Tools Needed:

Access to OpenSSH Client

1) WSL (recommended for Windows users)
2) Command Line / Terminal (Mac)
3) Linux … (If you're using this, I don't have to tell you.)

Accessing the Jenkins Server

1) Create an SSH Tunnel from your machine to the Jenkins Server
    a. Servers
        i. osc-mgmt.eng.auburn.edu
            1. username: comp2710
            2. password: comp2710
        ii. 192.168.0.178 (only accessible from osc-mgmt – the cloud gateway)
            1. Username: jenkins
            2. Password: jenkins_p@ssw0rd (the 0 is the number 0 not capitol O)

Example:

```
ssh -L 8002:localhost:8080
    -J comp2710@osc-mgmt.eng.auburn.edu
       jenkins@192.168.0.178
```

Command Breakdown:

-L forward a local port to a remote machine

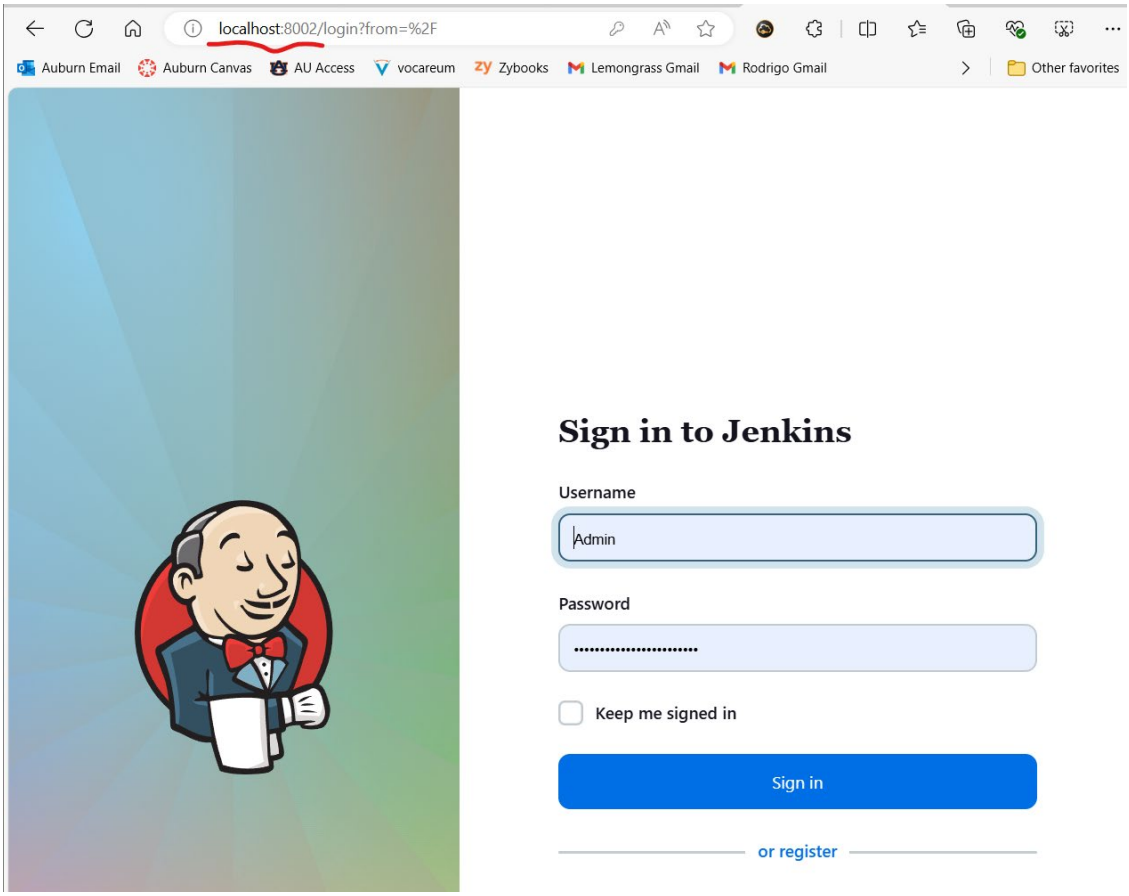
Blue (8002) is the local port

Orange (8080) is the port on the remote machine

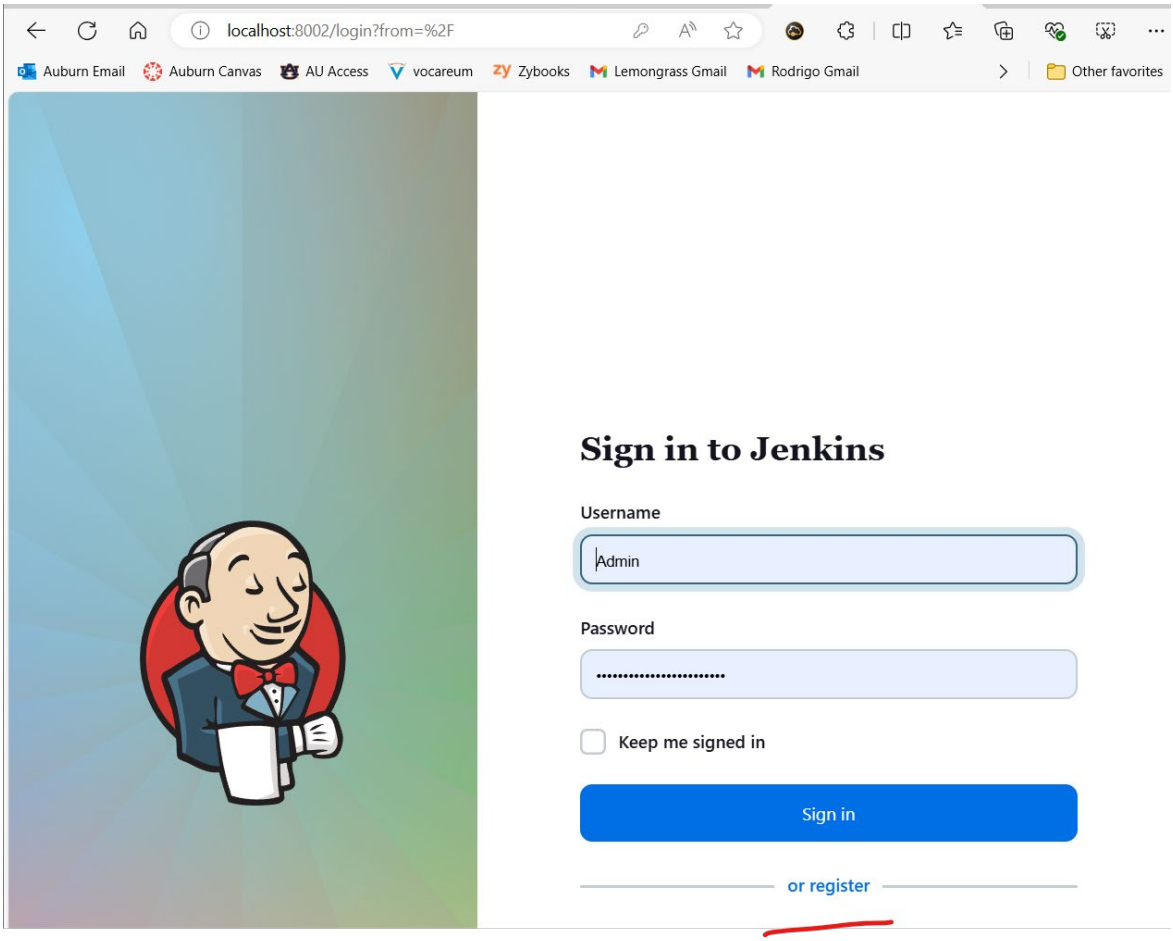
Localhost is the remote machine

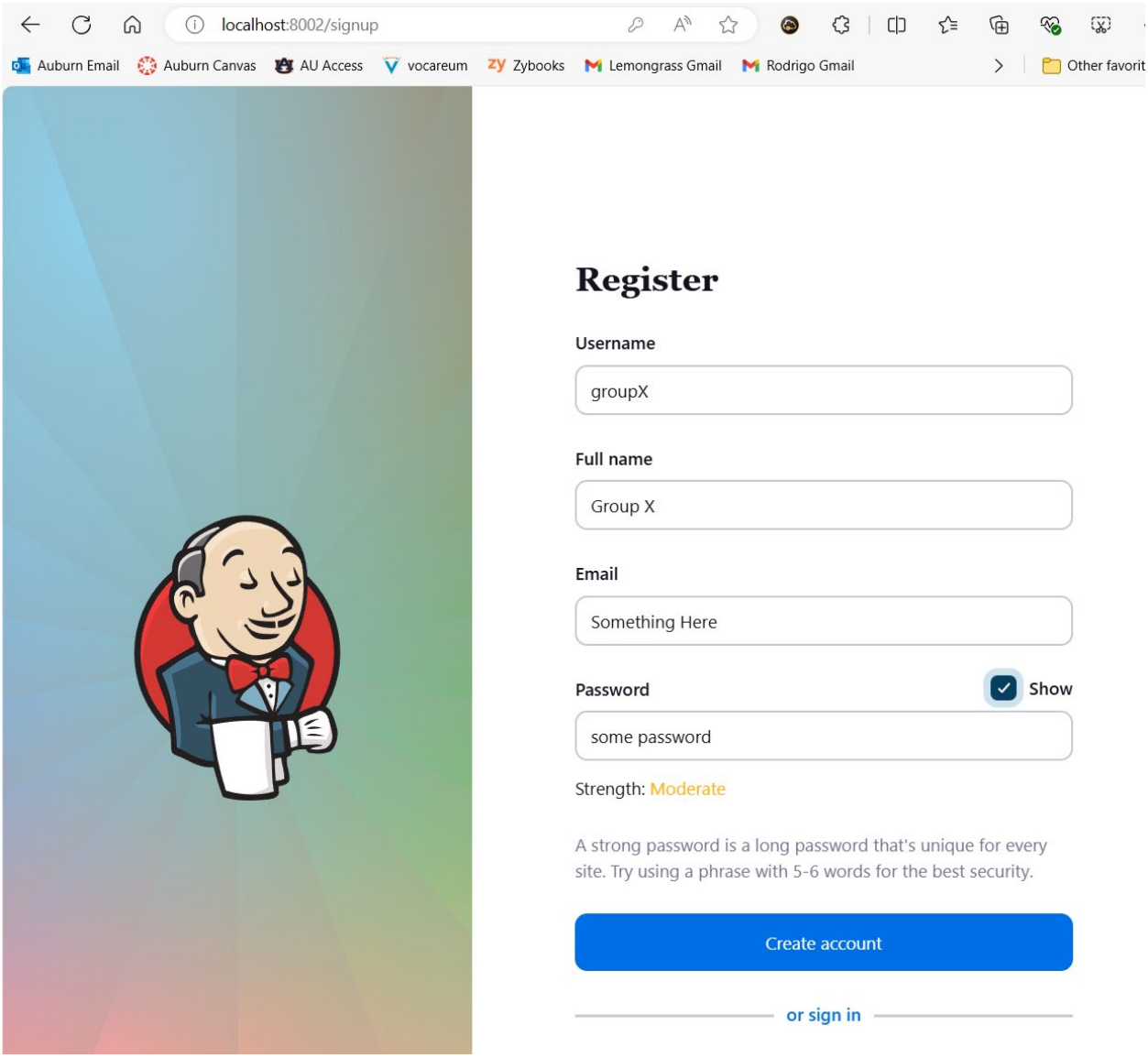
-J "jump" through multiple machines to get to the final destination. Format is comma separated list with final destination separated with space. *Ex: server1,server2,server3 final_server*

