Improving the Pipeline

Updated 5/5/2022

Overview

In this lab, students will identify a way to improve pipeline efficiency, make the changes, and then confirm the impact of their actions.

Estimated time: 1 hr

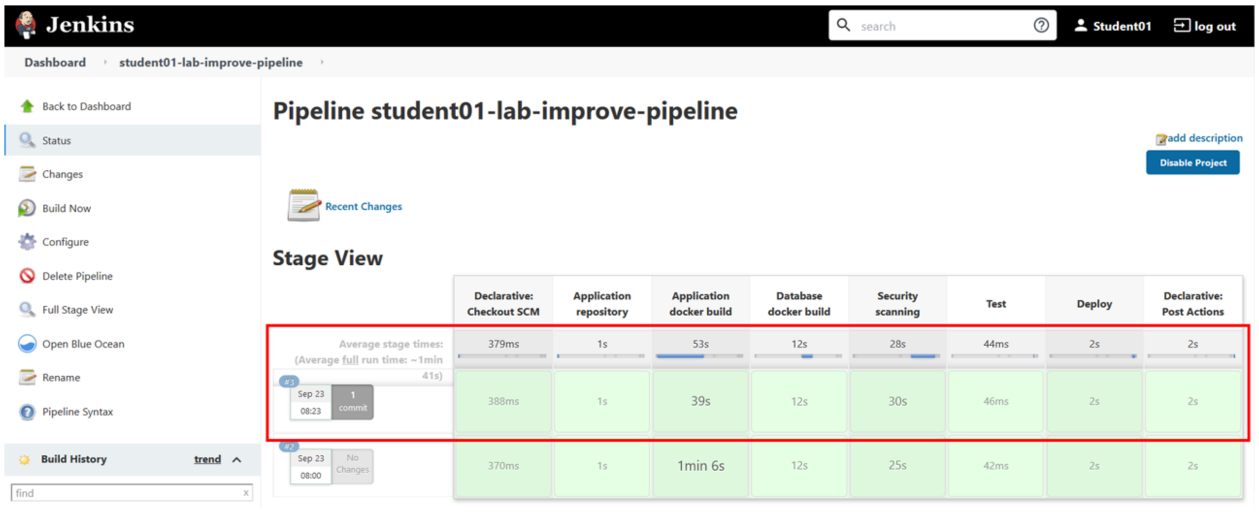
Learning Objectives

1. Monitor Jenkins "total" build times
2. Look for build time improvements
3. Make build improvements
4. Verify build is functioning properly
5. Verify "total" build time improved

Instructions

Identify Areas of Improvement

1. Monitor pipeline and identify the amount of time it currently takes for each stage to run



Pipeline student01-lab-improve-pipeline

Stage View

	Declarative: Checkout SCM	Application repository	Application docker build	Database docker build	Security scanning	Test	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 41s)	379ms	1s	53s	12s	28s	44ms	2s	2s
#3 Sep 23 08:23 1 commit	388ms	1s	39s	12s	30s	46ms	2s	2s
#2 Sep 23 08:00 No Changes	370ms	1s	1min 6s	12s	25s	42ms	2s	2s

Build times can be improved by running the pipelines *concurrently* instead of *sequentially*. We can also see that any changes in the application code will always trigger a build of both the application and database code/images which is unnecessary as the database code is not frequently changing. This separation will be the focus of this lab.

After separating them, changes in the database code will only trigger a build of the database.

Make Build Improvement

1. Create a new Jenkinsfile for the database pipeline in the `devops-camp-pipeline` folder called `devops-camp-db-jenkinsfile`.

