# Guide: How to Set Up a Jenkins Pipeline to Run After GitHub Commit

This document provides a step-by-step guide on how to configure Jenkins to run a pipeline automatically after a commit is pushed to a GitHub repository. Use the provided examples and add your own screenshots to enhance the guide.

## 1. How to Create a Personal Access Token in GitHub

Access the PAT Settings:

Go to GitHub and click on your profile picture in the upper-right corner.

Navigate to Settings > Developer settings > Personal access tokens > Fine-grained tokens (Preview).

Create a New Token:  
Click Generate new token.  
Provide a Token name (e.g., Jenkins Integration).

Set the Resource Owner:  
Ensure the correct resource owner is selected (your GitHub username).

Define Expiration:  
Choose an expiration period for the token, such as 90 days.

Add a Description (Optional):  
You may add a short description of what the token will be used for.

Configure Repository Access:  
Select Public repositories (read-only) or All repositories, depending on your requirement.

For more specific permissions, choose Only select repositories, then specify which repositories you want the token to access.

Set Permissions:  
Under Permissions, configure the desired scopes (e.g., read/write access for repository contents).

Generate the Token:  
Click Generate token.  
Copy the token immediately and store it securely, as it won’t be displayed again.

Use the Token:  
Use the generated token wherever required, such as in Jenkins or other integrations. Replace your GitHub password with this token for secure authentication.

