Git clone versus git fetch-
Clone will make local copy in our machine.
Fetch - will bring the repo till github not to local.

Https-
We need user name and password. Not good practice.

Ssh-
More secure.
Public key and private key concept.
Private will be in local system.
Public will be in github.

To check if git installed or not-
Git - - version

```
karan@LAPTOP-H56OVJTV MINGW64 ~
$ git -v
git version 2.50.1.windows.1
```

```
karan@LAPTOP-H56OVJTV MINGW64 ~
$ git --version
git version 2.50.1.windows.1
```

Type git and you come to know all the commands available.

```
karan@LAPTOP-H56OVJTV MINGW64 ~
$ git
usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
           [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
           [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
           [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
           [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
           <command> [<args>]

These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone      Clone a repository into a new directory
   init       Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add        Add file contents to the index
   mv         Move or rename a file, a directory, or a symlink
   restore    Restore working tree files
   rm         Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect     Use binary search to find the commit that introduced a bug
   diff       Show changes between commits, commit and working tree, etc
   grep       Print lines matching a pattern
   log        Show commit logs
   show       Show various types of objects
   status     Show the working tree status

grow, mark and tweak your common history
   backfill   Download missing objects in a partial clone
   branch     List, create, or delete branches
   commit     Record changes to the repository
   merge      Join two or more development histories together
   rebase     Reapply commits on top of another base tip
   reset      Reset current HEAD to the specified state
   switch     Switch branches
   tag        Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch      Download objects and refs from another repository
   pull       Fetch from and integrate with another repository or a local branch
   push       Update remote refs along with associated objects

'git help -a' and 'git help -g' list available subcommands and some
concept guides. See 'git help <command>' or 'git help <concept>'
to read about a specific subcommand or concept.
See 'git help git' for an overview of the system.

karan@LAPTOP-H56OVJTV MINGW64 ~
$ ~
```

Git log-
To check commit history.

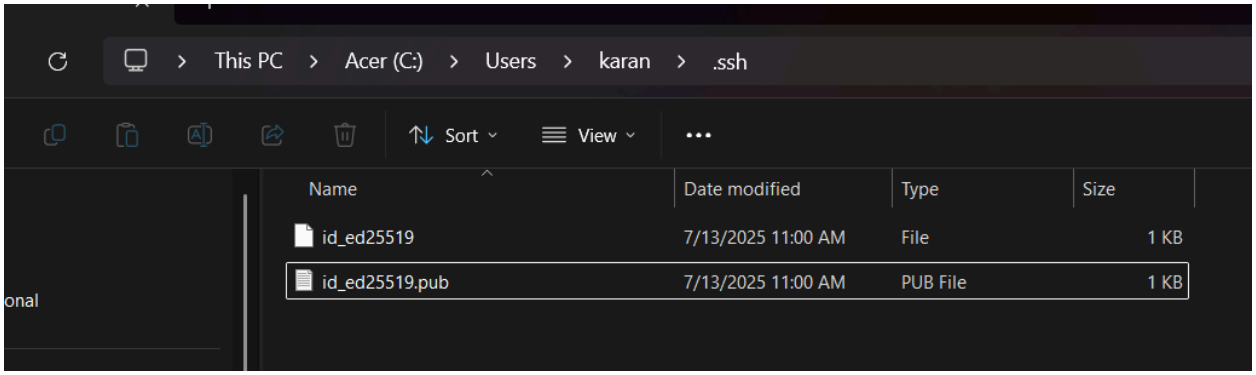
```
karan@LAPTOP-H56OVJTV MINGW64 ~/Desktop/cypressByMukesh (master)
$ git log
commit 2ffed03dfb6639f4334eefe59db6dca9ab239f05 (HEAD -> master, origin/master)
Author: karanAtreya1986 <karan1988@gmail.com>
Date:   Sat Jul 12 19:33:04 2025 +0530

    cypress web learnings part 1

karan@LAPTOP-H56OVJTV MINGW64 ~/Desktop/cypressByMukesh (master)
$
```

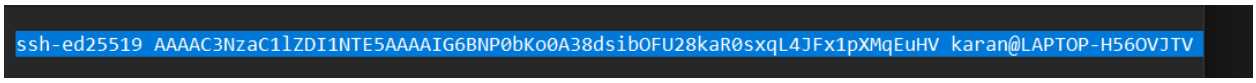
Lets try with ssh-
Project created on local.
Git init.
Git add.
Git commit.
Git log.

Open new cmd.
Ssh-keygen
It will generate private and public key.
Keep hitting Enter and finish.

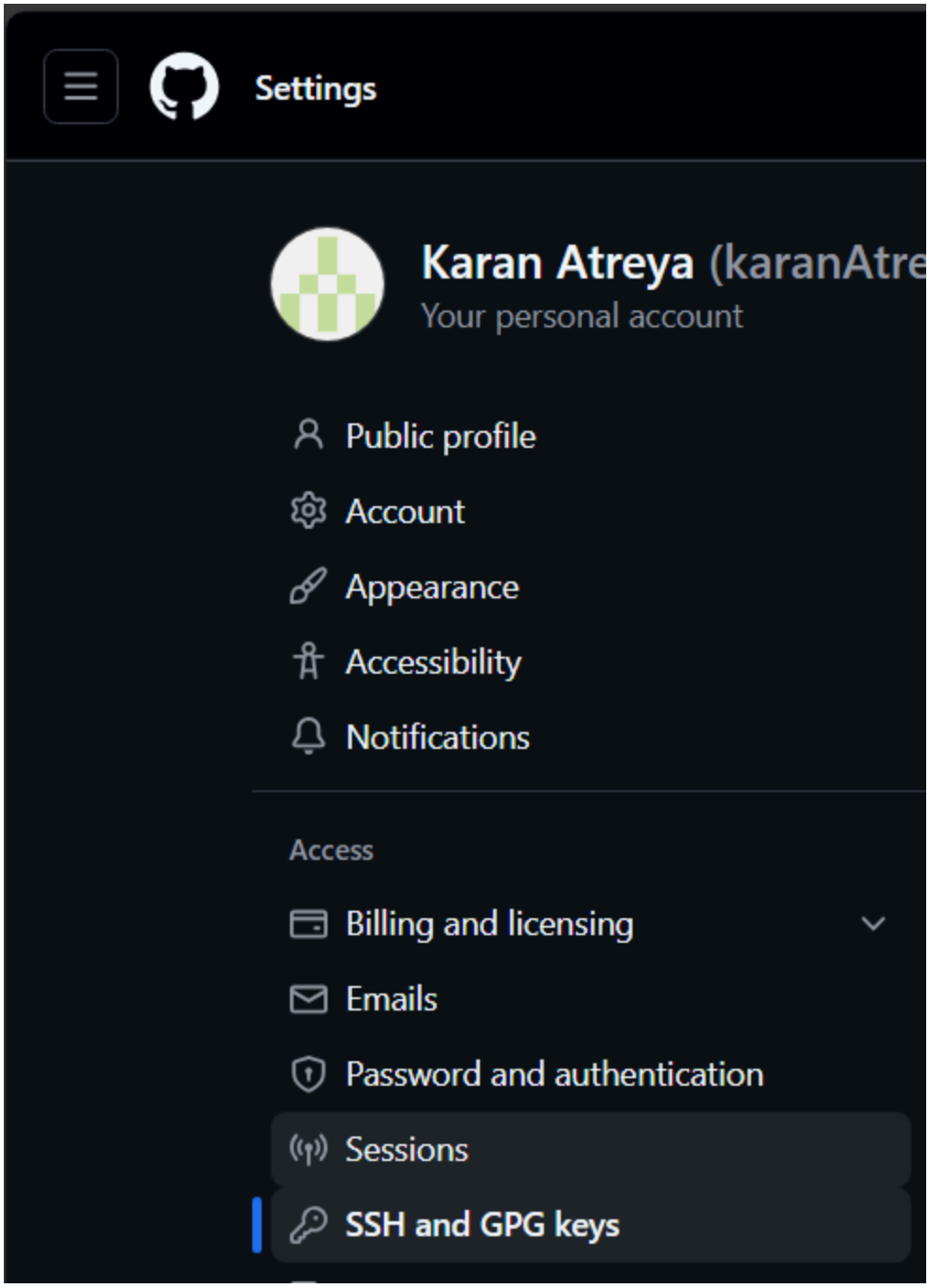
```
C:\Users\karan>ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (C:\Users\karan/.ssh/id_ed25519):
Created directory 'C:\\Users\\karan/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\karan/.ssh/id_ed25519
Your public key has been saved in C:\Users\karan/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:61b3604FS3X5p32uIEz6xMY8RmdeMnoAW8FFsezkgZ4 karan@LAPTOP-H56OVJTV
The key's randomart image is:
+--[ED25519 256]--+
|          ..o+.  +|
|           .+ . o.|
|         . o = o .|
|          = = o oo|
|         S E B ooo|
|          X.*.+..o|
|         o.%.+....|
|        ..= + o. .|
|        .. .  o=o |
+----[SHA256]-----+

C:\Users\karan>
```

