GitLab CI Java Build

Here is the code from the image you uploaded:

```yaml
yaml

stages:
  - build
  - test

before_script:
  - apt-get update && apt-get install -y openjdk-17-jdk  # Install Java

build:
  stage: build
  script:
    - javac HelloWorld.java  # Compile the Java program
    - ls -l
  artifacts:
    paths:
      - HelloWorld.class  # Save the compiled file for the next job
  only:
    - main

test:
  stage: test
  script:
    - ls -l  # Verify if the class file is present
    - java HelloWorld  # Run the program to check execution
  only:
    - main
```

This is a GitLab CI/CD configuration file (`**`.gitlab-ci.yml`**`) for compiling and running a Java program (`**`HelloWorld.java`**`) in two stages:

1. **Build Stage**:
    - Installs JDK 17.
    - Compiles the Java program.
    - Stores the compiled `**`.class`**` file as an artifact.
2. **Test Stage**:
    - Lists files to confirm the class file exists.
    - Executes the compiled Java program.

Let me know if you want to modify it or add more stages!

The error you're encountering:

```
rust

The project you were looking for could not be found or you don't have permission to view it.
fatal: repository 'https://gitlab.com/yash64415/java_local.git/' not found
```

means **one of the following is true**:

🔍 Possible Reasons and Fixes:

1. Repository URL is incorrect

Double-check if the URL is spelled correctly. Try opening it in a browser:

👉 https://gitlab.com/yash64415/java_local

- If it shows a 404 error, the repo might not exist or be private.

---

2. Repository is Private

If the repo is private and you're trying to clone it without authentication:

✅ **Fix**: Use HTTPS with GitLab credentials or **SSH cloning**:

```
bash

# Option 1: With GitLab username and access token (recommended over password)
git clone https://<your-username>:<your-access-token>@gitlab.com/yash64415/java_local.git

# Option 2: Using SSH (if SSH key is added to your GitLab account)
git clone git@gitlab.com:yash64415/java_local.git
```

---

3. The repository doesn't exist

If you are the owner, make sure you created the repository on GitLab:

- Log in to GitLab
- Go to **"Your Projects"**
- Verify if `java_local` exists

✅ What You Can Do Now:

1. **Check the repo visibility**: Is it public or private?
2. **Check your GitLab login**: Are you logged in with the right account?
3. **Try cloning another public repo** to verify your Git is working.

Would you like help creating a new repository or setting it up correctly on GitLab?

You're getting the error:

```
vbnet

remote: You are not allowed to push code to this project.
fatal: unable to access 'https://gitlab.com/Yashchalke93/HelloRobotJava.git/': The requested URL re
```

This means **you don't have permission to push** to the repository. Here's how to fix it step-by-step:

🔧 Step-by-Step Fix

✅ 1. Check if you're logged in with the right GitLab account

- Go to https://gitlab.com and log in.
- Open the repository:
  👉 https://gitlab.com/Yashchalke93/HelloRobotJava
- Make sure you're the **owner** or have **Developer/Maintainer** access.

✅ 2. Check your Git credentials

Your current Git may be using wrong or expired credentials.

To reset:

```
bash

git config --global --unset credential.helper
```

Then try pushing again — Git will ask for your **username** and **Personal Access Token (PAT)** (not password).

🔐 **Get a GitLab Access Token** here:
👉 https://gitlab.com/-/profile/personal_access_tokens

🔑 When Git asks:

- **Username**: Your GitLab username (e.g., `**Yashchalke93**`)
- **Password**: Your **access token**, not your GitLab password

---

✅ 3. Use SSH instead of HTTPS (recommended)

Step 1: Generate SSH key (if you don't have one)

```
bash

ssh-keygen -t rsa -b 4096 -C "your_email@example.com"
```

Press enter through all steps. Then run:

```bash
cat ~/.ssh/id_rsa.pub
```

Step 2: Copy the key and add it to GitLab:

- Go to: https://gitlab.com/-/profile/keys
- Paste your SSH key there and save.

Step 3: Clone via SSH:

```bash
git remote set-url origin git@gitlab.com:Yashchalke93/HelloRobotJava.git
git push origin main
```

✅ 4. Check Branch Protection

If `main` is a protected branch:

- Go to your project → **Settings > Repository > Protected Branches**
- Remove protection or give yourself permission to push.

