ATLANTIC TECHNOLOGICAL UNIVERSITY

ASSIGNMENT COVER SHEET

To Be Completed by The Student

Lecturer’s Name: Ms. Maria Griffin

Assessment Title: Continuous Integration (CI) Tool With Automated Pipelines

Submission Date: 30-Nov-2022

Student’s Name: Tony Mathew Thomas Id. Number: L00171045

Course / Stage Master’s in DevOps

Subject/Module: DevOps Software Engineering (2022/23)

Word Limit: Actual Word Count: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

I confirm that the work submitted has been produced solely through my own efforts.

Student’s signature: Tony Mathew Thomas Date: 30-Nov-2022

|  |
| --- |
| **Note**: **PENALTIES**   * The total marks available for an assessment will be reduced by 15% for work submitted up to one week late. The total marks available are reduced by 30% for work up to two weeks late. * Assessment work received more than two weeks late, without prior approval by the lecturer will receive a mark of zero. * Marks awarded will be reduced by 10 % if submitted work is greater than 10% above or below the assigned word limit. * A further hard or electronic copy of your submitted work may be requested, and therefore you must keep a copy on disc. * Incidents of alleged plagiarism and cheating are dealt with in accordance with the Institute’s Assessment Regulations   **Plagiarism:** Presenting the ideas, words of someone else without proper acknowledgement. Refer to the Institutes’ procedures and guidelines for the assessment of learners. |

# Introduction

The objective of this lab is to learn about a CI tool by configuring it to implement the Continuous Integration activities, build, test and deploy on a sample project. The tool used here for this purpose is Jenkins which is open source and freely available. There is also a comparison with another tool to highlight the benefits and limitations of each.

# Aims/Objectives

* Use Jenkins to learn about Continuous Integration
* Build a pipeline for a sample project to learn how a CI tool could be configured for building the target project
* Restrict usage of tools or framework to open-source ones than enterprise or licensed ones
* Executing Unit Tests through the built pipeline.
* Compare and contrast features available in another CI tool

# Methods

1. Write a docker compose file to setup Jenkins in a docker container (Panu, 2022) in cluster.
2. Setup docker and ensure it exposes the port 8080 to the outside and ensure its now browsable locally (<http://localhost:8080>)
3. Add the following plugins:

|  |  |
| --- | --- |
| **Plugin** | **Purpose** |
| .NET SDK Support | Required for building sample .Net project |
| Git plugin | Required for pulling source code from GitHub |
| MSBuild Plugin | Required for building sample .Net project |
| MSTestRunner plugin | Required for running Unit Tests |

Figure 1 shows these plugins added in.

1. Create new pipeline (Figure 3) and add tasks for:
   1. Pulling source code from GitHub (Figure 4)
   2. Poll source code repository for changes (Figure 5)
   3. Add build step for building the project (Figure 6)
   4. Add step for executing unit tests (Figure 7)
2. Try out a manual build and ensure it is working and the unit tests are executed and passed.
3. Try a code commit and check in to ensure it is detected and automatically trigger a build.

# Results & Analysis

* Setting up Jenkins in local docker was easy following Jenkins documentation (Jenkins, 2022). Instead of a command, a docker compose file was written with the relevant ports exposed, volumes and networks set up.
* Once running the docker compose file, the user interface of Jenkins is available at <http://localhost:8080> browsable locally.
* The password to login to Jenkins was found in file ../data/secrets/initialAdminPassword in the docker volume created for Jenkins home path.
* Searching for a plugin and installing it were easy, although the restarting of the containers might not always go smoothly. The compose file need to be ran again to bring the cluster back up again.
* There was a bit of configuration to be done for the .Net Sdk plugin in the Global Tool Configuration of Jenkins where the platform and .Net version are set. This is shown in Figure 2.
* First few attempts to build the project failed as one of the libraries had to be installed. For this, had to bash into the container and install the library ‘libicu-dev’ as explained in the article followed (Panu, 2022).
* Once this library issue was resolved, the builds started going through successfully.
* The unit test too could be seen getting executed and passing on each build.
* A summarised view of the overall builds could be seen in the Dashboard (Jenkins, 2022) view of Jenkins where the following details are listed below
  + Status of the last build: whether the last build was one of the following status:
    - In Progress
    - Passed
    - Unstable
    - Failed
    - Aborted
  + Aggregated status: this shows the aggregated status of the last five builds in a weather report format (Jenkins, 2022). These are the list of statuses:
    - Sunny: more than 80% of Runs passing
    - Partially Sunny: 61% to 80% of Runs passing
    - Cloudy: 41% to 60% of Runs passing
    - Raining: 21% to 40% of Runs passing
    - Storm: less than 21% of Runs passing
  + Name of the project that is built
  + Last Success: Date and time of the last run that has passed
  + Last Failure: Date and time of the last run that has failed
  + Last Duration: total time elapsed for the last build

Figure 8 shows how the Dashboard with the status of the sample project.

* There is also a build history view which could be seen by clicking on the project name of the list which, shows the past builds and their status. Figure 9 shows the build history of the sample project.
* Once a build is done there is also a console view that we could look into for detailed information of the run. This is helpful for troubleshooting when we are facing issues with the build. The details of initial error encountered in the first few builds could be seen as shown in Figure 12
* Similar to the failed builds, a success build too will have its logged which could be examined as shown in Figure 10.
* The status of the unit tests execution can also be seen in Console Output view where it lists the number of tests passed or failed. This is shown in Figure 11
* Could not get the Azure App Service plugin (Jenkins, 2022) working to deploy the app into an Azure App Service. It is not listing resource groups although the credentials were set in the Credential Manager and is verified successfully. The plugin comes with a warning that its is lacking update and is up for adoption.

