5. On the **Select Certificate** page, do one of the following:

- If you have a certificate from AWS Certificate Manager, select **Choose an existing certificate from AWS Certificate Manager (ACM)**, select the certificate from **Certificate**, and then choose **Save**.

Note: This option is available only in regions that support AWS Certificate Manager.

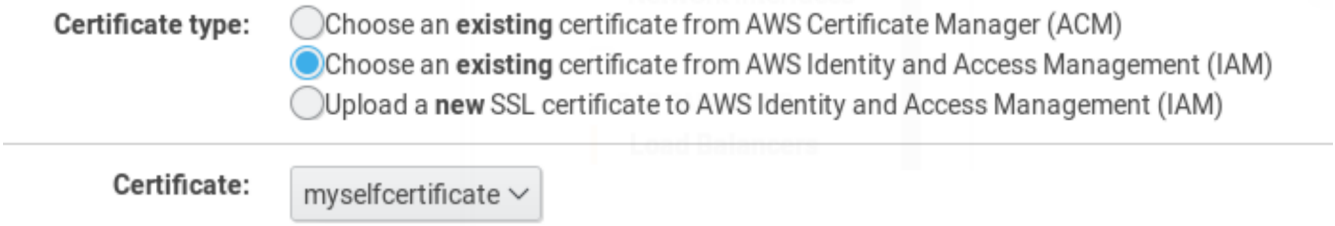
- If you have already uploaded a certificate using IAM, select **Choose an existing certificate from AWS Identity and Access Management (IAM)**, select the certificate from **Certificate**, and then choose **Save**.
- If you have an SSL certificate to upload, select **Upload a new SSL Certificate to AWS Identity and Access Management (IAM)**. Enter a name for the certificate, copy the required information to the form, and then choose **Save**. Note that the certificate chain is not required if the certificate is a self-signed certificate.

Method 1: **Using awscli (Aws Command Line Interface) upload to IAM**

Command:

```
[root@ip-172-31-27-189 ~]# aws iam upload-server-certificate --server-certificate-name mys
elfcertificate --certificate-body file://mycertificate.pem --private-key file://privatekey
.pem
{
    "ServerCertificateMetadata": {
        "ServerCertificateId": "ASCAJDAVWTXEVWR7RPHY2",
        "ServerCertificateName": "myselfcertificate",
        "Expiration": "2018-06-11T04:44:50Z",
        "Path": "/",
        "Arn": "arn:aws:iam::982673618411:server-certificate/myselfcertificate",
        "UploadDate": "2017-06-11T04:57:17.458Z"
    }
}
```

aws iam upload-server-certificate –server-certificate-name myselfcertificate –certificate-body file://mycertificate.pem –private-key file://privatekey.pem



Note: in this certificate Stored under IAM existing certificate option

Method 2: Upload a new SSL Certificate to AWS Identity and Access Management (IAM) using cat command we can see the key something like this