> **Note:** Like the application pipeline, the database pipeline will be cloning the application repository (i.e., `afs-labs-student`). However, the pipeline will only build the database image when there are changes to `database.sql`

a. In your new `devops-camp-db-jenkinsfile`, paste the following template:

```
pipeline {
    agent {
        label 'jenkins-agent'
    }
    //TODO: add environment variables
    stages {
        //TODO: add stages to pipeline
    }
    post {
        cleanup {
            echo "Clean workspace"
            sh 'rm -rf .git ./*'
        }
    }
}
```

b. Add environment variables for Harbor to the `pipeline {}` by pasting the below code snippet *after* the `agent {}` block (and *before* the `stages {}` block) to the `devops-camp-db-jenkinsfile`

> **Tip:** *Replace* `<YOUR HARBOR PROJECT NAME>` to match your Harbor project name.

```
environment {
        PIPELINE_HASH = sh(script: 'git rev-parse --short HEAD', returnStdout: true).trim()
        HARBOR_REGISTRY = 'registry.dev.afsmtddso.com'
        HARBOR_PROJECT = '<YOUR HARBOR PROJECT NAME>'
        DB_IMAGE_NAME = 'db'
    }
```

c. Add a stage to clone the application repository *inside* the `stages {}` block.

i. *Replace* `<YOUR AFS-LABS-STUDENT REPO URL>` with your GitHub application repository URL.

ii. In the snippet below, we modify the script for the environmental variable `COMMIT_HASH` to track any changes in the `database.sql` file and grab its commit hash.

```
stage('Application repository') {
            steps {
                echo "Cloning application repository"
                sh 'git clone <YOUR AFS-LABS-STUDENT REPO URL>'
                dir('afs-labs-student') {
                    script {
                        env.COMMIT_HASH = sh(script: 'git log --format=format:%h -1 --follow
                    }
                }
            }
        }
```

> **Note:** `COMMIT_HASH` and `PIPELINE_HASH` will be used to uniquely "tag" the database Docker image.

2. Now we will create a Python script called `check_harbor_db.py`. The script will compare the `database.sql` commit hash and Harbor database image tag.

   a. In your terminal, navigate to the `devops-camp-pipeline` folder

   b. Download the `check_harbor_db.py` script by running the following command:

```
curl -O 'https://raw.githubusercontent.com/khaledAFS/sample-files/main/post_improving_pipeli
```

   c. The `check_harbor_db.py` script will return a true/false value representing if there have been any changes made to `database.sql`. The return value will be used to determine whether or not to build the database image. If no changes have been made, we can continue using a previous build.

> **Note:** Please take the time to read through the script and understand how this comparison is made.

3. In your `devops-camp-db-jenkinsfile`, paste the following code snippet to the `stage('Application repository') {}`, *after* the `dir('afs-labs-student') {}` block to utilize the `check_harbor_db.py` script.

> **Note:** We're adding an environment variable, `BUILD_DB`, and assigning it to execute the `check_harbor_db.py` script and store either true or false.

```
withCredentials([usernamePassword(credentialsId: '<YOUR CREDENTIAL ID NAME>', usernameVariab
                    script {
                        env.BUILD_DB = sh(script: 'python check_harbor_db.py -h $COMMIT_HASH
                    }
                }
```

> **Tip:** *Replace* the `credentialsID` value with the ID of the credentials you created in Jenkins (e.g., `<first initial + lastname>-harbor-auth`). Do **not** update the `USERNAME` and `PASSWORD` values.

a. Add the following code snippet *after* the `stage('Application repository') {}` block.

> **Note:** Our new environment variable `BUILD_DB` will be used in the `stage('DB changes: true') {}` block. When there are changes to `database.sql`, `BUILD_DB` will be set to `true`. Then the stages to build, scan, and deploy the database will execute. If `BUILD_DB` is set to `false`, the pipeline will skip this stage.

```
stage('DB changes: true') {
           when {
              environment name: 'BUILD_DB', value: 'true'
           }
           stages {
              stage('Database docker build') {
                  steps {
                      echo "Building database image"
                      //TODO: build docker image & push to Harbor
                  }
                  post {
                      //TODO: clean local docker images
                  }
              }
              stage('Deploy') {
                  steps {
                      echo "Deployment stage"
                      //TODO: deploy database
                  }
              }
           }
        }
```

> **Explanation:**
>
> The `stage('DB changes: true') {}` is a conditional stage (i.e., the `when {}` block) that contains a `stages {}` block. Think of this as a nested concept, so

