DevOps

Caltech | Center for Technology & Management Education

Post Graduate Program in DevOps

simplilearn

Caltech | Center for Technology & Management Education

CI/CD Pipeline with Jenkins

Jenkins Build Triggers

Learning Objectives

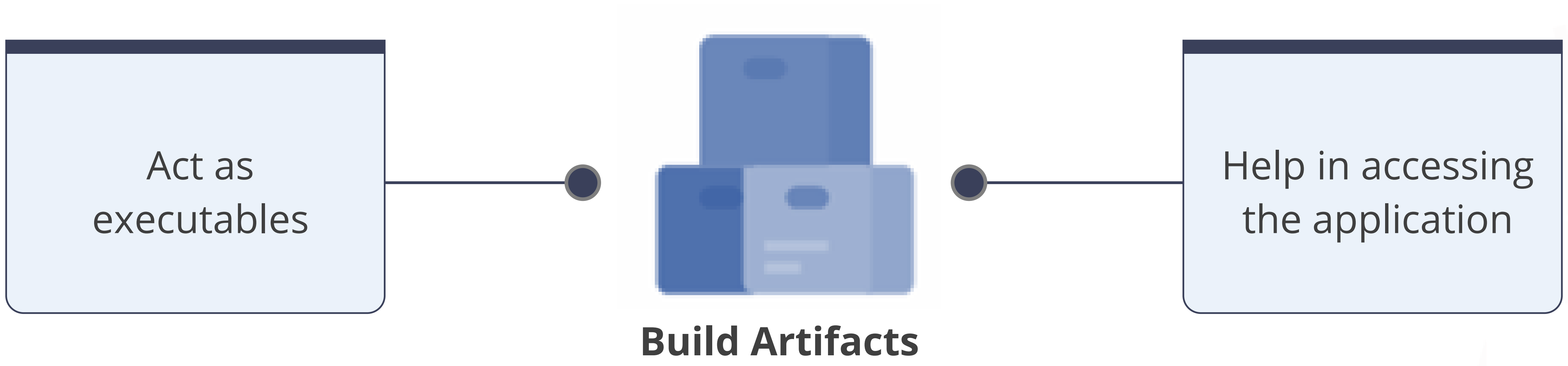
By the end of this lesson, you will be able to:

- Outline artifacts and fingerprints feature of Jenkins
- Discuss setting up upstream and downstream Jenkins jobs
- Creating Build Periodically triggers
- Setting up Poll SCM and Webhook configuration triggers

Build Triggers

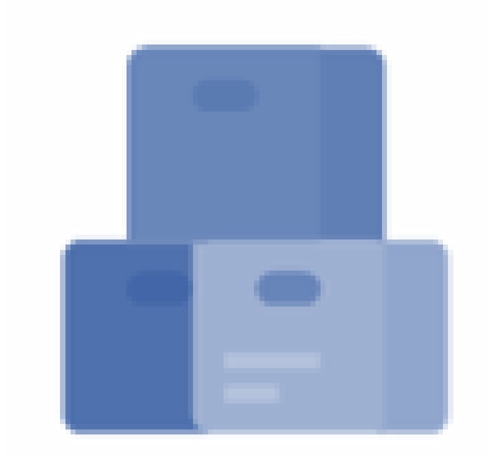
Introduction to Build Artifacts

Build Artifacts are files produced by a Build.



Introduction to Build Artifacts

Following are the types of artifacts that various programming languages support:



Types of Build Artifacts

- WAR
- EAR
- EXE
- DLL

Introduction to Build Artifacts

Before building tools, developers used to:

Prepare artifacts in their local IDEs

Share artifacts with the operations team to perform deployment

Consumed more manual efforts

Note

The artifacts can also be generated once the Build is done, so that it can be preserved permanently in Jenkins.

Artifact Management with Jenkins

An artifact is the end result of any Build process executed in Jenkins.

Artifacts are generated in the form of binaries and executables.

They can be generated by running Build scripts like Maven, Ant, and Gradle to compile source code.

Artifact Management with Jenkins

Jenkins can be used to archive artifacts for any Build.

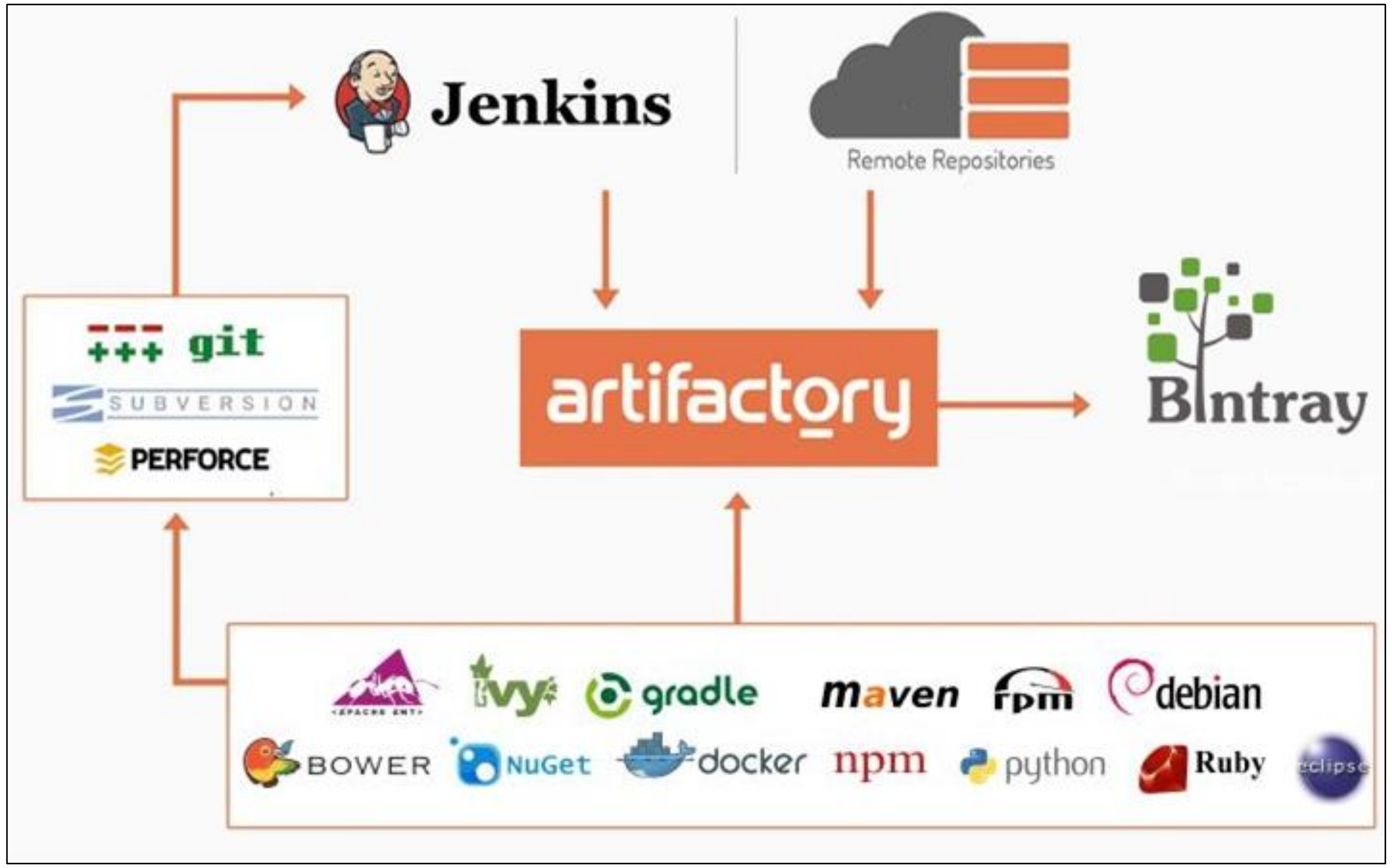
Artifacts take disk space on the local Jenkins server.

They can be stored in an artifactory.

Artifactory is the place where the various project artifacts are stored in a single location to avoid the issue of disk space.