cat privatekey.pem

----------BEGIN RSA PRIVATE KEY----------

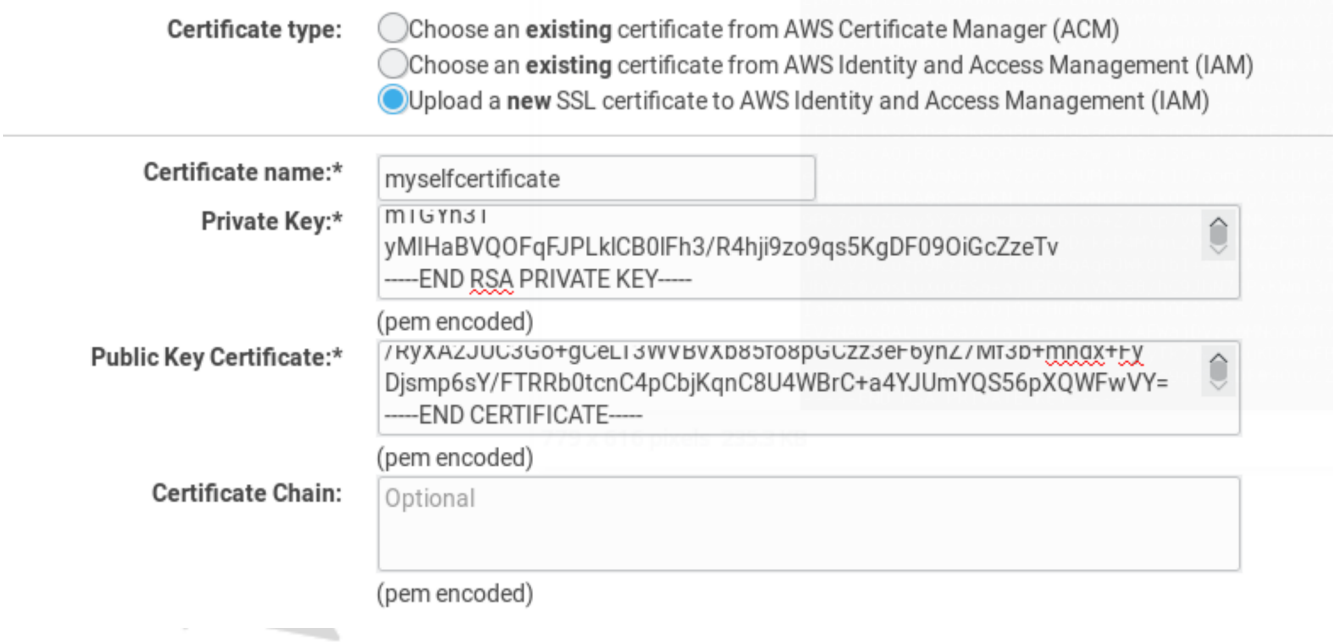
----------END RSA PRIVATE KEY----------

cat mycertificate.pem

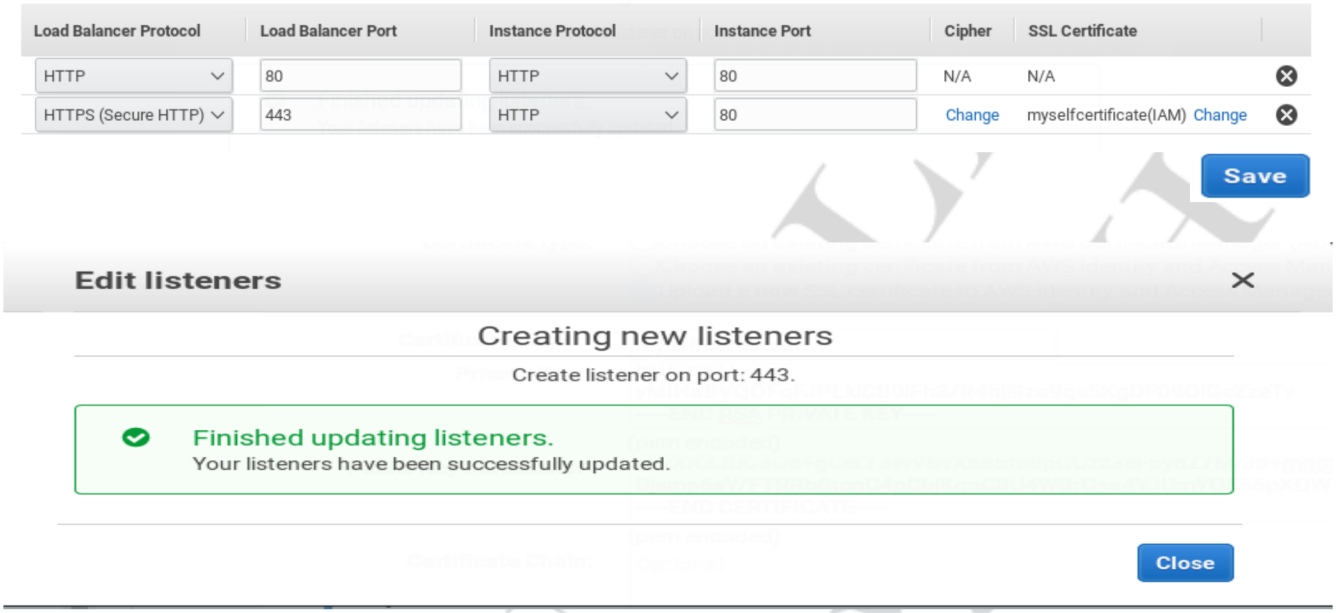
----------BEGIN CERTIFICATE----------

----------END CERTIFICATE----------

- Click on SAVE → we can see like this
- Enter the certificate name
- Copy the privatekey.pem file and Paste in Private Key Box
- Copy the mycertificate.pem file and Paste in Public Key Certificate Box