2) Open any web browser and go to the remote machine part of the address in the tunnel you made, followed by the local port that you are forwarding. For this example, this would be **localhost:8002**



3) If this is your first time accessing the server, register and create an account.





a. Be sure to use your auburn ID as your username.
   Example: **ras0054**
b. For full name, you can put your name

Using Jenkins

The Jenkins server is a shared server. Please use responsibly and DO NOT modify or delete or run other people's jobs. You will all be using the same Jenkins server, meaning that you will have access to everyone's jobs.

Please Limit the number of jobs created to 1 job per assignment per group. For example, for homework 1, each group will create a single job. That's 24 jobs. For the second assignment, each group will create another job. That would be a total of 48 jobs.

Naming Conventions:

In order to help the TA and I with grading, please name your jobs as follows:

Group**X**-HW**Y** Where the **X** is your group number and the **Y** is the assignment number. Use the same case as the example here. For example:

Group1-HW1 is good.

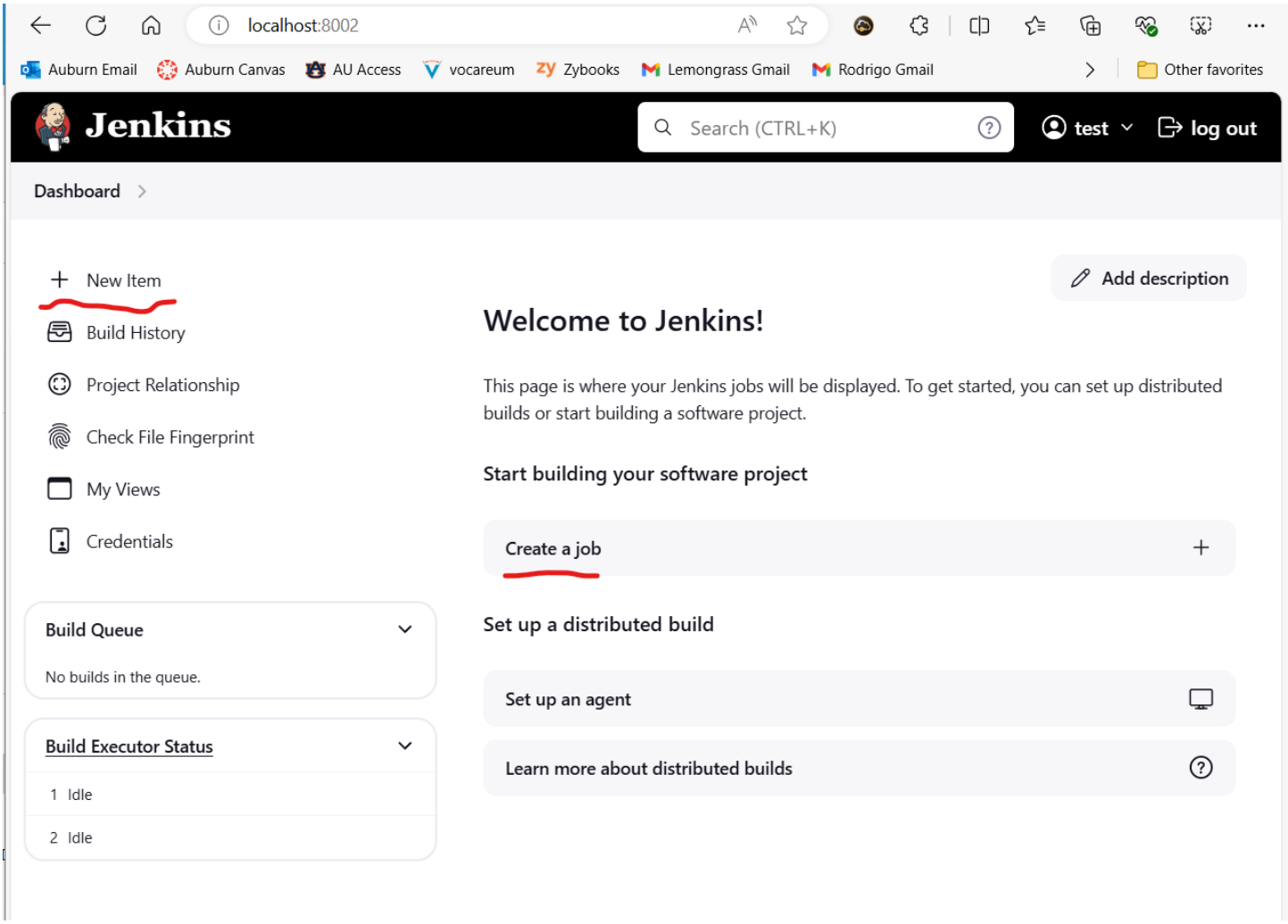
group1-hw1 is NOT good because the case is different.

This will make it easier for the TA to grade your assignments as he/she can easily create a filter for all assignments starting with your group.

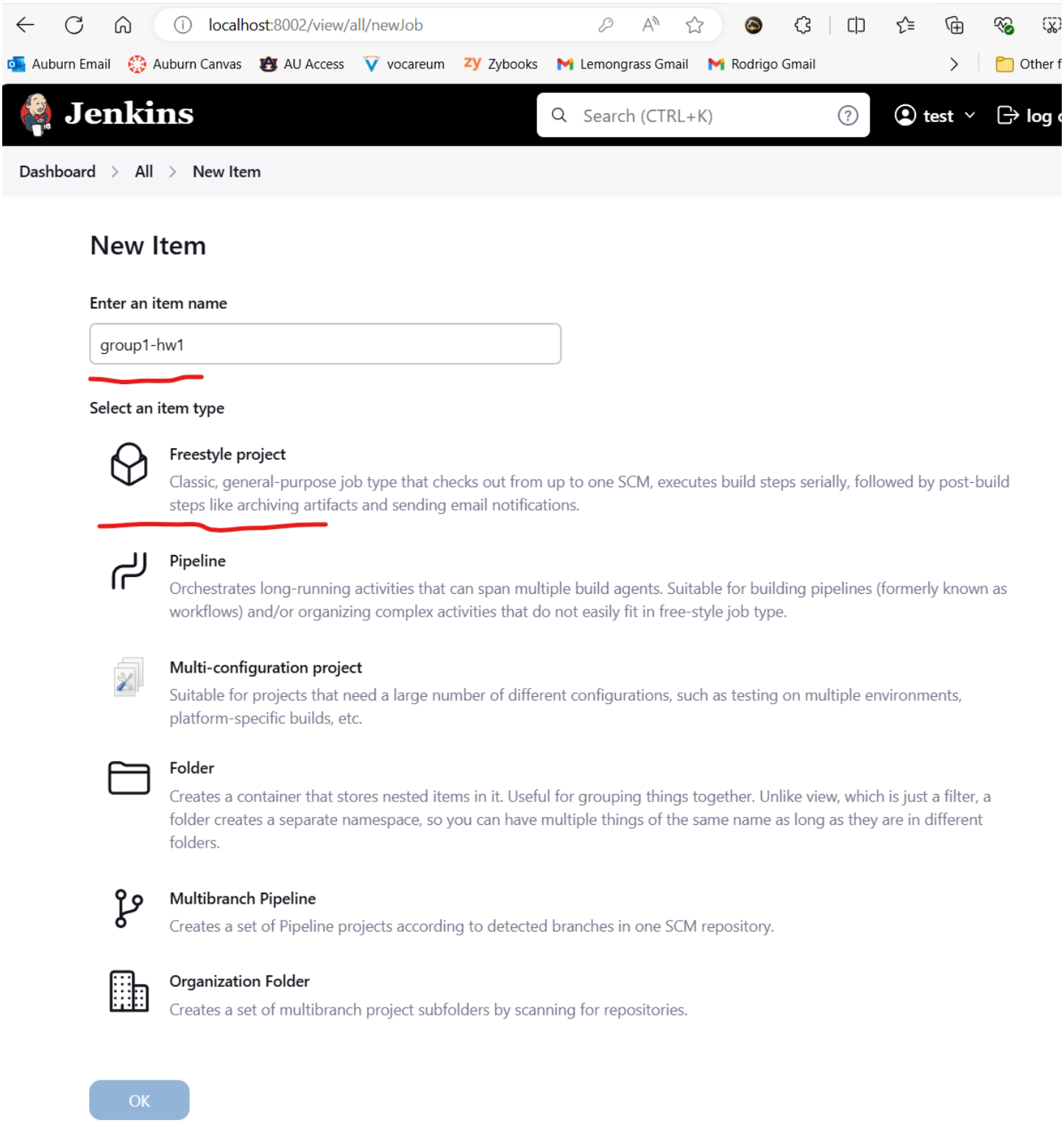
Creating a Jenkins Job

Jenkins is a tool that has the ability to automate tasks for you. For example, we are going to be using it to fetch code from a github repo, pull it locally, compile it, and run it. It can do other things (like launch docker containers, create vms, etc...