Open public key.
Copy public key.

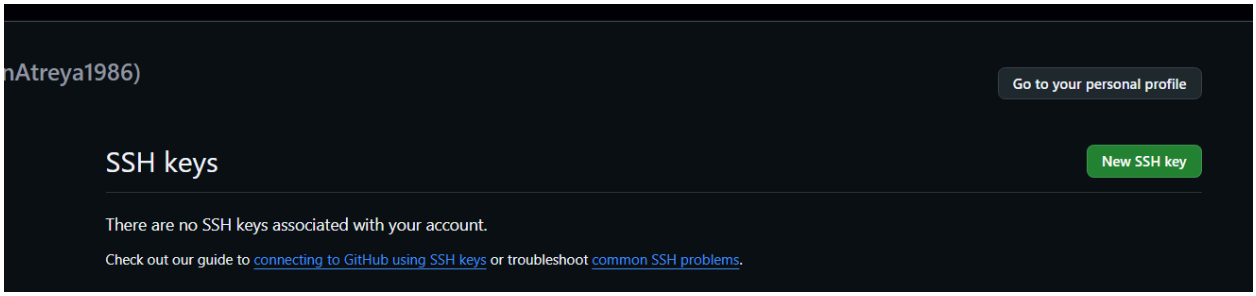




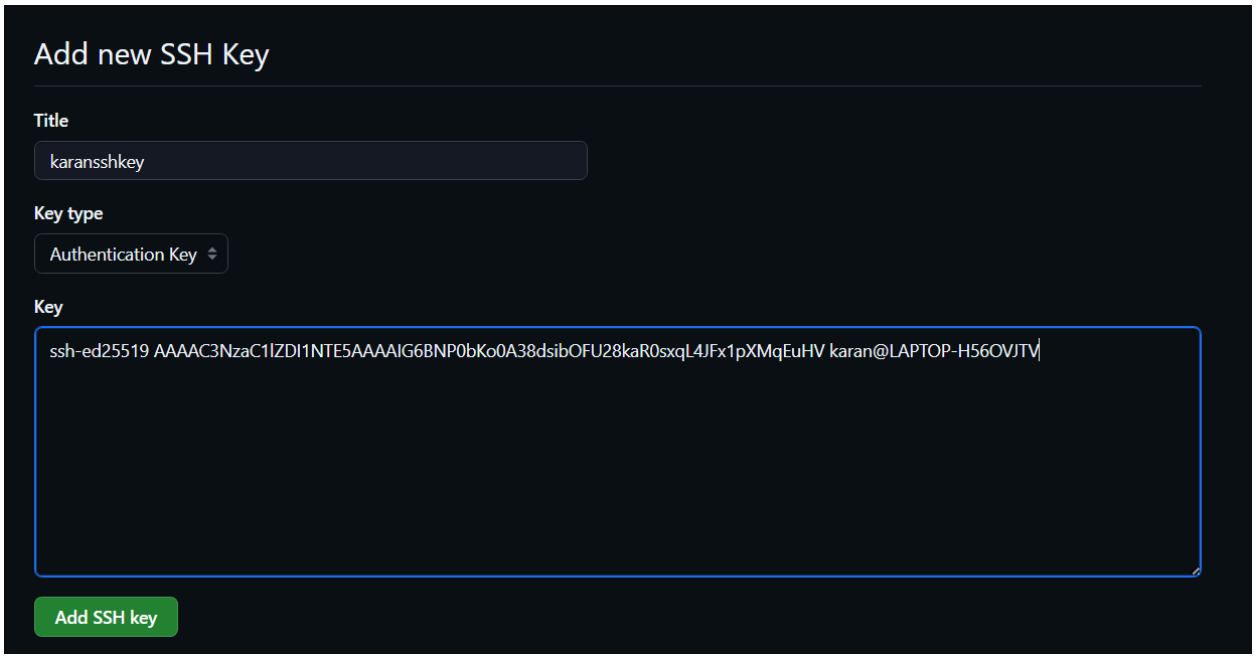
Github - settings.
Click ssh.

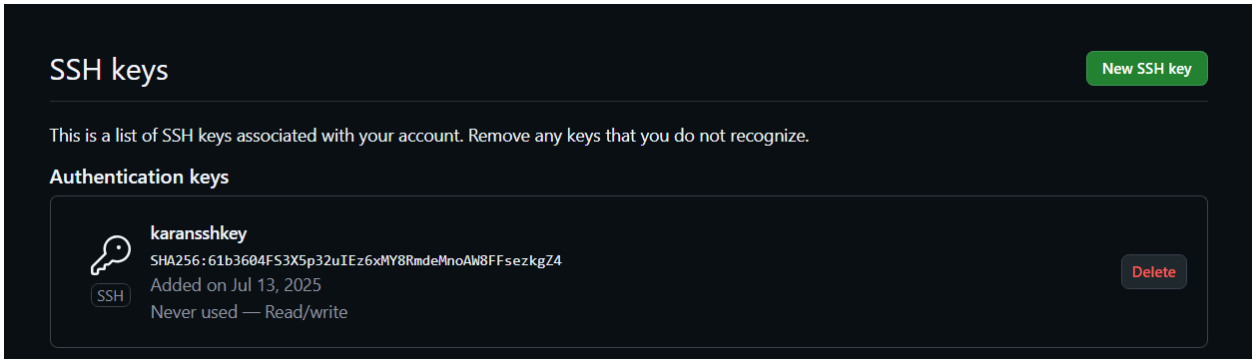

Settings
Karan Atreya (karanAtre
Your personal account
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and licensing
Emails
Password and authentication
Sessions
SSH and GPG keys

New ssh key.



