

Steps to Projects to HDAP

Create Jenkins Job for the Project

Go to Jenkins application <https://apps2.hdap.gatech.edu/jenkins/>

Login to Jenkins



Press the **log in** link in the upper right corner of the screen

Use your GT credentials to log in to the system.

Create a New Item



-  New Item
-  People
-  Build History
-  Project Relationship
-  Check File Fingerprint
-  Manage Jenkins
-  My Views

Select **New Item**

Configure a New Pipeline Project

Enter an item name

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

Orchestrates long-running activities that can span multiple build slaves. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Bitbucket Team/Project**

Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

if you want to create a new item from other existing, you can use this option:



Copy from

OK

For the **Enter an item name** field type a name for the student project

Select **Pipeline** as the project type

Press the **OK** button

Configure the Pipeline Project

Add a Project Name and Description

The screenshot shows the 'General' tab of a Pipeline configuration interface. At the top, there are four tabs: 'General', 'Build Triggers', 'Advanced Project Options', and 'Pipeline'. The 'General' tab is active. It contains a 'Pipeline name' text input field and a 'Description' text area. Below these is a '[Plain text] Preview' link. A list of checkboxes follows: 'Discard old builds', 'Do not allow concurrent builds', 'Do not allow the pipeline to resume if the master restarts.', 'GitHub project', 'Pipeline speed/durability override', 'This project is parameterized', and 'Throttle builds'. Each checkbox has a help icon to its right. Below this list is the 'Build Triggers' section, which includes checkboxes for 'Build after other projects are built', 'Build periodically', 'GitHub hook trigger for GITScm polling', 'Poll SCM', 'Disable this project', 'Quiet period', and 'Trigger builds remotely (e.g., from scripts)'. Each checkbox also has a help icon. The 'Advanced Project Options' section is empty, with an 'Advanced...' button on the right. The 'Pipeline' section at the bottom has a 'Definition' dropdown menu set to 'Pipeline script'. Below it is a 'Script' tab with a text area containing 'try sample Pipeline...' and a help icon. At the bottom left are 'Save' and 'Apply' buttons.

Pipeline name will contain the name given on the previous page

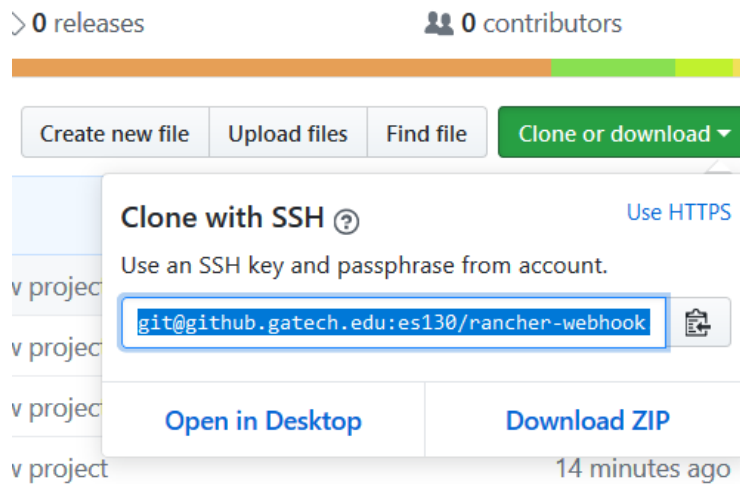
Enter a **Description** for the project

Add a GitHub Repository

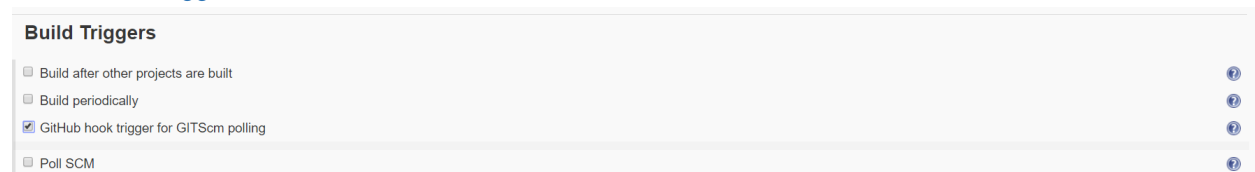
The screenshot shows the 'GitHub project' section of the configuration interface. It has a checkbox labeled 'GitHub project' which is checked. Below it is a 'Project url' text input field containing the value 'git@github.gatech.edu:es130/rancher-webhook.git'. A help icon is to the right of the input field. At the bottom right is an 'Advanced...' button.