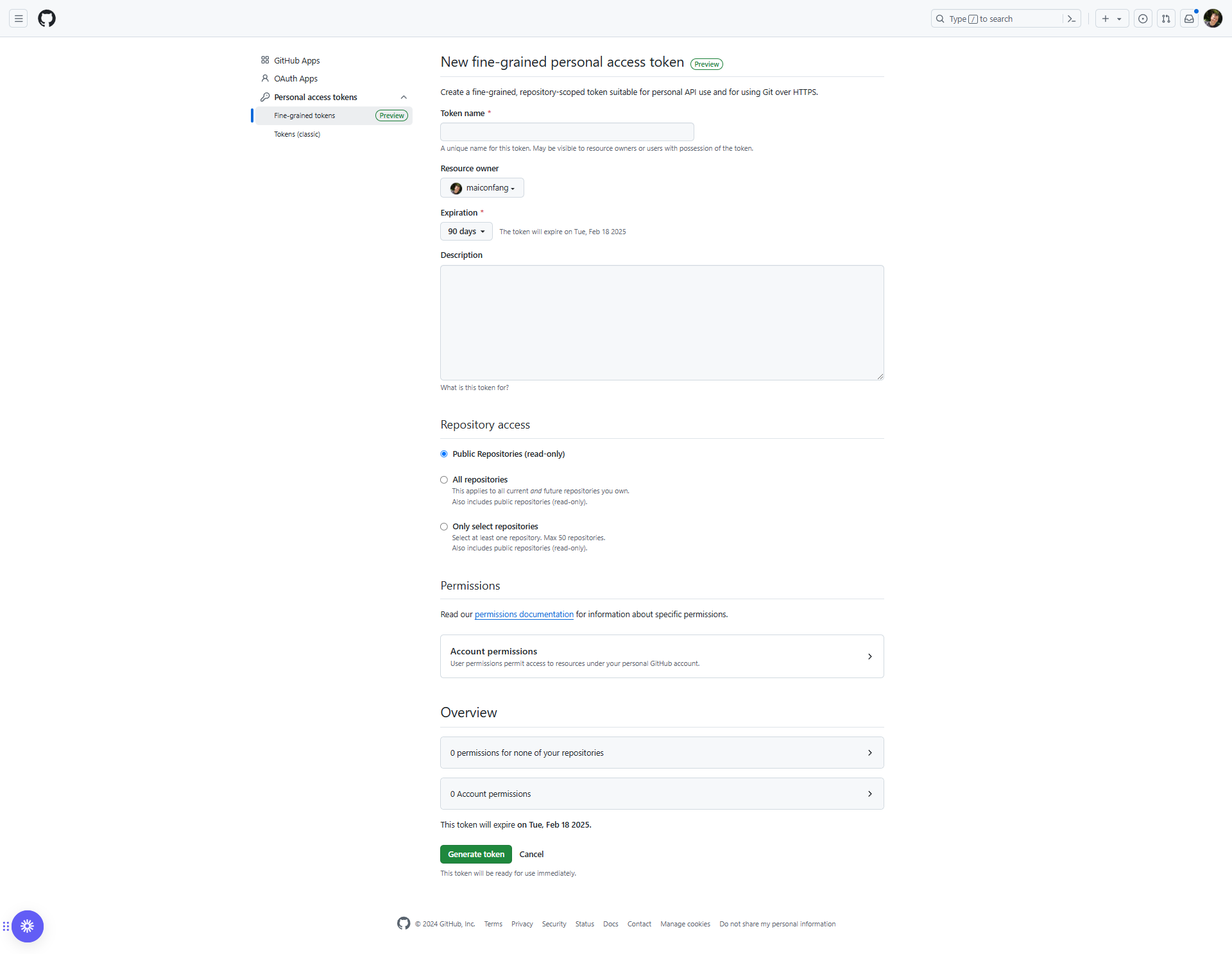


Image personal access token

## 2. Configure ngrok Authtoken

To authenticate your ngrok agent and establish secure tunnels for Jenkins, follow these steps:

1. Go to the ngrok dashboard (https://dashboard.ngrok.com/get-started/your-authtoken).  
2. Copy the authtoken displayed on the page.  
3. Open your terminal or command prompt.