One way that you specify what kind of tasks you want it to do for you is by creating a job. A job is one or more tasks that you want to automate.



1) If there are no jobs on the server, your dashboard will look something like this. Select Create a job. You can also select “New Item”.



2) For now, select “Freestyle project”. Note that the “item name” is groupX-hwX. The item name is also the job name. Be sure to check “Discard old builds” to save disk space on the server.

Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- Build Steps
- Post-build Actions

General

Enabled

Description

Plain text Preview

Discard old builds

Strategy

Log Rotation

Days to keep builds

if not empty, build records are only kept up to this number of days

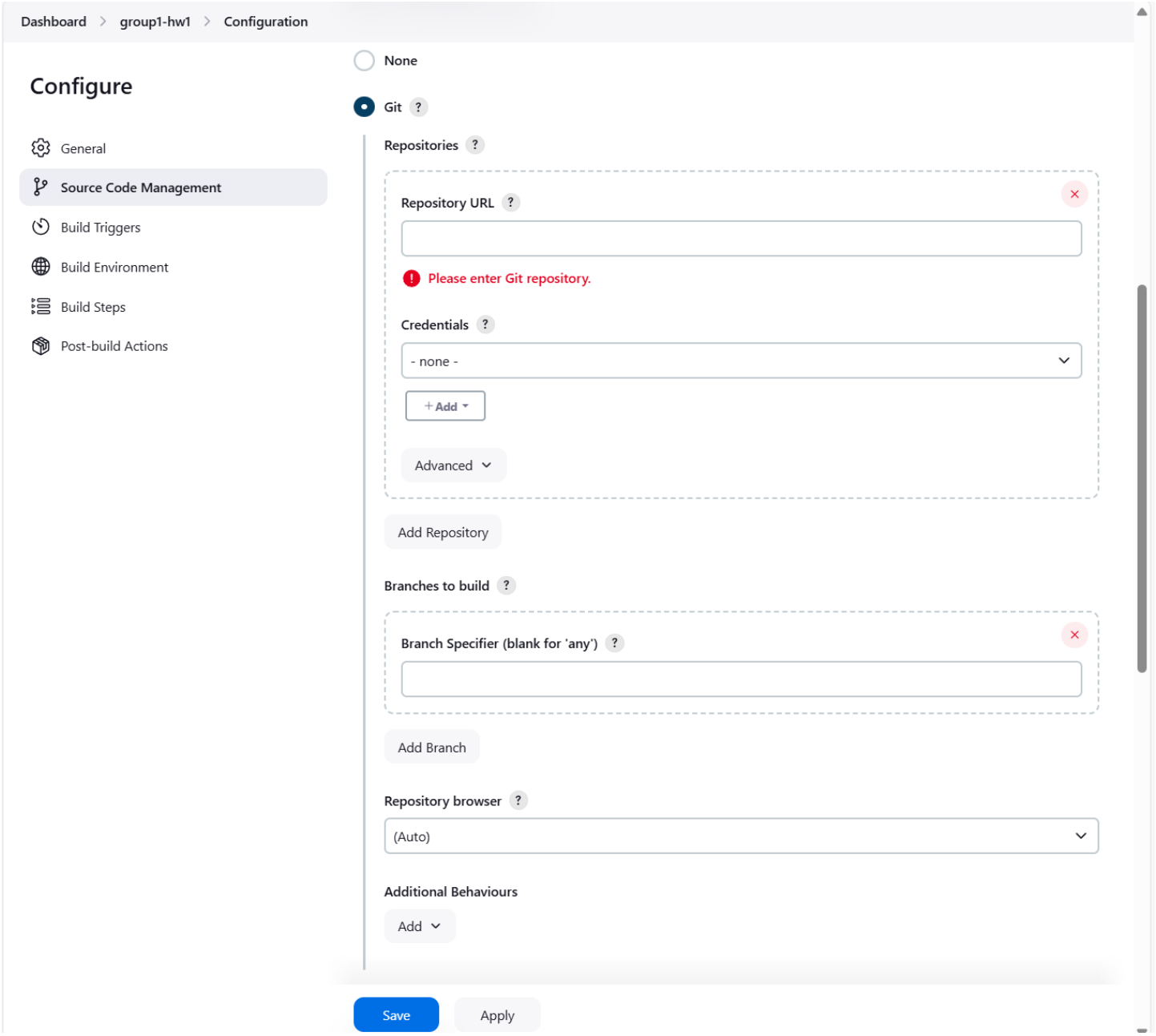
Max # of builds to keep

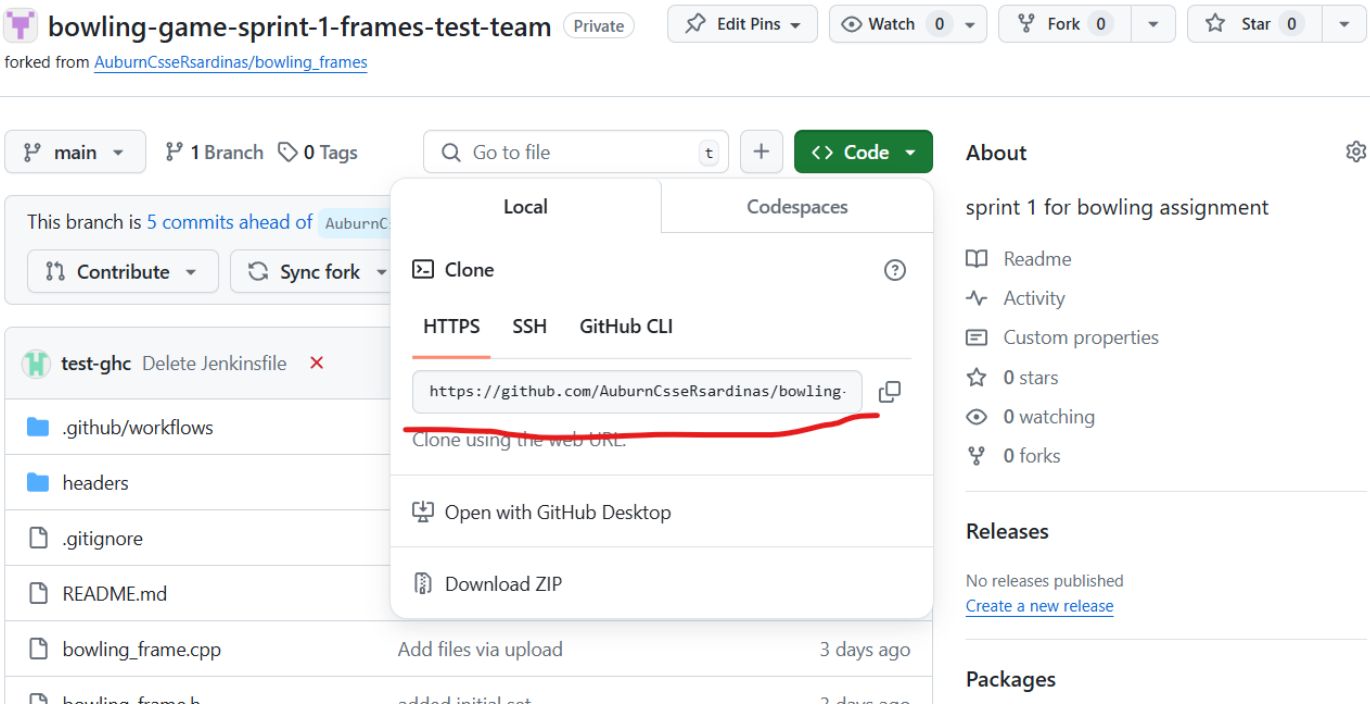
if not empty, only up to this number of build records are kept