Give title.
Paste the key in the key.
Click add.



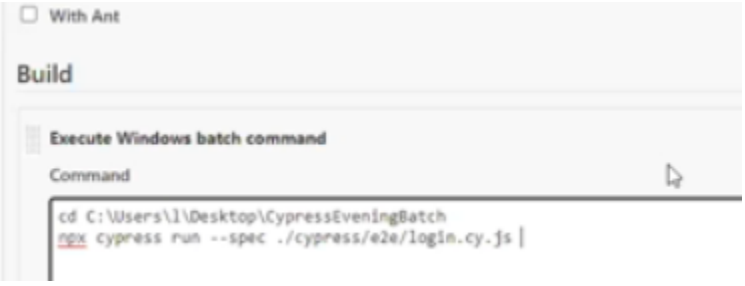


Then gitbash.
Git remote add origin.
Git push -u origin master

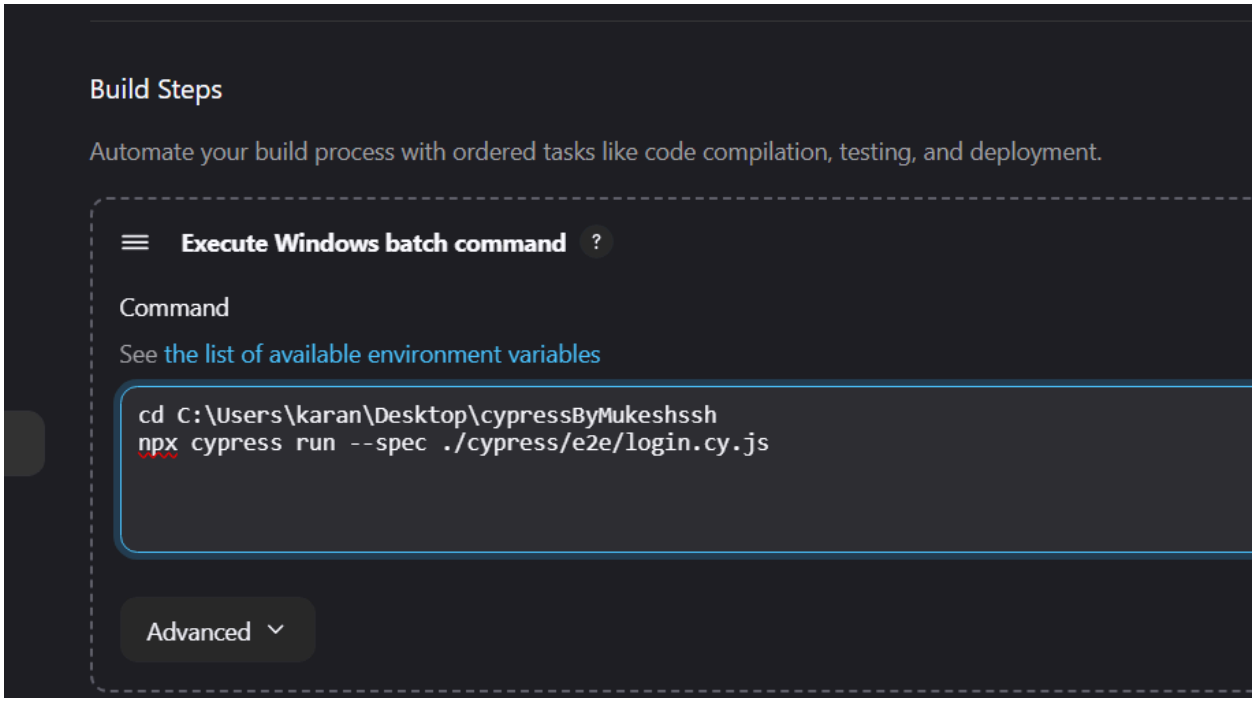
Say yes to the question. It will first time when pushing, it confirms if this key is virus free.

```
karan@LAPTOP-H56OVJTV MINGW64 ~/Desktop/cypressByMukeshssh (master)
$ git push -u origin master
The authenticity of host 'github.com (20.207.73.82)' can't be established.
ED25519 key fingerprint is SHA256:+DiY3wvvV6TuJJhbpZisF/zLDA0zPMSvHdkr4UvCOqU.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
Enumerating objects: 100, done.
Counting objects: 100% (100/100), done.
Delta compression using up to 12 threads
Compressing objects: 100% (97/97), done.
Writing objects: 100% (100/100), 3.20 MiB | 1.92 MiB/s, done.
Total 100 (delta 6), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (6/6), done.
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:      https://github.com/karanAtreya1986/MO_CypressLearningWithSSHPart2/pull/new/master
remote:
To github.com:karanAtreya1986/MO_CypressLearningWithSSHPart2.git
 * [new branch]      master -> master
branch 'master' set up to track 'origin/master'.
```

Github jenkins-
Create free style project.



Clear picture-



Save.
Build.

Segregated test -

Package.json file–

```
"description": "",
"main": "index.js",
▷ Debug
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1",
  "smoke": "cypress run --spec ./cypress/e2e/first.cy.js",
  "regression": "cypress run --spec ./cypress/e2e/createCourse.cy.js",
  "e2e": "cypress run",
  "smoke-chrome": "cypress run --spec ./cypress/e2e/first.cy.js --browser=chrome --headed"
},
"author": "",
"license": "ISC",
```

Clear picture-
In package.json no need to give npm or npx at start as its understood.

```json
{
  "name": "cypressbymukesh",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "smoke": "cypress run --spec ./cypress/e2e/first.cy.js",
    "regression": "cypress run --spec ./cypress/e2e/createCourse.cy.js",
    "e2e": "cypress run",
    "smokeonchromebrowserheaded": "cypress run --spec ./cypress/e2e/apitest.cy.js --browser=chrome --headed"
  },
  "author": "",
  "license": "ISC",
  "description": "",
  "devDependencies": {
    "cypress": "^14.5.1",
    "cypress-file-upload": "^5.0.8",
    "cypress-iframe": "^1.0.1",
    "cypress-mochawesome-reporter": "^3.8.2",
    "cypress-xpath": "^2.0.1"
  },
  "dependencies": {
    "cypress-real-events": "^1.14.0"
  }
}
```

If you give npm cypress run, it is error. It should be npm run <filename>
Only with npx we add cypress. Npx cypress run <filename>

package.json

package.json > {} devDependencies

```json
{
  "name": "cypressbymukesh",
  "version": "1.0.0",
  "main": "index.js",
  ▷Debug
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "smoke": "cypress run --spec ./cypress/e2e/first.cy.js",
    "regression": "cypress run --spec ./cypress/e2e/createCourse.cy.js",
    "e2e": "cypress run",
    "smokeonchromebrowserheaded": "npm cypress run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed"
  },
  "author": "",
```

Jenkins-

```
> cypressbymukesh@1.0.0 smokeonchromebrowserheaded
> npm cypress run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed

Unknown command: "cypress"

To see a list of supported npm commands, run:
  npm help
Build step 'Execute Windows batch command' marked build as failure
Email was triggered for: Always
Sending email for trigger: Always
Sending email to: seleniumforkaran@gmail.com
Finished: FAILURE
```