> within the stages there will be multiple `stage {}` blocks.

b. Add the following code snippet to the `stage('Database docker build') {}`, *inside* the `steps {}` block to build the database image:

```
withCredentials([usernameColonPassword(credentialsId: '<YOUR CREDENTIAL ID NAME>', variable:
                          script {
                              sh 'docker build -t $DB_IMAGE_NAME-$COMMIT_HASH -f ./db/Docker
                              docker.withRegistry('https://$HARBOR_REGISTRY', '<YOUR CREDENT
                                  sh 'docker tag $DB_IMAGE_NAME-$COMMIT_HASH $HARBOR_REGISTR
                                  sh 'docker push $HARBOR_REGISTRY/$HARBOR_PROJECT/$DB_IMAGE
                              }
                          }
                      }
```

> **Tip:** *Replace* both occurences of `<YOUR CREDENTIAL ID NAME>` to match your Harbor credential ID name (i.e., `<first initial + last name>-harbor-auth`).

c. Add the following code to the `stage('Database docker build') {}`, *inside* the `post {}` block to clean the Docker images built:

```
always {
                          echo "Clean local $DB_IMAGE_NAME image"
                          script {
                              try {
                                  sh 'docker rmi $DB_IMAGE_NAME-$COMMIT_HASH:latest'
                                  sh 'docker rmi $HARBOR_REGISTRY/$HARBOR_PROJECT/$DB_IMAGE_
                              } catch (err) {
                                  echo err.getMessage()
                              }
                          }
                      }
```

d. Add the following line to the `stage('Deploy') {}`, *inside* the `steps {}` block to deploy the containers to EKS:

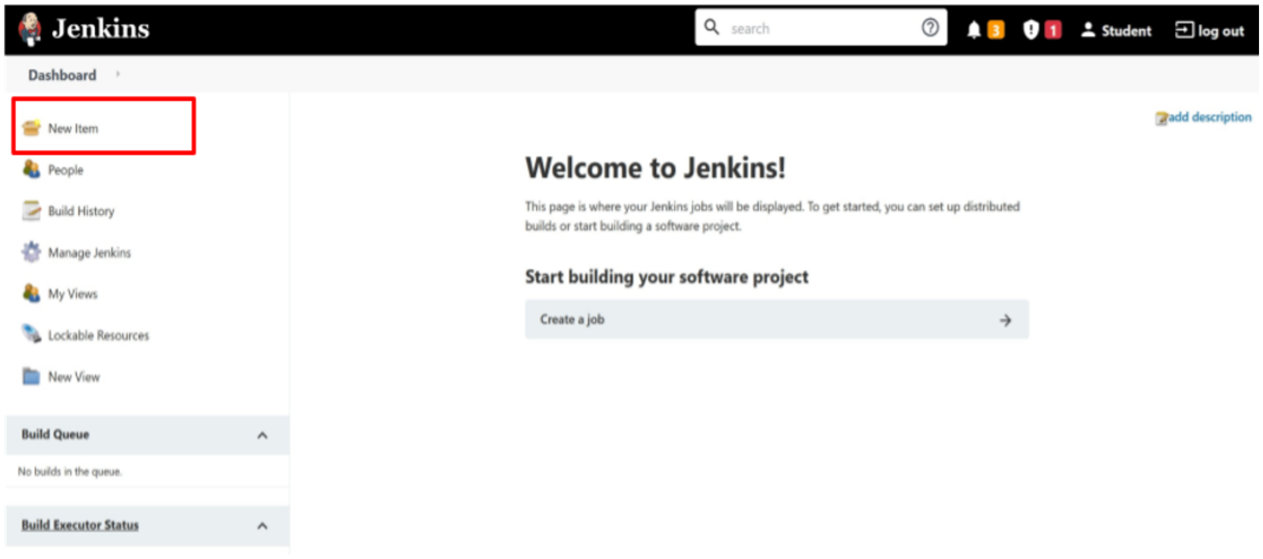
```
sh 'kubectl -n <YOUR EKS NAMESPACE> set image deployment/db-deployment db-deployment=$HARBOR
```