4. Execute the following command, replacing `$YOUR\_AUTHTOKEN` with your copied authtoken:

```bash

ngrok config add-authtoken $YOUR\_AUTHTOKEN

```

5. After running the command, ngrok will be configured to use the authtoken automatically for secure tunnels.

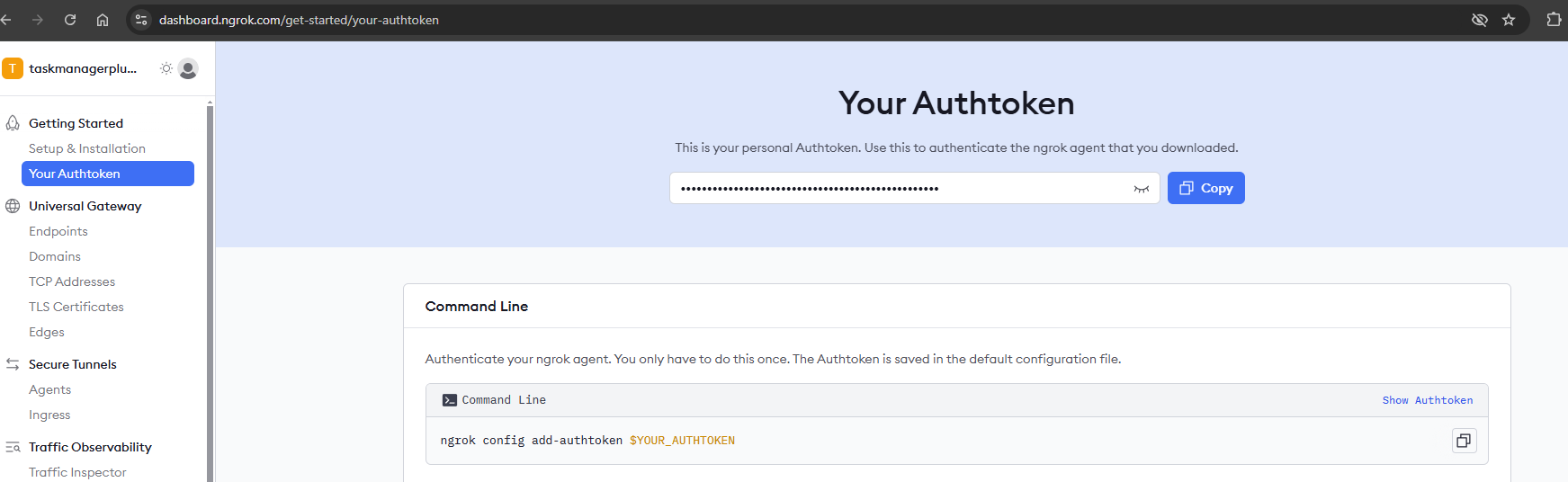
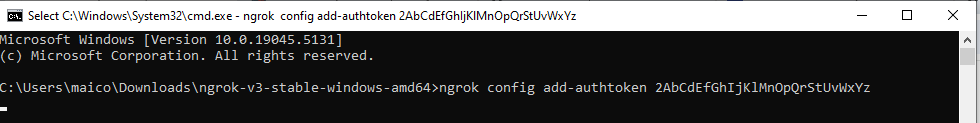


