

Jenkins Integration

Connectors

1.Jenkins Integration and trigger build

First Go to Jenkins and create the API Token

Click your Name (top right) → security → Add new Token → Save.

11023af9b726ca61c25a16c1c435778434

The screenshot shows the Jenkins Security page. On the left, there's a sidebar with links like Status, Builds, My Views, Account, Appearance, Preferences, Security (which is highlighted), Experiments, and Credentials. The main area is titled 'Security' and has a sub-section 'API Token'. It shows a table with one row named 'TriggerAPI'. The table includes columns for 'Created' (0 day(s) ago), 'Used' (2 time(s), last time was 0 day(s) ago), and a trash icon. Below the table is a button 'Add new Token'. At the bottom of the page are 'Save' and 'Apply' buttons.

Now you have setup the API token . Let's go to Dynatrace Automation workflow.

1. Allow Jenkins for outbound connections

- 1) Open the **Settings** app and go to **Preferences > Limit Outbound Connections**.
- 2) Select **Add item** and **add the domain of your publicly accessible Jenkins instance**.
- 3) Select **Save changes**.

The screenshot shows the Dynatrace Settings app with the 'Limit outbound connections' configuration. The left sidebar lists various monitoring and automation settings. The main panel shows a 'Limit outbound connections' section with a description: 'You can limit the accessibility of public endpoints from functions running in the Dynatrace JavaScript runtime, for example, the backends of apps and functions written in the Dashboards, Notebooks and Automations app.' It has a 'Use defaults' button and an 'Allow-list' section. Under 'Allow-list', there's a 'Add Item' button and a table with two entries: 'Allowed Host' (35.212.30.80) and 'team.microsoft.com'. There are 'Delete' and 'x' buttons next to each entry.

2.Grant permissions to Workflows

Some permissions are required by Workflows to run actions on your behalf. Other permissions are required by actions that come bundled with Jenkins Connector itself.

1. Go to **Workflows** and select **Settings > Authorization settings**.
2. Select the following permissions besides the general Workflows permission.
 - Permissions needed for Jenkins workflow actions:
 - **app-settings:objects:read**

Open the Settings app and go to Dynatrace Apps > Jenkins Connections.

Jenkins

① Changing this connection setting also affects the usage of this connection in other contexts.

② For security reasons, Dynatrace blocks outgoing traffic by default. Therefore, you need to add the URL to the allow-list [?.](#)

Connection name*

jenkins

A unique and clearly identifiable connection name to your Jenkins instance.

Jenkins instance URL*

<http://35.212.30.180:8080/>

Base URL of your Jenkins instance (e.g. [https://\[YOUR_JENKINS_DOMAIN\]/](https://[YOUR_JENKINS_DOMAIN]/)).

User*

admin

The name of your Jenkins user (e.g. Jenkins).

Password*

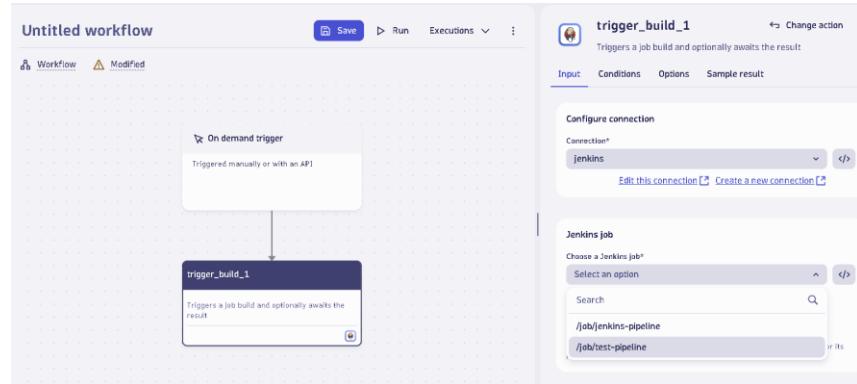
The password of the user or API token obtained from the Jenkins UI (Dashboard > User > Configure > API Token). [Change](#)