Artifact Management with Jenkins

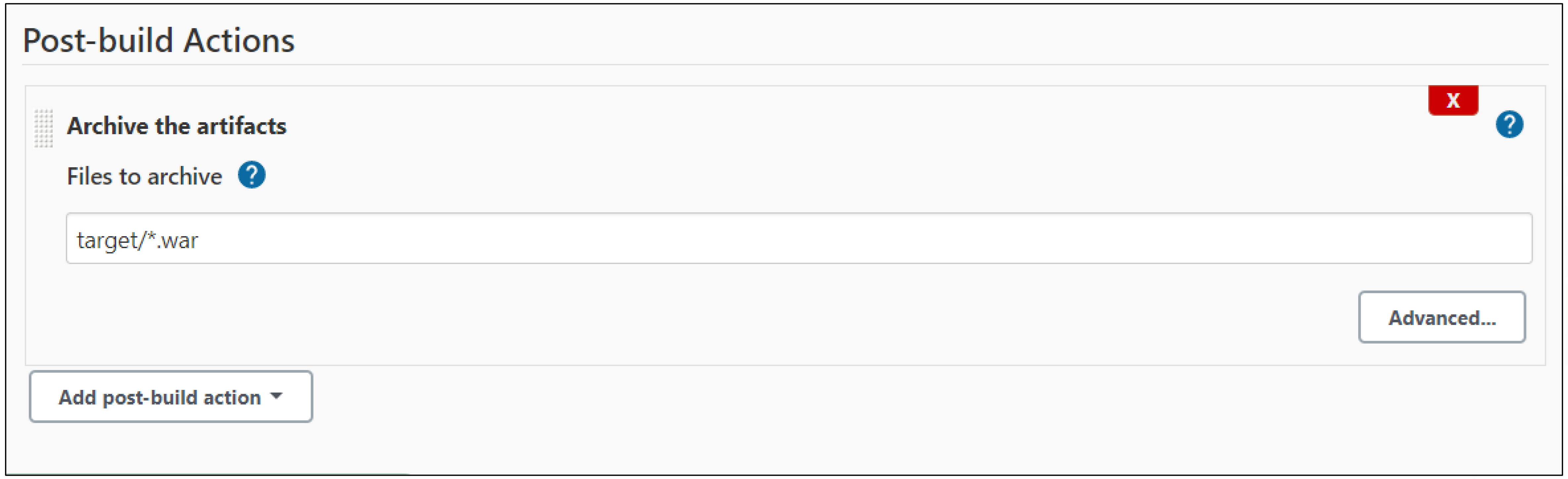
While running Build like Maven, developers must install the artifactory plugin. This plugin helps in uploading the artifacts into Jfrog and Nexus artifactory.



Source: https://www.jfrog.com/

Artifacts Archiving with Jenkins

Shown below is a screenshot of the Post-build Actions section, which is used to archive the artifacts.



Post-build Actions

Archive the artifacts

Files to archive: target/*.war

Introduction to Fingerprints

Jenkins Fingerprinting is the mechanism used by Jenkins to track a file across different Builds and jobs.

Jenkins Fingerprint

- Is the MD5 checksum of a file that will be stored in Jenkins.
- Is an xml file stored in Jenkins Master's $JENKINS_HOME/fingerprints directory.

Jenkins Fingerprints

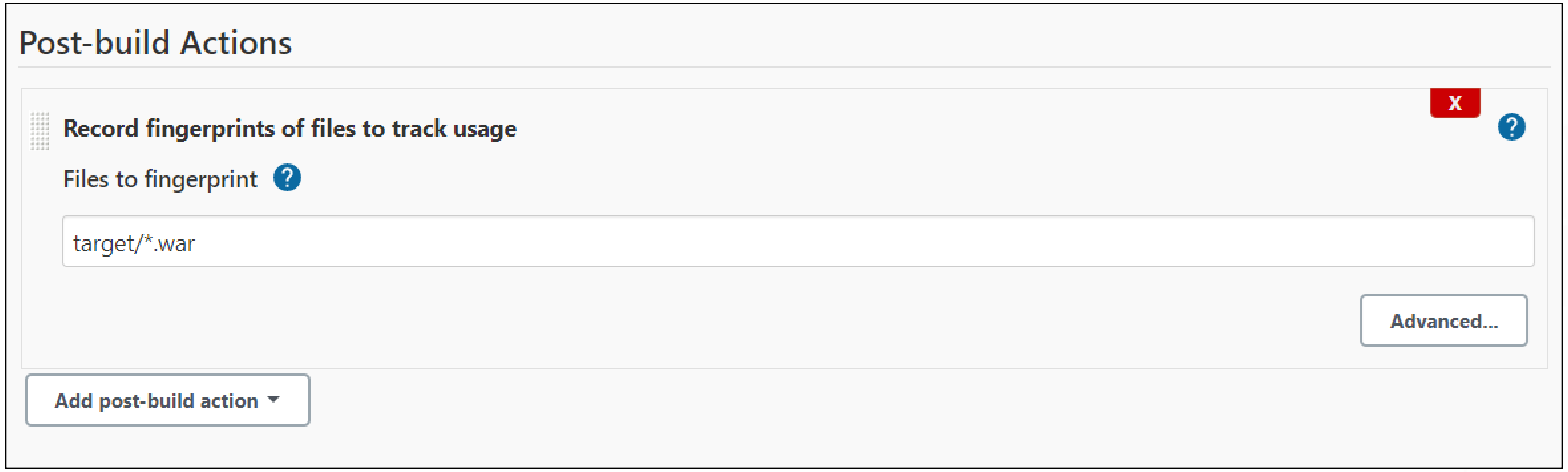
Jenkins maintains an md5sum database, which gets updated every time a Build gets executed.

Jenkins stores md5sum values instead of saving the file, in order to preserve disk usage.

Developers track the artifacts based on md5sum values.

Jenkins Fingerprints

Shown below is a screenshot of the Post-build Actions section, which is used to record the fingerprints of the files to track usage.



Post-build Actions

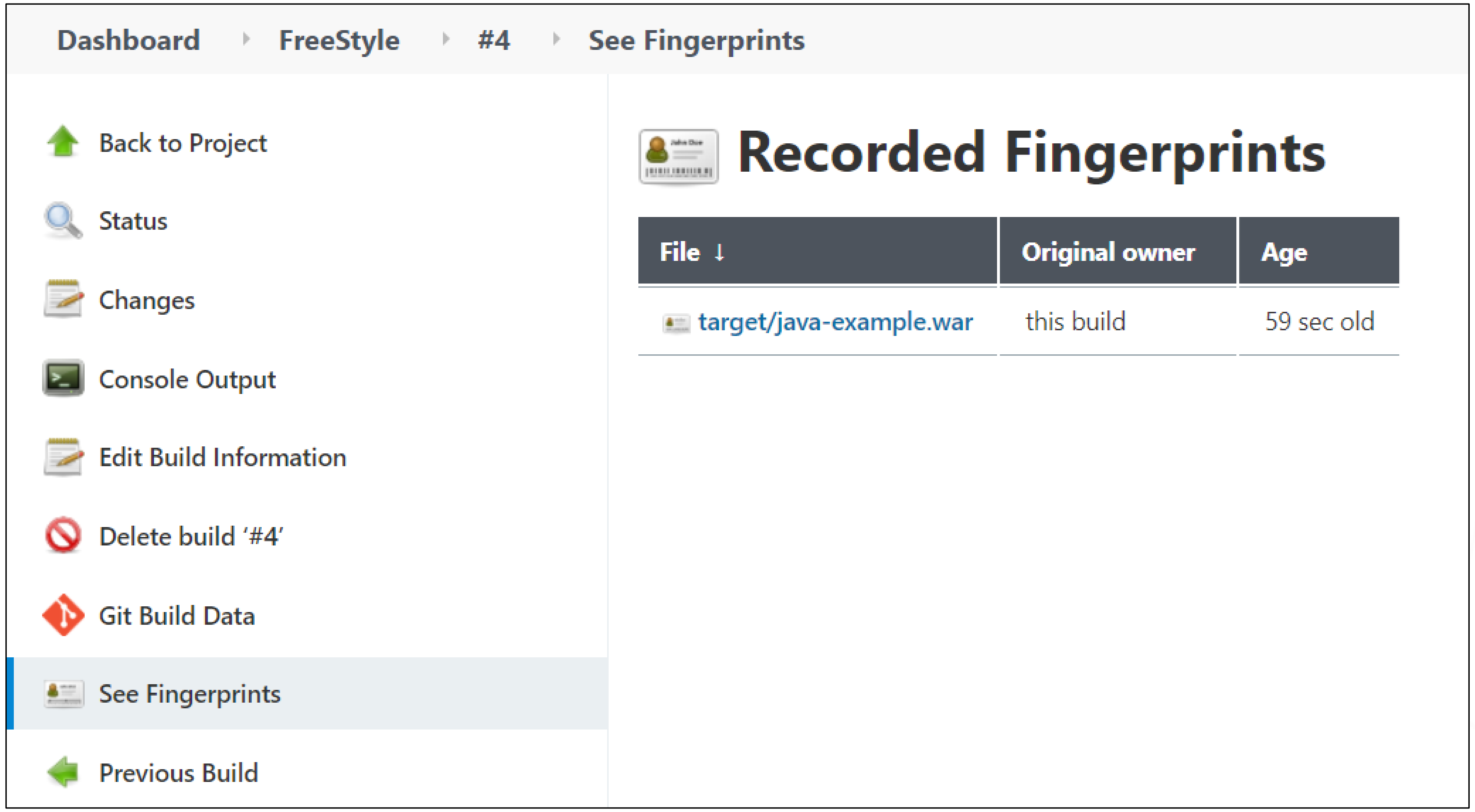
Record fingerprints of files to track usage