Image - Configure ngrok Authtoken  
  
  
Image – Cmd

## 3. Configure the Webhook on GitHub

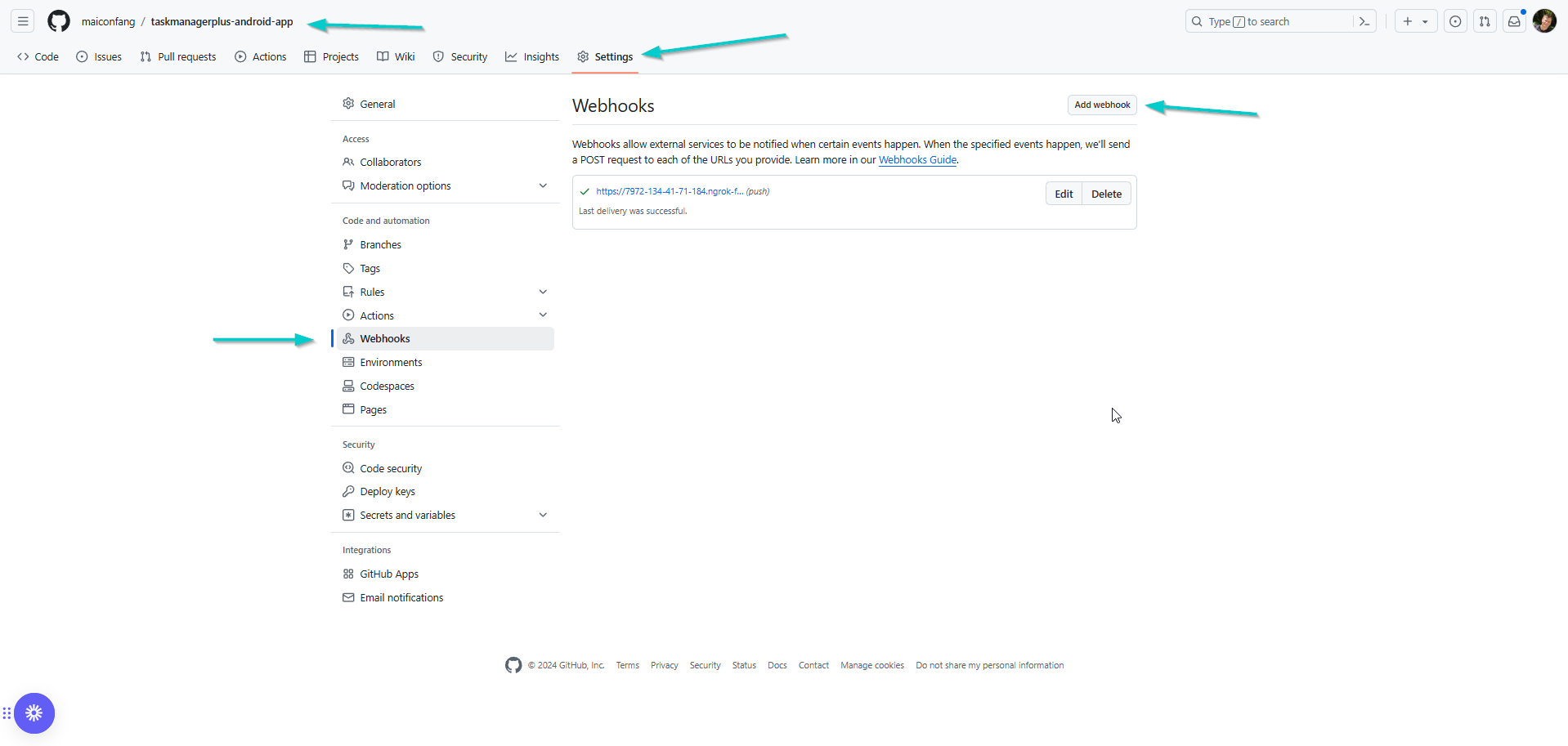
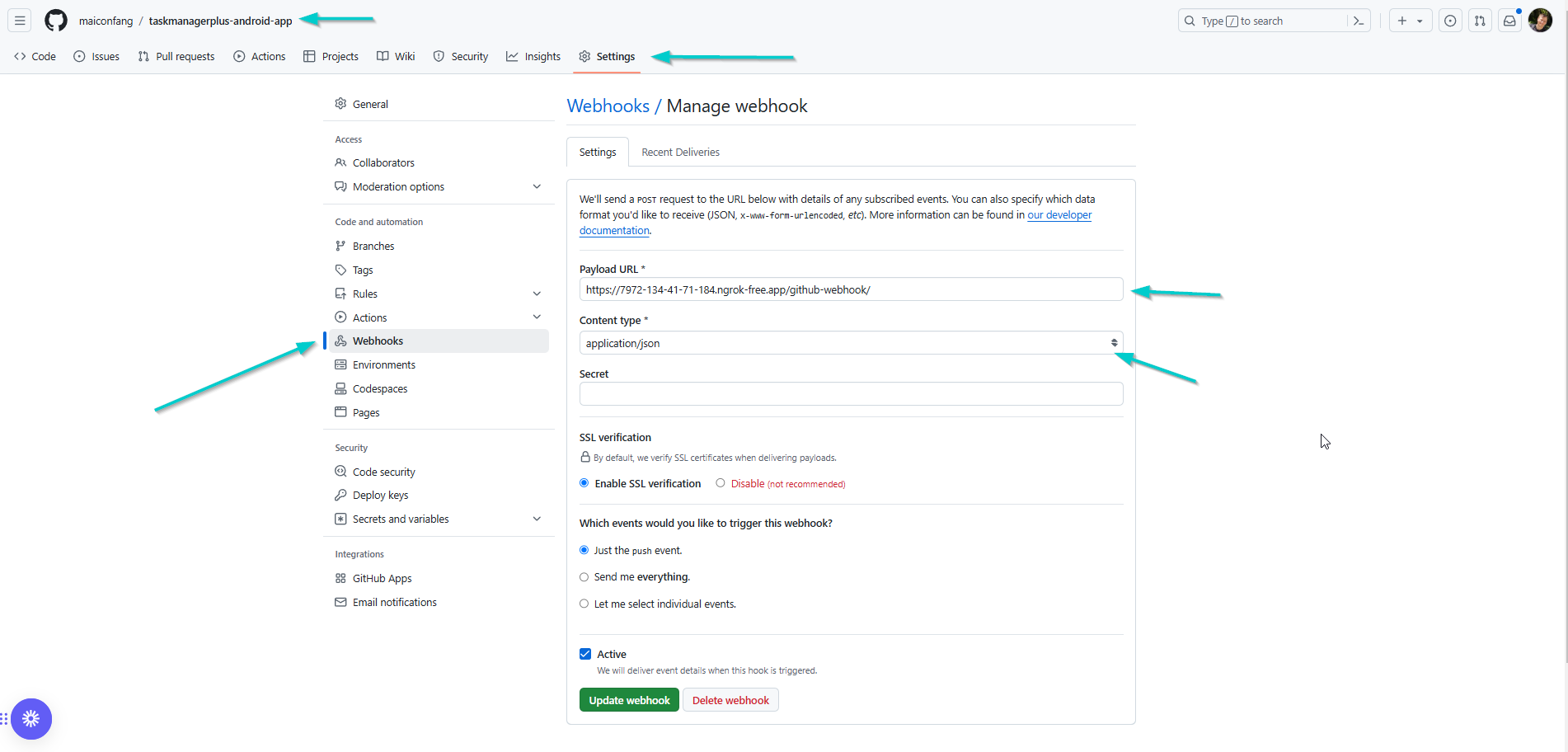
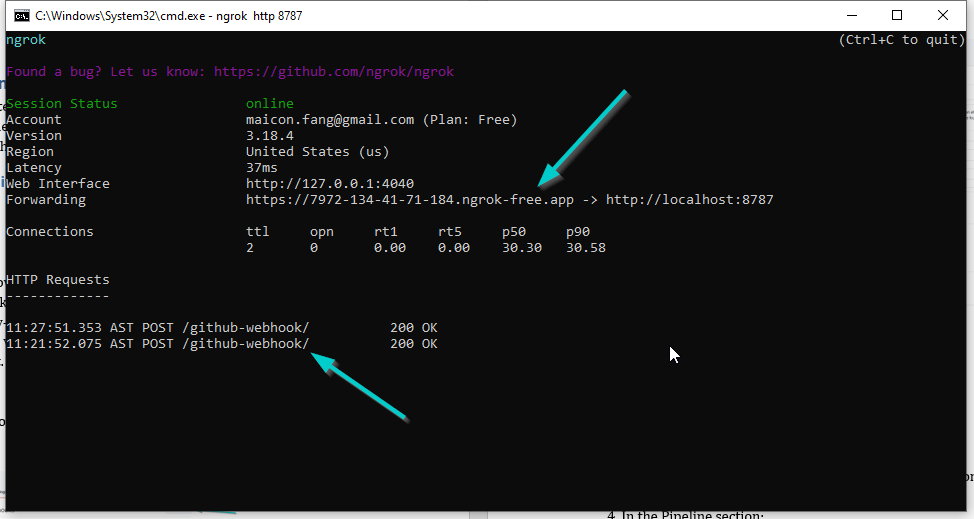