1

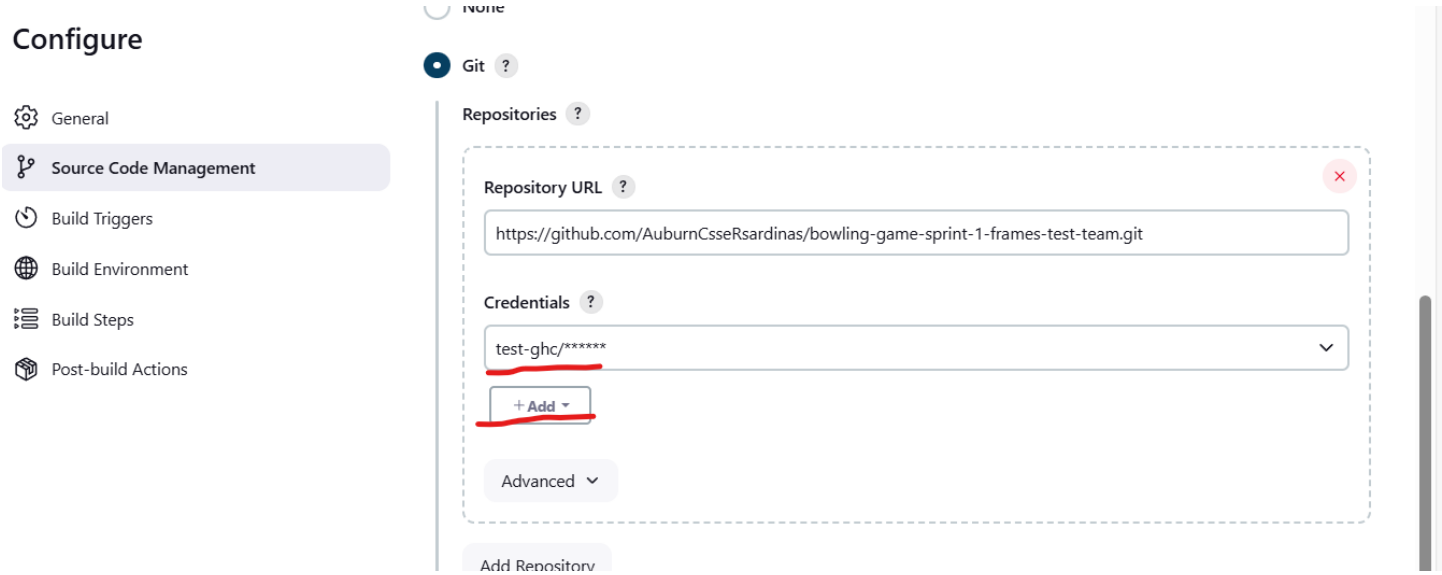
Advanced

3) Selecting ok will take you to a page where you can configure your job. For now, we want to configure several things.
    a. Our github repo (each assignment you'll be assigned a new github repo that you can use to upload your work for that specific assignment).

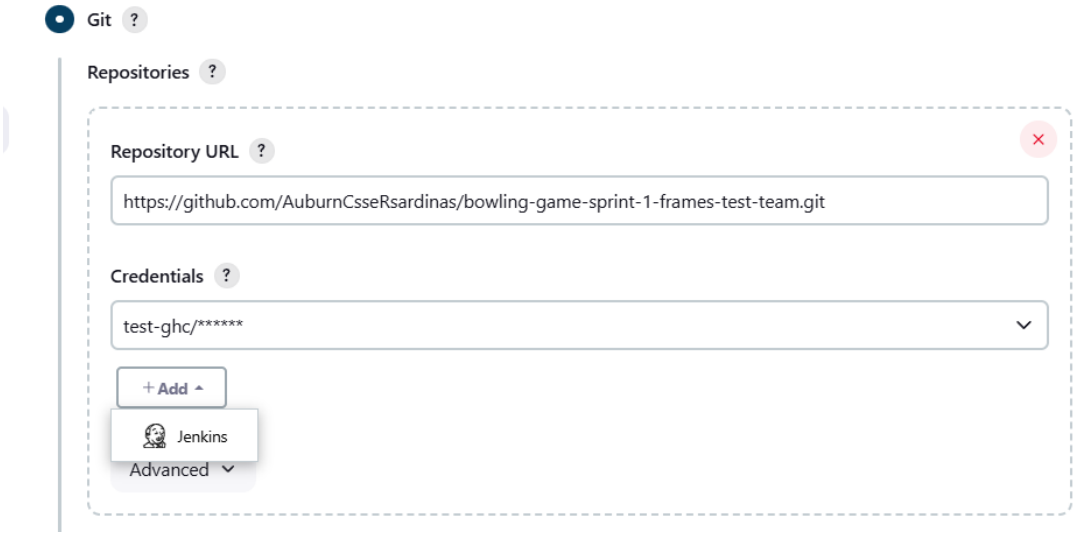


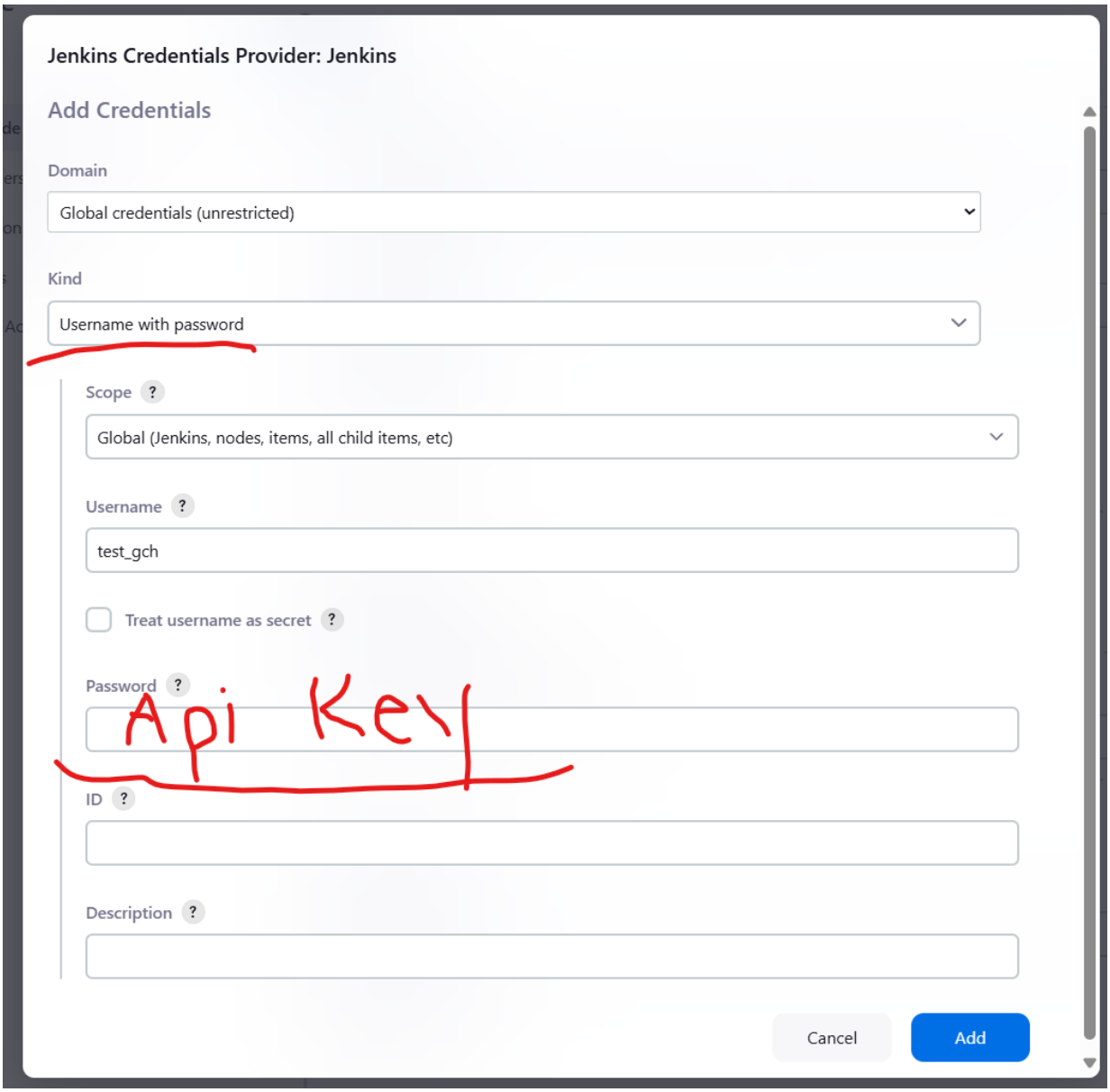


i. The repository url is the url you would use to clone the github repo.