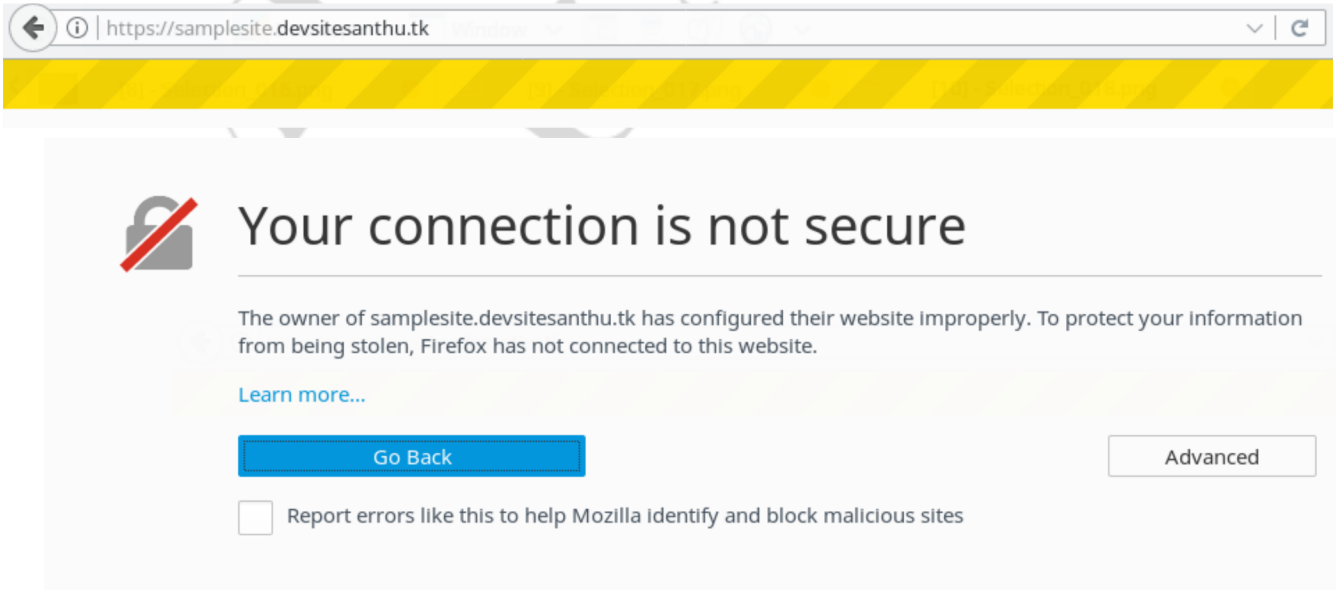
Click on Save →



Next Click on Close

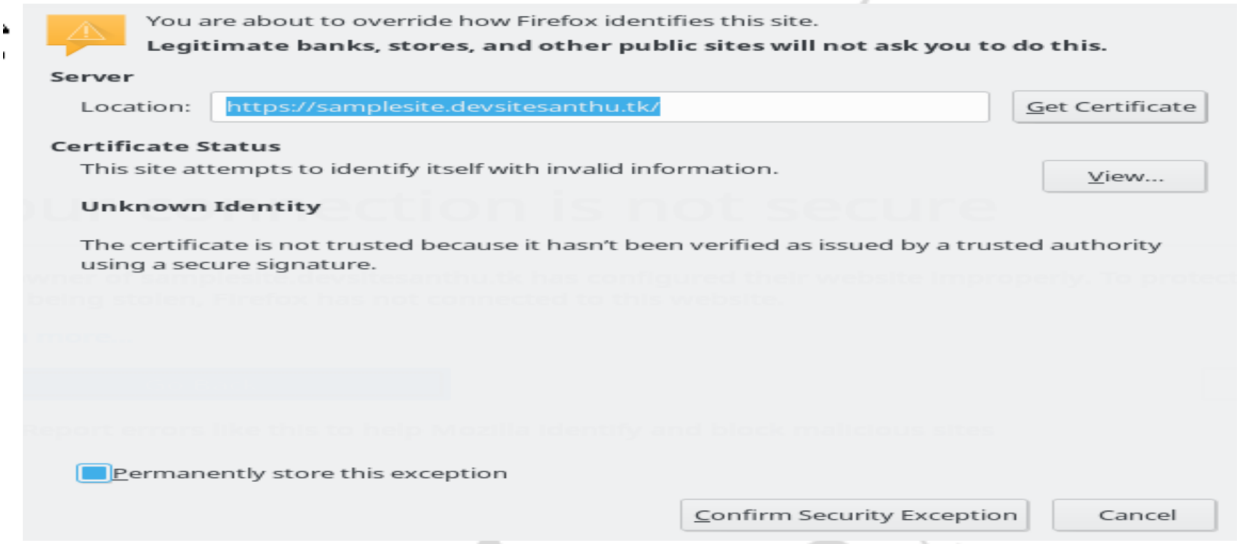
8. Verification

Step5: Now Try to open website with https protocol



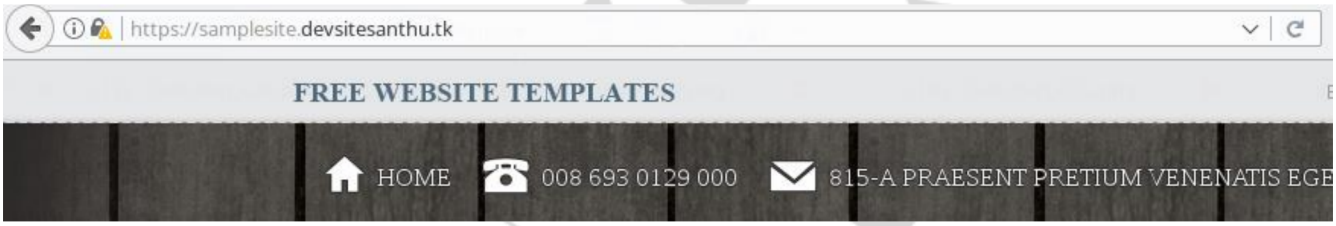
Browser knows the website under self signed or issued by CA.

Next click on Advanced Option → click on AddException



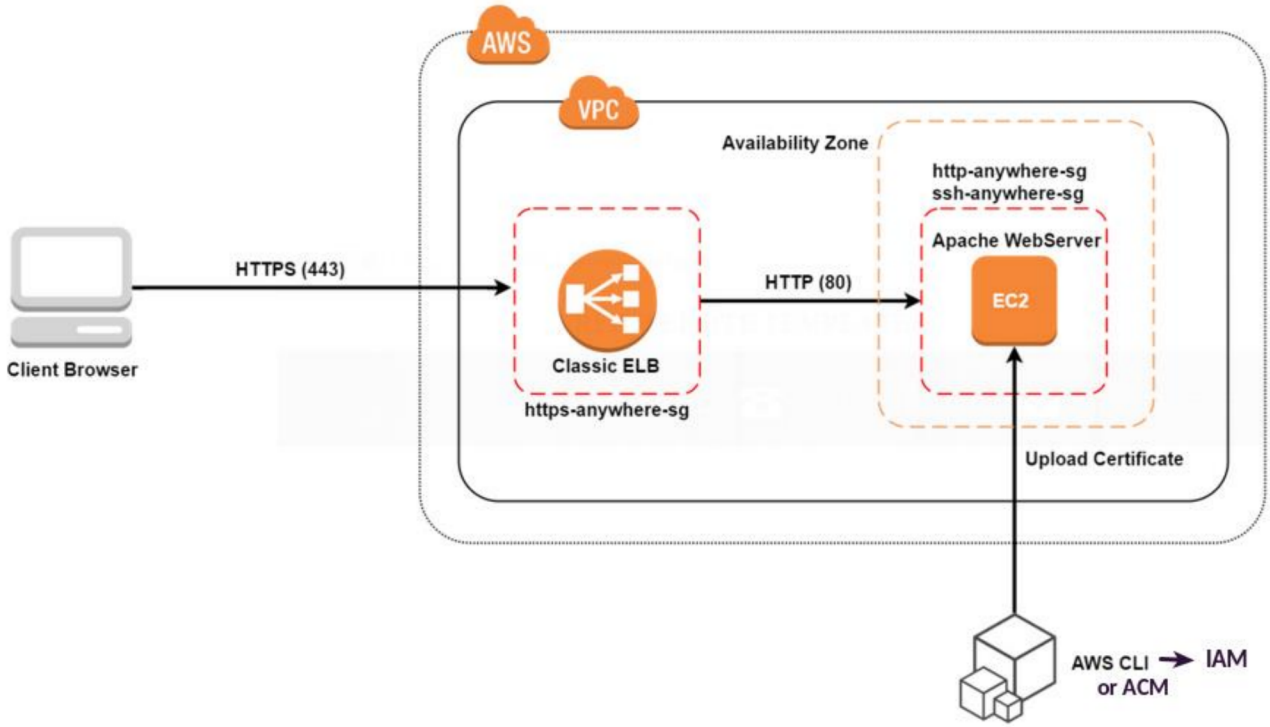
If we want to see the details of issued CA. Cick on view