Files to fingerprint

```
target/*.war
```

Jenkins Fingerprints

Shown below is a screenshot of the See Fingerprints tab, which displays details of the recorded fingerprints.


Dashboard › FreeStyle › #4 › See Fingerprints

Recorded Fingerprints

File ↓	Original owner	Age
target/java-example.war	this build	59 sec old

Build Triggers

A Jenkins job can be triggered by clicking on the Build Now option to execute a Build step.

Configuring Automated Build Triggers

- Enables automatic triggering of Jenkins job
- Implements Continuous Integration and Continuous Deployment
- Ensures every Build's job is triggered

Build Triggers

With the help of Build triggers, developers can get new Builds triggered once the source code is committed in the repository.

Build Triggers

Shown below are the different Build triggers supported by Jenkins for triggering Build automatically:

- Trigger Builds remotely
- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM

Upstream and Downstream Jenkins Job

Jenkins provides one of the important features in which developers can split complex Jenkins job into pieces. These are upstream and downstream Jenkins jobs.

- Create upstream and downstream jobs on an original job so that the developer can execute the steps before and after original job.
- The original job will not be triggered if the upstream job is not executed successfully.
- The original job will not be triggered if the downstream job is not executed successfully.

Upstream and Downstream Jenkins Job

Using upstream and downstream Jenkins job configuration, developers can interlink different Jenkins job from CI/CD automation.

Upstream Jobs

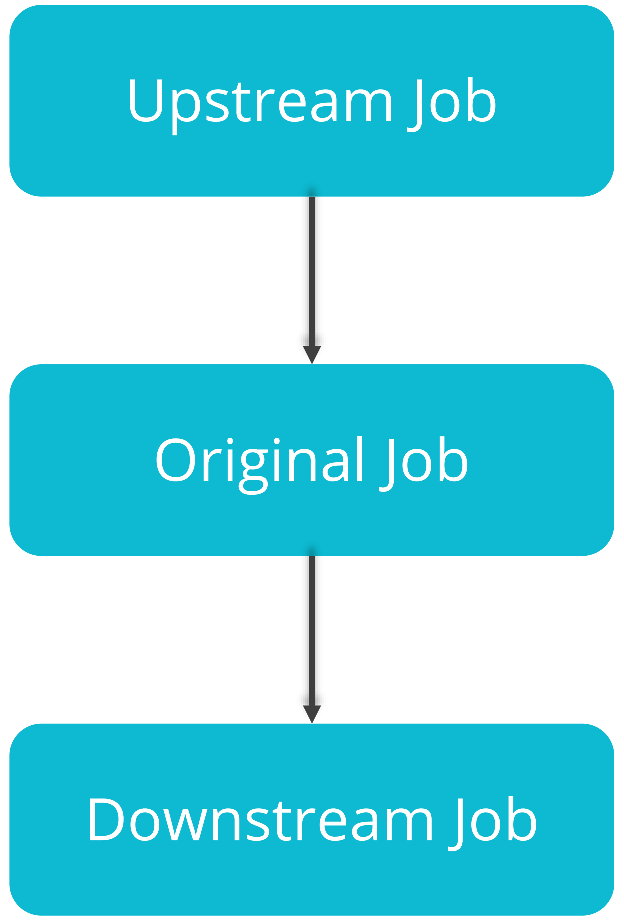
- Upstream job is one that is triggered before the original job is triggered.
- Configure execution of the original job depending on upstream job Build status.

Downstream Jobs

- Downstream job is one that is triggered after the original job is triggered.
- Configure downstream job execution depending on the Build status of the original job.

Upstream and Downstream Jenkins Job

Given below is the order of job triggers:



Upstream Jenkins Job Configuration

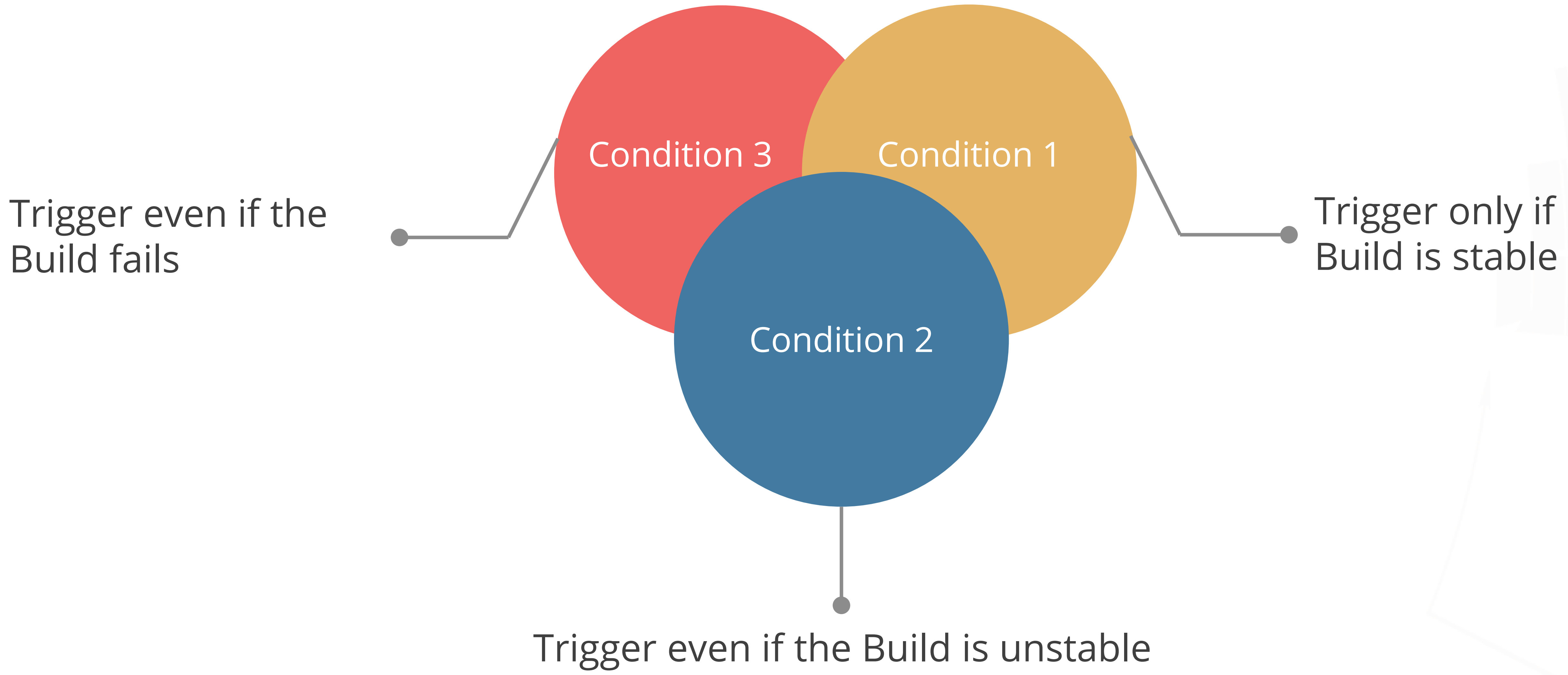
Upstream job refers to the job that triggers first.