Select the **GitHub project** checkbox

For **Project url** enter the **Clone with SSH** URL for the GitHub repository to use for this job. It is the URL displayed in modal dialog when you press the **Clone or download** button in the GitHub repository.

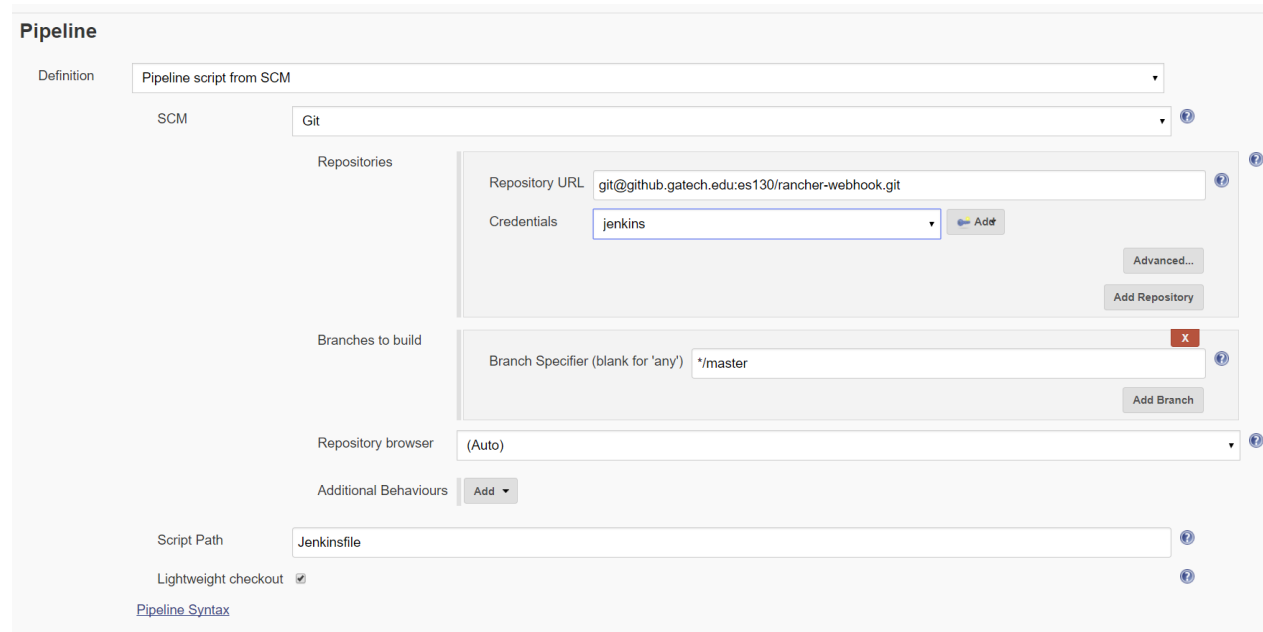


Add a Build Trigger



In the **Build Triggers** section select the **GitHub hook trigger for GITScm polling** checkbox. This option will allow GitHub to tell Jenkins to build the project each time an update is pushed to the GitHub repository.

Configure the Pipeline Build



For **Definition** select the **Pipeline script from SCM** option.

For **SCM** select **Git**

For **Repository URL** enter the same Git SSH URL used for **GitHub project** -> **Project url**.

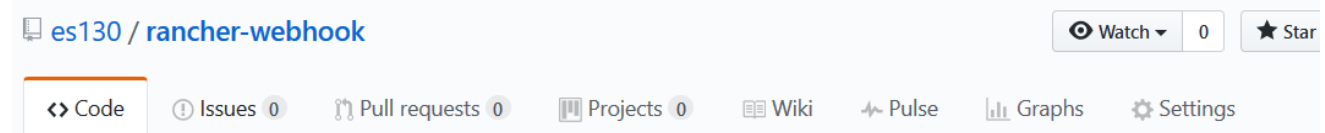
For **Credentials** select your Jenkins key. This is the key you added to your GitHub profile.

For **Script Path** select **Jenkinsfile**. This option tells Jenkins to use the Jenkinsfile, included in the root directory of the source code, to define the Pipeline process.

Press the **Save** button.