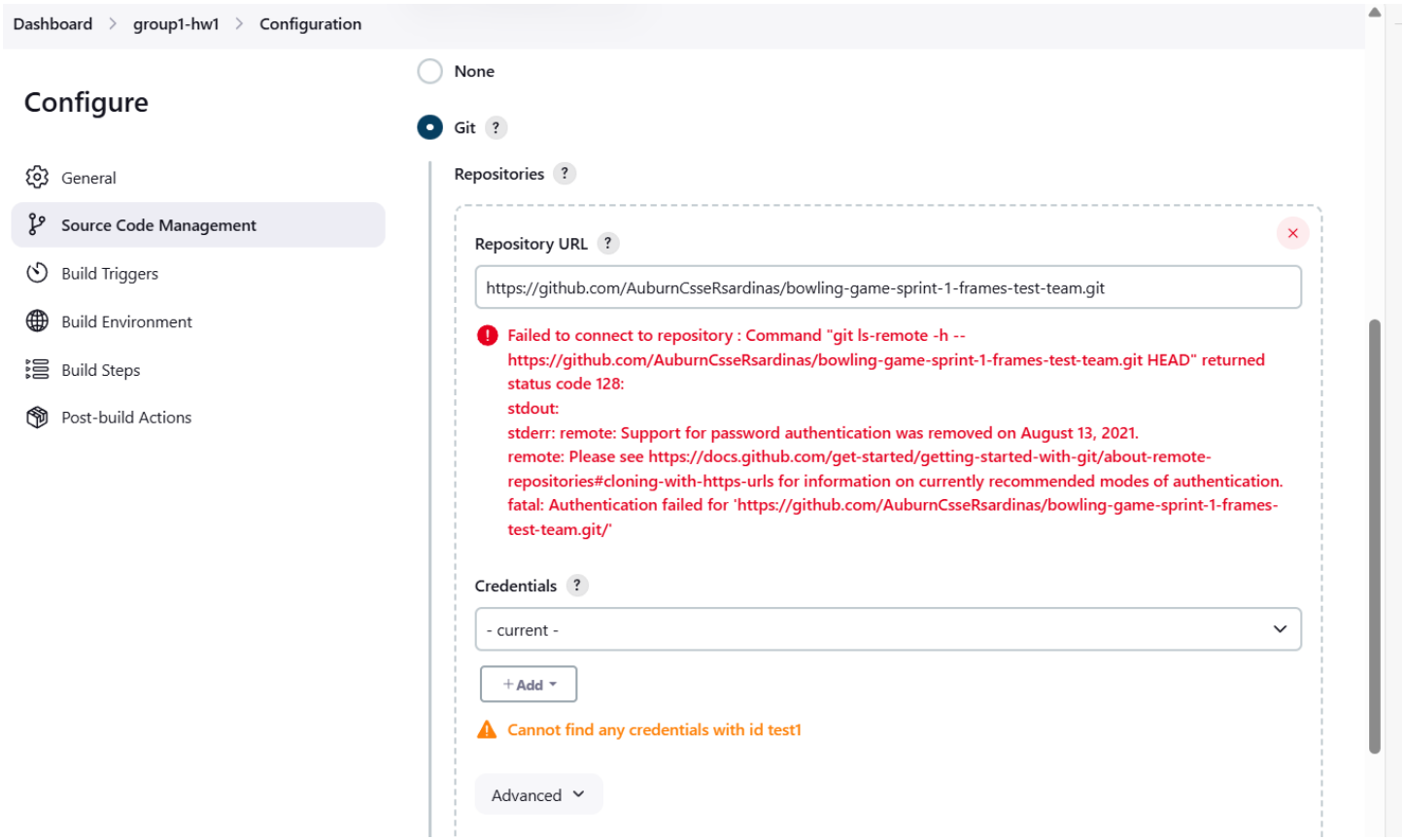


b. Next you need to either select some credentials that will give you access to that repo (if you've already created them), or you need to create them. I have found that it is easiest to create an api key and use that to access all of your github repos. This will be covered in another section of this document.

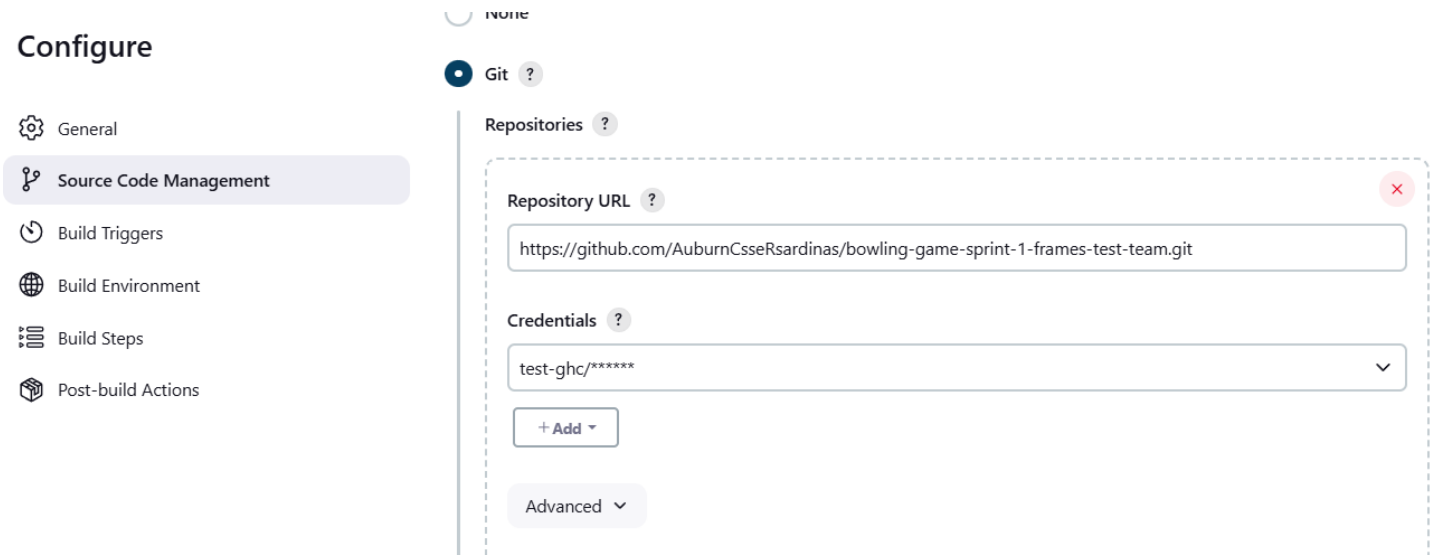




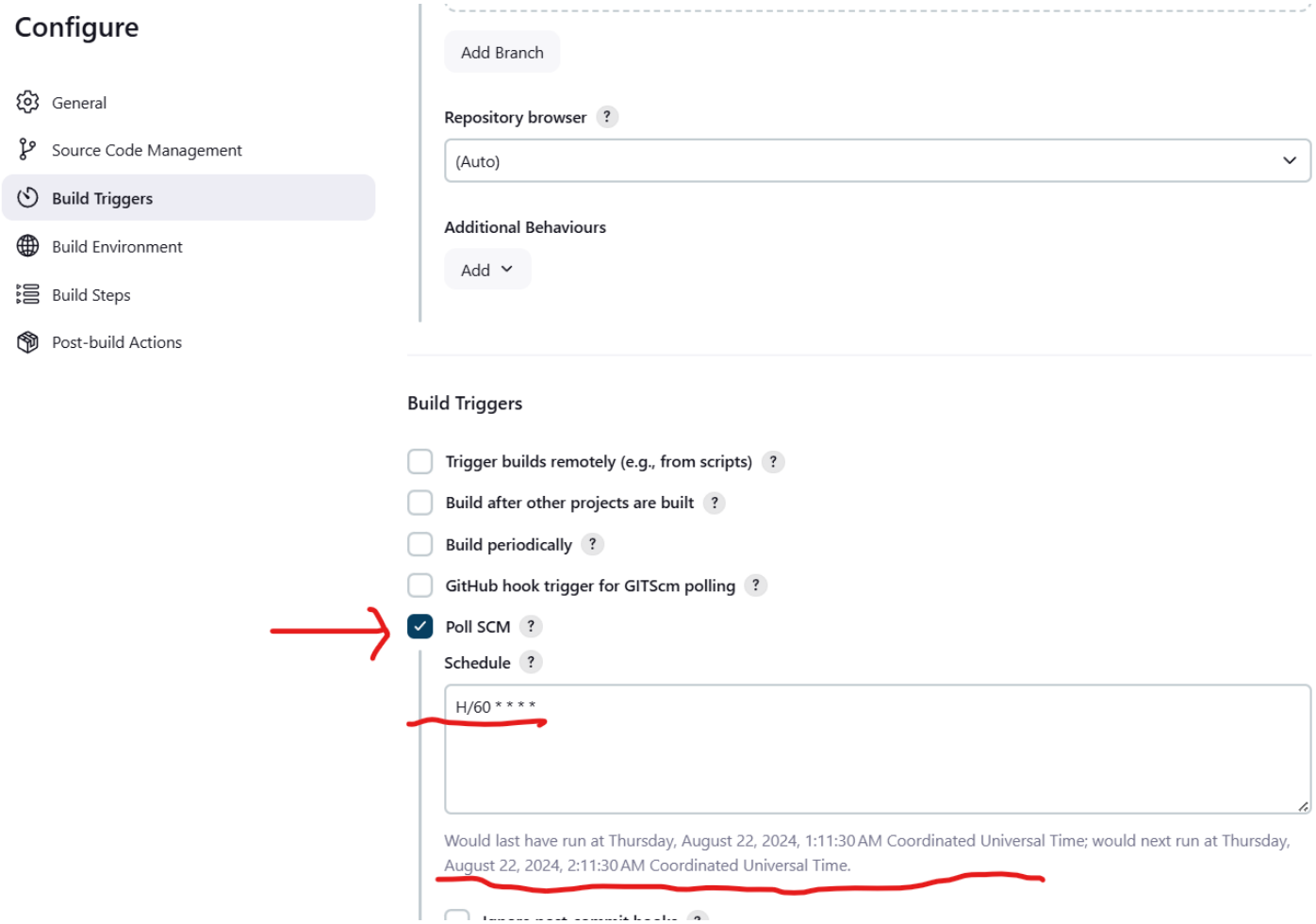
c. Username should be your github username (only one of the group members needs to do this).
d. Password should be the API key you created in github.



e. If you see this screen, (all the red text) then most likely your credentials aren't working correctly.



f. Note how the warning goes away once the correct credentials are selected.



4) Next we want to configure what triggers a "build". A build is an activation of our job.
    a. Since our Jenkins server doesn't have a public URL we can't configure it to work with Github hook Triggers. Ideally this is what you would want for a CI/CD type job. With this type of trigger, everytime some event happens in your github repo (like someone pushing new code to a specific branch), github would notify your Jenkins server and this would trigger some job to run.
    b. Since we can't do this, we will have to settle for checking periodically. The schedule uses CRON notation (automated linux task scheduler) to determine how often to run. For now, just configure it to run hourly (you can use exactly what I did above). That means that each hour, go to check the github repo, and if there were any changes, trigger the build.
    c. Once you enter a schedule, you will see text below detailing exactly how often it is going to run by way of an example.
    d. You also want to select the option to abort the job if the log file gets too big. This usually happens (in this class) with infinite loops.

Build Environment

- [ ] Delete workspace before build starts
- [ ] Use secret text(s) or file(s) ?
- [x] Abort the build if its log file size is too big

  Use job specific maximum log size instead global value (0 MB)

  - [x] Use job specific size

  Job specific size ?

  Job specific max log size in MB

  5

  - [x] Fail the build ?

  If checked, the build will be marked as failed rather than marked as aborted. (In any case, the build does not continue when the maximum file size is reached).

