Integrate Dynatrace + Jenkins + Ansible

Install Jenkins Server :

https://github.com/Sumanth17-git/APMTrianing.git

cd APMTraining

chmod +x *

./setup_jenkins.sh

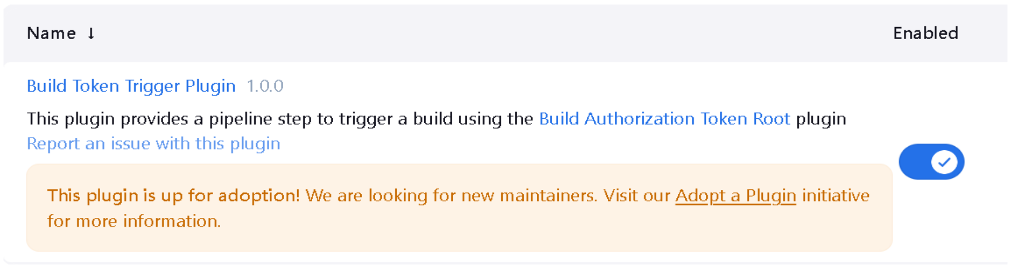
===This will setup the Jenkins setup and copy the password===

Open the Jenkins Portal : http://34.21.69.137:8080/

Install Suggested Plugins

Manage Jenkins ➔Plugins ➔ Available Plugins ➔

Step 1: Add Webhook Plugins : https://plugins.jenkins.io/build-token-trigger/

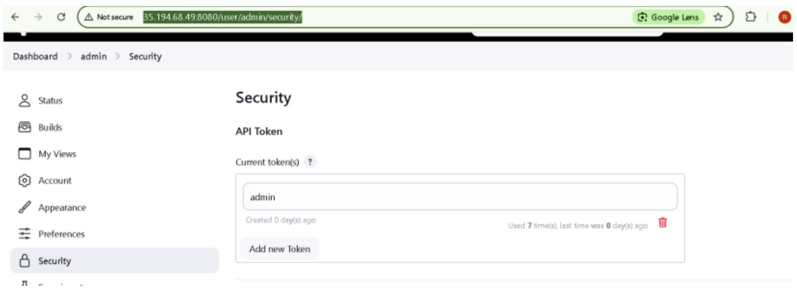


Name ↓	Enabled
Build Token Trigger Plugin 1.0.0 This plugin provides a pipeline step to trigger a build using the Build Authorization Token Root plugin Report an issue with this plugin This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.	✓

Now Let's create the API-Token in Jenkins

Click Your Profile ➔ Security ➔ API Token ➔Add new Token

http://35.194.68.49:8080/user/admin/security/



Security

API Token

Current token(s) ?

admin

Created 0 day(s) ago

Used 7 time(s), last time was 0 day(s) ago

Add new Token

Copy the API Token

jenkinsUrl = "http://34.21.69.137:8080/" # Your Jenkins server URL

username = "admin"

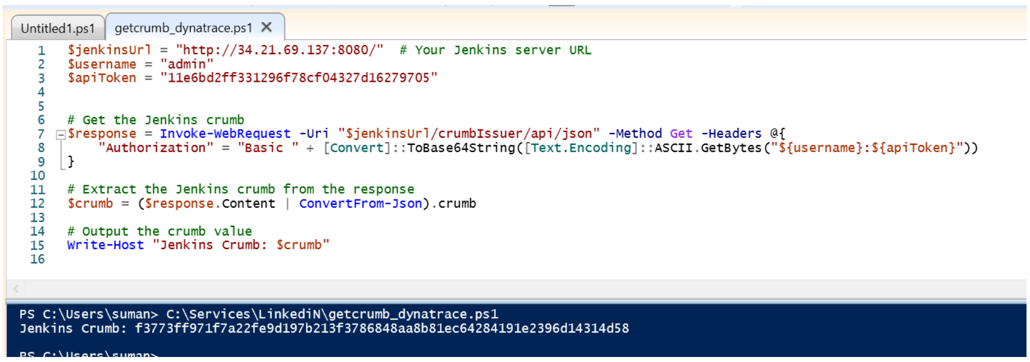
apiToken = "11e6bd2ff331296f78cf04327d16279705"

Inorder to integrate this with Dynatrace , we need some more details to integrate the dynatrace with Jenkins , we need to generate crumb token.