Configuring Upstream Job

- Depends on the status of the current triggered Jenkins job
- Executes before running Jenkins job configuration

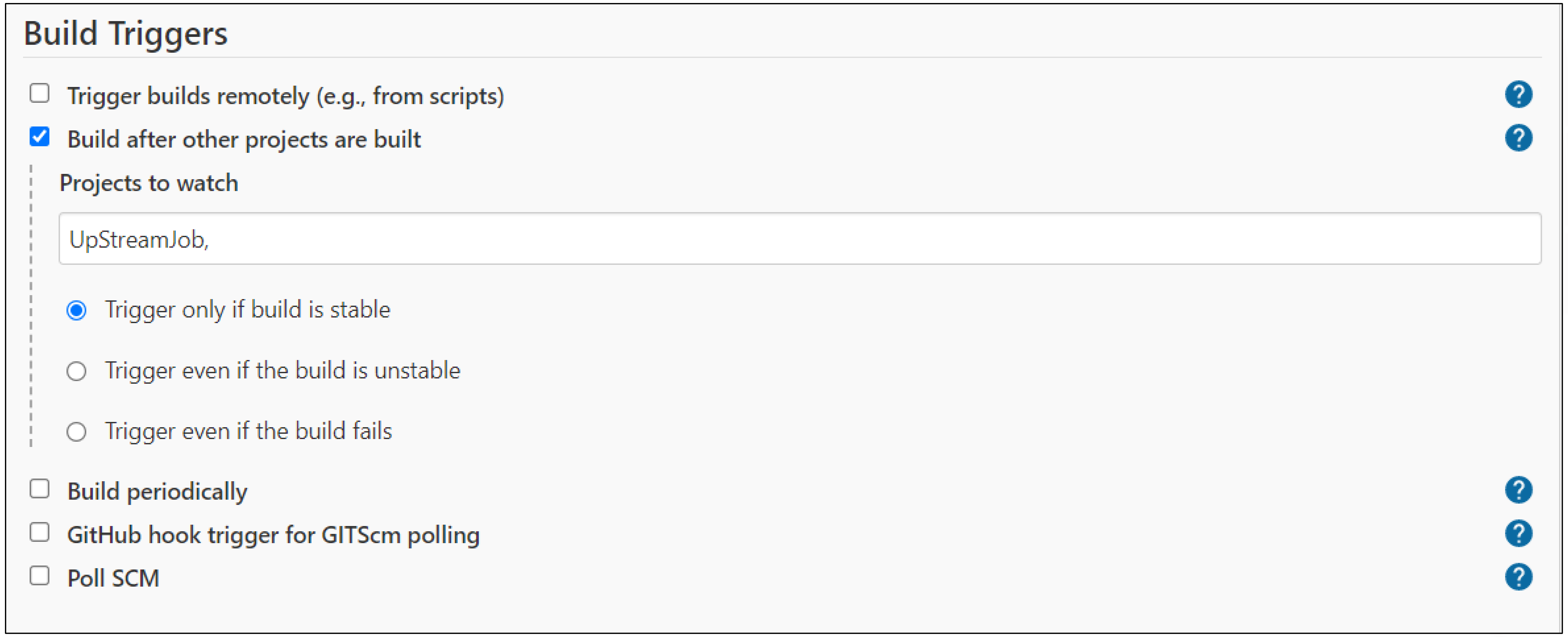
Upstream Jenkins Job Configuration

Stated below are some of the conditions under which developers can trigger current job, depending on the upstream Jenkins job status:


Trigger even if the Build fails
Condition 3
Condition 1
Trigger only if Build is stable
Condition 2
Trigger even if the Build is unstable

Upstream Jenkins Job Configuration

To configure Build condition for an upstream Jenkins job, select the appropriate condition in the Build Triggers tab as shown.



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [✓] Build after other projects are built

Projects to watch

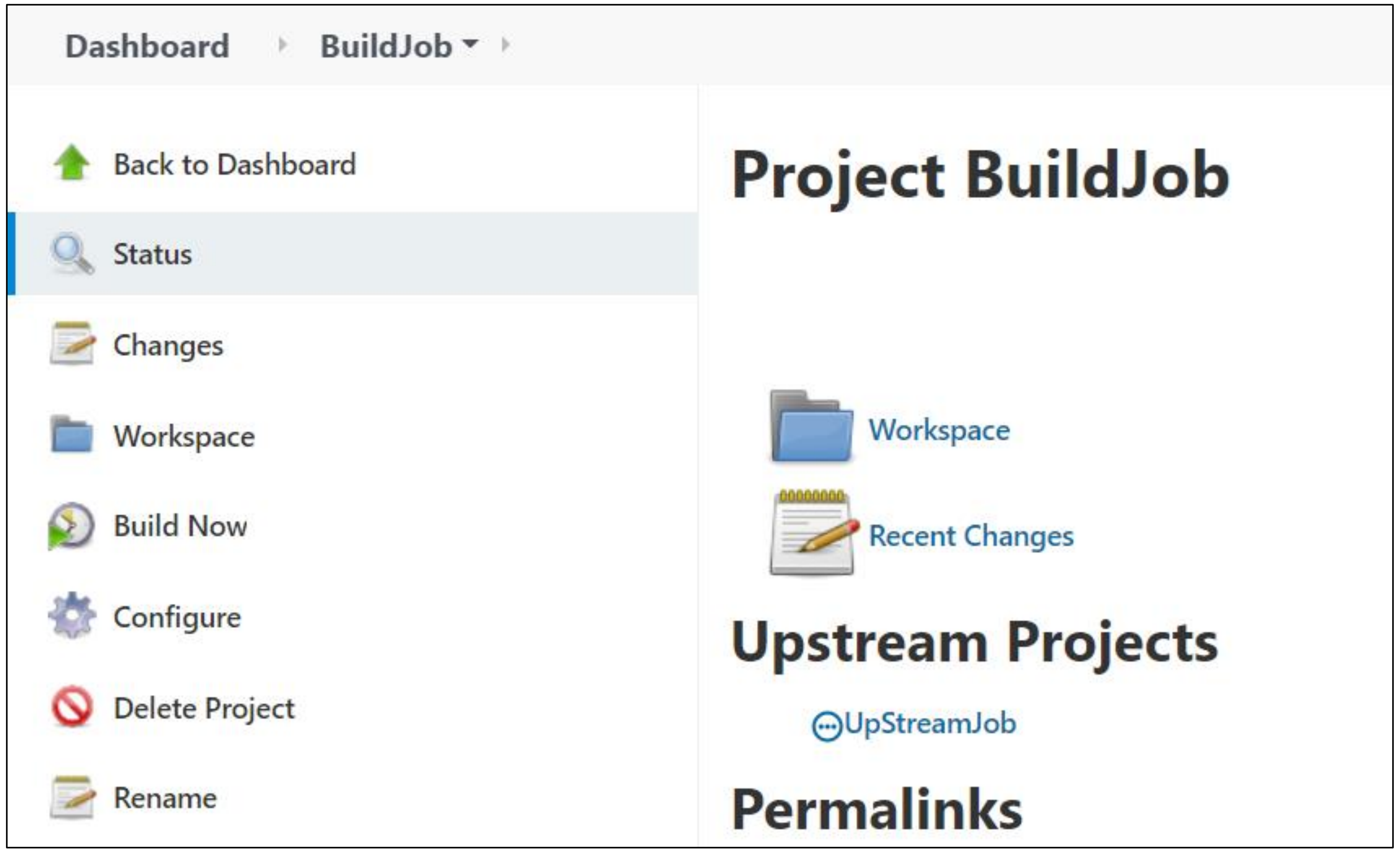
UpStreamJob,

- ● Trigger only if build is stable
- ○ Trigger even if the build is unstable
- ○ Trigger even if the build fails

- [ ] Build periodically
- [ ] GitHub hook trigger for GITScm polling
- [ ] Poll SCM

Upstream Jenkins Job Configuration

This is a screenshot of the Dashboard screen, which displays the status of upstream projects.