> **Tip:** Make sure you replace `<YOUR EKS NAMESPACE>` with your personal EKS namespace (e.g., first initial + last name)

e. Save your changes

4. Clean up your `devops-camp-jenkinsfile` (i.e., the original one) by removing everything associated with the database, including environment variables, the DB build stage, the DB deployment stage, etc.
5. In the terminal, make sure you're in the right directory before adding, committing, and pushing your files to GitHub like you've done previously
6. **Navigate to the Jenkins website and verify that your pipeline builds successfully; if not, debug and resolve the issue(s) before moving on**
7. Create a new pipeline for building your database by following the instructions below:

a. On the Jenkins website, under the "Dashboard", click on "New Item" on the left



Welcome to Jenkins!

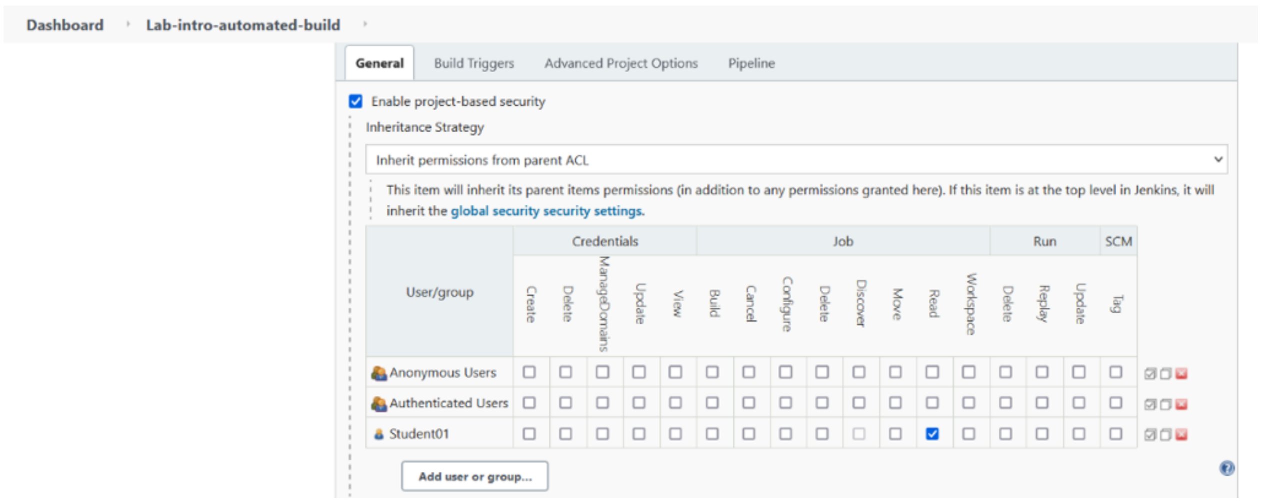
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.

Start building your software project

No builds in the queue.

b. Name the pipeline `<YOUR FIRST INITIAL + LAST NAME>-db-pipeline`, click "Pipeline," then click "OK" at the bottom to create the pipeline.

c. On the next page, if "Enable project-based security" isn't checked already, click on the checkbox and ensure that the options match those of the image below:



Dashboard › Lab-intro-automated-build ›

General | Build Triggers | Advanced Project Options | Pipeline

☑ Enable project-based security

Inheritance Strategy

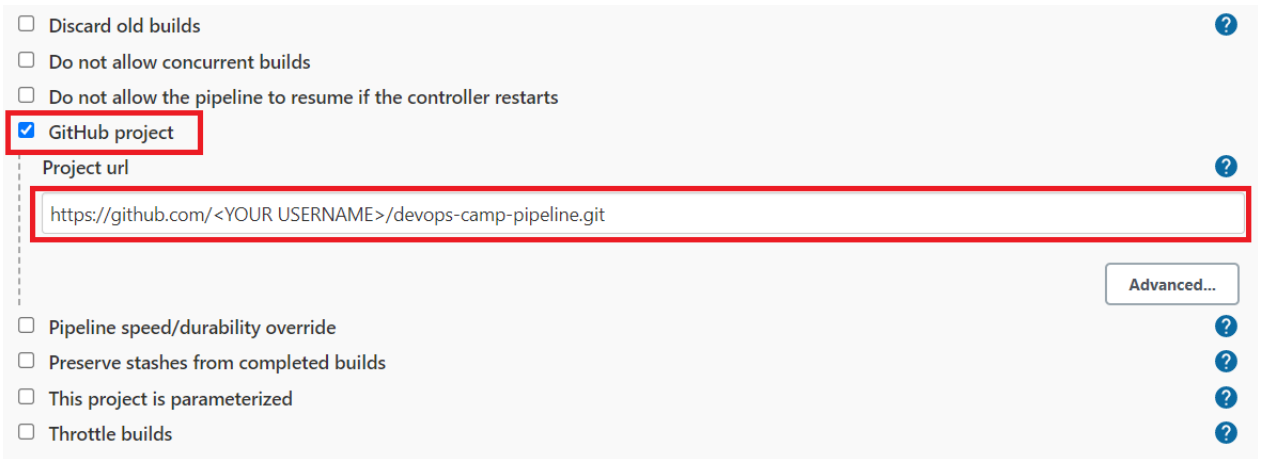
Inherit permissions from parent ACL

This item will inherit its parent items permissions (in addition to any permissions granted here). If this item is at the top level in Jenkins, it will inherit the global security security settings.

User/group	Credentials					Job								Run			SCM
	Create	Delete	ManageDomains	Update	View	Build	Cancel	Configure	Delete	Discover	Move	Read	Workspace	Delete	Replay	Update	Tag
Anonymous Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Authenticated Users	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Student01	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐

Add user or group...

d. A few options underneath, check "GitHub project" and paste the URL (including the .git portion) of your `devops-camp-pipeline` repository.

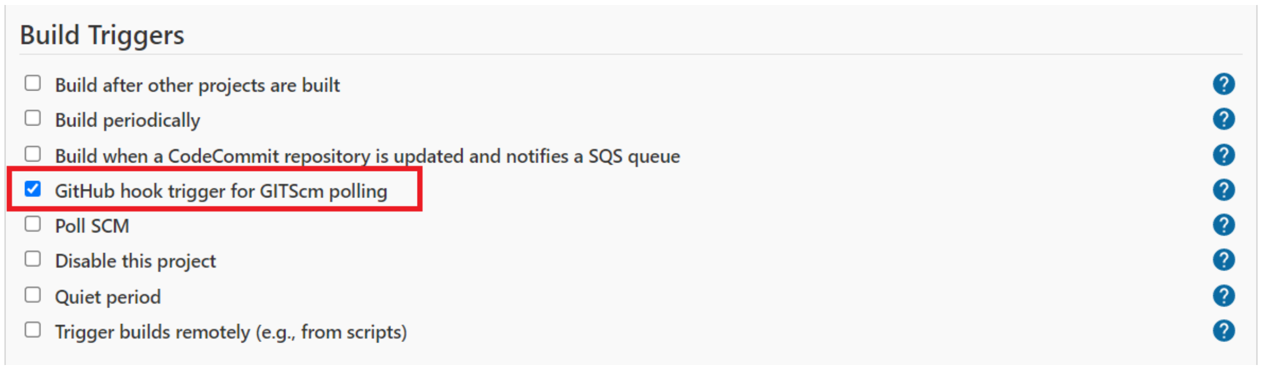


☐ Discard old builds
☐ Do not allow concurrent builds
☐ Do not allow the pipeline to resume if the controller restarts
☑ GitHub project
Project url
https://github.com/<YOUR USERNAME>/devops-camp-pipeline.git

Advanced...