Another use case-
I have this in my package.

```
package.json ×
package.json > {} scripts > smokeonchromebrowserheaded
1  {
2    "name": "cypressbymukesh",
3    "version": "1.0.0",
4    "main": "index.js",
     ▷Debug
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1",
7      "smoke": "cypress run --spec ./cypress/e2e/first.cy.js",
8      "regression": "cypress run --spec ./cypress/e2e/createCourse.cy.js",
9      "e2e": "cypress run",
10     "smokeonchromebrowserheaded": "npm run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed"
11   }
```

My jenkins has this command:
Npm run –spec ……. We get error.

```
C:\Users\karan\Desktop\cypressByMukeshssh>npm run smokeonchromebrowserheaded

> cypressbymukesh@1.0.0 smokeonchromebrowserheaded
> npm run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed

npm error Missing script: "./cypress/e2e/login2.cy.js"
npm error
npm error To see a list of scripts, run:
npm error   npm run
npm error A complete log of this run can be found in: C:\Users\karan\AppData\Local\npm-cache\_logs\2025-07-13T06_10_40_579Z-debug-0.log
Build step 'Execute Windows batch command' marked build as failure
Email was triggered for: Always
Sending email for trigger: Always
Sending email to: seleniumforkaran@gmail.com
Finished: FAILURE
```

Why we get error -
Your Jenkins build failed because of a **badly formed npm command**. Let's break down the issue and how to fix it.

You ran-
npm run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed

But `npm run` expects the **name of a script defined in your `package.json`**, like:
"scripts": {
  "test": "cypress run",
  "smoke": "cypress run --spec ./cypress/e2e/login2.cy.js"
}

So, the error:
npm error Missing script: "./cypress/e2e/login2.cy.js"

...means you're treating the test file path as a script name — which is incorrect.

How to Fix It

You have two options:

Option 1: Use `npx cypress run` Directly (recommended for Jenkins)

Replace this line in your Jenkins batch command:

npm run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed

With:
npx cypress run --spec "./cypress/e2e/login2.cy.js" --browser chrome --headed

This runs Cypress directly without relying on a script in `package.json`.

Option 2: Define a Custom Script in `package.json`

In your `package.json`, add:

"scripts": {
  "smokeonchromebrowserheaded": "cypress run --spec ./cypress/e2e/login2.cy.js --browser chrome --headed"
}

Then, in Jenkins, run:
npm run smokeonchromebrowserheaded

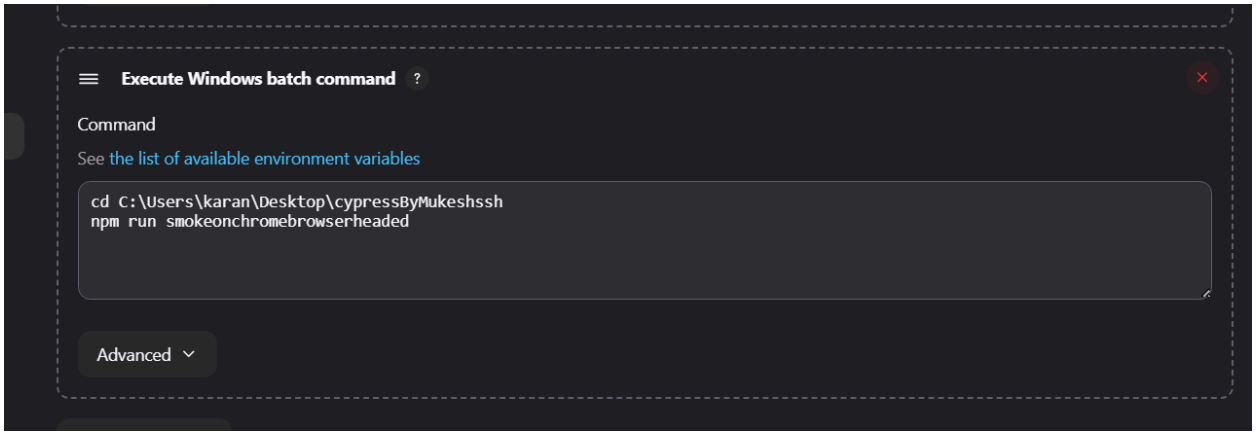
Bonus Tip: View Available Scripts

To see which scripts are available in your `package.json`, run:

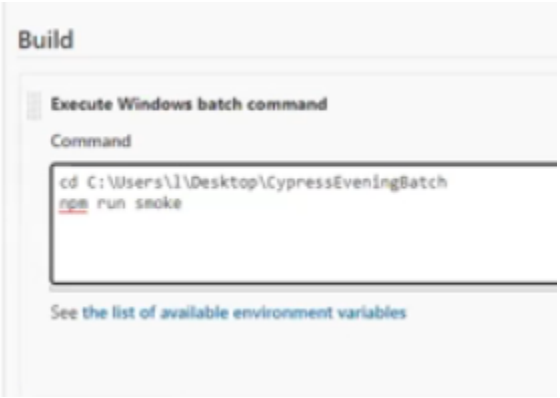
npm run

So in short, in script section dont include the npx or npm word.