Downstream Jenkins Job Configuration

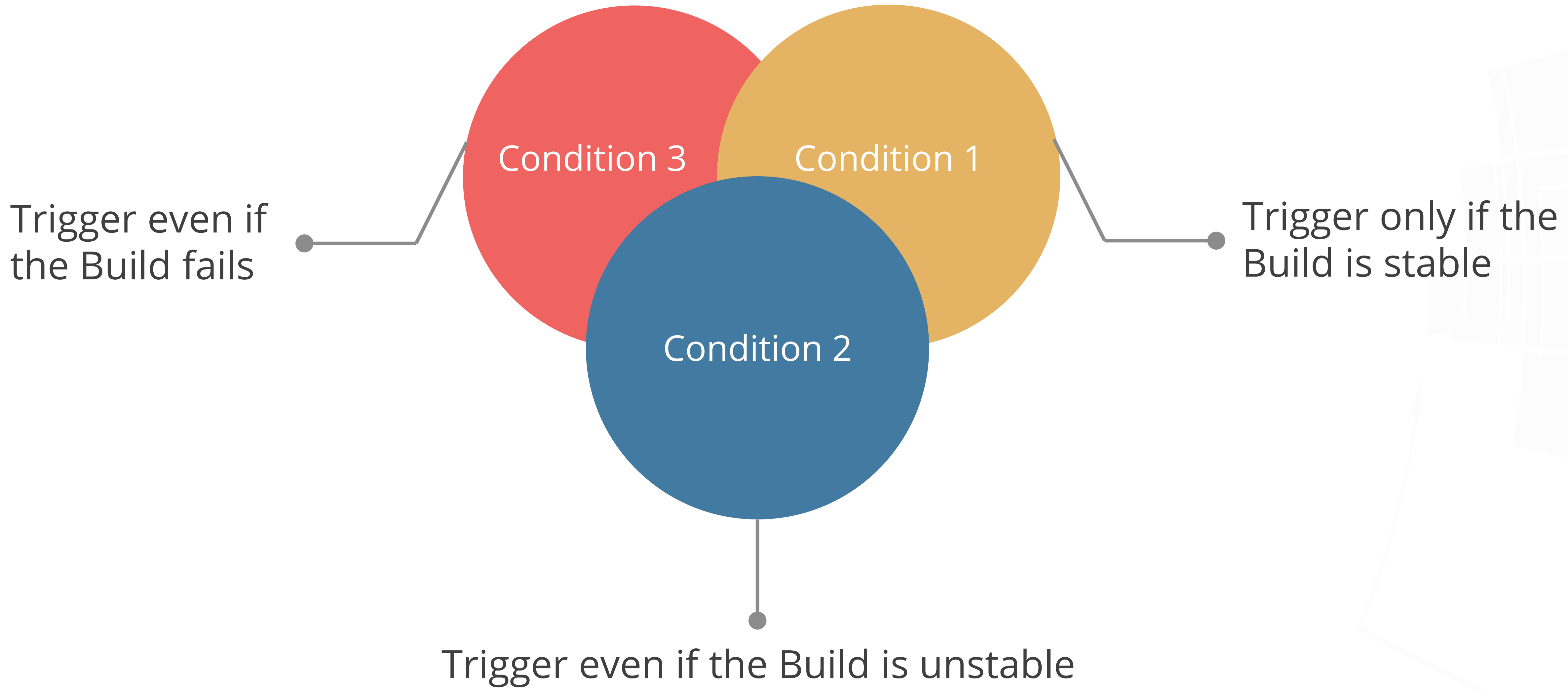
Downstream job refers to the job that triggers after upstream job.

Configuring Downstream Job

- Executes after running Jenkins job configuration
- Triggers irrespective of the status of the Build

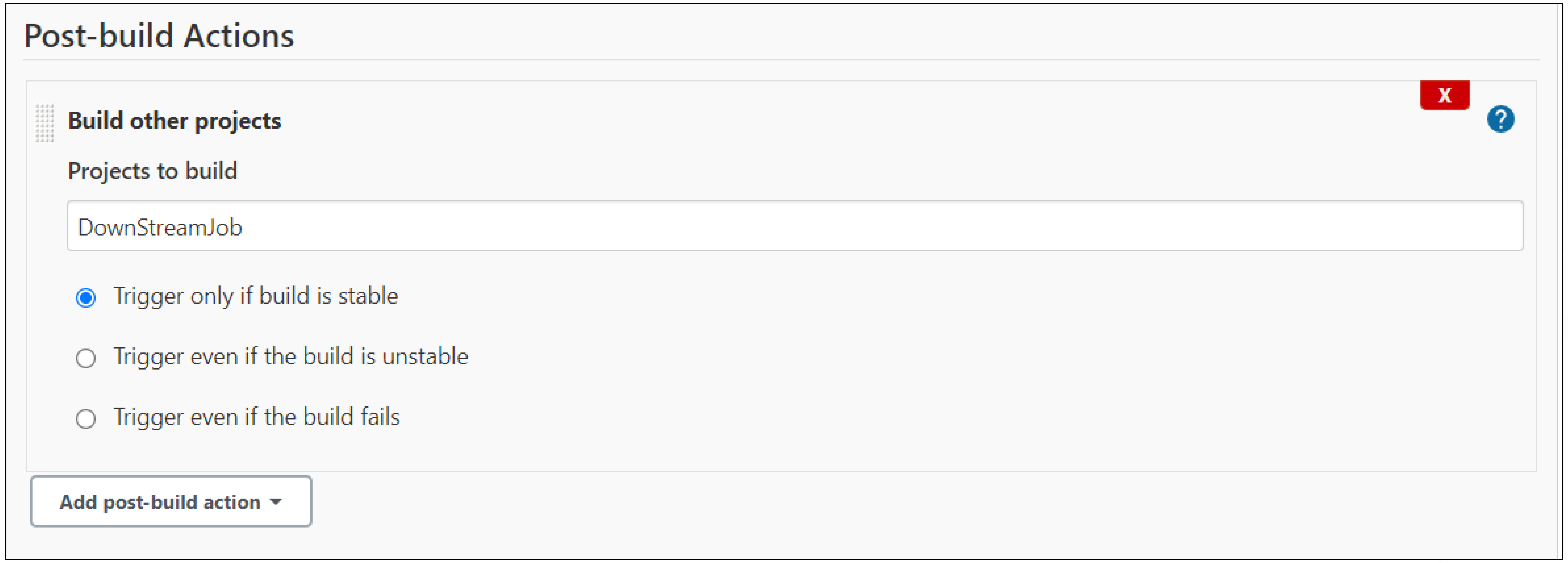
Downstream Jenkins Job Configuration

Stated below are some of the conditions under which you can trigger downstream Jenkins job from the current Jenkins job:


Condition 3
Condition 1
Condition 2
Trigger even if the Build fails
Trigger only if the Build is stable
Trigger even if the Build is unstable

Downstream Jenkins Job Configuration

To configure Build condition for a downstream Jenkins job, select the appropriate condition in the Post-build Actions tab as shown:



Post-build Actions

Build other projects

Projects to build

DownStreamJob

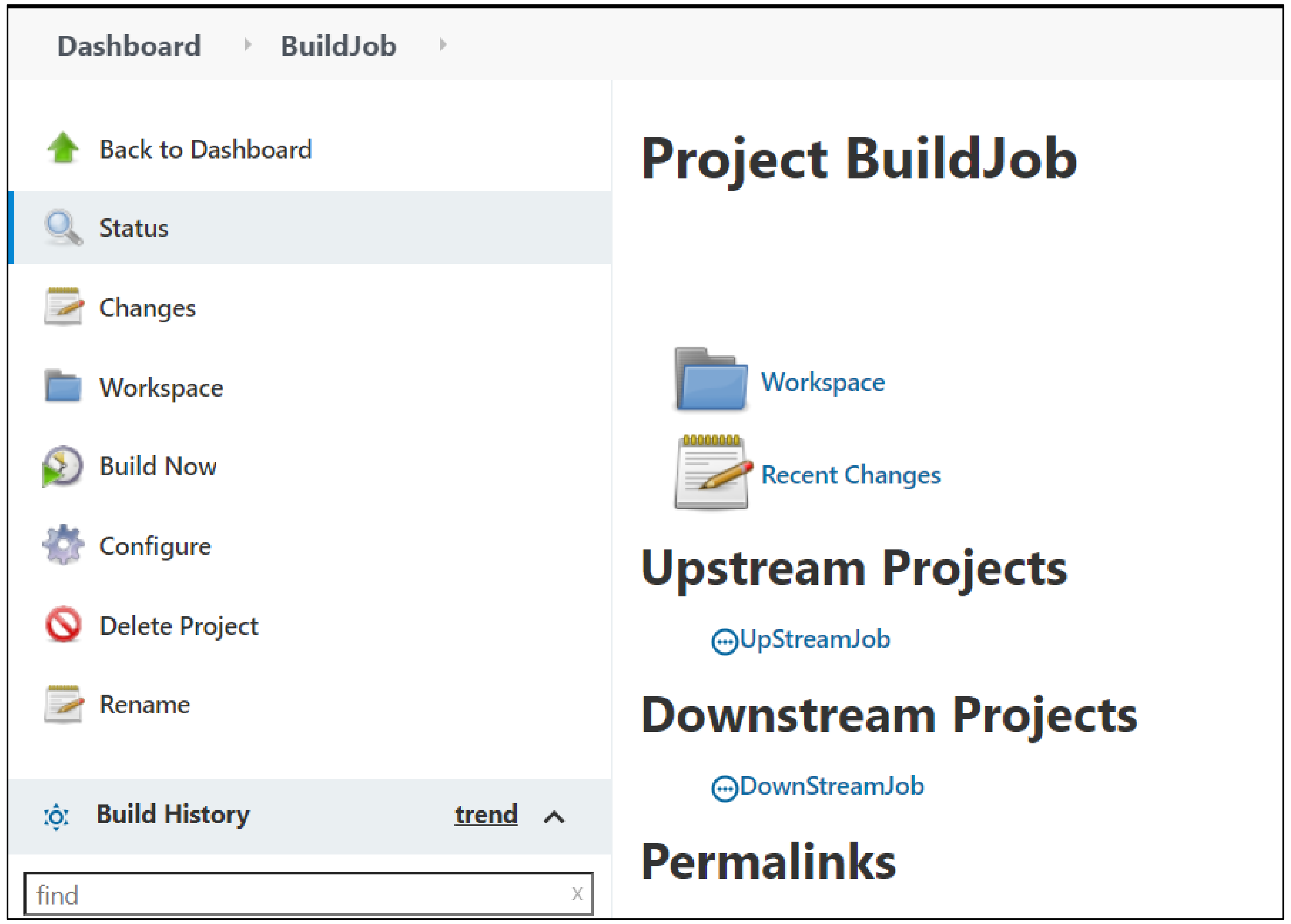
[✓ Trigger only if build is stable]

Trigger even if the build is unstable

Trigger even if the build fails

Downstream Jenkins Job Configuration

Shown below is the screenshot of the Dashboard screen which displays the status of downstream projects:


Dashboard › BuildJob ›

Back to Dashboard
Status
Changes
Workspace
Build Now
Configure
Delete Project
Rename

Build History trend

find

Project BuildJob

Workspace
Recent Changes

Upstream Projects

UpStreamJob

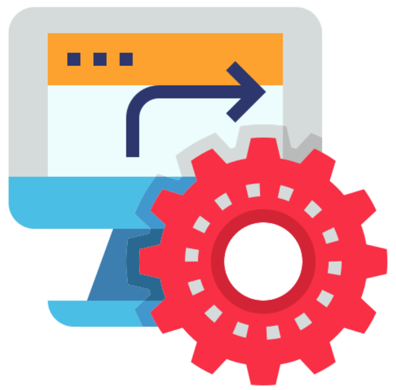
Downstream Projects

DownStreamJob

Permalinks

Build Periodically

To schedule a project periodically at different schedules, use the Build Periodically trigger.



Build Periodically trigger executes Builds for a Jenkins job, depending on the CRON pattern that is specified in Jenkins job.

It executes Jenkins Build even if there are no new commits performed in the Git repository.

Build Periodically

Build Periodically configuration can be used for:

- Scheduling test cases execution
- Performing deploys to test environments
- Scanning static source code

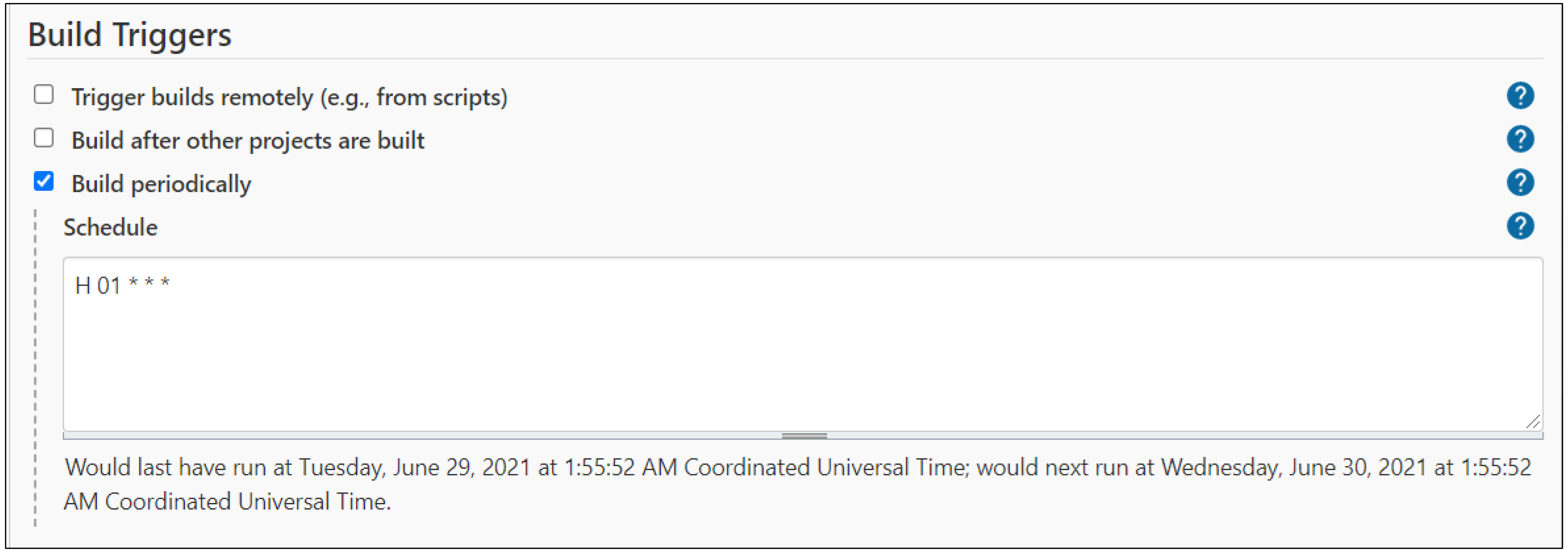
CRON Pattern

A CRON pattern is a string that is used as a schedule to execute a routing. Each CRON pattern specified in Jenkins job consists of five fields separated by a tab or a whitespace. The fields are:

1. **MINUTE:** Minutes within the hour (0–59)
2. **HOUR:** The hour of the day (0–23)
3. **DOM:** The day of the month (1–31)
4. **MONTH:** The month (1–12)
5. **DOW:** The day of the week (0–7) where 0 and 7 are Sunday

Build Periodically

To schedule a Build periodically, select the Build periodically option in the Build Triggers section, as shown below:



Build Triggers

- [ ] Trigger builds remotely (e.g., from scripts)
- [ ] Build after other projects are built
- [x] Build periodically

Schedule

```
H 01 * * *
```

Would last have run at Tuesday, June 29, 2021 at 1:55:52 AM Coordinated Universal Time; would next run at Wednesday, June 30, 2021 at 1:55:52 AM Coordinated Universal Time.