Jenkins instance URL : <http://35.212.30.180:8080/>

User* : admin

Password: 11023af9b726ca61c25a16c1c435778434 (your API token copied from jenkins side)

Go Back to Workflow to Trigger the Jenkins Build



Await result:

Only Trigger build

Click SAVE and RUN

==

UseCase 1: Trigger the Build directly from Dynatrace

Pre-Requisites:

1. Connect Jenkins with Dynatrace

Open Jenkins , Create Pipeline job

This is optional : Plugin :



Triggers

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 ?
 - Generic Webhook Trigger ?
 - GitHub Branches
 - GitHub Pull Requests ?
 - GitHub hook trigger for GITScm polling ?
 - Poll SCM ?
 - Trigger builds remotely (e.g., from scripts) ?
- Authentication Token
- java-token
- Use the following URL to trigger build remotely: JENKINS_URL/job/restart-java-app/build?token=TOKEN_NAME or

Pipeline:

```
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"  
        }  
    }  
}
```

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 'memoryleaksimulator-0.0.1-SNAPSHOT.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 -jar /home/sumanth_suman17/memoryleak/target/memoryleaksimulator-0.0.1-SNAPSHOT.jar 2>&1 &" args:
        chdir: "/home/sumanth_suman17/memoryleak/target"
      async: 10
      poll: 0

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

    - name: Verify Java process is running
      shell: "jps | grep 'memoryleaksimulator-0.0.1-SNAPSHOT.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
```

inventory.ini

```
[mytargets]
10.150.0.12 ansible_user=ansible ansible_ssh_private_key_file=~/ssh/id_rsa
```

```
ansible-playbook restart_java_app.yml -i inventory.ini
```

UseCase 2: Trigger the Build from GitHub

Before we configure this from Dynatrace , first we need to make Github to connect Jenkins and push the build

<https://github.com/Sumanth17-git/restart-java-ansible>

1.Configure Webhook in GitHub

Go to repository→ Settings→ Webhook

□ Open GitHub Repository → Click on Settings.

□ Go to Webhooks → Click Add Webhook.

□ Payload URL: **Enter Jenkins Webhook URL**

http://your-jenkins-server/github-webhook/

- Content Type: application/json
- Trigger Events: Select "Just the push event".
- Click "Add Webhook".

[http://34.48.3.231:8080/github-webhook/ \(Your Jenkins Host URL\)](http://34.48.3.231:8080/github-webhook/)

Step2: Setup GITHUB Connection in Jenkins

Create GIT-Hub Credentials in Jenkins

Click Manage Jenkins → Credentials → Click Global

Once it is setup , next step is to configure the Git Credentials on Jenkins

How to Add GitHub Credentials in Jenkins

To allow Jenkins to pull code from a **private GitHub repository**, you need to store your **GitHub credentials** in Jenkins securely.

Step 1: Generate a Personal Access Token (PAT) in GitHub

1. Go to GitHub and navigate to Settings → Developer settings → Personal access tokens.
2. Click Generate new token (or Fine-grained tokens for new GitHub versions).
3. Select Expiration (Choose "No Expiration" if you don't want it to expire).
4. Select Scopes:
 - repo (Full control over private repositories)
 - admin:repo_hook (For webhooks)

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

5. Click **Generate token**.
6. Copy the **token** and save it somewhere safe.

The screenshot shows the GitHub Developer Settings page under Personal access tokens (classic). It lists three tokens:

- Jenkins-webhook**: Never used, Delete. Description: "jenkins-webhook — admin:enterprise, admin:pgp_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete_packages, gist, notifications, project, repo, user, workflow, write:discussion, write:network_configurations, write:packages". Expires on Sun, Mar 23 2025.
- Jenkins-token**: Never used, Delete. Description: "jenkins-token — admin:enterprise, admin:pgp_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete_packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:network_configurations, write:packages". Expires on Sun, Mar 23 2025.
- github-token**: Last used within the last 2 weeks, Delete. Description: "github-token — admin:enterprise, admin:pgp_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, admin:ssh_signing_key, audit_log, codespace, copilot, delete_packages, delete_repo, gist, notifications, project, repo, user, workflow, write:discussion, write:network_configurations, write:packages". Expires on Tue, Apr 22 2025.

Step 2: Add GitHub Credentials to Jenkins

1. Login to Jenkins and go to Manage Jenkins → Manage Credentials.
2. Click (global) > Add Credentials.
3. Select:
 - Kind: Username with password**
 - Username:** Your GitHub username (Sumanth17-git)
 - Password:** Paste the **GitHub PAT token**
 - ID:** (Optional) Set an identifier like github-token
 - Description:** "GitHub Credentials for Jenkins"

The screenshot shows the Jenkins Manage Credentials page for GitHub credentials. The fields filled are:

- Scope**: Global (Jenkins, nodes, items, all child items, etc)
- Username**: Sumanth17-git
- Password**: Concealed
- ID**: github-token

4. Click **Save**.

All set , lets update/create the Jenkinsfile in your Github Repo

credentialsId: 'github-token'