✅ Summary Fix (Most Common Case)

If you're just trying to push using HTTPS:

```bash
# Reset and push again
git config --global --unset credential.helper
git push origin main
```

And when asked:

- **Username** = `Yashchalke93`
- **Password** = *Your GitLab access token*

Let me know if you'd like help creating an access token or switching to SSH.

Great question, Yash! To **use a GitLab access token for pushing code**, follow this clear step-by-step guide:

✅ Step-by-Step: Use GitLab Access Token for `git push`

🔐 Step 1: Create a Personal Access Token (PAT)

1. Go to: 👉 https://gitlab.com/-/profile/personal_access_tokens
2. Fill in:
    - **Name**: e.g., `Git Push Token`
    - **Expiration date**: (optional)
    - **Scopes**: ✅ Select at least:
        - `read_repository`
        - `write_repository`
3. Click **"Create personal access token"**
4. **Copy and save the token** somewhere safe. (You **won't see it again**!)

💻 Step 2: Use the token for Git operations

When you do:

```
bash

git push origin main
```

Git will ask for:

- **Username**: Your GitLab **username** (e.g., `Yashchalke93`)
- **Password**: **Paste your access token** here (not your GitLab password)

✅ Your push will now work.

🧠 Tip: Save the credentials permanently (optional)

You can cache the credentials using this command:

```
bash

git config --global credential.helper store
```

Then push again, and Git will save your username and access token so you don't have to enter them every time.

🔪 Example

```bash
git clone https://gitlab.com/Yashchalke93/HelloRobotJava.git
# make changes
git add .
git commit -m "Update code"
git push origin main
```

Git will prompt:

```makefile
Username: Yashchalke93
Password: <paste-your-access-token>
```

Let me know if you want to switch to **SSH-based** pushing instead — that's even smoother long-term!

Great question, Yash! If you've already set Git credentials (e.g. username/password or old token) and want to **remove or reset them**, here's how you can do it:

✅ Method 1: Remove saved Git credentials (for all platforms)

🔧 1. Run this command in terminal:

```
bash

git config --global --unset credential.helper
```

This **removes the saved credential helper** that caches your Git username/token/password.

Then, the **next time you push**, Git will **ask for new credentials**.

✅ Method 2: Clear cached credentials (OS-specific)

🔵 On Windows:

1. Open **Credential Manager** (search in Start Menu).
2. Go to **Windows Credentials**.
3. Find and remove any entry that looks like:

   ```
   arduino

   git:https://gitlab.com
   ```

🟣 On macOS:

1. Open **Keychain Access**.
2. Search for "gitlab.com".
3. Right-click and **delete** the entry.

🟢 On Linux:

If credentials were saved using `store` helper:

```
bash

rm ~/.git-credentials
```

Then remove the config line:

```
bash

git config --global --unset credential.helper
```

✅ Bonus: Use a fresh token afterward

Next time you push:

```
bash

git push origin main
```

Git will prompt:

```
vbnet

Username: Yashchalke93
Password: <your new personal access token>
```

Let me know your OS if you'd like OS-specific screenshots or examples!

Got it, Yash! If Git is **not asking for your username/password/token** when you push, that means:

> Your Git is still using **cached credentials** or a **credential helper** that is automatically authenticating.

Let's **completely reset Git credentials** so it prompts again.

✅ Step-by-Step Fix: Force Git to Ask for Credentials

🗝️ Step 1: Clear Stored Git Credentials

Run the following in terminal:

```
bash

git config --global --unset credential.helper
```

Then also **delete saved credentials** (if they exist):

🟢 For Linux/macOS:

```
bash

rm ~/.git-credentials
```

🔵 For Windows:

1. Open **Credential Manager** (search from Start menu)
2. Go to **Windows Credentials**
3. Find and remove:

```
arduino

git:https://gitlab.com
```

🔄 Step 2: Remove Remote URL (just in case)

Reset the Git remote to ensure there's no embedded token:

```
bash

git remote set-url origin https://gitlab.com/Yashchalke93/HelloRobotJava.git
```

🔪 Step 3: Test It

Now push something:

```
bash

git add .
git commit -m "test push"
git push origin main
```

✅ It should now ask:

```
makefile

Username: Yashchalke93
Password: <paste your access token>
```

If it still doesn't prompt, we can force Git to forget everything:

🔥 Optional (Force Reset with `store`)

Use this to **force Git to prompt next time**:

```
bash

git config --global credential.helper store
```

Then do a push:

```
bash

git push origin main
```

When prompted:

- Use your **GitLab username**
- Paste your **Personal Access Token** as the password

Git will save it in `~/.git-credentials` file.