1. **Generate the Crumb token**

   <u>PowerShell Script to Retrieve the Crumb token</u>

```
$jenkinsUrl = "http://35.194.68.49:8080"  # Your Jenkins server URL
$username = "admin"
$apiToken = "119e2eaec5345a94d408ea6a816f0d0fc7"
# Get the Jenkins crumb
$response = Invoke-WebRequest -Uri "$jenkinsUrl/crumbIssuer/api/json" -Method Get -
Headers @{
    "Authorization" = "Basic " +
[Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes("${username}:${apiToken}"))
}

# Extract the Jenkins crumb from the response
$crumb = ($response.Content | ConvertFrom-Json).crumb

# Output the crumb value
Write-Host "Jenkins Crumb: $crumb"
```

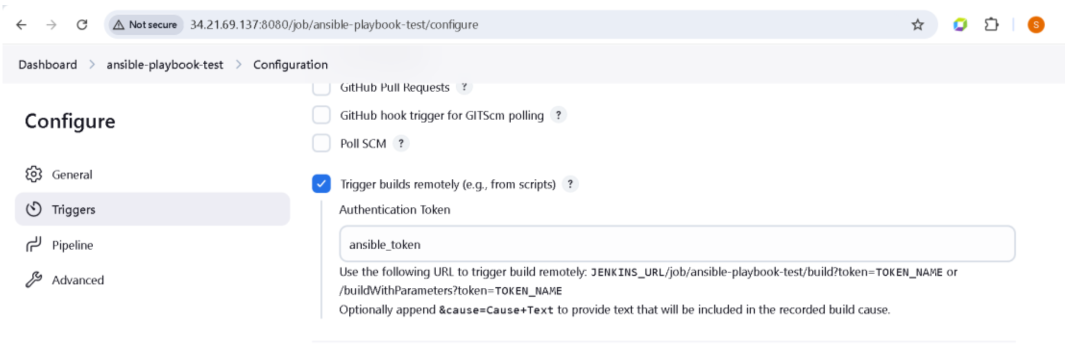


Untitled1.ps1 | getcrumb_dynatrace.ps1

```
$jenkinsUrl = "http://34.21.69.137:8080/"  # Your Jenkins server URL
$username = "admin"
$apiToken = "11e6bd2ff331296f78cf04327d16279705"


# Get the Jenkins crumb
$response = Invoke-WebRequest -Uri "$jenkinsUrl/crumbIssuer/api/json" -Method Get -Headers @{
    "Authorization" = "Basic " + [Convert]::ToBase64String([Text.Encoding]::ASCII.GetBytes("${username}:${apiToken}"))
}

# Extract the Jenkins crumb from the response
$crumb = ($response.Content | ConvertFrom-Json).crumb

# Output the crumb value
Write-Host "Jenkins Crumb: $crumb"
```

```
PS C:\Users\suman> C:\Services\LinkediN\getcrumb_dynatrace.ps1
Jenkins Crumb: f3773ff971f7a22fe9d197b213f3786848aa8b81ec64284191e2396d14314d58

PS C:\Users\suman>
```

Create the Jenkins job ➔ Pipeline job



Not secure 34.21.69.137:8080/job/ansible-playbook-test/configure

Dashboard > ansible-playbook-test > Configuration

- [ ] GitHub Pull Requests ?
- [ ] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [x] Trigger builds remotely (e.g., from scripts) ?

Authentication Token

ansible_token

Use the following URL to trigger build remotely: `JENKINS_URL`/job/ansible-playbook-test/build?token=`TOKEN_NAME` or /buildWithParameters?token=`TOKEN_NAME`

Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

http://34.21.69.137:8080/job/ansible-playbook-test/build?token=ansible_token

Now we have complete details on Dynatrace integration side. Go back to Dynatrace ➔ integration➔ Problem notification.