Click **on Close → ConfirmSecurityException → Now your website is opened under https**



Now Everything is done site is under works HTTPS Protocol also.

9. Algorithm Diagram:



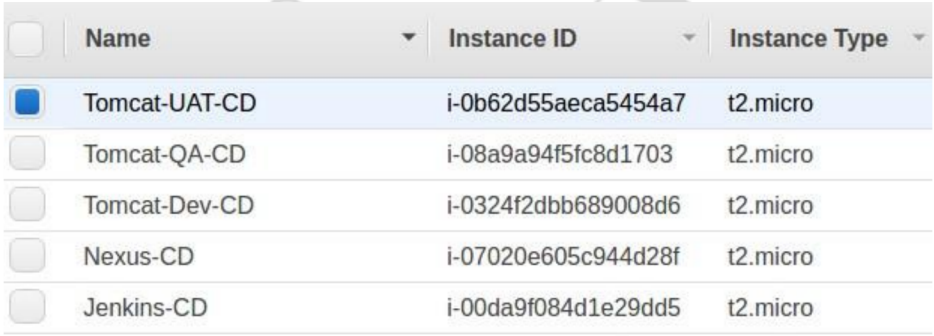
XIX. A Sample Continuous Delivery Project.

It's a very simple CD project but will give you an idea of Continuous Delivery Pipeline. Everything has been kept to very minimum to reduce the complexity of project.

PREREQUISITES:

Infrastructure can be setup with vagrant VM's or AWS EC2 instances.

- One vm/ec2 instance Jenkins server (Ubuntu).
- Three vm's/ec2 instances for deploying artefacts, tomcat node(Centos-6).
- One vm/ec2 instance for nexus server(Centos-6).



Name	Instance ID	Instance Type
Tomcat-UAT-CD	i-0b62d55aeca5454a7	t2.micro
Tomcat-QA-CD	i-08a9a94f5fc8d1703	t2.micro
Tomcat-Dev-CD	i-0324f2dbb689008d6	t2.micro
Nexus-CD	i-07020e605c944d28f	t2.micro
Jenkins-CD	i-00da9f084d1e29dd5	t2.micro

NOTE: If using EC2 instances, refer to the last page for Security group rules of Jenkins, Nexus & Tomcat Nodes.

1. Jenkins server on ubuntu system: -

Vagrant vm or Ec2 instance.

- Install JDK 1.7 on Jenkins server

  # sudo add-apt-repository ppa:openjdk-r/ppa

  # sudo apt-get update

  # sudo apt-get install openjdk-7-jdk

- Install Jenkins by following steps mentioned in below link.
  https://wiki.jenkins-ci.org/display/JENKINS/Installing+Jenkins+on+Ubuntu

- Install git client and maven in Jenkins server.
  # sudo apt-get install git
  # sudo apt-get install maven

- Install Ansible on Jenkins server.
  https://docs.ansible.com/ansible/intro_installation.html#latest-releases-via-apt-ubuntu

2. Tomcat node on centos: -

- Three tomcat nodes are required for Dev, QA and UAT deployment.
- Vagrant centos vm or Ec2 instance with Centos 6 OS
- Installation of tomcat and configuration would be taken care by ansible playbook, so we just need three centos vm's.
- Please note its IP and credentials which we are going to add in ansible inventory file.

3. Nexus server on Centos.

- Nexus installation steps below.
- Create a centos vagrant vm/Ec2 instance with Centos 6 OS and login into it.

```
# yum install -y java-1.8.0-openjdk.x86_64 vim wget
# yum install -y java-1.8.0-openjdk-devel.x86_64
# export RUN_AS_USER=root
# wget http://www.sonatype.org/downloads/nexus-latest-bundle.tar.gz
# sudo cp nexus-latest-bundle.tar.gz /usr/local/
# cd /usr/local
# sudo tar xvzf nexus-latest-bundle.tar.gz
# sudo ln -s nexus-2.13.0-01 nexus
# /usr/local/nexus/bin/nexus start
```

- From browser hit URL <Nexus server IP>:8081/nexus.
- Click login button and enter the credentials. (admin/admin123)
- Create hosted repository named “gameoflife-repo” with all default settings.

4. In total we should have below mentioned vm/instances ready.

- One vm/instance Jenkins server (Ubuntu).

- Three vm's for deploying artefacts, tomcat node(Centos-6).
- One vm for nexus server(Centos-6).

JENKINS DEV JOB SETUP.