Note

Select Build periodically to schedule a Build that must be executed periodically or on a specified date and time.

Poll SCM

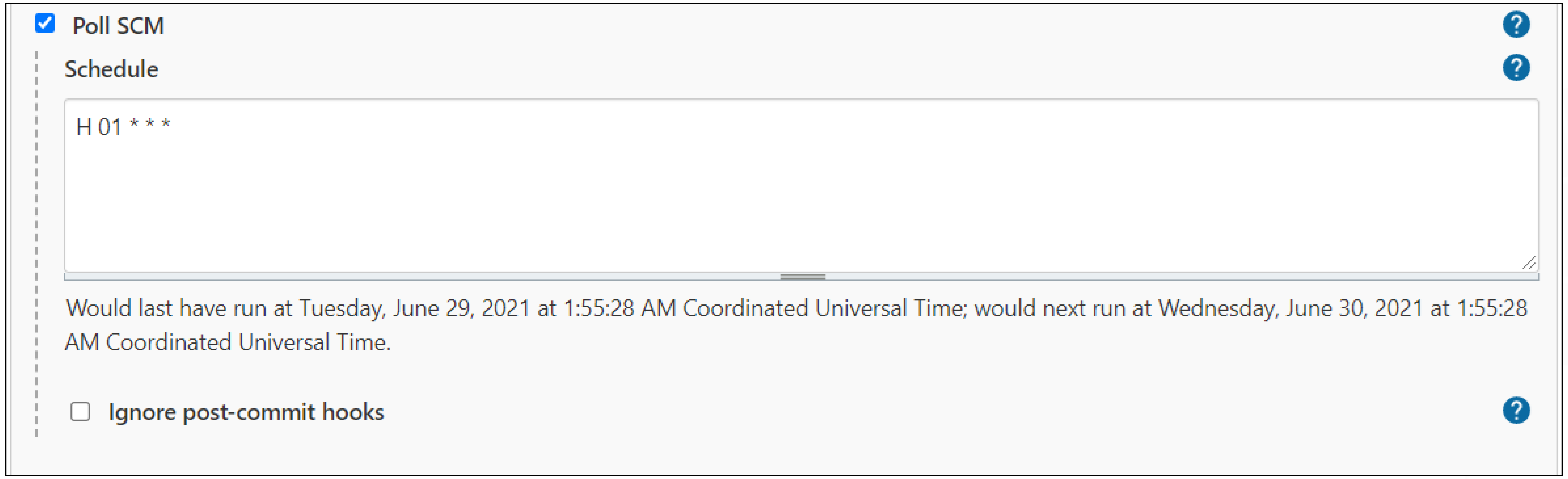
Poll SCM is the next best option to Build Periodically for automating Jenkins job trigger.

Poll SCM

- Configures a Build trigger only when there is a new commit in the repository
- Uses a CRON pattern, but triggers Jenkins job only when there are new commits since last Build
- Results in having less number of Build requests getting executed for a Jenkins job

Poll SCM

This is a screenshot of the Poll SCM screen that allows the developer to schedule the Build.



- [✓] Poll SCM

Schedule

```
H 01 * * *
```

Would last have run at Tuesday, June 29, 2021 at 1:55:28 AM Coordinated Universal Time; would next run at Wednesday, June 30, 2021 at 1:55:28 AM Coordinated Universal Time.

- [ ] Ignore post-commit hooks

Git Webhook Configuration

Git Webhook is an HTTP POST that occurs when developers perform push event with remote repositories like:

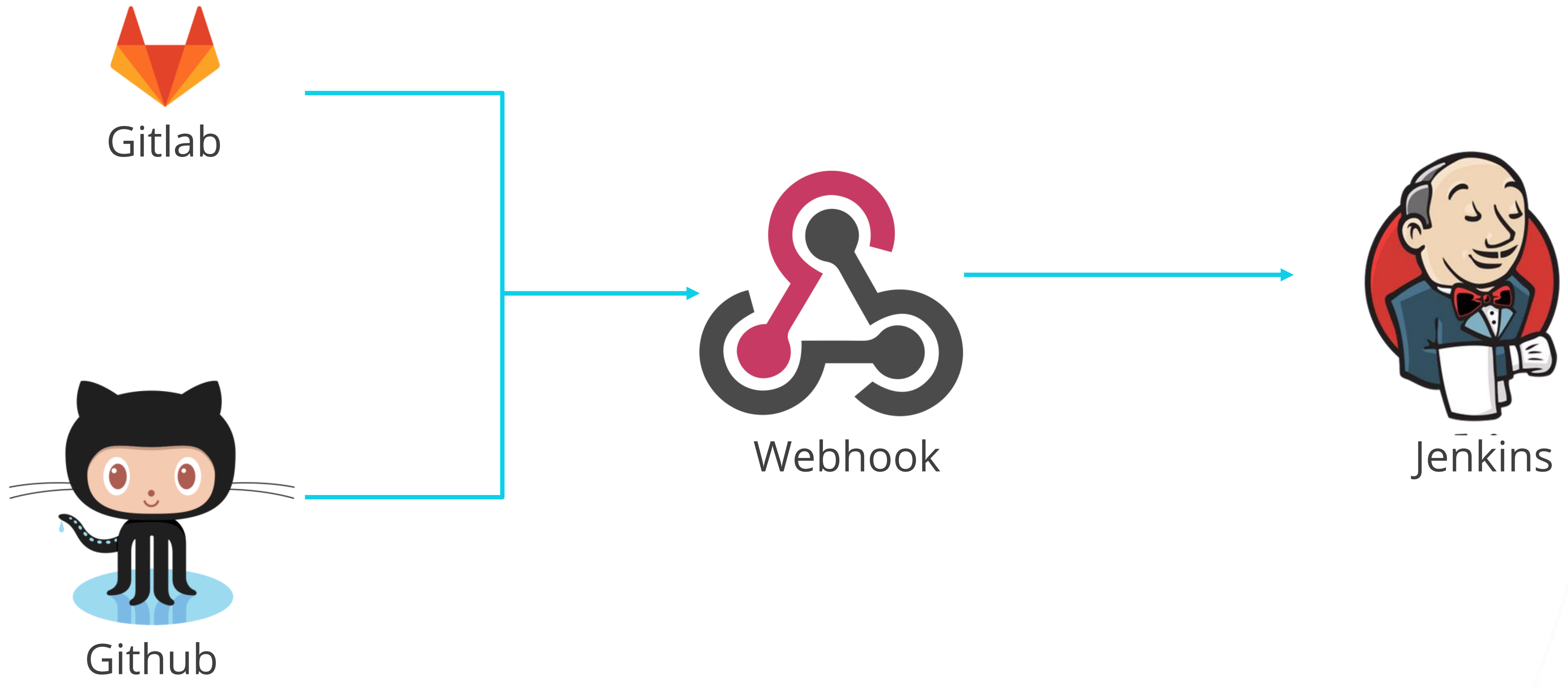
GitHub

Bitbucket

Gitlab

Git Webhook Configuration

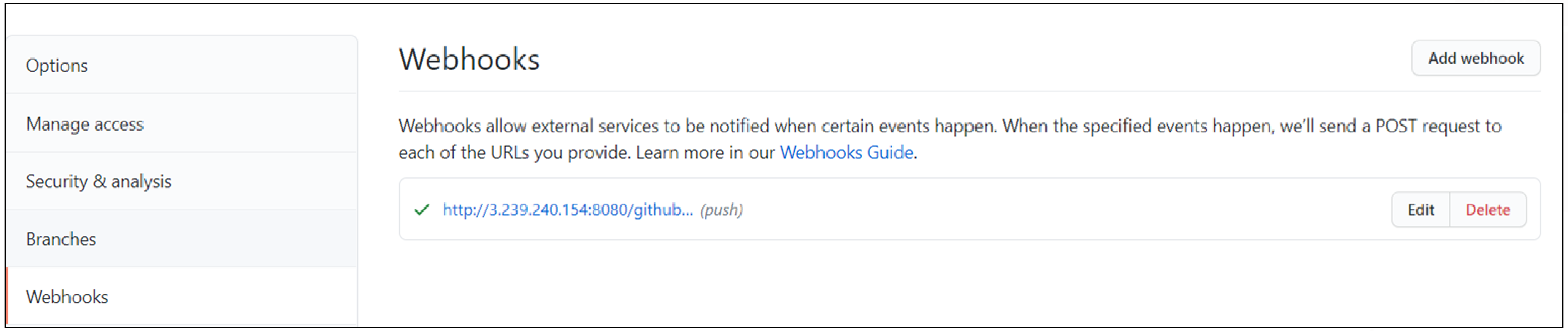
Git Webhook trigger is more effective than other triggers as it triggers the Jenkins job immediately.