Notification Type: Custom Integration

Display Name : Jenkins-integration

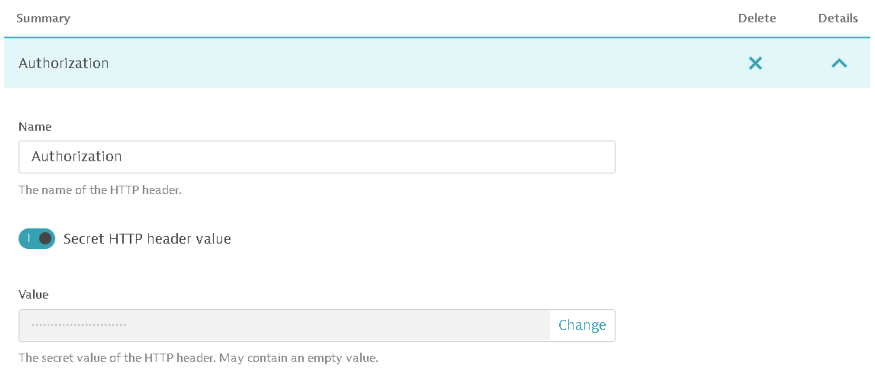
Webhook URL : http://34.21.69.137:8080/job/ansible-playbook-test/build?token=ansible_token

1.Choose Create basic authorization header

Username: admin

Password:11e6bd2ff331296f78cf04327d16279705   (i.e. jenkins API Token)

Once you have added this , this will show like this.



Summary	Delete	Details
Authorization	✕	^

Name

Authorization

The name of the HTTP header.

Secret HTTP header value

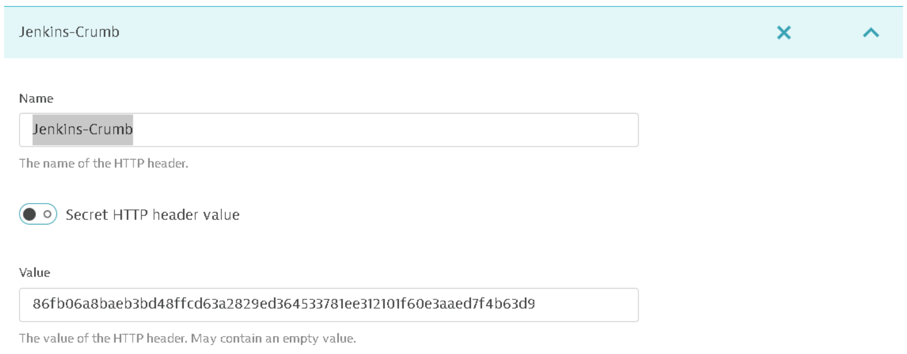
Value

························ Change

The secret value of the HTTP header. May contain an empty value.

2.Click Add Item

Jenkins-Crumb



Jenkins-Crumb ✕ ^

Name

Jenkins-Crumb

The name of the HTTP header.

Secret HTTP header value

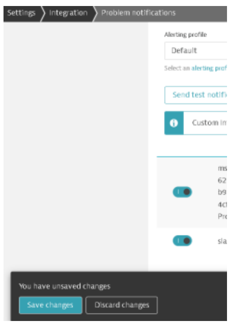
Value

86fb06a8baeb3bd48ffcd63a2829ed364533781ee312101f60e3aaed7f4b63d9

The value of the HTTP header. May contain an empty value.

Choose Alerting Profile and Click Send test notifications

Click Save Changes.