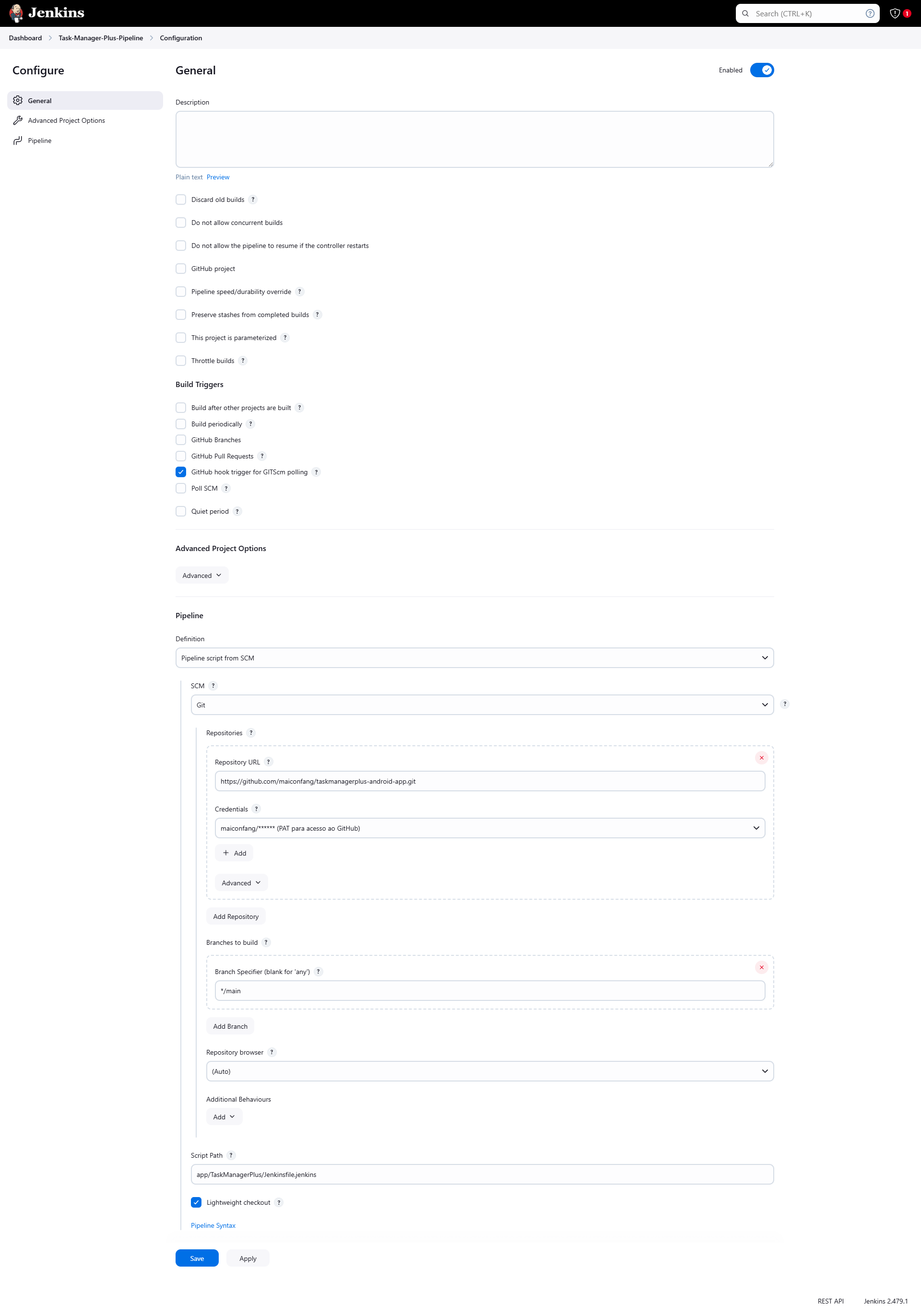
1. Go to your GitHub repository.  
2. Navigate to Settings > Webhooks.  
3. Click Add webhook.  
4. Fill in the fields:  
 - Payload URL: The ngrok URL followed by /github-webhook/ (e.g., https://<your-ngrok-url>.ngrok-free.app/github-webhook/).  
 - Content type: application/x-www-form-urlencoded.  
 - SSL verification: Keep Enable SSL verification checked.  
 - Events: Select Just the push event.  
5. Click Add webhook.  
  