1. Login to Jenkins server.

2. Prerequisites for the job execution mentioned below.

Install plugins: -

- git
- zentimestamp
- Parameterized Trigger plugin
- Nexus
- Ansible

Configure plugin: - Setup zentimestamp variable from Jenkins global settings page.
Manage Jenkins → Configure System → Global properties →
Check mark Date pattern for the BUILD_TIMESTAMP → Enter value

yyyyMMddHHmm

3.Create Jenkins Jobs

Create three empty Jenkins job (Freestyle project) with below mentioned name and run all three jobs. Once you run these three empty jobs it's going to fail but that's okay we want the jobs to **create workspace directory** so that we can place our ansible script in those directories.

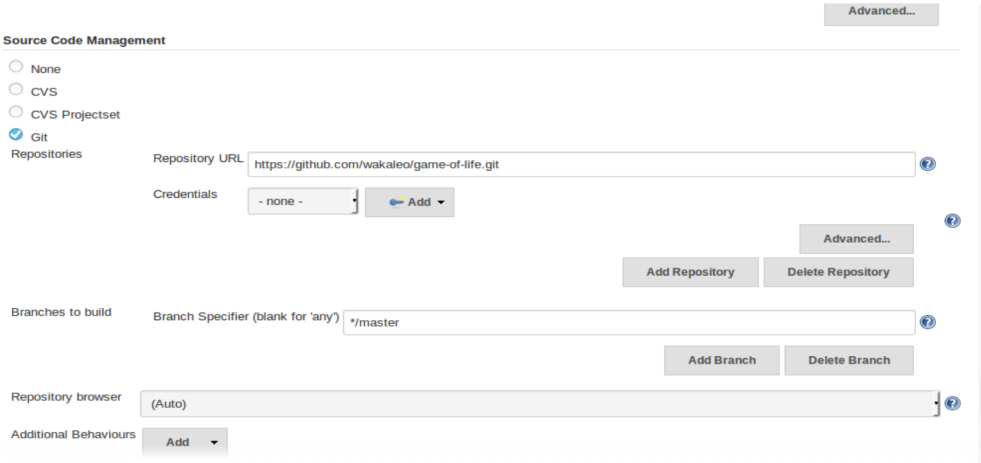
Job Names: - GameOfLifeOf_Dev
GameOfLifeOf_QA
GameOfLifeOf_UAT

4. Configure Dev job

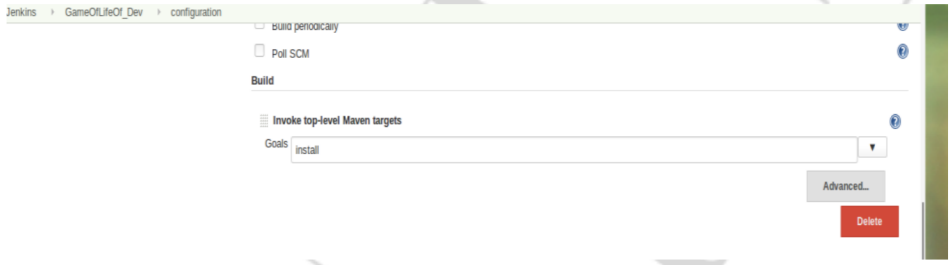
As specified in the screenshots below.

Configure Jenkins job.

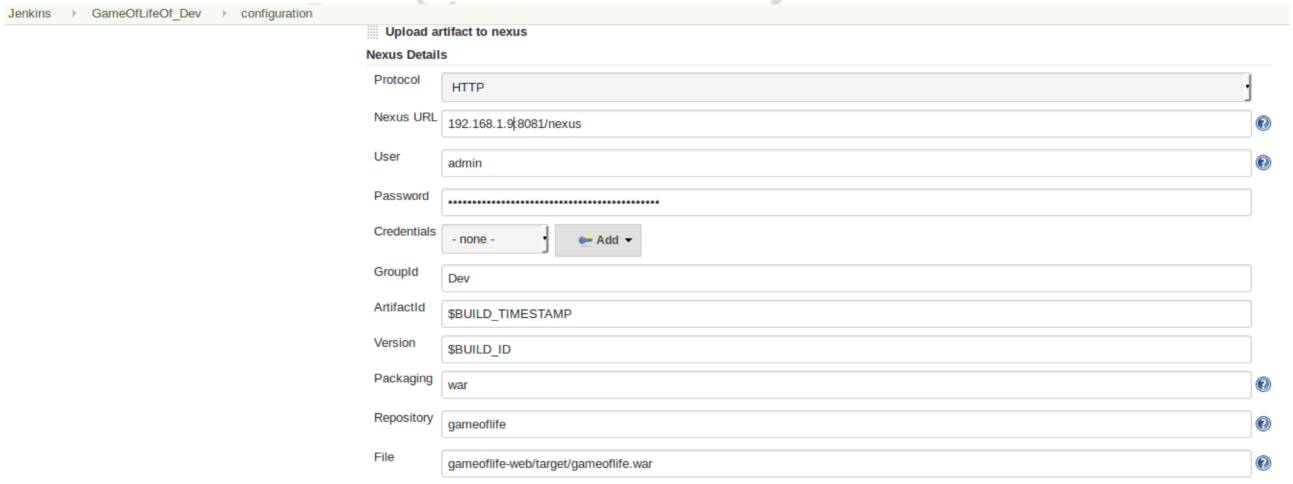
Configure GitHub plugin.