Settings › Integration › Problem notifications

Alerting profile

Default

Select an alerting prof

Send test notifi

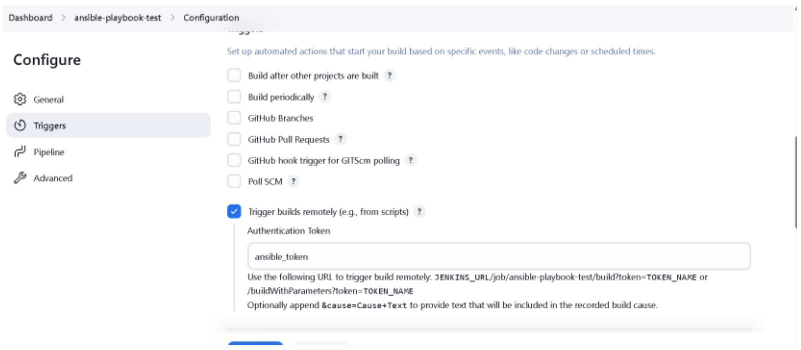
Custom In

You have unsaved changes

Save changes Discard changes

Visit : **https://www.genzgurukul.com/** for advanced Dynatrace courses

Pipeline Script



Dashboard > ansible-playbook-test > Configuration

Configure

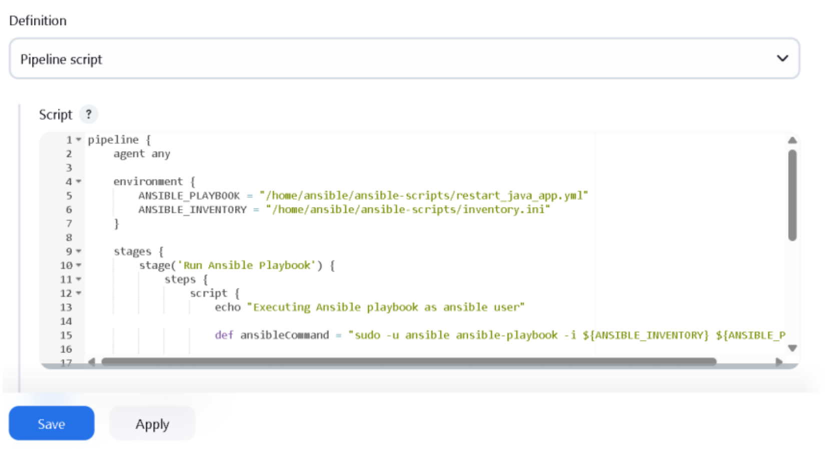
- General
- Triggers
- Pipeline
- Advanced

Set up automated actions that start your build based on specific events, like code changes or scheduled times.

- [ ] Build after other projects are built ?
- [ ] Build periodically ?
- [ ] GitHub Branches
- [ ] GitHub Pull Requests ?
- [ ] GitHub hook trigger for GITScm polling ?
- [ ] Poll SCM ?
- [✓] Trigger builds remotely (e.g., from scripts) ?

Authentication Token: ansible_token

Use the following URL to trigger build remotely: `JENKINS_URL`/job/ansible-playbook-test/build?token=`TOKEN_NAME` or /buildWithParameters?token=`TOKEN_NAME`
Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.



Definition: Pipeline script

Script ?

```
pipeline {
    agent any

    environment {
        ANSIBLE_PLAYBOOK = "/home/ansible/ansible-scripts/restart_java_app.yml"
        ANSIBLE_INVENTORY = "/home/ansible/ansible-scripts/inventory.ini"
    }

    stages {
        stage('Run Ansible Playbook') {
            steps {
                script {
                    echo "Executing Ansible playbook as ansible user"

                    def ansibleCommand = "sudo -u ansible ansible-playbook -i ${ANSIBLE_INVENTORY} ${ANSIBLE_P
```

```
pipeline {
    agent any

    environment {
        ANSIBLE_PLAYBOOK = "/home/ansible/ansible-scripts/restart_java_app.yml"
        ANSIBLE_INVENTORY = "/home/ansible/ansible-scripts/inventory.ini"
    }

    stages {
        stage('Run Ansible Playbook') {
            steps {
                script {
                    echo "Executing Ansible playbook as ansible user"

                    def ansibleCommand = "sudo -u ansible ansible-playbook -i ${ANSIBLE_INVENTORY}
${ANSIBLE_PLAYBOOK}"

                    sh ansibleCommand
                }
            }
        }
    }

    post {
        success {
            echo "Ansible playbook executed successfully!"
        }
        failure {
            echo "Ansible playbook execution failed!"
            error "Stopping pipeline due to failure"
        }
    }
}
```

This pipeline script is created.