Tip: After setting up, test the webhook by clicking Redeliver on a test event and check the status under Recent Deliveries.   
  


Image - Add Webhooks  
  
  
  
Image – Edit Webhooks  
  
  
Image – ngrok cmd  
Cmd command: ngrok http 8787 // This command sets up a tunnel to my local Jenkins instance running on port 8787.

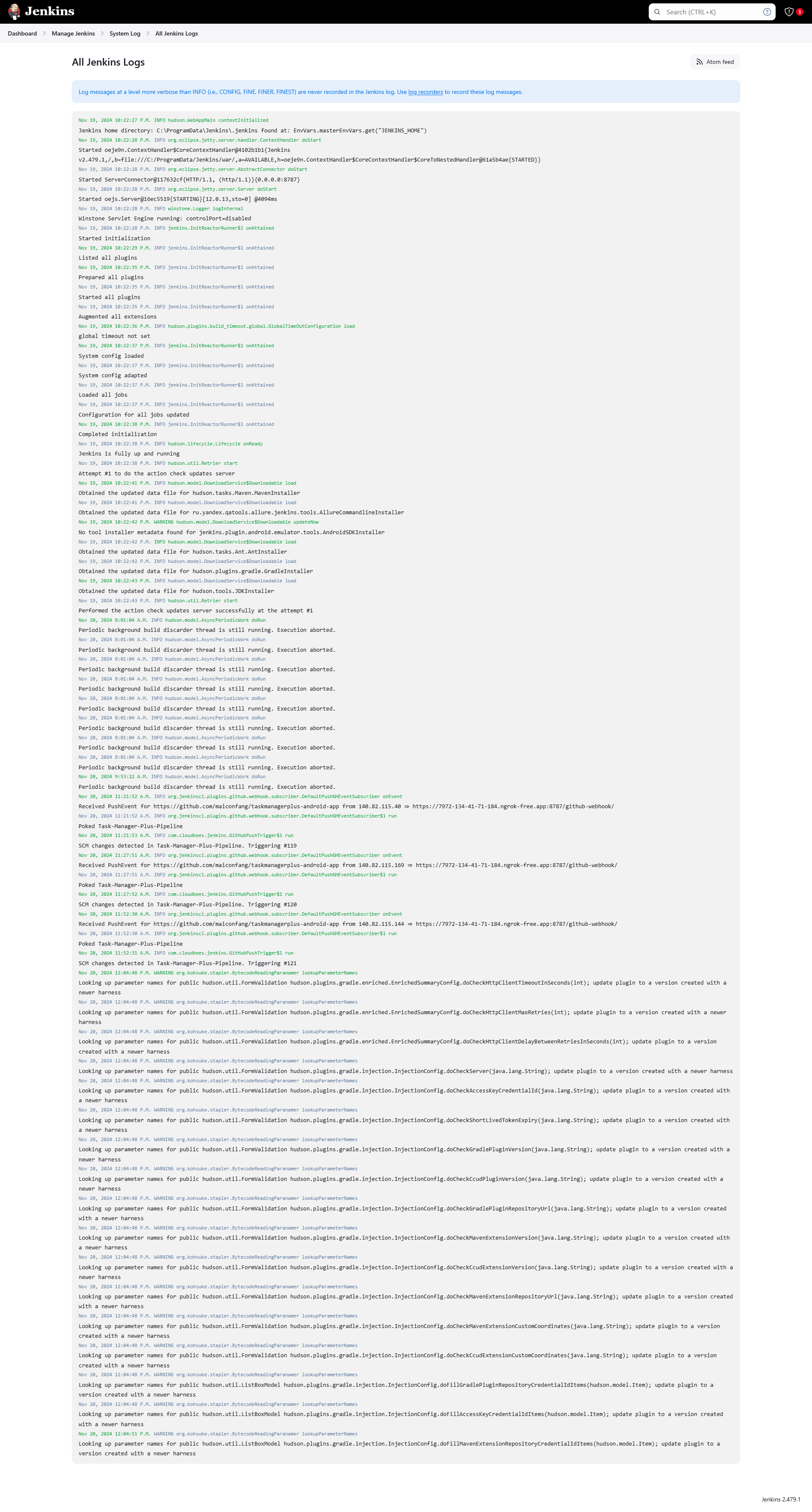
## 4. Configure the Job in Jenkins

1. Access Jenkins (http://localhost:8787).  
2. Create or edit the job you want to configure.  
3. In Configure > Build Triggers:  
 - Check the GitHub hook trigger for GITScm polling option.  
 - Uncheck Poll SCM, if enabled.  
4. In the Pipeline section:  
 - Select Pipeline script from SCM.  
 - SCM: Choose Git.  
 - Repository URL: Enter your GitHub repository URL (e.g., https://github.com/your-repo.git).  
 - Credentials: Select the credentials you configured (e.g., github-pat).  
 - Branch Specifier: Ensure you are monitoring the correct branch (e.g., \*/main).  
5. Save the configuration by clicking Save.  
Image - Task-Manager-Plus-Pipeline – configure  
Obs: In the image, I’m not selecting the Poll SCM option, which means the process is running every minute. If you prefer, you can configure it to run, for example, every day after 6:00 PM, using the cron expression: "0 18 \* \* \*"

## 5. Ensure the Jenkinsfile is in the Repository

1. Confirm that the Jenkinsfile is in the correct branch of your repository (e.g., main branch).  
2. Ensure the Jenkinsfile contains all the necessary pipeline stages.

## 6. Test the Configuration

1. Make a small change in your local repository and push it to GitHub:  
 ```bash  
 git add .  
 git commit -m "Testing webhook automation"  
 git push origin main  
 ```  
2. Check Jenkins to see if the pipeline was triggered automatically.  
3. If it doesn’t work, go to Manage Jenkins > System Log to review logs related to the webhook.  
  
  
Image – Jenkins Log

## 7. Add GitHub PAT Credential to Jenkinsfile

To ensure your Jenkins pipeline can access the GitHub repository, follow these steps to securely configure and integrate your GitHub Personal Access Token (PAT) with your pipeline:

1. **Navigate to Credentials Management**
   * Open Jenkins and go to **Manage Jenkins > Credentials > System > Global credentials (unrestricted)**.
2. **Add a New Credential**
   * Click on **Add Credentials** and fill in the following:
     + **Kind:** Secret text
     + **Scope:** Global (Jenkins, nodes, items, all child items, etc.)
     + **Secret:** Paste your GitHub Personal Access Token (PAT).
     + **ID:** Set it to github-pat-global.
     + **Description:** Add a descriptive name for easier identification, e.g., "GitHub PAT for Task Manager Plus".
   * Click **Create** to save the credential.
3. **Update Your Jenkinsfile**
   * Add the credential ID (github-pat-global) to the checkout stage in your Jenkinsfile. Ensure it looks like this:

Copy code

stage('Checkout') {

steps {

echo 'Starting Checkout stage'

git branch: 'main',

url: 'https://github.com/maiconfang/taskmanagerplus-android-app.git',

credentialsId: 'github-pat-global'

