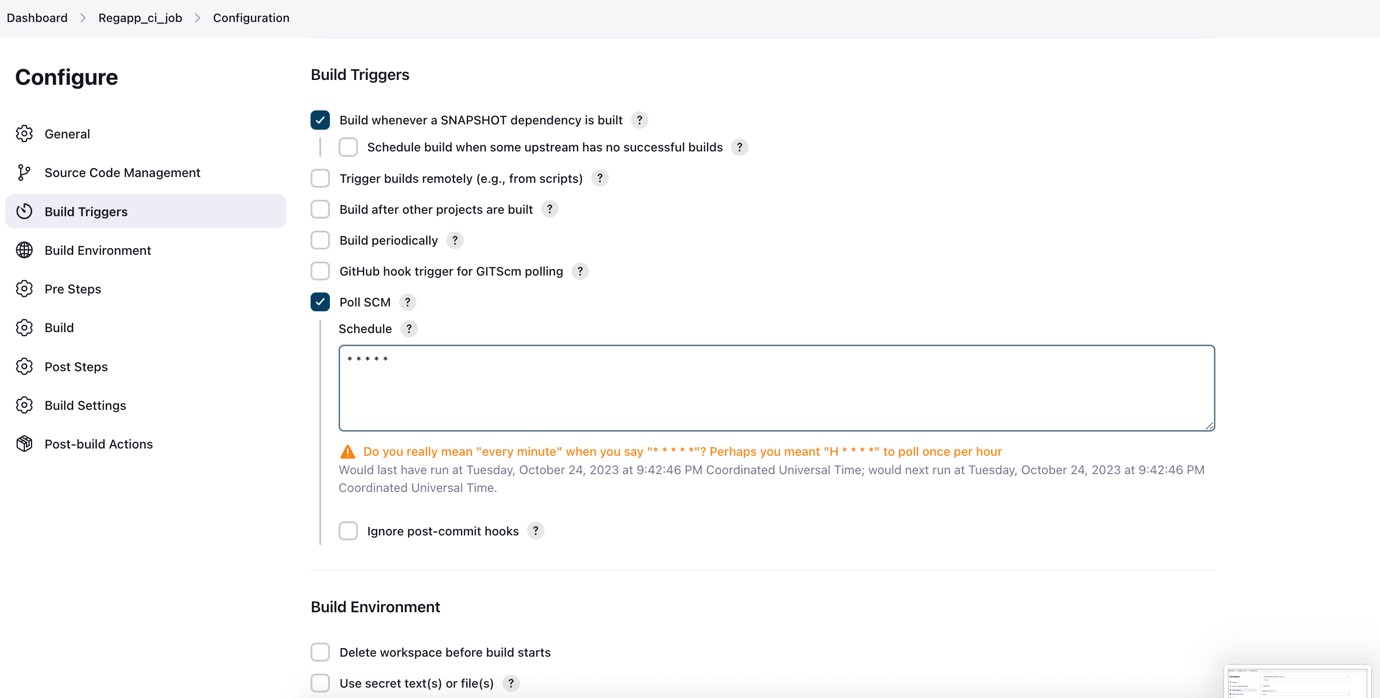
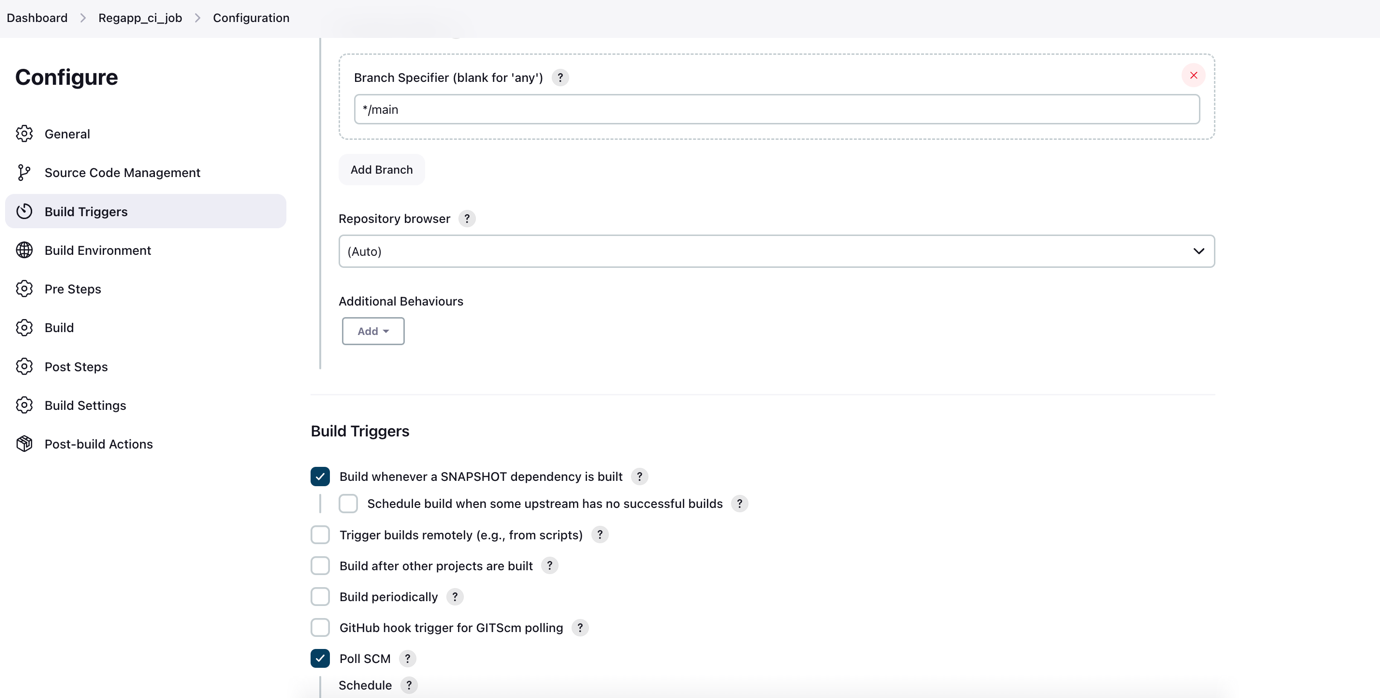
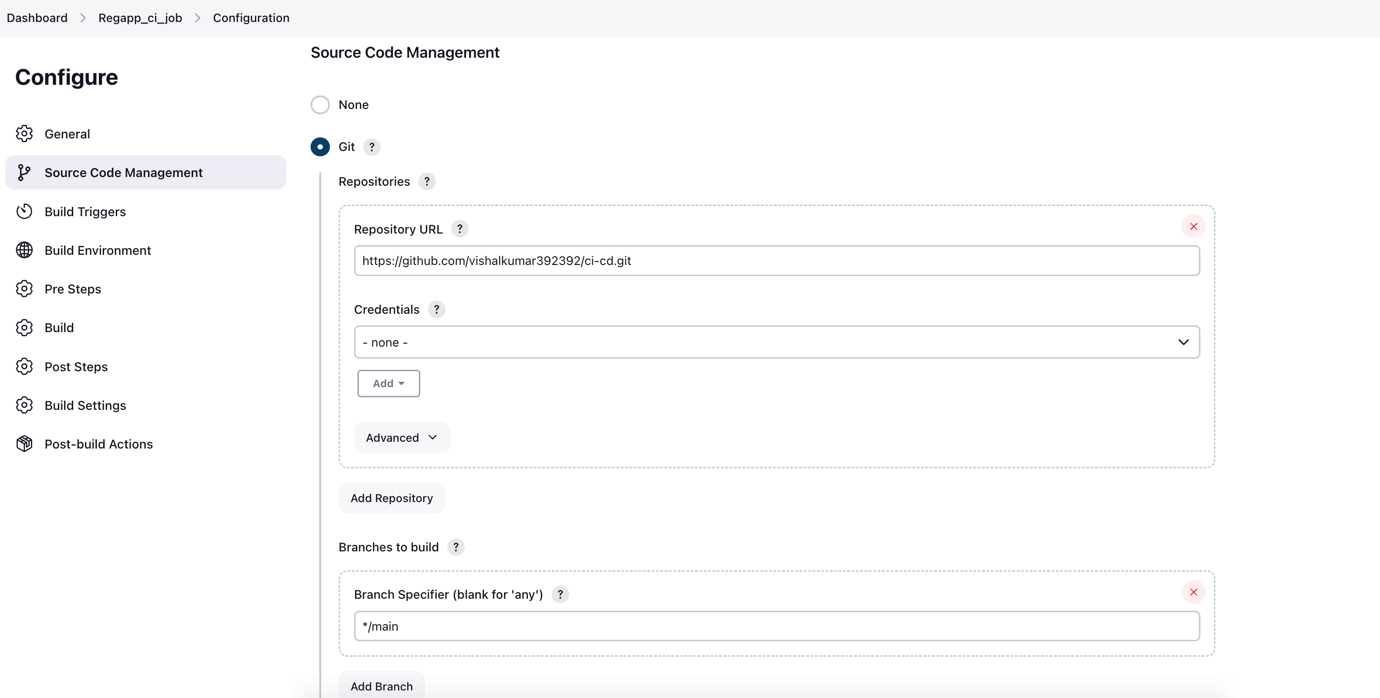
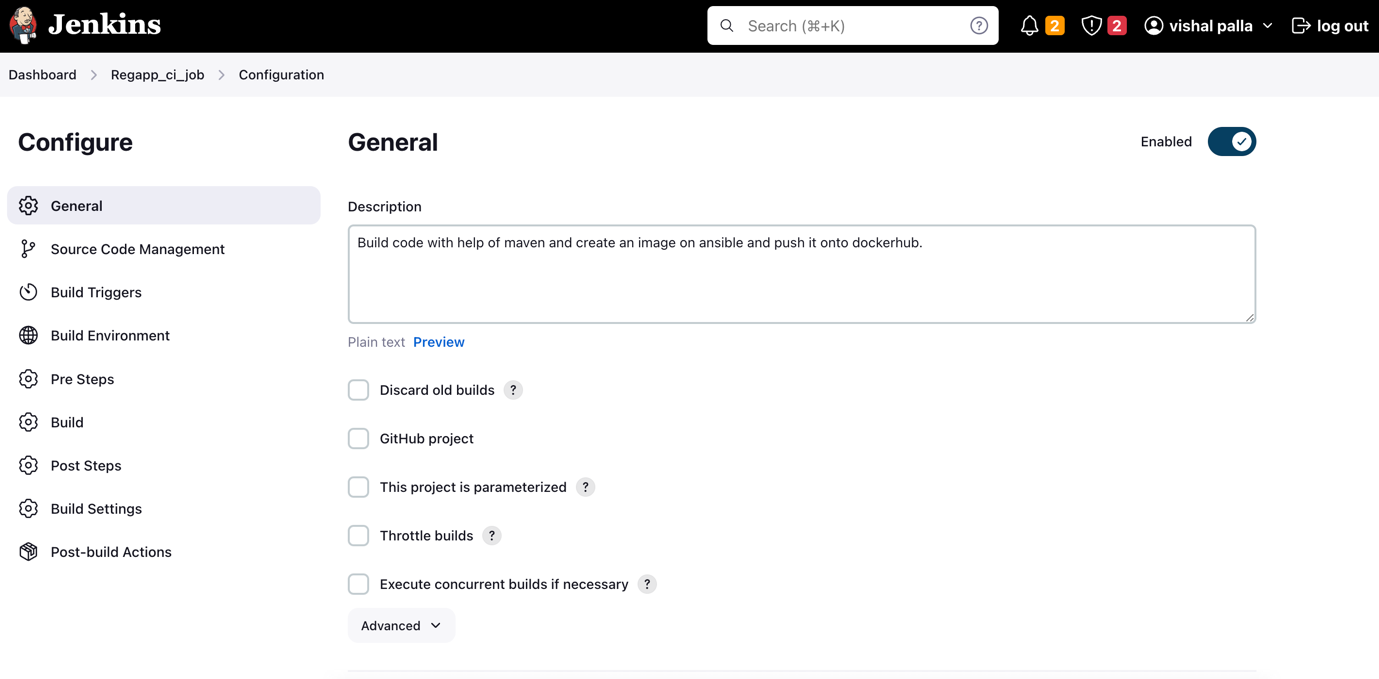
ansible-playbook -i /opt/docker/hosts kube\_deploy.yml

**CI/CD – Jenkins/Ansible/Kubernetes**

Open Jenkins and create a new item for CI







Configuration for CI item is finished.

In the above screenshot, we can observe a post build action, which will trigger Regapp\_cd\_job once current item succeded.

**Creating a new item for CD:**

Open Jenkins and create a new item for CD