Git Webhook Configuration

To configure Webhook, developers need to configure the Jenkins Webhook URL in Git repository in Webhooks section.

Webhook URL http://3.239.240.154:8080/github-webhook/



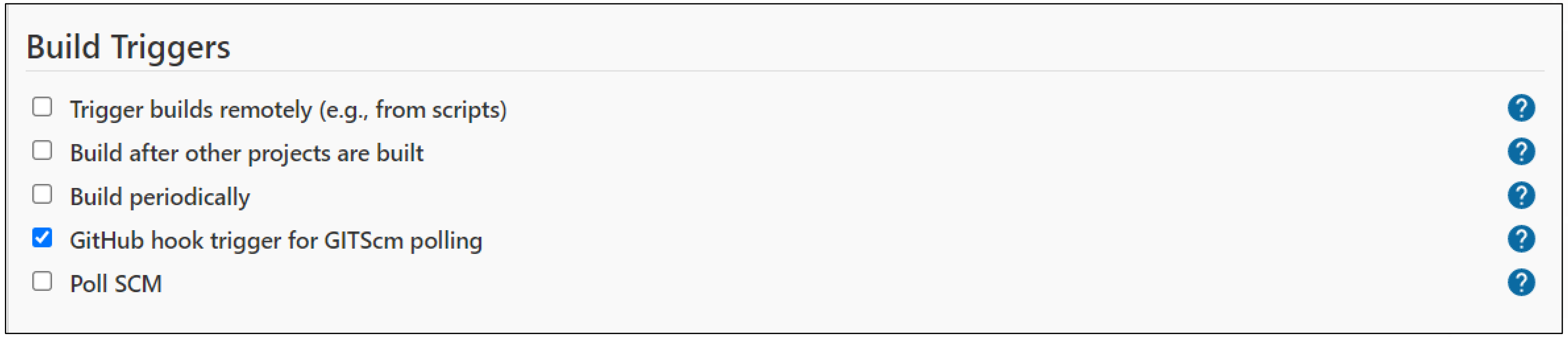
Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://3.239.240.154:8080/github... *(push)*

Git Webhook Configuration

To configure a GitHub hook trigger, select GitHub hook trigger for GITScm polling in the Build Triggers section, as shown below:

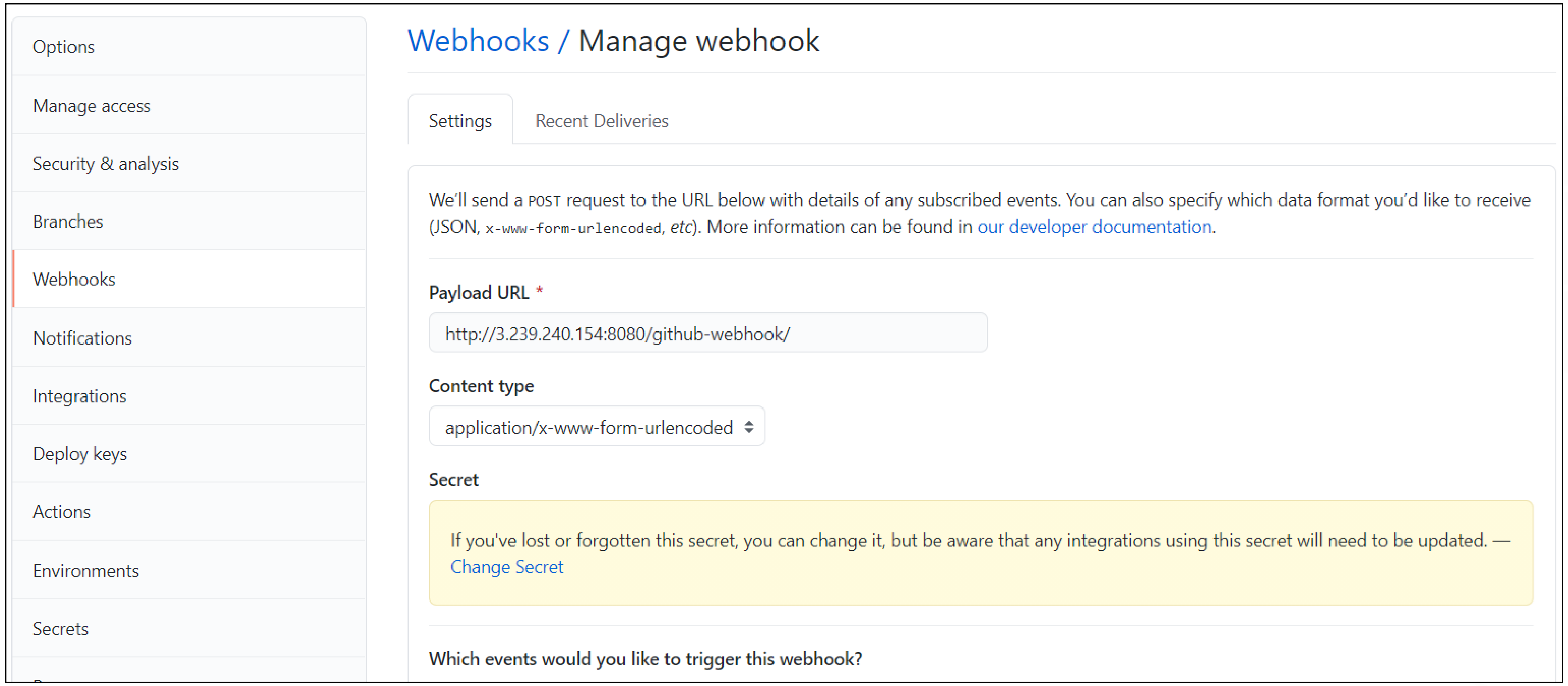


Build Triggers

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

Git Webhook Configuration

Webhook can be managed through the Webhooks / Manage webhook screen shown below:



We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in our developer documentation.

Payload URL *

http://3.239.240.154:8080/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

If you've lost or forgotten this secret, you can change it, but be aware that any integrations using this secret will need to be updated. — Change Secret

Which events would you like to trigger this webhook?

Assisted Practice

Setting up Poll SCM Configuration in Jenkins

Duration: 15 min

Problem Statement:

1. Set up the poll SCM configuration in Jenkins.

Assisted Practice: Guidelines

Steps to demonstrate how to set up Poll SCM configuration in Jenkins:

1. Log in to Jenkins CI tool and configure freestyle job to trigger build using Poll SCM trigger.

Assisted Practice

Configuring Webhook Trigger for Jenkins Job

Duration: 25 min

Problem Statement:

1. Configure the Webhook trigger for a Jenkins job.

Assisted Practice: Guidelines

Steps to demonstrate how to configure the Webhook trigger for a Jenkins job:

1. Log in to Jenkins CI tool and configure freestyle job to trigger build using webhook trigger.

Key Takeaways

- Jenkins can be used to archive artifacts for any Build but it takes disk space on the local Jenkins server.
- Jenkins Fingerprinting is a mechanism to track a file across different Builds and jobs.
- Jenkins job can be triggered by clicking on the Build Now option to execute a Build step.
- Build periodically configuration can be used for scheduling execution of test cases and performing deploys to test environments.
- In Jenkins, a Git Webhook trigger is more effective than other triggers.

Lesson-End Project

Configure GitHub Webhook

Problem Statement:

Perform the following:
- Configure GitHub Webhook with Jenkins.
- Trigger a downstream job for deployment.

Access: Click on the **Labs** tab on the left side panel of the LMS. Copy or note the username and password that is generated. Click on the **Launch Lab** button. On the page that appears, enter the username and password in the respective fields, and click **Login**.

Thank You