☐ Pipeline speed/durability override
☐ Preserve stashes from completed builds
☐ This project is parameterized
☐ Throttle builds

e. Scroll down to the "Build Triggers" subsection and check "GitHub hook trigger for GITScm polling"



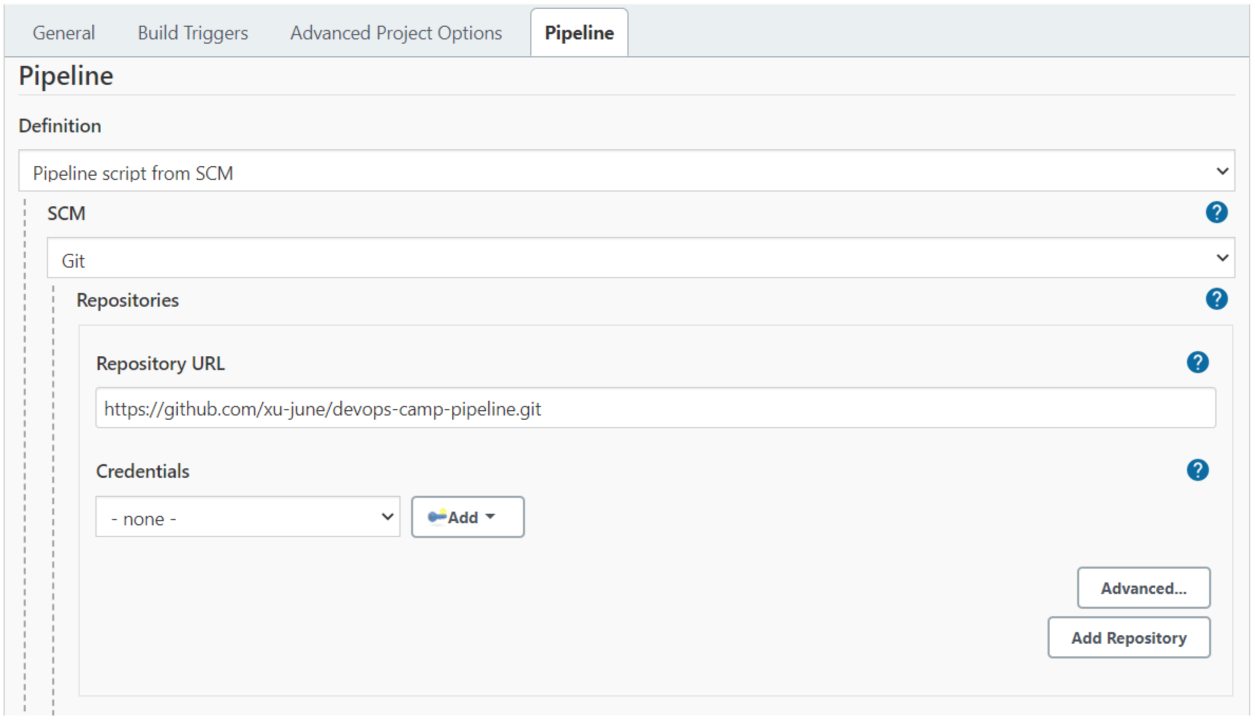
Build Triggers

☐ Build after other projects are built
☐ Build periodically
☐ Build when a CodeCommit repository is updated and notifies a SQS queue
☑ GitHub hook trigger for GITScm polling
☐ Poll SCM
☐ Disable this project
☐ Quiet period
☐ Trigger builds remotely (e.g., from scripts)

f. Select the "Pipeline" tab at the top. Under "Definition," click "Pipeline script" to open the dropdown menu; select "Pipeline script from SCM"

g. In the SCM dropdown menu, select "Git"

h. In "Repositories" section, paste your `devops-camp-pipeline` repository URL (including the .git portion)