  (from Jenkins build log file size checker plugin)

- [ ] Add timestamps to the Console Output
- [ ] Inspect build log for published build scans
- [ ] Terminate a build if it's stuck
- [ ] With Ant ?

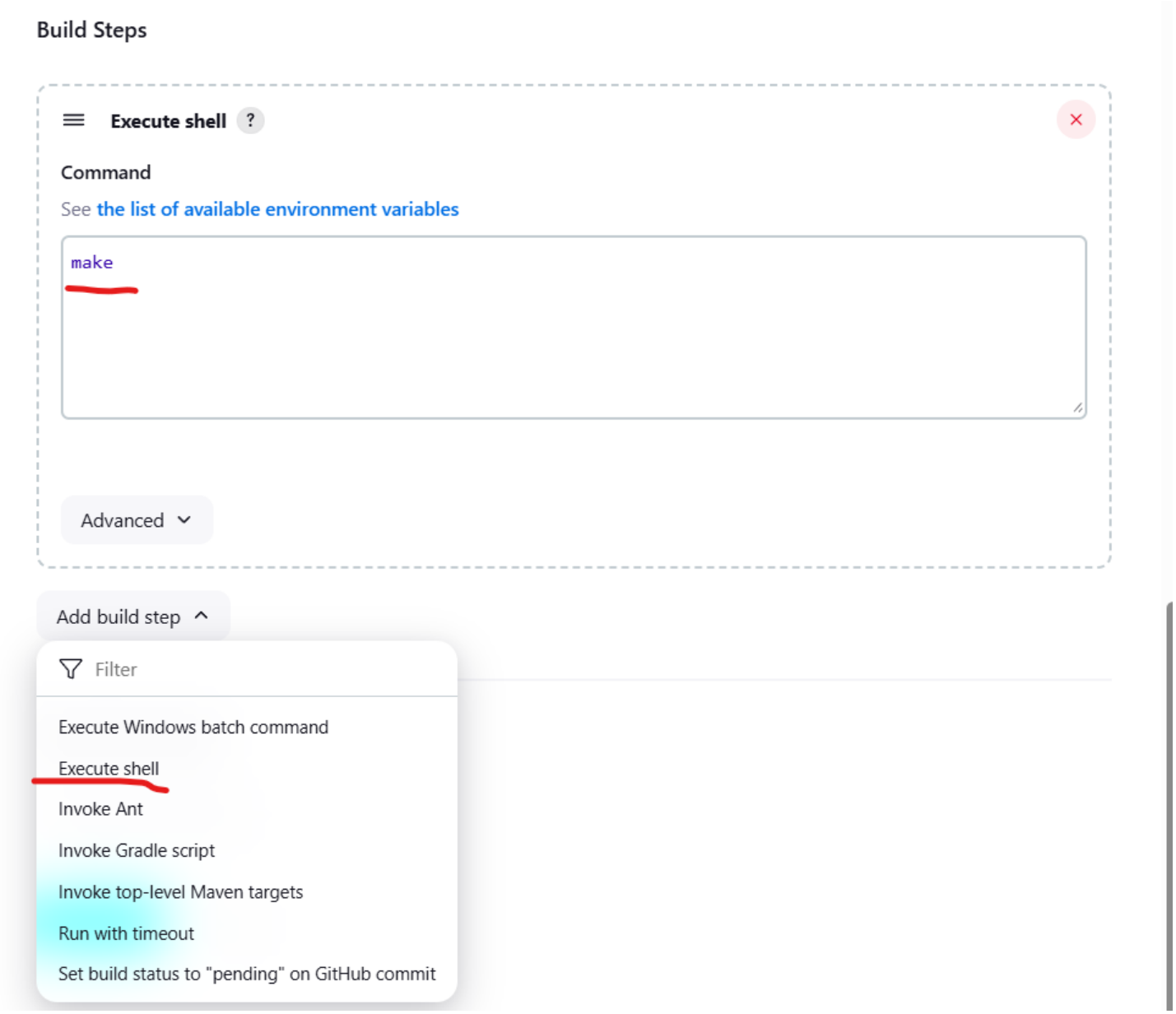
Build Steps

Add build step

Filter

Execute Windows batch command
Execute shell
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit

5) Next, we want to configure what actions the job is going to take. For now, we will simply execute shell commands. For all of your homework assignments, you should create makefiles (covered later) that will compile your code for you. We will simply execute those files, then run the executable file that was created.



Configure

- General
- Source Code Management
- Build Triggers
- Build Environment
- **Build Steps**
- Post-build Actions

Build Steps

Execute shell ?

Command

See the list of available environment variables

```
make
```

Advanced

Execute shell ?