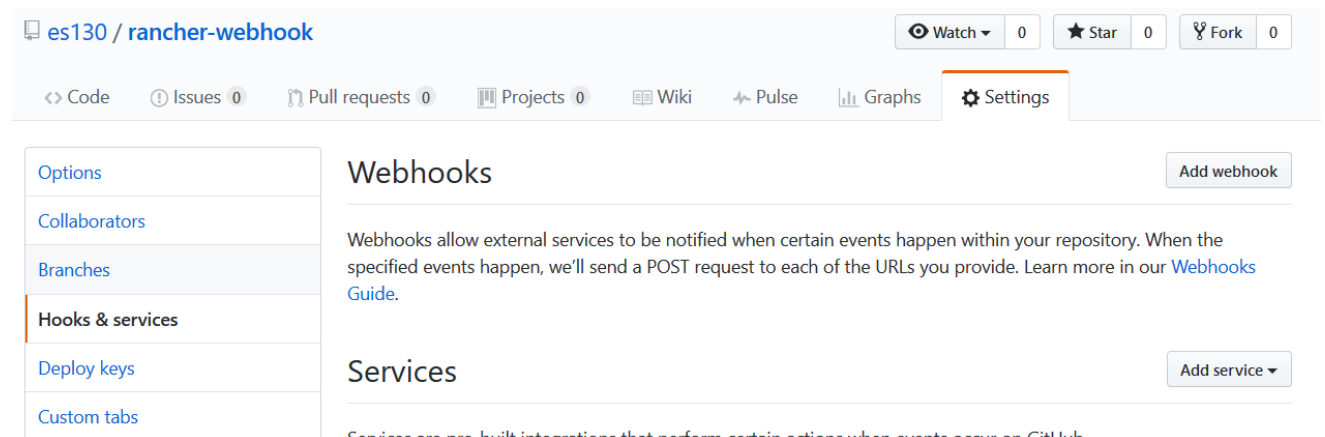
Add a Webhook to the GitHub Repository

Go to the Setting of the GitHub Repository



Select the **Settings** tab

Add a Webhook



Select the **Hooks & services** menu option

Press the **Add webhook** button

Configure Webhook

es130 / rancher-webhook

Watch 0 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Pulse Graphs Settings

Options

Collaborators

Branches

Hooks & services

Deploy keys

Custom tabs

Webhooks / Add webhook

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 *

https://example.com/postreceive

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Add webhook

For **Payload URL** enter <https://apps2.hdap.gatech.edu/jenkins/github-webhook/>

For **Content type** select **application/x-www-form-urlencoded**

For **Which events would you like to trigger this webhook?** Select **Just the push event**.

Make sure the **Active** checkbox is selected

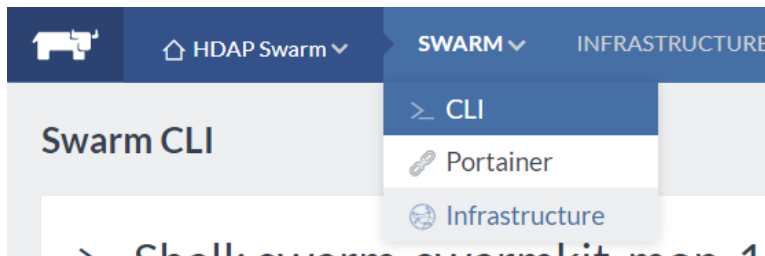
Press the **Add webhook** button

Now, any updates to the GitHub repo will notify the Jenkins server.

Create a Rancher Application Stack

Go to the Rancher Application <https://apps3.hdap.gatech.edu/>

Add a new stack



From the Navigation bar select **Swarm -> Infrastructure**

On the Infrastructure Stacks page press the **Add Stack** button.

Configure New Stack

A screenshot of the 'Add Stack' form in the HDAP Swarm interface. The form has a light gray background and a white content area. It contains two main sections: 'Name' and 'Description'. The 'Name' field has a placeholder 'e.g. myapp'. The 'Description' field has a placeholder 'e.g. MyApp Stack'. Below these fields is a section titled 'OPTIONAL: IMPORT COMPOSE' which contains two sub-sections: 'Optional: docker-compose.yml' and 'Optional: rancher-compose.yml'. Each sub-section has a text area with a placeholder 'Contents of docker-compose.yml' and a blue upload button with an upward arrow icon. At the bottom of the form is a section titled 'ADVANCED OPTIONS' with a downward arrow icon. At the very bottom are two buttons: 'Create' and 'Cancel'.