Pipeline

Definition: Pipeline script from SCM

SCM: Git

Repositories

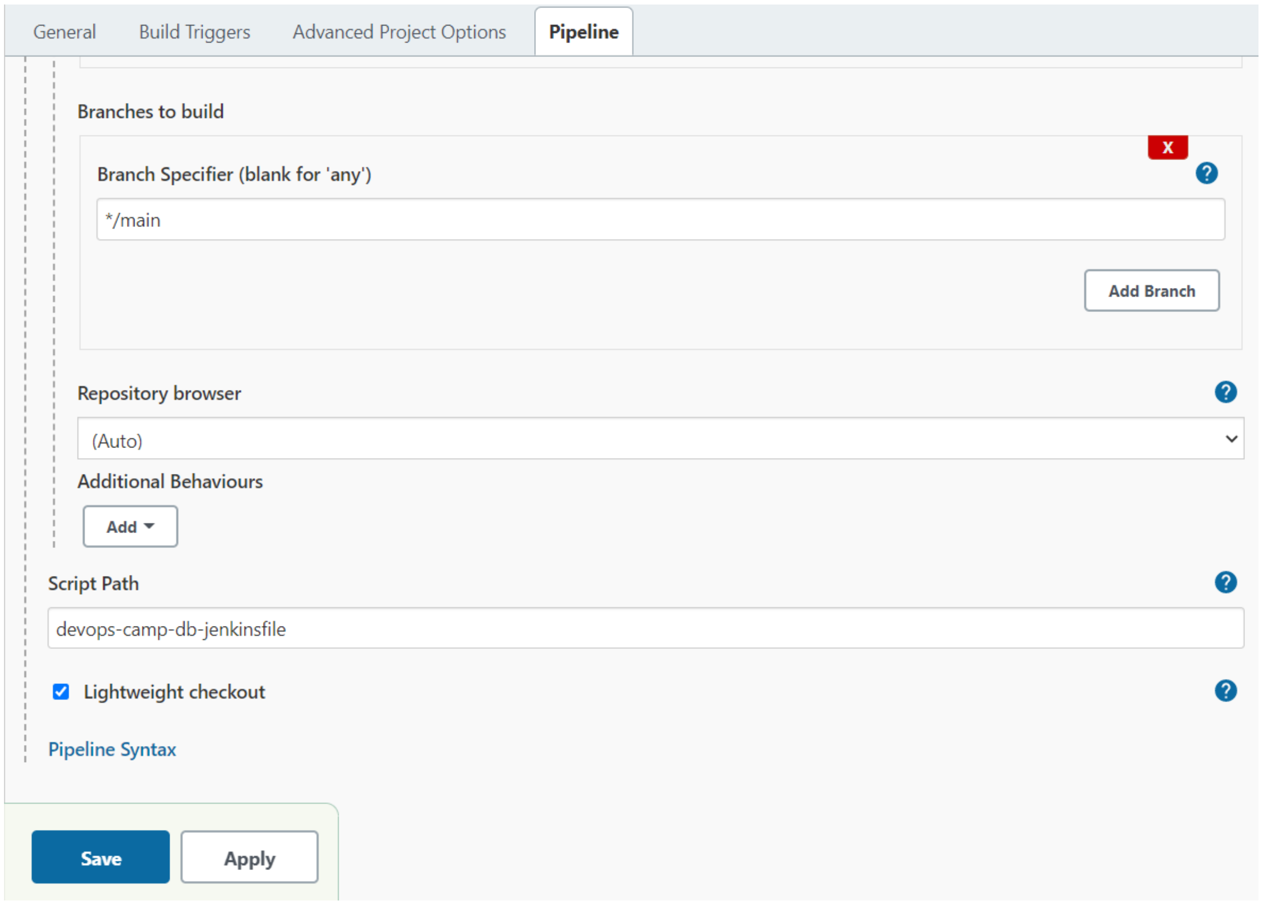
Repository URL: https://github.com/xu-june/devops-camp-pipeline.git

Credentials: - none -

i. In "Branches to build," make sure you put your main branch; this could be either `main` or `master`

j. Replace the default "Script Path" with your updated Jenkinsfile name, `devops-camp-db-jenkinsfile`

k. Save your new pipeline by clicking the "Save" button at the bottom



Branches to build

Branch Specifier (blank for 'any')

*/main

Repository browser

(Auto)