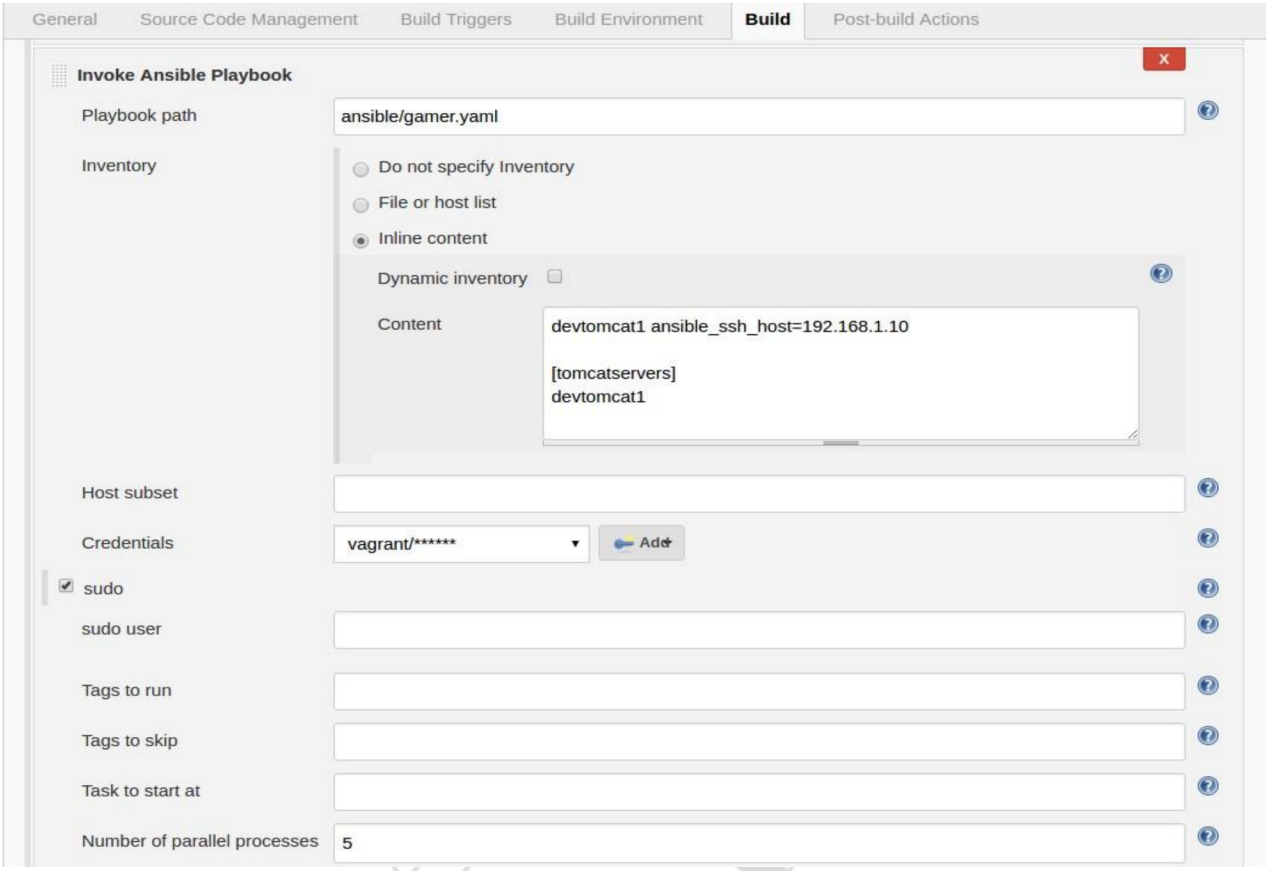
Add build step.



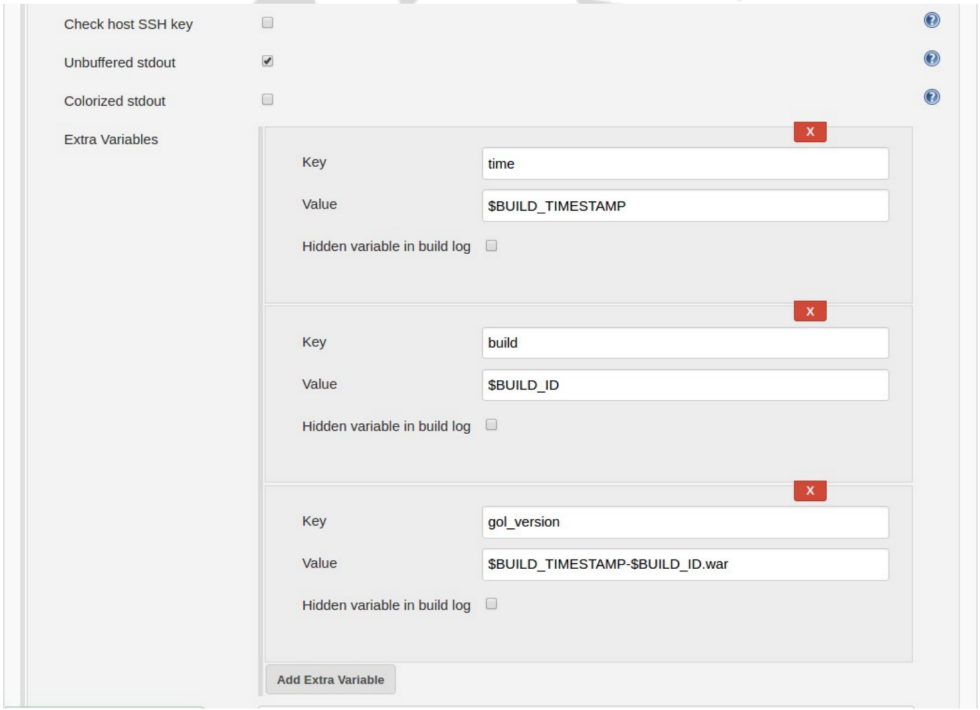
Configure nexus plugin.



Build step for ansible execution.



Playbook is given at the end of the page.



5. Ansible code modifications:

- Login to Jenkins server and place ansible directory in the workspace of the job. Workspace path specified below.
  - */var/lib/Jenkins/workspace/GameofLife_Dev/*

- Open gamer.yaml playbook and replace nexus server IP.