Enter a Name and Description

For **Name** enter a name for the project. It cannot have special characters or spaces. Try to make it identify the project. Probably just use the project name without punctuation or spaces.

For **Description** enter a description for the project. It is ok to use the same description you used in Jenkins.

Configure the Stack with Docker Compose

Students were asked to use Docker Compose version 2 to define how to create and link the containers for their applications. Use the **Optional: docker-compose.yml** field to upload the students docker-compose.yml file to configure the stack.

How to Upload and Modify Project docker-compose.yml

1. Use Git to clone the project to your computer
2. Press the blue upload button, , of the **Optional: docker-compose.yml** field to bring up a file upload dialog.
3. Navigate to the directory with the project and select the **docker-compose.yml** file
4. The **Optional: docker-compose.yml** text field will now be populated with the content of the docker-compose.yml file
5. Update the docker-compose.yml text in the textbox to use the Docker images for the project that were pushed to the HDAP Docker registry by Jenkins.
 - a. On your computer, open the Jenkinsfile for the project
 - b. Select the tag name used in the docker.build command of the Deploy stage

```
//Define the deploy stage
stage('Deploy'){
    steps{
        //The Jenkins Declarative Pipeline does not provide functionality to deploy to a private
        //Docker registry. In order to deploy to the HDAP Docker registry we must write a custom Groovy
        //script using the Jenkins Scripting Pipeline. This is done by placing Groovy code with in a "script"
        //element. The script below registers the HDAP Docker registry with the Docker instance used by
        //the Jenkins Pipeline, builds a Docker image of the project, and pushes it to the registry.
        script{
            docker.withRegistry('https://apps2.hdap.gatech.edu'){
                //Build and push the web application image
                def applicationImage = docker.build("rancherwebhook:1.0", "-f ./Dockerfile .")
                applicationImage.push('latest')
            }
        }
    }
}
```

- c. Update the docker-compose text to use the **image** directive with the URL and tag used in the Jenkinsfile

Original docker-compose text

```
version: '2'
services:
  rancher-webhooks-db:
    image: postgres
    restart: always
    expose:
      - "5432"
    volumes:
      - /usr/local/share/postgresql/rancher-webhooks:/var/lib/postgresql/data
  rancher-webhooks:
    build: .
    restart: always
    ports:
      - "12000:8080"
    depends_on:
      - rancher-webhooks-db
    command: ["/usr/local/bin/wait-for-postgres.sh", "rancher-webhooks-db", "catalina.sh", "run"]
```

New docker-compose text

```
version: '2'
services:
  rancher-webhooks-db:
    image: postgres
    restart: always
    expose:
      - "5432"
    volumes:
      - /usr/local/share/postgresql/rancher-webhooks:/var/lib/postgresql/data
  rancher-webhooks:
    image: apps2.hdap.gatech.edu/rancherwebhook:latest
    restart: always
    ports:
      - "12000:8080"
    depends_on:
      - rancher-webhooks-db
    command: ["/usr/local/bin/wait-for-postgres.sh", "rancher-webhooks-db", "catalina.sh", "run"]
```

Notice the difference. The **build** directive of the **rancher-webhooks** service is no longer being used. Now the **image** directive is used for the **rancher-webhooks** service to tell Rancher to pull the image from the HDAP Docker registry. Also **no change is needed** for the **rancher-webhooks-db** service because its image is being pulled from Dockerhub. If an image directive is used and no URL is used in front of the image name, then Docker goes to Dockerhub to retrieve the image.

6. Press the **Create** button
7. Containers will begin to activate

Stack: rancherwebhook ▼

Description: Application to accept notifications from a private docker

Activating	rancher-webhooks (In Progress) ⓘ
Activating	rancher-webhooks-db (In Progress) ⓘ

8. Containers will activate

Stack: rancherwebhook ▼

Description: Application to accept notifications from a private docker registry and call a webhook on a rancher server

Active	rancher-webhooks ⓘ
Active	rancher-webhooks-db ⓘ

9. Test navigation to the application by selecting the link for the host port number

Active	rancher-webhooks ⓘ	Image: apps2.hdap.gatech.edu/rancherwebhooklatest Ports: 82000
Active	rancher-webhooks-db ⓘ	Image: postgres