Command

See the list of available environment variables

```
./a.out
```

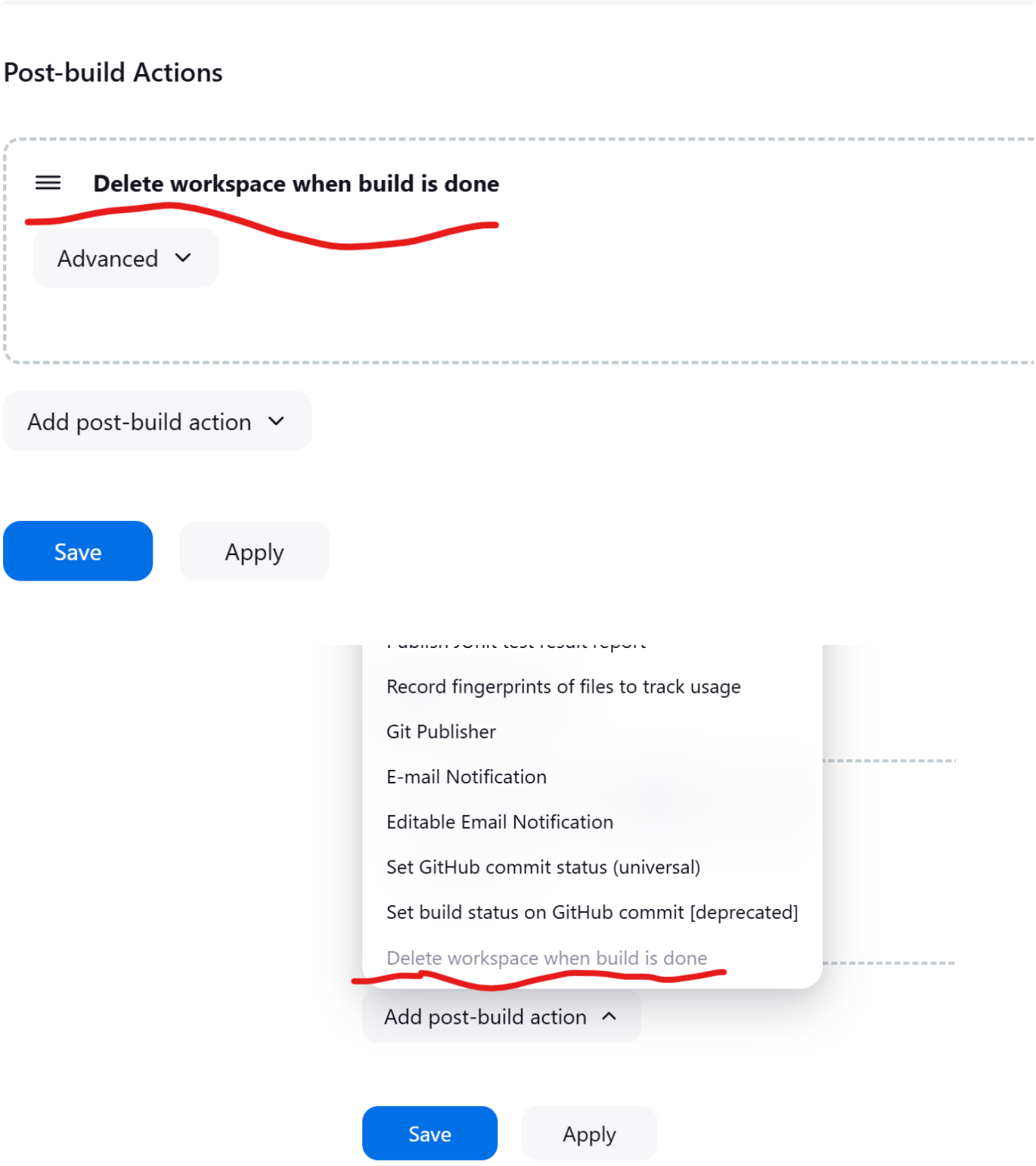
Advanced

Add build step

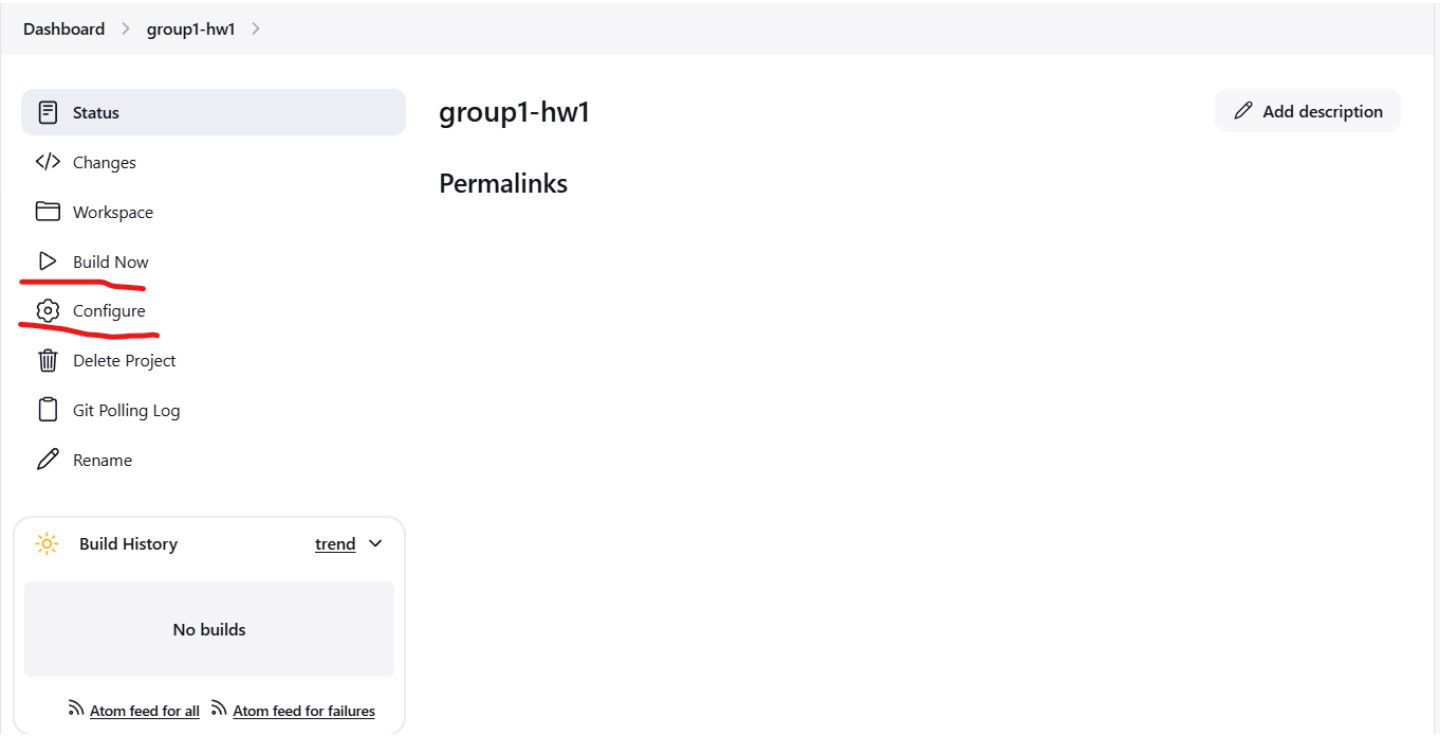
Post-build Actions

Save Apply

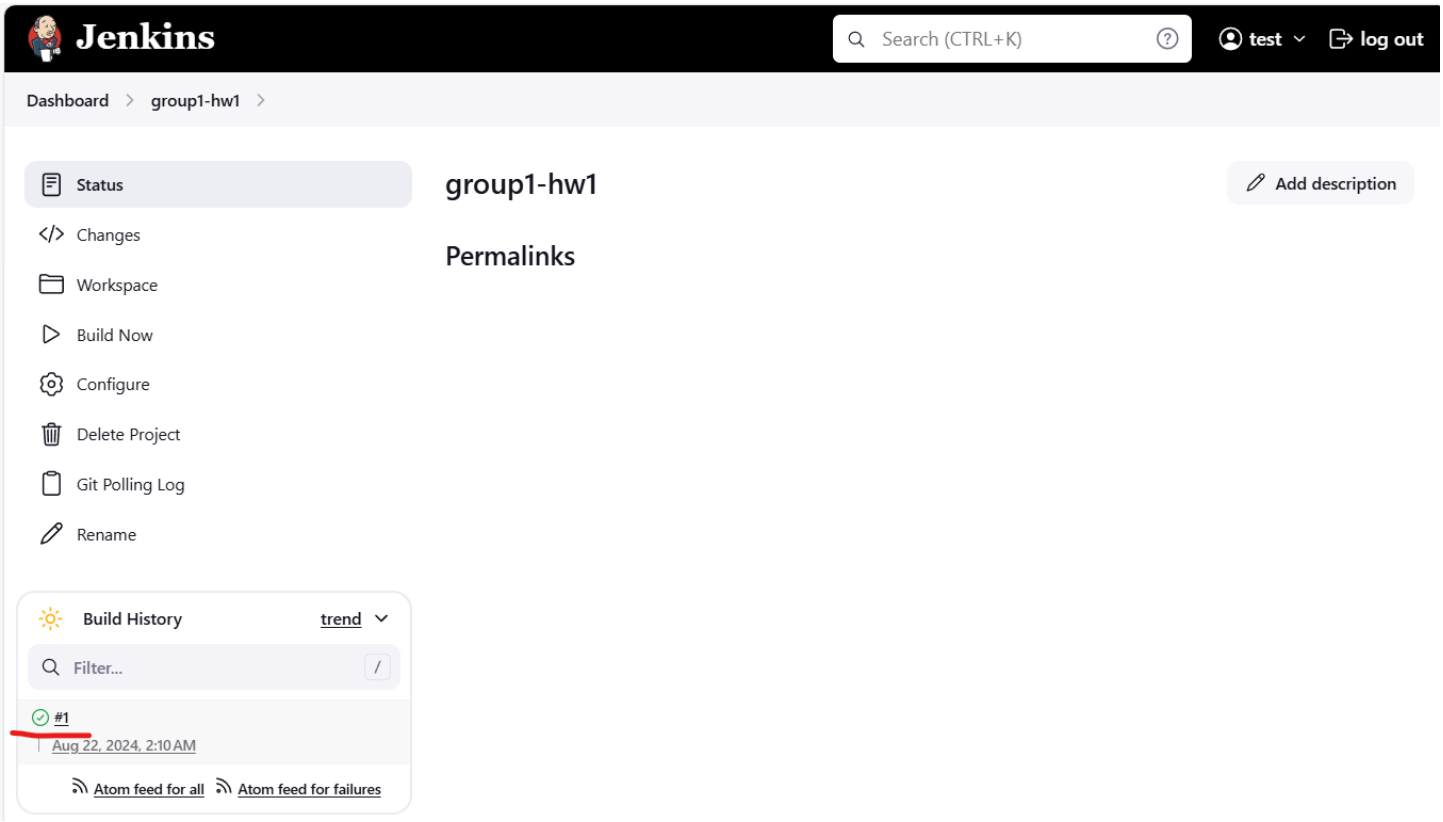
6) Finally, select the option to clean up the workspace when the build is complete. Again, this is to save disk space on the Jenkins server.



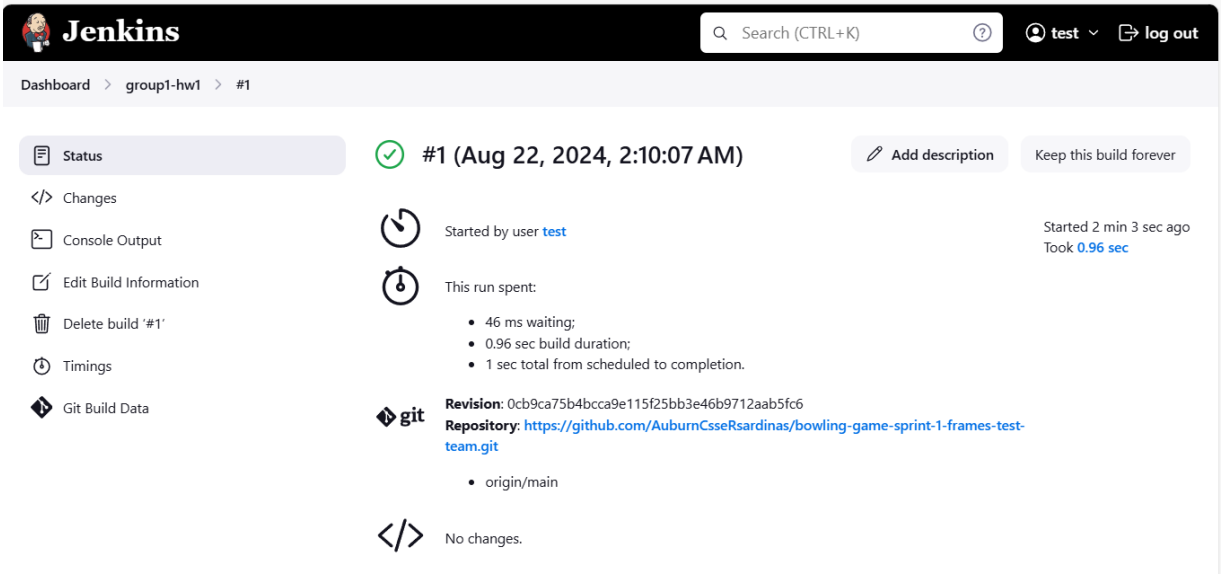
7) When you're done, you can click save to save the job.