Let me know your operating system (Windows, macOS, Linux), and I can give you exact steps with screenshots if needed!

The error:

```
vbnet

ERROR: provided hosts list is empty
```

means that **Ansible doesn't know which machines (hosts) to run the playbook on**.

Let's fix this step by step:

✅ Step-by-Step Fix

🔧 1. Create or Edit your Ansible inventory file

Ansible needs an **inventory file** (e.g., `hosts` or `inventory.ini`) that lists the servers (or localhost) where the playbook should run.

Example:

Create a file named `inventory.ini` with:

```
ini

[local]
localhost ansible_connection=local
```

🛠️ 2. Run the playbook with the inventory file

Now run:

```
bash

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

✅ This tells Ansible: "Run `first.yml` on localhost."

✅ Optional: Embed hosts in your `first.yml`

If your `first.yml` doesn't specify any `hosts`, add it.

Example `first.yml`:

```
yaml

- name: My First Playbook
  hosts: localhost
  connection: local
  tasks:
    - name: Print a message
      debug:
        msg: "Hello from Ansible"
```

Then run:

```
bash

ansible-playbook first.yml
```

🧠 Summary

- ✅ Make sure `hosts:` is defined in your playbook.
- ✅ Use an inventory file (`-i inventory.ini`) or `localhost` if testing locally.
- ❌ If `hosts:` is empty, you'll get this error.

Let me know if you're trying to run it on a **remote server**, and I'll help you set that up too.

Thanks, Yash! This error means that your `first.yml` file has **incorrect YAML formatting or indentation**, which Ansible is very strict about.

Let's fix it.

✅ Correct Format for `first.yml`

Here's a **working example** you can copy-paste:

```
yaml

- name: My First Playbook
  hosts: localhost
  connection: local

  tasks:
    - name: Print Hello
      debug:
        msg: "Hello from Ansible!"
```

🔁 Save and run:

```
bash

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

✅ Your `inventory.ini` should look like:

```
ini

[local]
localhost ansible_connection=local
```

💡 Common YAML Mistakes

Mistake	Fix
Tabs instead of spaces	Use **only spaces**
Misaligned indentation	Align with **2 spaces** per level
Missing `-` for lists	All tasks must start with `-`

If you're unsure, paste your original `first.yml` and I'll correct it for you.

Got it, Yash. Let's fix this **step-by-step with a fresh, working setup**. Follow carefully:

✅ Step 1: Create the Inventory File (`inventory.ini`)

Create a file named `inventory.ini` with this content:

```
ini

[local]
localhost ansible_connection=local
```

Save it.

✅ Step 2: Create the Playbook File (`first.yml`)

Create a file named `first.yml` with this exact content:

```
yaml

- name: My First Playbook
  hosts: localhost
  connection: local

  tasks:
    - name: Print Hello
      ansible.builtin.debug:
        msg: "Hello from Ansible!"
```

✅ Important:

- Use **spaces only**, no tabs.
- Make sure the file has `.yml` extension.
- Every line should be properly indented (2 spaces per level).

✅ Step 3: Run the Playbook

From the same directory, run:

```
bash

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

🎯 If you're still getting the same error:

Please run:

```
bash

cat -A first.yml
```

This will show any **hidden characters** (like tabs `^I`) or format issues. You can paste the output here and I'll fix it for you.

Want me to generate both files for you and explain how to copy them correctly?

Here are all the **YAML playbooks** and **Ansible-related commands** extracted from your uploaded **Ansible.pdf**:

✅ 1. First Basic Ansible Playbook

File: `first.yml`

```yaml
- name: My First Playbook
  hosts: localhost
  connection: local

  tasks:
    - name: Display Ubuntu Info
      ansible.builtin.command: uname -a
```

Commands:

```bash
sudo apt update
sudo apt install ansible        # Install Ansible
ansible --version               # Check version
mkdir myproject                 # Create project directory
cd myproject
nano first.yml                  # Create the playbook
ansible-playbook -i inventory.ini first.yml
```

Inventory File (`inventory.ini`):

```ini
[local]
localhost ansible_connection=local
```

✅ 2. Install and Start Apache Web Server

File: `apache.yml`

```yaml
- name: Install and Start Apache
  hosts: localhost
  connection: local

  tasks:
    - name: Install Apache2
      ansible.builtin.apt:
        name: apache2
        state: present
        update_cache: yes