6. Execute the Jenkins dev job to validate.

Execution will happen in below phases

- Checkout git code from GitHub repository.
- Run maven build and unit test execution.
- Upload artefact to nexus server.
- Execution of ansible playbook to install java, tomcat on tomcat node and deploy artefacts from nexus to tomcat node.

JENKINS QA JOB SETUP.

1. Configure QA job same as Dev job, step 4 & 5.

- Make sure to edit the ansible inventory to specify QA tomcat node IP and nexus groupid to QA
- Change Nexus artefact URL in gamer.yaml playbook: - In URL change Dev to QA.

Note: All the Jenkins job will have its own workspace directory. For QA job, the workspace path would be as mentioned below.

- /var/lib/Jenkins/workspace/GameofLife_QA/

2. Adding QA automation script.

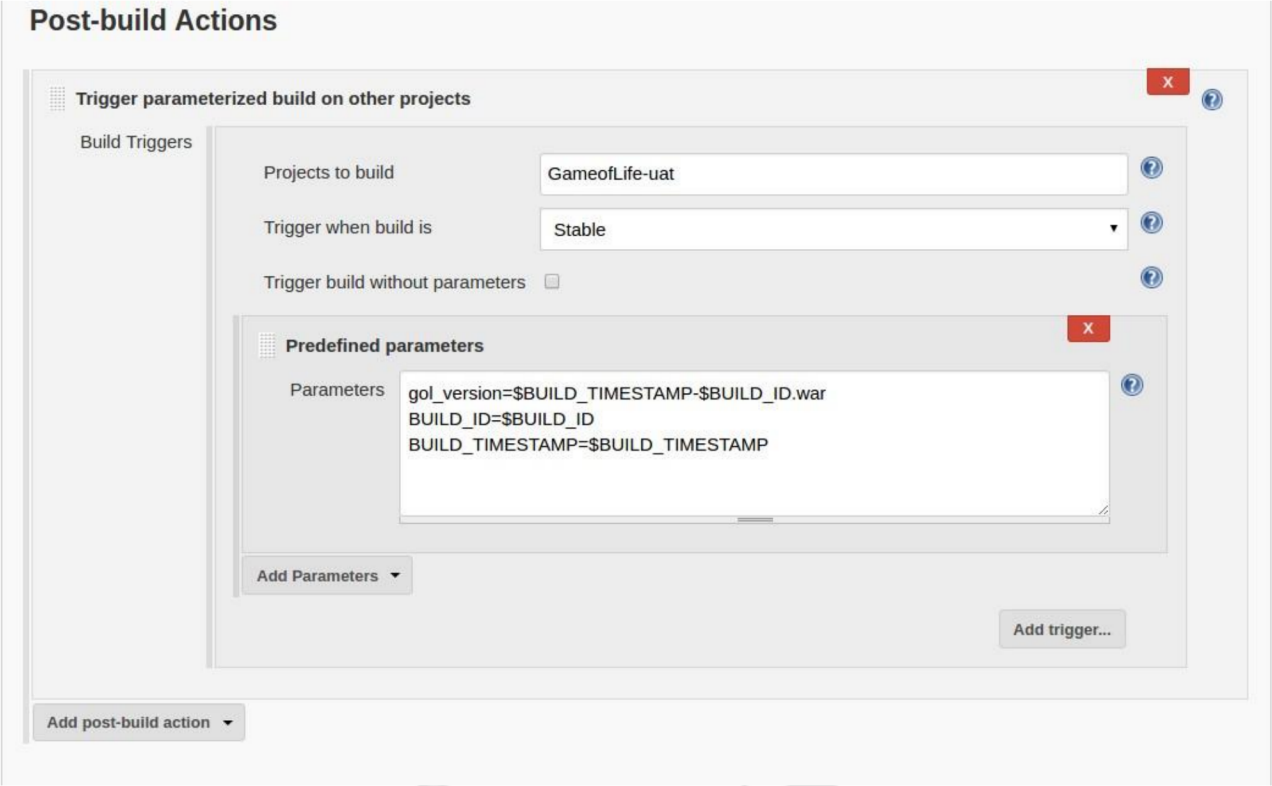
Add build step → Execute Shell → add below mentioned commands

```
echo "Executing test automation script."
sleep 5
```

Note: - We do not have test automation script so we are **imitating** it.