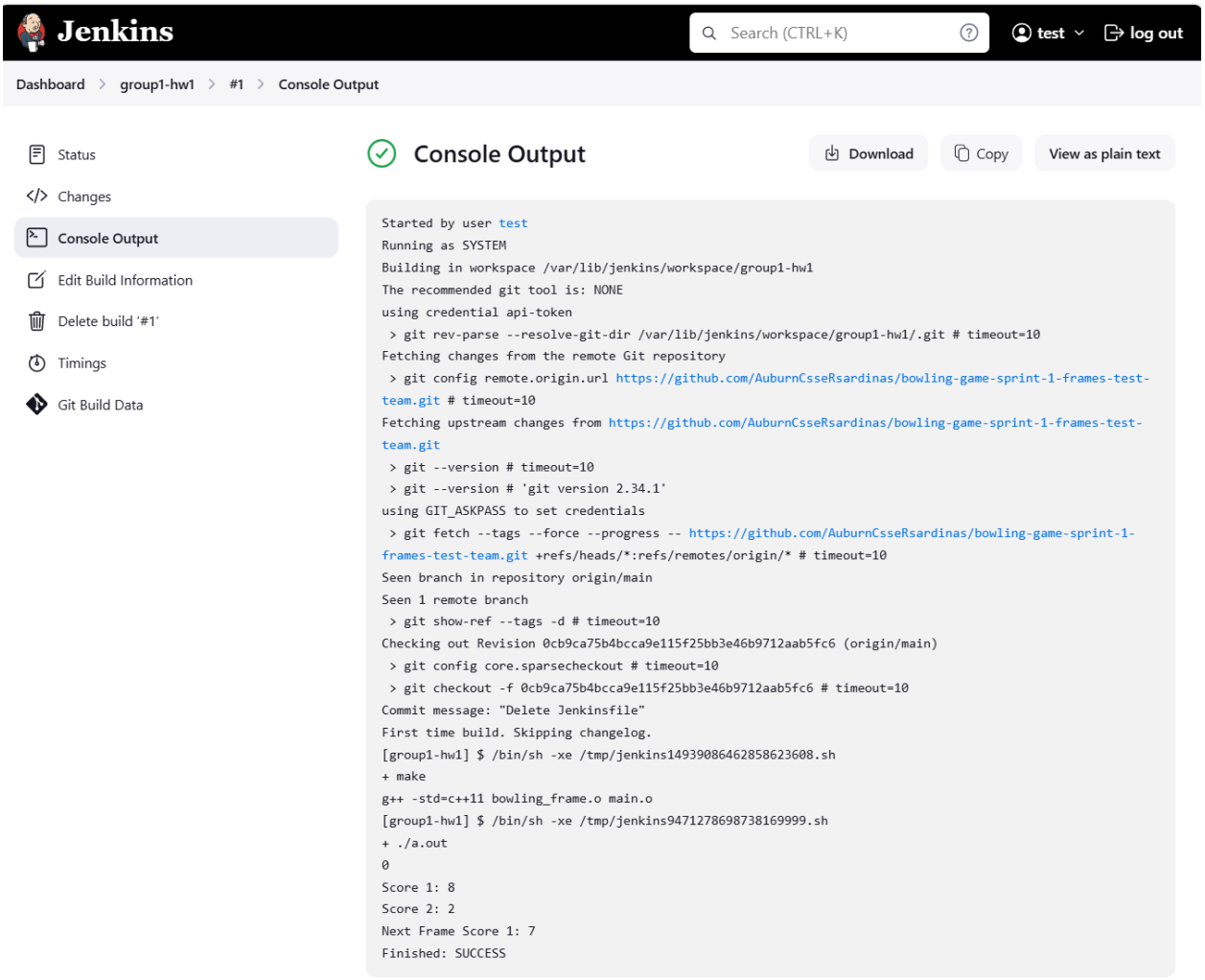
Once you save, it should take you to this screen where you can see information about the job, among other things. Note that you can manually trigger the build (job) and modify it (configure) among other things.



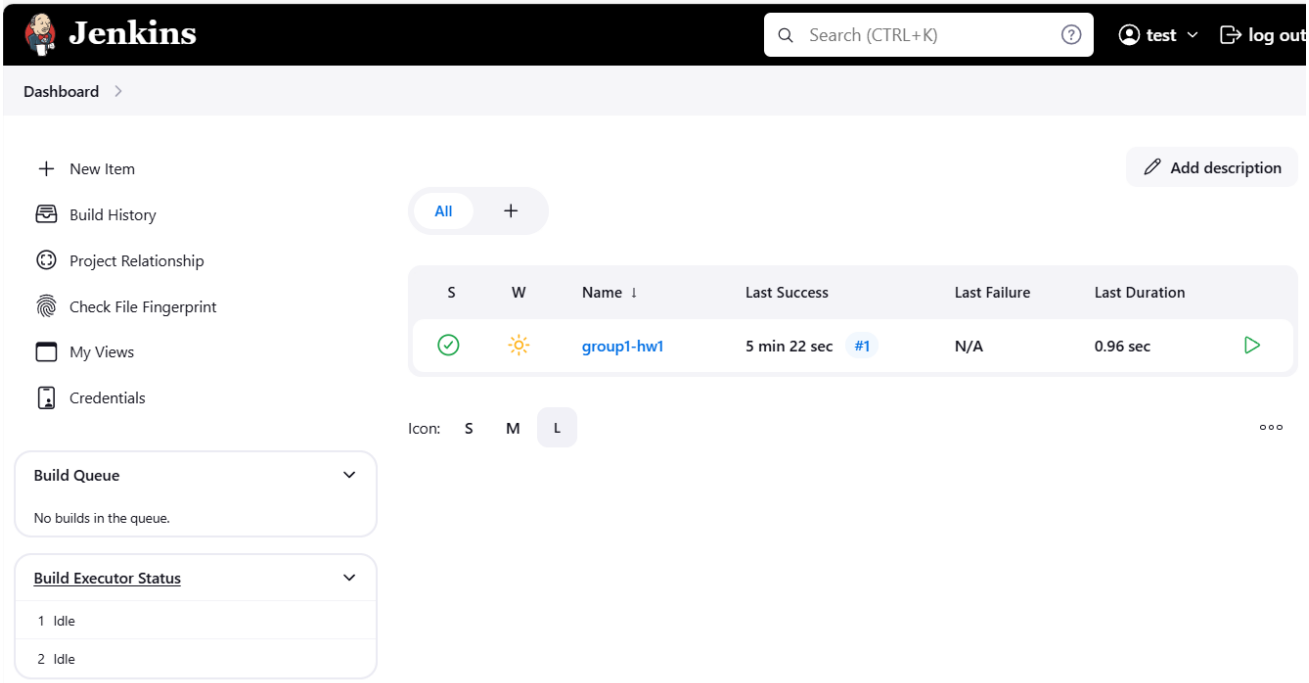
Once the build has run at least once, you will see a build history (each time the build was triggered). If no changes are made, it should not trigger again. If you make changes to your github repo, then whenever it polls your github repo, it should trigger a new build.



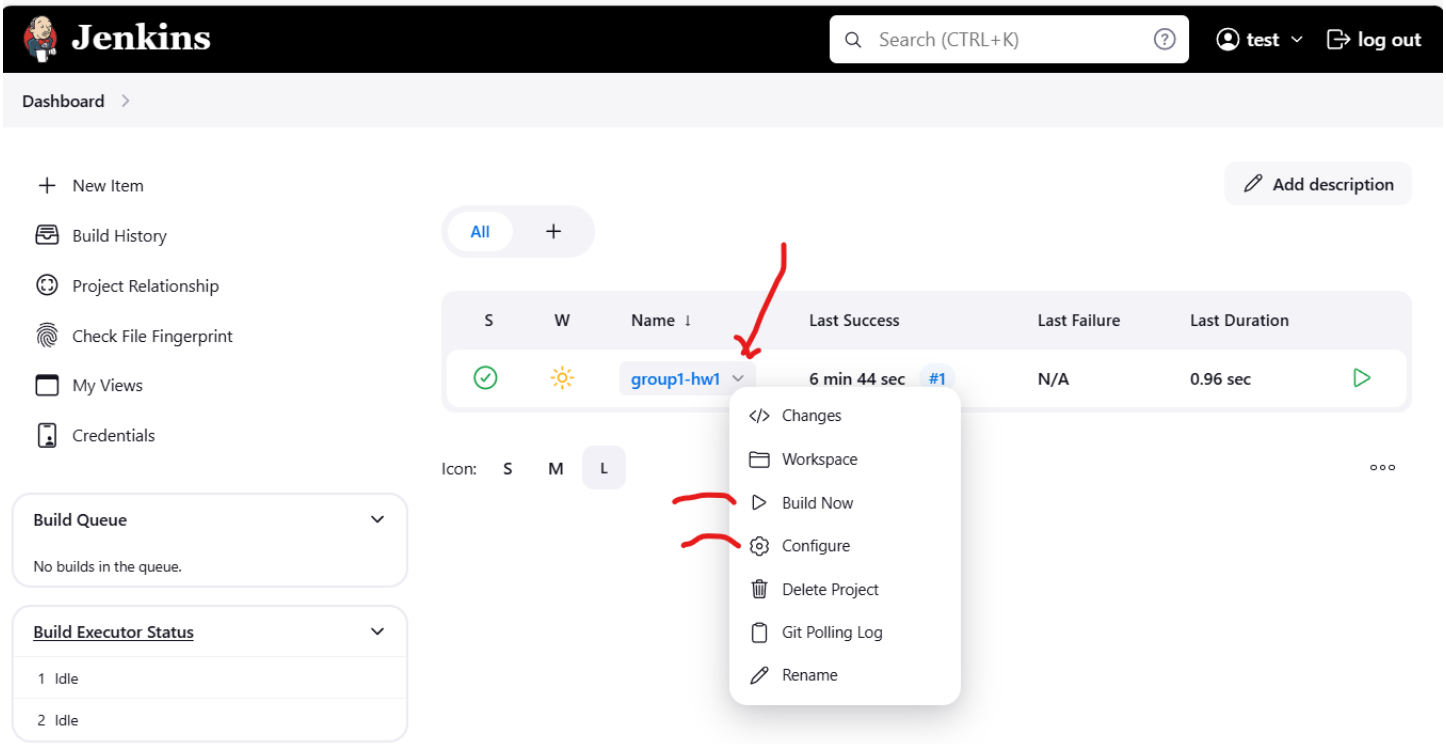
Selecting a build number will take you to a page that will show you information about that instance of the job running.



The tab we really care about will be the console output tab. Your submitted assignments should always produce output. When your executable is run, we can see that output here. Your assignments will be graded based on the output here.



If you go back to your dashboard, your default view will show you all jobs in the Jenkins server. This is one reason we want to have some kind of naming convention for jobs, to ensure that you ONLY work with or modify your jobs or even the job for the specific assignment. From here, you can select a job, rerun it, make modifications, etc...



Please be careful NOT to delete anyone else's jobs.