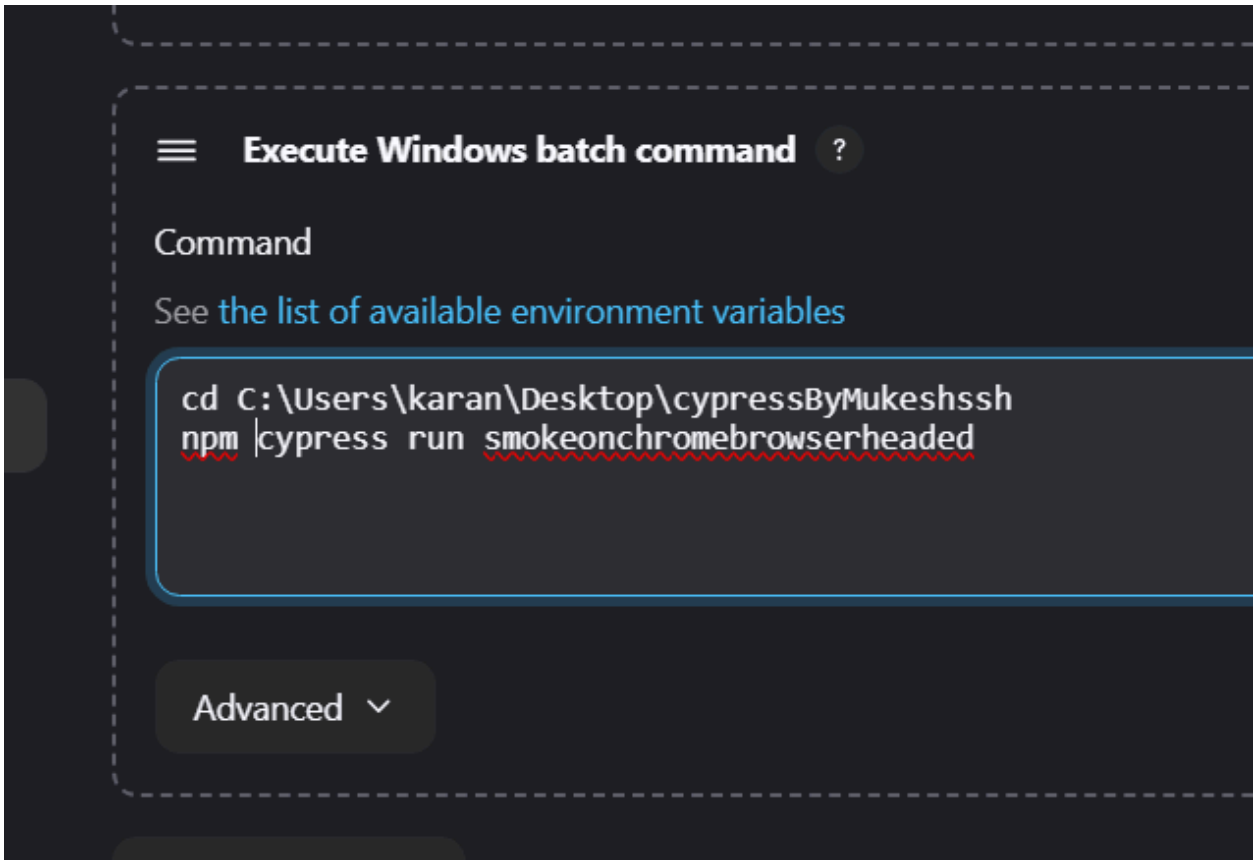
```
package.json ×
package.json > {} scripts > smokeonchromebrowserheaded
1  {
2    "name": "cypressbymukesh",
3    "version": "1.0.0",
4    "main": "index.js",
     ▷Debug
5    "scripts": {
6      "test": "echo \"Error: no test specified\" && exit 1",
7      "smoke": "cypress run --spec ./cypress/e2e/first.cy.js",
8      "regression": "cypress run --spec ./cypress/e2e/createCourse.cy.js",
9      "e2e": "cypress run",
10     "smokeonchromebrowserheaded": "cypress run --spec ./cypress/e2e/login2.cy.js --browser=chrome --headed"
11   },
```



Go to build and update-

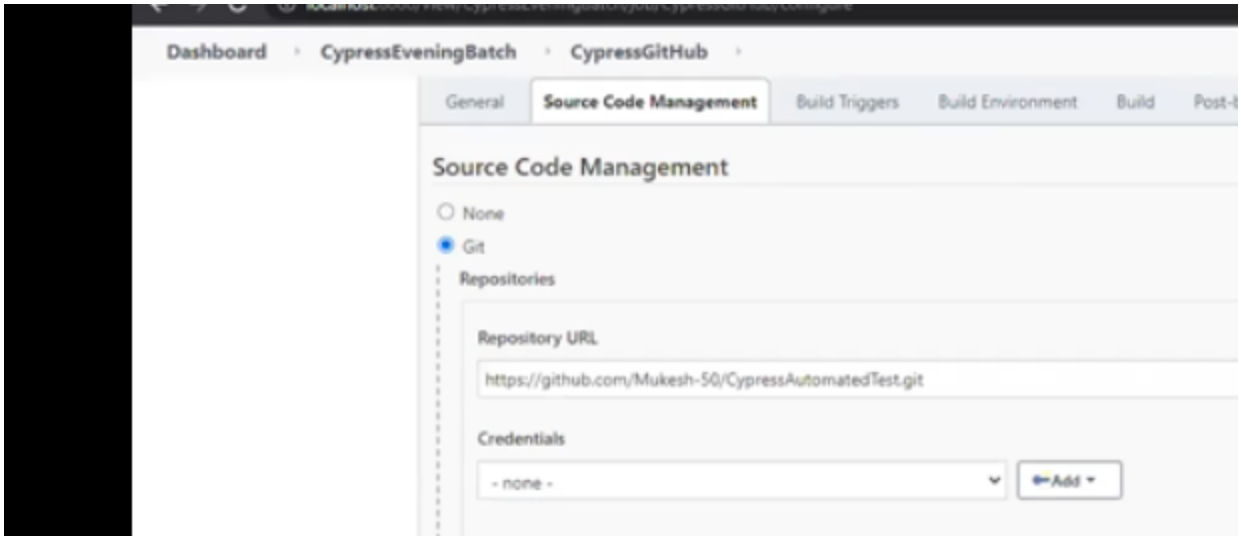


Clear picture -

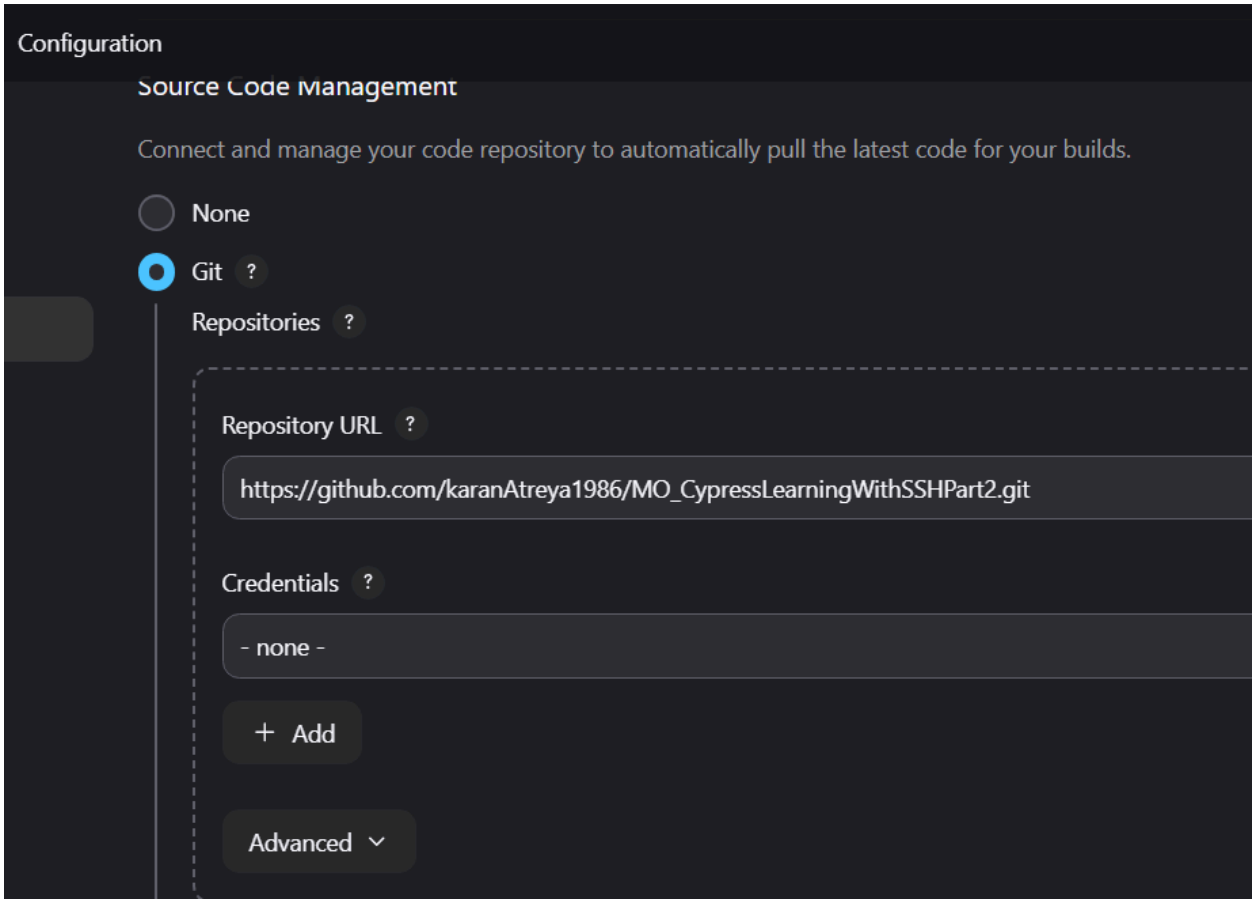


Build and it works.