Additional Behaviours

Script Path

devops-camp-db-jenkinsfile

☑ Lightweight checkout

8. In GitHub, create a second webhook for your `afs-labs-student` repository by following the instructions below:

a. Navigate to your GitHub `afs-labs-student` repository

b. Under Settings → Webhooks, click on "Add webhook"

c. In the "Payload URL" box, paste in the following URL filled in with your information:

```
https://<JENKINS USERNAME>:<API TOKEN>@jenkins.dev.afsmtddso.com/job/<JENKINS PIPEL
```

> **Tip:** Your API token is the token that you created and saved in one of the first labs. Your Jenkins pipeline name should be `<YOUR FIRST INITIAL + LAST NAME>-db-pipeline`

c. Make sure the "Content type" is application/json

d. Click on "Add webhook" to create the webhook

> **Note:** If done correctly, adding the webhook will trigger a build in db pipeline.

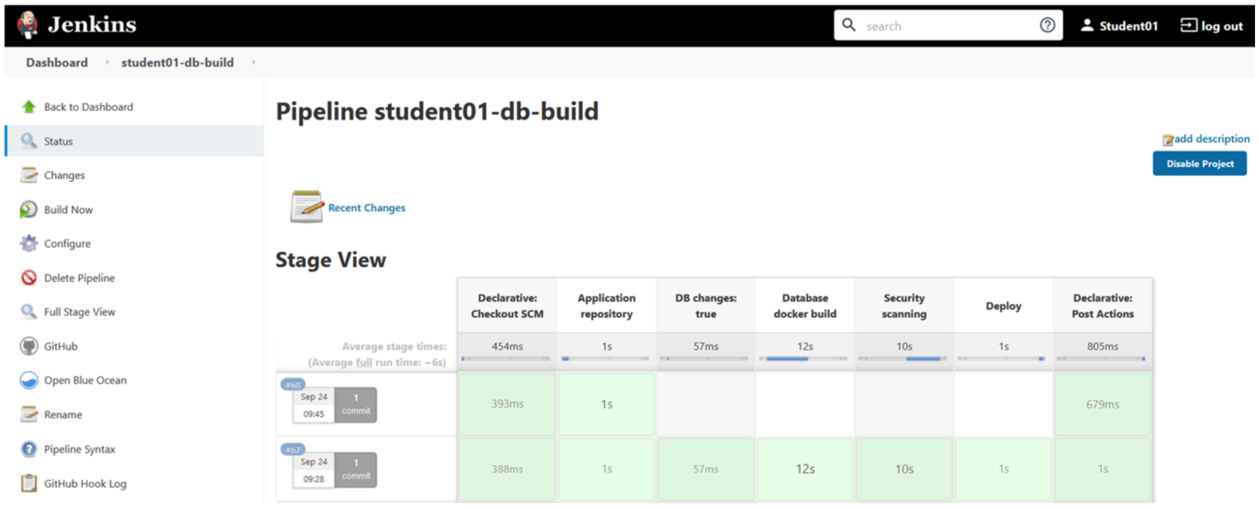
9. In VSCode, find and open `database.sql` in your `afs-labs-student/database` folder
10. Add a comment to the first line (comments are preceded by `--`)
11. In the terminal, make sure you're in the `afs-labs-student` directory before commiting and pushing the updated code

    a. Check to make sure that your new database pipeline automatically builds when changes are made (and pushed) in `database.sql`
12. **Ensure that both of your pipelines build successfully (i.e., all the stages are green)**
13. **Navigate to your website and ensure that it displays all features before moving on**

Verify Improvements

1. Make a change to your application code, in any file besides `database.sql`, and ensure that the application pipeline builds after the change is pushed to your GitHub repository.

    a. Ensure that the database pipeline doesn't reach the database build stage; it should look like the one shown in the image below:



Pipeline student01-db-build

Stage View

	Declarative: Checkout SCM	Application repository	DB changes: true	Database docker build	Security scanning	Deploy	Declarative: Post Actions
Average stage times: (Average full run time: ~6s)	454ms	1s	57ms	12s	10s	1s	805ms
#68 Sep 24 09:45 1 commit	393ms	1s					679ms
#67 Sep 24 09:28 1 commit	388ms	1s	57ms	12s	10s	1s	1s