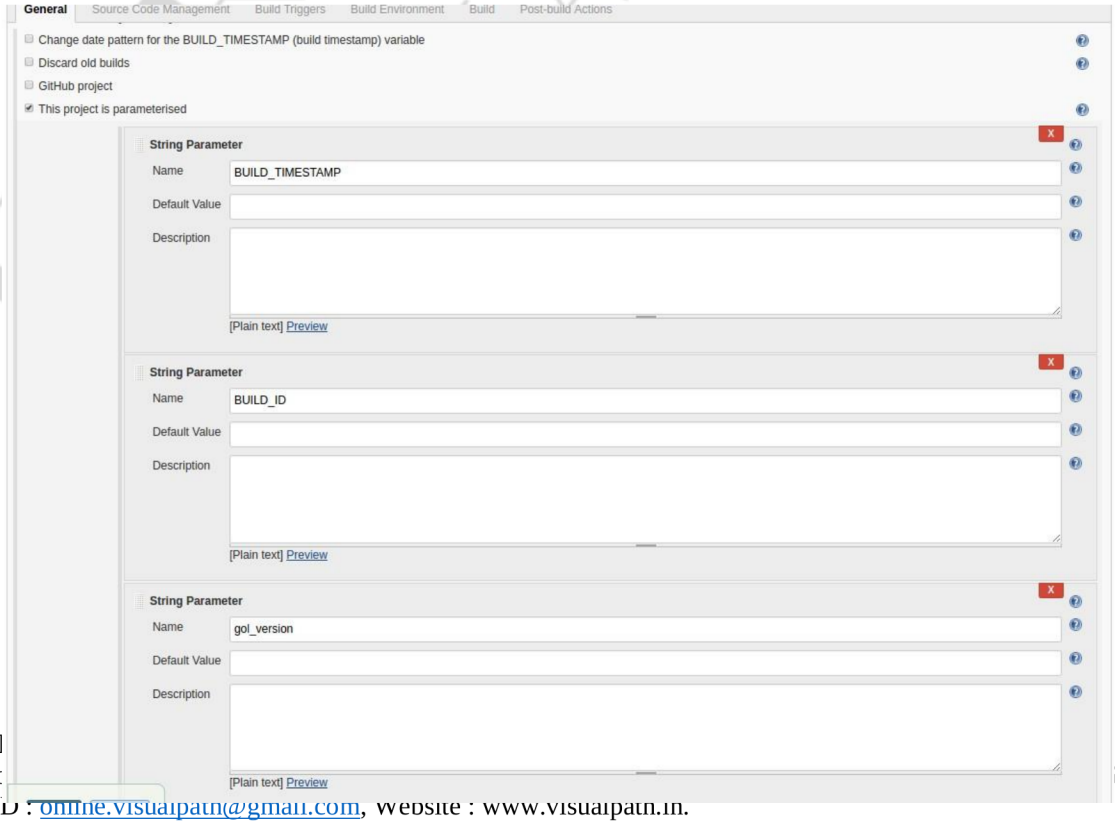
3. Add post build step which will pass parameter to the UAT job

- Click Add post-build action → Trigger parameterized build on other properties
  → Click on Add Parameter
  → Predefined parameters
  → fill everything as shown in screenshot below.
  → Put a check mark on "Block until the triggered projects finish their builds."



JENKINS UAT JOB SETUP.

1. Configure UAT job, put a check mark on “This build is parameterized”.
2. Add three String parameters, fill in only “Name” section as shown in the screenshot.



3. Ansible execution for UAT job.
   Add build step → Execute Shell → Add below mentioned content from screenshot.

4. Ansible Code modifications for UAT job.
   - Place ansible directory in UAT workspace. Path for workspace mentioned below. - */var/lib/Jenkins/workspace/GameofLife_QA/*
   - Make sure to edit the ansible inventory to specify UAT tomcat node IP.
   - Change Nexus artefact URL in gamer.yaml playbook: - In URL change it to QA.

Deployment Playbook:

```
---
- hosts: tomcatservers
  become: yes
  tasks:
  - name: Install EPEL-release
    yum: name=epel-release state=present
  - name: Install java_1.7
    yum: name=java-1.7.0-openjdk.x86_64 state=present
  - name: Install tomcat
    yum: name=tomcat state=present
  - name: Download latest gameoflife.war file
    get_url: url=http://<Nexus
IP>:8081/nexus/content/repositories/gameoflife-repo/<Group ID
name>/{{time}}/{{build}}/{{gol_version}} dest=/tmp/ mode=755
  - name: Stop tomcat service
    service: name=tomcat state=stopped
  - name: Copy artefact to tomcat folder
    shell: cp /tmp/{{gol_version}} /var/lib/tomcat/webapps
  - name: Delete link to existing gol version
    file: path=/var/lib/tomcat/webapps/gameoflife state=absent
  - name: Start tomcat service
    service: name=tomcat state=started
  - wait_for: path=/var/lib/tomcat/webapps/{{time}}-{{build}}
  - name: Link latest GOL version
    file: src=/var/lib/tomcat/webapps/{{time}}-{{build}}
dest=/var/lib/tomcat/webapps/gameoflife state=link
  - name: Stop iptables
    service: name=iptables state=stopped
```

Connect Dev and QA job together, use build pipeline plugin, execute the Dev job from Build Pipeline plugin and see everything in action.

XX. A word about security

In previous chapters while we were practicing scripting and service deployments we did not take care about the security much. We have stopped firewall (iptables) at some time just to get things done quickly. But this is not recommended in productions and real time services.

We should edit firewall rules and not shut them down to allow access.
Redhat family OS firewall name :- iptables
Ubuntu family OS firewall name:- ufw

Please refer below links to read in detail about these firewalls.
https://wiki.centos.org/HowTos/Network/IPTables
https://www.howtoforge.com/tutorial/ufw-uncomplicated-firewall-on-ubuntu-15-04/

At some places I have mentioned commands in ansible and puppet about modifying firewall rules. We have modules in ansible to edit firewall rules and also in puppet we have seen using firewall module from puppetforge. Its highly recommened to use these modules to update firewall rules.

http://docs.ansible.com/ansible/firewalld_module.html
http://docs.ansible.com/ansible/ufw_module.html
https://docs.ansible.com/ansible/iptables_module.html

https://forge.puppet.com/puppetlabs/firewall

In AWS we have seen updating security group rules. Make sure you assign the SG rules carefully and "**don't just say source Anywhere"** for protocols like SSH, MYSQL etc.

If you are hosting your website under Security group then **you can allow 443/80 from source anywhere.**

Lastly, be carefull about AWS account details.
Setup Multi factor authentication for console login and take very good care of IAM users Access key and Secret key. AWS accounts are under contant radar of hackers, if you give them a slightest of chance they will take over you AWS account completely.

KEEP LEARNING.

LEARNING IS A TREASURE THAT WILL FOLLOW ITS OWNER EVERYWHERE.