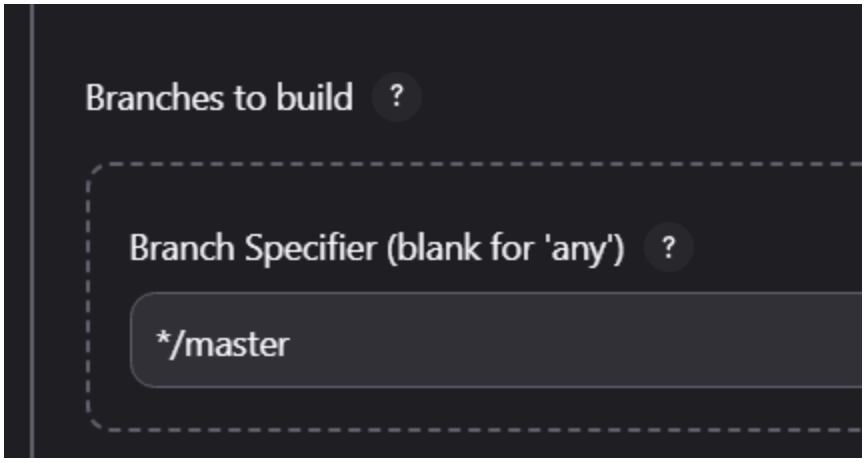
Now lets do from git-
Give repo url.

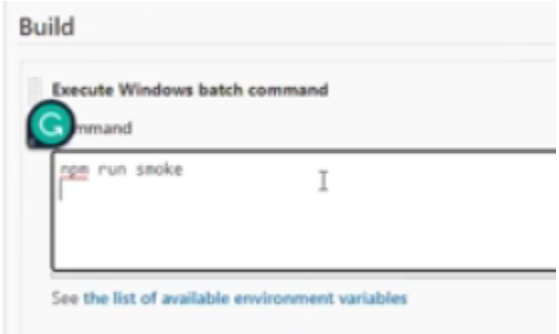


Clear picture -
For clone no need of credentials.