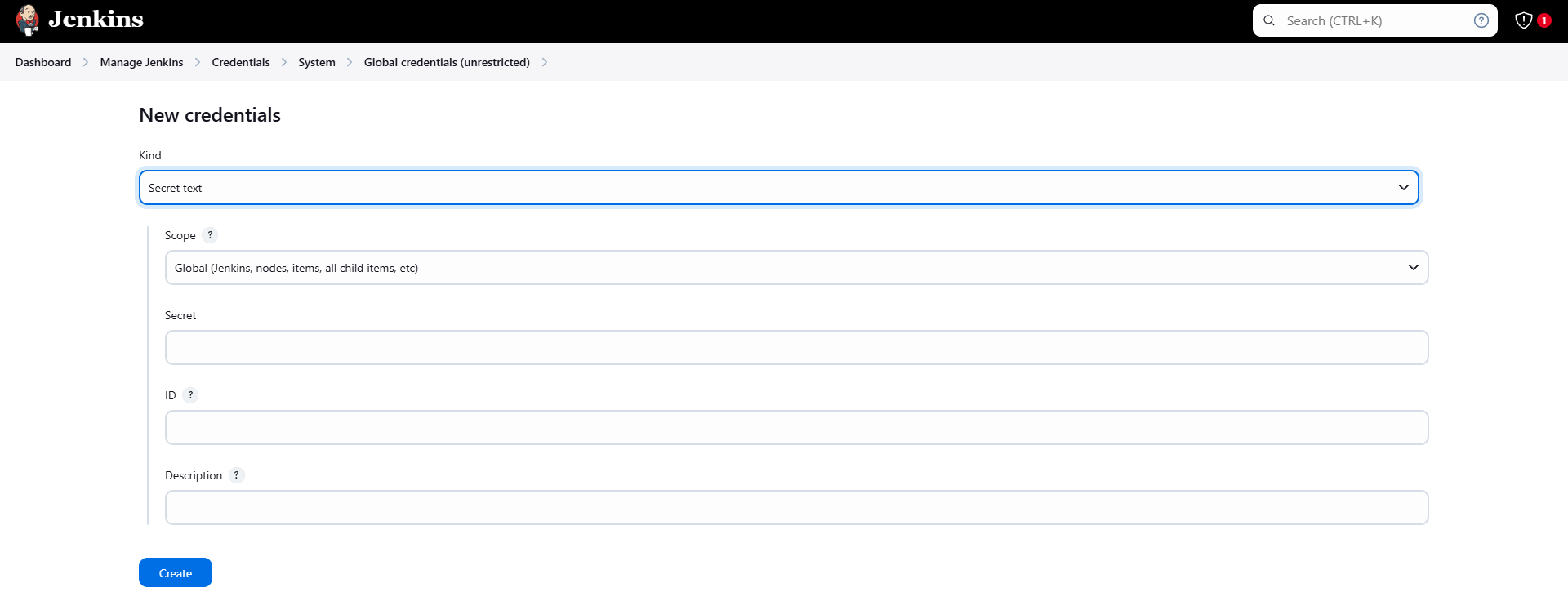
echo 'Checkout completed successfully'

}

}

1. **Save the Jenkinsfile**
   * Save the updated Jenkinsfile to your GitHub repository. This ensures the pipeline fetches it during execution.
2. **Verify Credential ID Consistency**
   * Double-check that the credentialsId in your Jenkinsfile (github-pat-global) matches the one you configured in Jenkins. Any mismatch will cause the pipeline to fail.
3. **Test the Pipeline**
   * Trigger the Jenkins pipeline and monitor the logs to confirm that the Checkout stage completes successfully, indicating that the repository was accessed using the configured credential.

This ensures a seamless connection between Jenkins and your GitHub repository while securely managing access credentials.

  
Image - Add GitHub PAT Credential to Jenkinsfile

Congratulations! After following these steps, your Jenkins should be configured to automatically run pipelines after commits are pushed to GitHub. Add complementary screenshots to create a more complete visual guide.

## 8. Troubleshooting

1. If the pipeline is not triggered, check:  
 - Whether the webhook is properly configured in GitHub.  
 - Whether Jenkins is accessible via the ngrok URL.  
 - Whether the Jenkinsfile is in the correct branch and functional.  
2. On GitHub, verify the event status under Settings > Webhooks > Recent Deliveries.  
3. In Jenkins, review the job configuration and logs in the System Log.