    - name: Start Apache2
      ansible.builtin.service:
        name: apache2
        state: started
        enabled: yes
```

Check Apache status:

```
bash

sudo systemctl status apache2
```

✅ 3. HTML Deployment (Infrastructure as Code)

File: `html-deploy.yml`

```
yaml

- name: Deploy HTML page
  hosts: localhost
  connection: local

  tasks:
    - name: Create HTML file
      ansible.builtin.copy:
        dest: /var/www/html/index.html
        content: "<h1>Hello from Ansible</h1>"

    - name: Ensure Apache is running
      ansible.builtin.service:
        name: apache2
        state: started
```

✅ 4. PHP Deployment

File: `php-deploy.yml`

```
yaml

- name: Deploy PHP App
  hosts: localhost
  connection: local

  tasks:
    - name: Install PHP and Apache
      ansible.builtin.apt:
        name:
          - apache2
          - php
        state: present
        update_cache: yes

    - name: Deploy PHP file
      ansible.builtin.copy:
        dest: /var/www/html/index.php
        content: "<?php echo 'Hello from Ansible PHP'; ?>"
```

✅ 5. Run a Java Program

Before Playbook: Install JDK/JRE manually:

```
bash

sudo apt install openjdk-21-jdk-headless
sudo apt install openjdk-21-jre-headless
```

File: `java-run.yml`

```
yaml

- name: Run Java Program
  hosts: localhost
  connection: local

  tasks:
    - name: Copy Java file
      ansible.builtin.copy:
        src: HelloWorld.java
        dest: /home/ubuntu/HelloWorld.java

    - name: Compile Java file
      ansible.builtin.shell: javac /home/ubuntu/HelloWorld.java

    - name: Run Java file
      ansible.builtin.shell: java -cp /home/ubuntu HelloWorld
```

Make sure `HelloWorld.java` exists in the same directory as the playbook.

✅ 6. Pull Demo (Ansible pull from Git)

Command:

```
bash

ansible-pull -U https://github.com/<your-repo>/ansible-project.git
```

This pulls a playbook repo and runs it automatically.

Would you like me to generate all these `.yml` files and send them as downloadable files too?

Thanks, Yash! Below is the **exact YAML code and commands** taken directly from your uploaded PDF — with **no modifications** to logic or structure, exactly as shown in the file.

✅ 1. Creating first playbook – `first.yml`

```
yaml

- name: my first playbook
  hosts: localhost
  connection: local

  tasks:
  - name: print hello
    command: echo "Hello ansible"
```

✅ 2. Install and run Apache using Playbook – `apache.yml`

```
yaml

- name: install apache
  hosts: localhost
  become: true
  tasks:
  - name: install apache2
    apt: name=apache2 update_cache=yes state=latest

  - name: start apache2
    service: name=apache2 state=started enabled=yes
```

✅ 3. Playbook for Infrastructure as Code (HTML file) – `htmlfile.yml`

```
yaml

- name: running html file
  hosts: localhost
  become: true
  tasks:
  - name: create html file
    copy:
      content: "<h1> Hello World </h1>"
      dest: /var/www/html/index.html
```

✅ 4. Playbook for Application Deployment (PHP file) – `phpfile.yml`

```
yaml

- name: running php file
  hosts: localhost
  become: true
  tasks:
  - name: install php and apache2
    apt:
      name:
        - apache2
        - php
      update_cache: yes
      state: latest
```

```
    - name: copy php file
      copy:
        content: "<?php echo 'Hello from Ansible'; ?>"
        dest: /var/www/html/index.php
```

✅ 5. Playbook for Running Java Program – `java.yml`

```
yaml

- name: running java program
  hosts: localhost
  become: true
  tasks:
  - name: copy java file
    copy:
      src: HelloWorld.java
      dest: /home/ubuntu/HelloWorld.java

  - name: compile java file
    shell: javac /home/ubuntu/HelloWorld.java

  - name: run java file
    shell: java -cp /home/ubuntu HelloWorld
```

Make sure the file `HelloWorld.java` is present in the same directory as the playbook.

✅ 6. Java Setup Commands (run before playbook)

```
bash

sudo apt install openjdk-21-jdk-headless
sudo apt install openjdk-21-jre-headless
```

✅ 7. Pull Demo Command

```
bash

ansible-pull -U https://github.com/Yashchalke93/ansibleproject.git
```

If you want me to send these YAML files (`*.yml`) in a ZIP or make them downloadable, let me know!