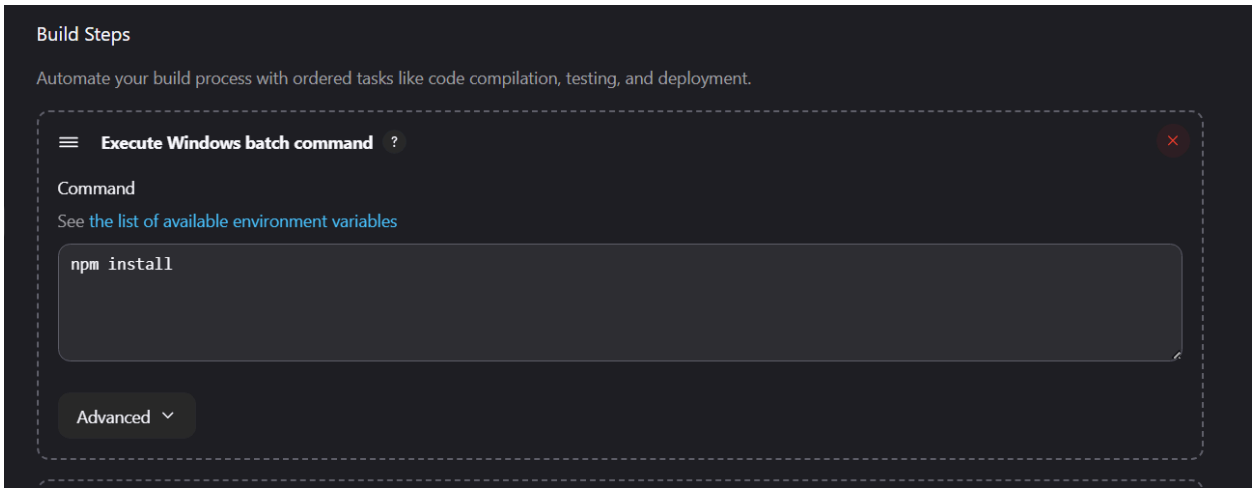


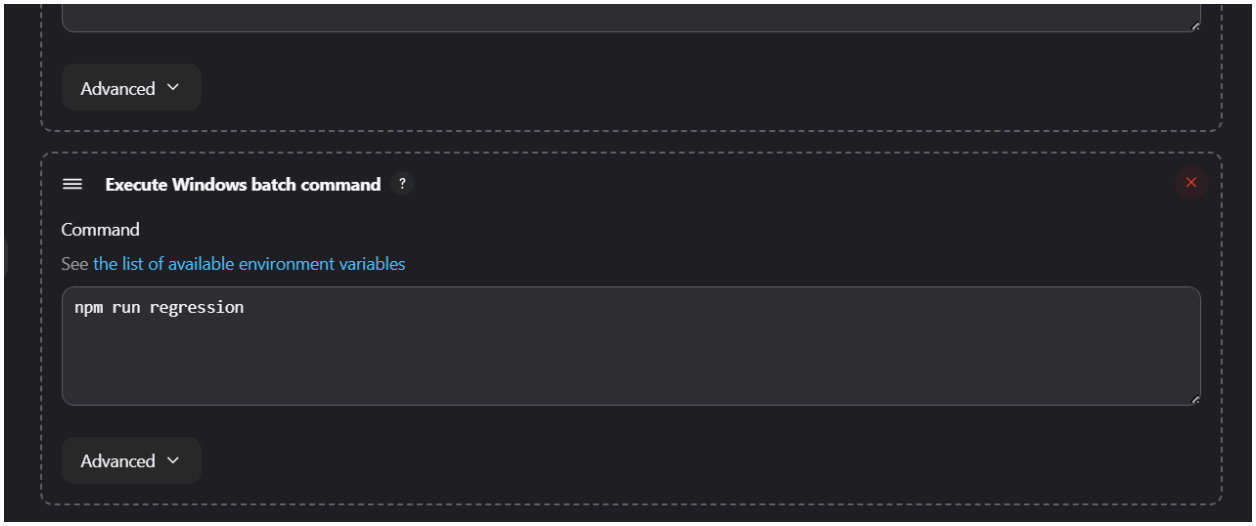
Select the branch where to run from-



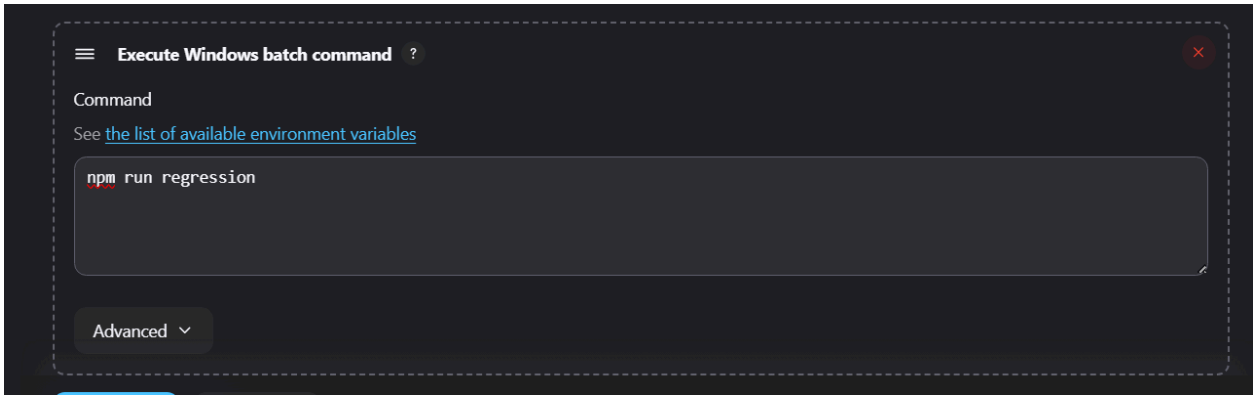


Clear picture-
Always keep installation and running commands separate, else it only runs the first command.





Same code pasted below:

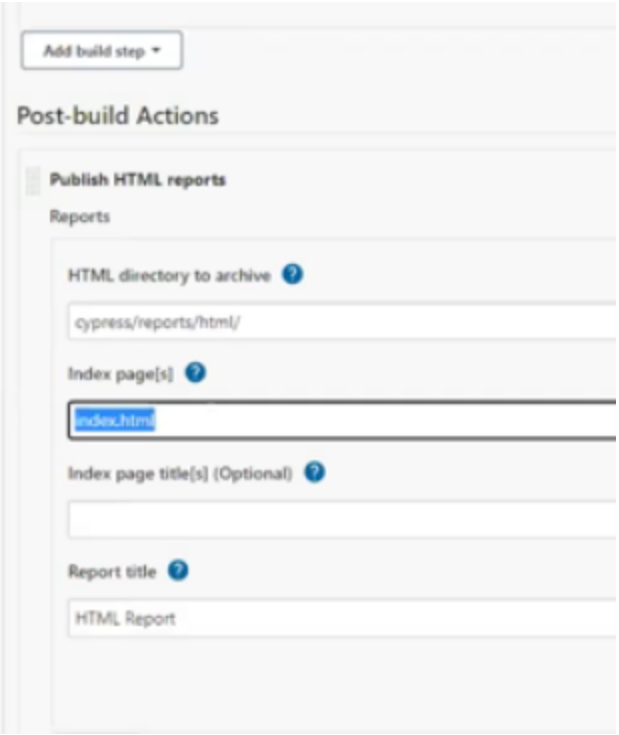


Click save. No need to give path of project as its in git.

Click build now.

NOTE-
In javascript, the package.lock.json and package.json should not be added to .gitignore because when we run from CI those files are need to specify npm installations and to run specific scripts mentioned in package.json.

Publish html reports via jenkins-

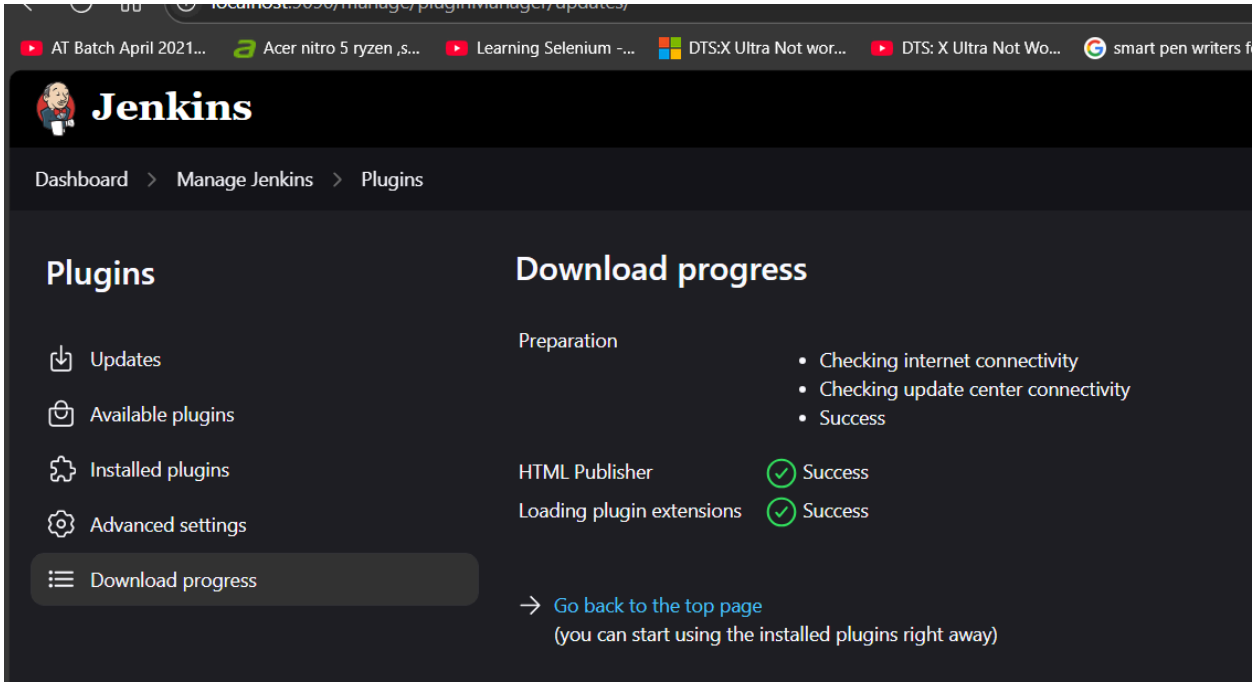


Clear picture is down.