Now let’s setup the Ansible maser and target server

On Master server

https://github.com/Sumanth17-git/APMTrianing.git

cd APMTraining

chmod +x *

./setup.ansible_master.sh

Copy the public key and save it for future use.

Setup the Ansible Target instance

Now We need to setup ansible target server where your java application is running , now I need to setup this as ansible target instance.

https://github.com/Sumanth17-git/APMTrianing.git

cd APMTraining

chmod +x *

./setup_ansible_target.sh

Paster the Public Key which is copied from ansible master server.Once this is successful.

Validate

Go back to ansible master server ,try to connect ansible target server.

ssh ansible@10.150.0.12

Click yes

Allow jenkins to switch ansible user without password.

sudo vi /etc/sudoers

Add the following line at the end:

jenkins ALL=(ansible) NOPASSWD: ALL

Save the file.

Verify the changes by running:

sudo -l -U jenkins

Your Jenkins job should now execute the Ansible playbook **without prompting for a password**.

On Ansible Master server , we need to create the ansible playbook and inventory file.

```
mkdir ansible-scripts

cd ansible-scripts
```

Create inventory.ini

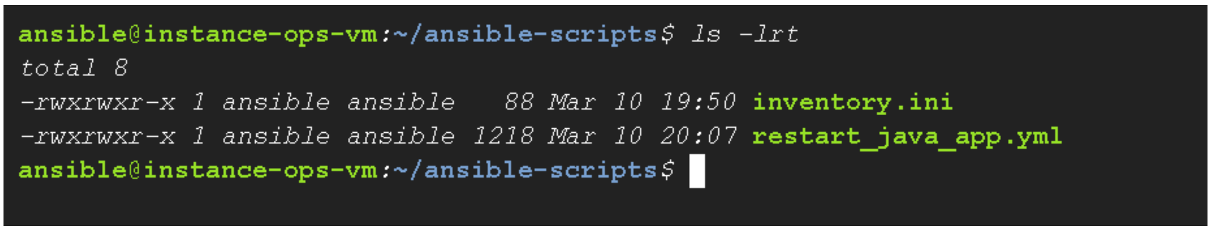
```
[mytargets]

10.150.0.12 ansible_user=ansible ansible_ssh_private_key_file=~/.ssh/id_rsa
```

Create restart_java_app.yml

```
---
- name: Restart Java Spring Boot Microservice
  hosts: mytargets
  become: yes
  become_user: root
  tasks:

    - name: Find if Java process is running
      shell: "jps | grep 'buggyApp.jar' | awk '{print $1}'"
      register: java_pid
      changed_when: false

    - name: Kill the Java process if running
      shell: "kill -9 {{ java_pid.stdout }}"
      when: java_pid.stdout | length > 0

    - name: Start Java Spring Boot microservice
      shell: "nohup java -Xmx512m -jar /home/jyothichandrasowreddy/buggyApp/buggyApp.jar
PROBLEM_MEMORY 2>&1 &"
      args:
        chdir: "/home/jyothichandrasowreddy/"
      async: 10
      poll: 0

    - name: Wait for Java application to start
      pause:
        seconds: 5

    - name: Verify Java process is running
      shell: "jps | grep 'buggyApp.jar'"
      register: java_status
      changed_when: false

    - name: Print success message
      debug:
        msg: "Java service restarted successfully!"
      when: java_status.stdout | length > 0

    - name: Print failure message
      debug:
        msg: "Java service failed to start!"
      when: java_status.stdout | length == 0
```

Once created these 2 files , enable the executable permission of two files.

chmod +x *



```
ansible@instance-ops-vm:~/ansible-scripts$ ls -lrt
total 8
-rwxrwxr-x 1 ansible ansible   88 Mar 10 19:50 inventory.ini
-rwxrwxr-x 1 ansible ansible 1218 Mar 10 20:07 restart_java_app.yml
ansible@instance-ops-vm:~/ansible-scripts$ 
```

ansible-playbook restart_java_app.yml -i inventory.ini