```
stages {
  stage('Checkout Code') {
    steps {
      script {
        git branch: 'main',
        credentialsId: 'dynatrace-github', // Use the ID you set in Jenkins
        url: 'https://github.com/Sumanth17-git/restart-java-ansible.git'
      }
    }
  }
}
```

Create the Jenkins Pipeline

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

The screenshot shows the Jenkins configuration interface for a project named "restart.java.app-github". In the "Triggers" section, several options are listed under "Get up automated actions that start your build based on specific events, like code changes or scheduled times". The "GitHub hook trigger for GITScm polling" option is checked. Other options include "Preserve stasher from completed builds", "This project is parameterized", "Throttle builds", "Build periodically", "Generic Webhook Trigger", "GitHub Branches", "GitHub Pull Requests", "Pull SCM", and "Trigger builds remotely (e.g., from scripts)".

Provide these details

The screenshot shows the "Pipeline script from SCM" configuration. Under the "SCM" dropdown, "Git" is selected. In the "Repositories" section, a single repository is defined with the URL "https://github.com/Sumanth17-git/restart-java-ansible.git" and credentials "Sumanth17-git/******** (GitHub Webhook)". Below the form are "Save" and "Apply" buttons.

Update the branch as **main**

The screenshot shows the "Branch Specifier (blank for any)" field containing the value "*main". Other settings visible include "Add Branch", "Repository browser (Auto)", "Additional Behaviours", and "Script Path Jenkinsfile". Below the form are "Save" and "Apply" buttons.

Script path : **Jenkinsfile**

Click Apply And SAVE

If everything works fine , when you make some changes in Github repo and commit to Main branch . this jenkins will get trigger automatically.

Update the GitHub Task

 **create_or_replace_file_1** [Change action](#) [⋮](#)

Creates or replaces a file

[Input](#) [Conditions](#) [Options](#) [Sample result](#)

Configure file creation or replacement [ⓘ](#)

Repository details

Owner*

The account owner of the repository (private user or organization). The name is not case sensitive.

Repository*

The repository without the .git extension. The name is not case sensitive.

Branch details

Commit on a new branch
 Commit on an existing branch

Branch*

The existing branch you want to commit to.

File details

Update the pull request

 **create_pull_request_1** [Change action](#) [:](#)

Creates a pull request

[Input](#) [Conditions](#) [Options](#) [Sample result](#)

Repository details

Owner*
Sumanth17-git

The account owner of the repository (private user or organization). The name is not case sensitive.

Repository*
restart-java-ansible

The repository without the .git extension. The name is not case sensitive.

Branch details

Source branch*
feature-branch

The branch where the changes are implemented.

Target branch*
main

The branch you want the changes pulled into.

Pull request details

Pull request title*