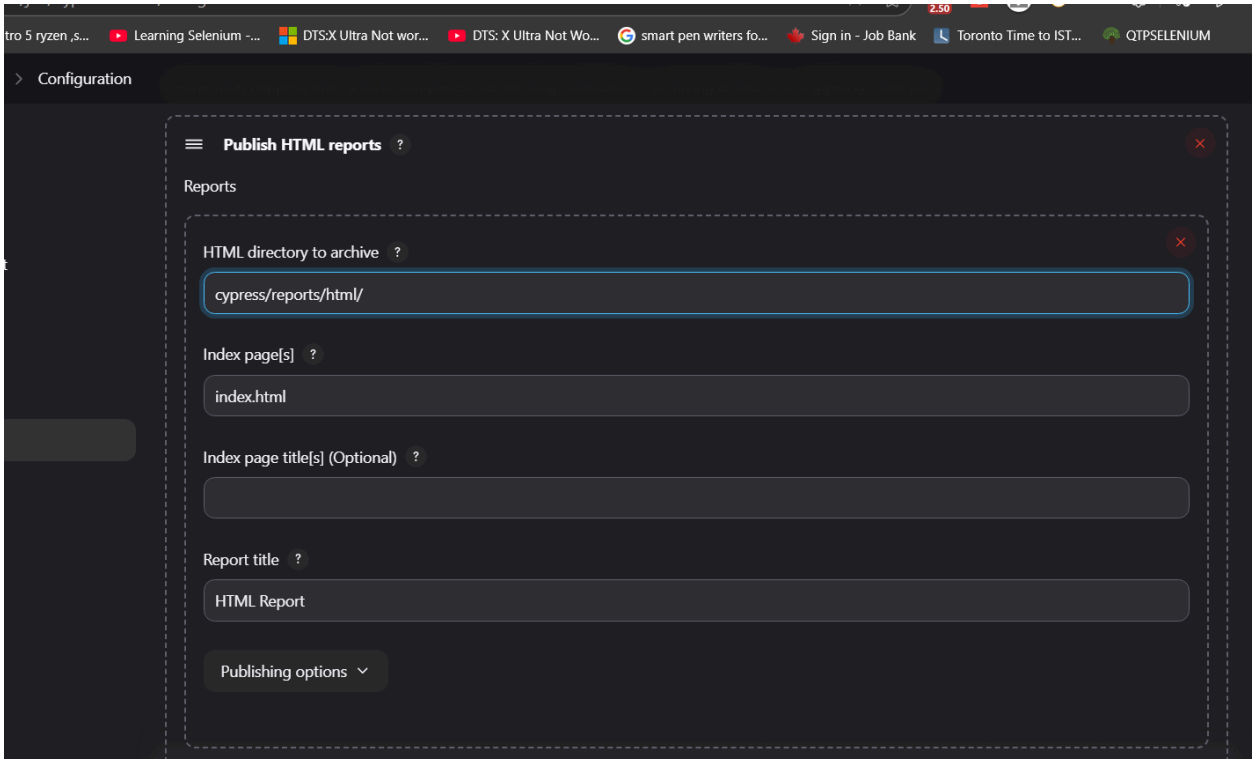
Click save.

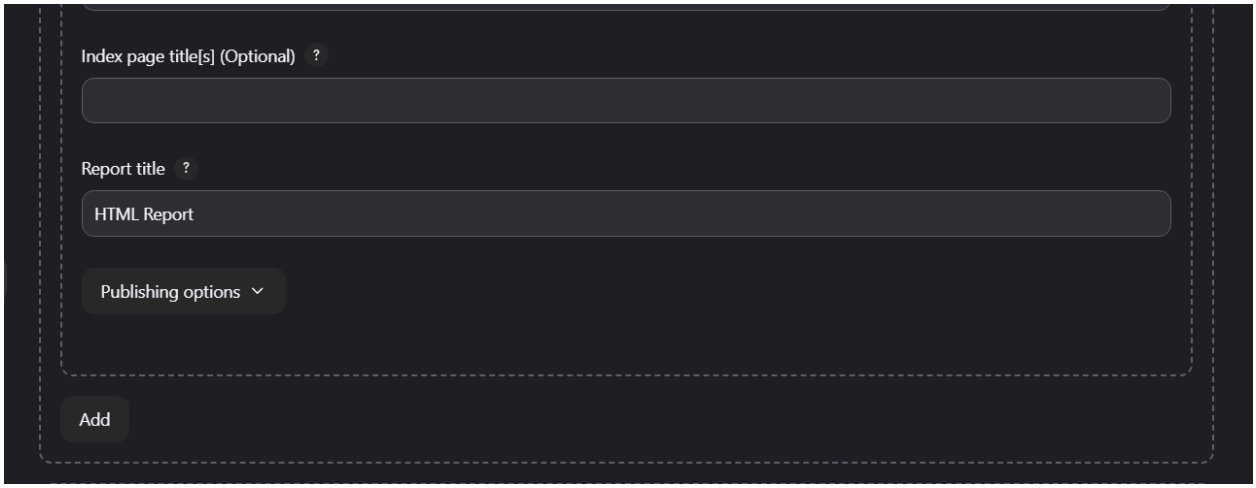
Click build now.

NOTE-
Html report not directly available.
Go to manage jenkins. Plugins. Search for html publisher. Install.



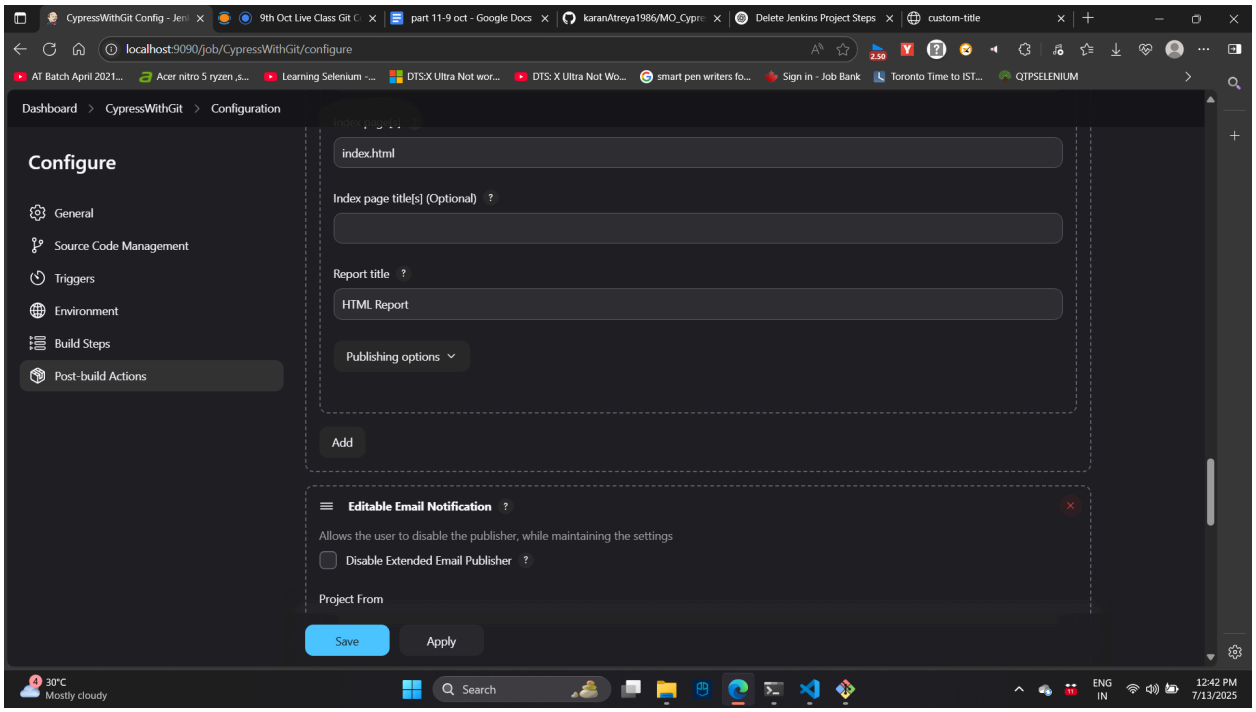
Now come back to project.
Configure.
Post build action.
Add html report.





Click save.

ocd-



Bad news is the html doesnt work directly.
Mukesh will tell later.