# Conclusion

The Jenkins installation done for this lab was setup in a docker container by writing a dockercompose.yml file which is free of any cost. Jenkins, being open source and freely available in contrast to other CI tools is encouraging for anyone to try their hands on. In addition to the managed service offerings which are available in various cloud platforms like Azure and AWS, Jenkins gives the flexibility of hosting it on our own premises. Having a docker image means it can be hosted in a container in a local docker cluster as well as in an orchestrated container service like Azure Kubernetes Service (AKS) or Elastic Container Service in AWS. The components needed to perform each task are available as plugins which can be configured and added into the pipeline which is straightforward to understand. These plugins are written by third parties, the .Net Sdk or the MSBuild used here for instance is written by Microsoft. The Plugin Manager has a wide variety of plugins made available by other companies to integrate with Jenkins pipelines.

Another feature is the Manage Credentials in Jenkins which provides the ability to store credentials. For a task like GitHub source control management, if it setup to be authenticated to connect, the plugin has the ability read from the credential store. The plugin’s task would list the credentials stored in the Credential store in a drop down from which we could select the right credential. The kinds of credentials supported too are configurable. This will depend on the plugins that we have installed. Some of the credential available by default are:

* SSH Username with private key
* Secret file
* Secret text
* X.509 Client Certificate

Besides these there will be ones that gets added through plugins. For instance, Azure Web App plugin will bring in Azure Key Vault Secret, Azure Managed Identity and Azure Service Principal.

The Output console provides a detailed level of logging which is sufficient for troubleshooting any issues and provides some level of statistics for instance the unit tests success and failures. Issue encountered during the initial build was further looked into the console as shown in Figure 12 to understand the cause and resolve. Although the steps provided by Jenkins are plugins, there are options to write and execute code. “Execute Windows Batch command” and “Execute Shell” are two of the available steps available by default where we could write executable scripts as tasks. This will be helpful if we need to do any custom tasks which may not be made available through any plugin.

Jenkins is among the very few CI tools that offer a docker image and on premise setup in a container. Azure DevOps which I am familiar with and worked with has its own CI/CD system having us to write pipelines has many similarities with Jenkins. Both of them has their own provision for adding third party plugins. Like the Jenkins’ Plugin Manager, Azure DevOps has its Market Place where 3 parties can supply their extensions that could be added into the pipelines. The console output that Jenkins provides us with also is comparable to the logging view provided by Azure DevOps. Although, Jenkins is free and open source, it has the power enough to be comparable with any other enterprise level CI tools available like the Azure DevOps. One downside noted while using Jenkins is the warning shown for many of the plugins saying “This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.” This means there is limited support from contributors of the plugins and the plugins are not receiving updates as their platforms are getting updated. The Azure App Service plugin failed to fetch resource groups in the subscription specified in Credential Manger. This is probably because of the obsolete plugin as discussed in certain forums as this one here: <https://github.com/MicrosoftDocs/azure-docs/issues/42402>. Another limitation noted is the easiness in writing a yaml script. Jenkins is configurable through the plugins whereas most of the CI tools like Azure DevOps and CircleCI use yaml scripts as their primary way of scripting pipelines. Azure DevOps initially had these plugin style of configuring a pipeline but they have eventually moved to the newer style of writing a pipeline completely in yaml. But the future looks promising for Jenkins as there is already a plugin (Jenkins, 2022) being made available though it is right now in incubation stage which will allow us to write a pipeline fully in yaml.

From my personal experience, even though Jenkins has certain limitations and is plugin oriented compared to the fully scripted styles employed by enterprise level CI tools like Azure DevOps and CircleCI, it should be encouraged since its open source and freely available to anyone. Using Jenkins has no limitation and does not require any licensing. CI tools like CircleCI and Azure DevOps may be ahead with respect to third party contributions and frequent updates to their offerings on paid platforms. But Jenkins prove to be more versatile as it can be setup on premise as well as in cloud platforms. Moreover, it is available free and open source.

# References

Jenkins, 2022. *Azure App Service.* [Online]   
Available at: https://plugins.jenkins.io/azure-app-service/  
[Accessed 30 11 2022].

Jenkins, 2022. *Dashboard.* [Online]   
Available at: https://www.jenkins.io/doc/book/blueocean/dashboard/  
[Accessed 30 11 2022].

Jenkins, 2022. *Docker.* [Online]   
Available at: https://www.jenkins.io/doc/book/installing/docker/  
[Accessed 30 11 2022].

Jenkins, 2022. *Pipeline As YAML (Incubated).* [Online]   
Available at: https://plugins.jenkins.io/pipeline-as-yaml/  
[Accessed 28 11 2022].

Panu, N., 2022. *Building an ASP.Net 6 Application on Docker based Jenkins.* [Online]   
Available at: https://medium.com/@nestor.panu/building-an-asp-net-6-application-on-docker-based-jenkins-d6c8aaf94756  
[Accessed 30 11 2022].

# Appendix

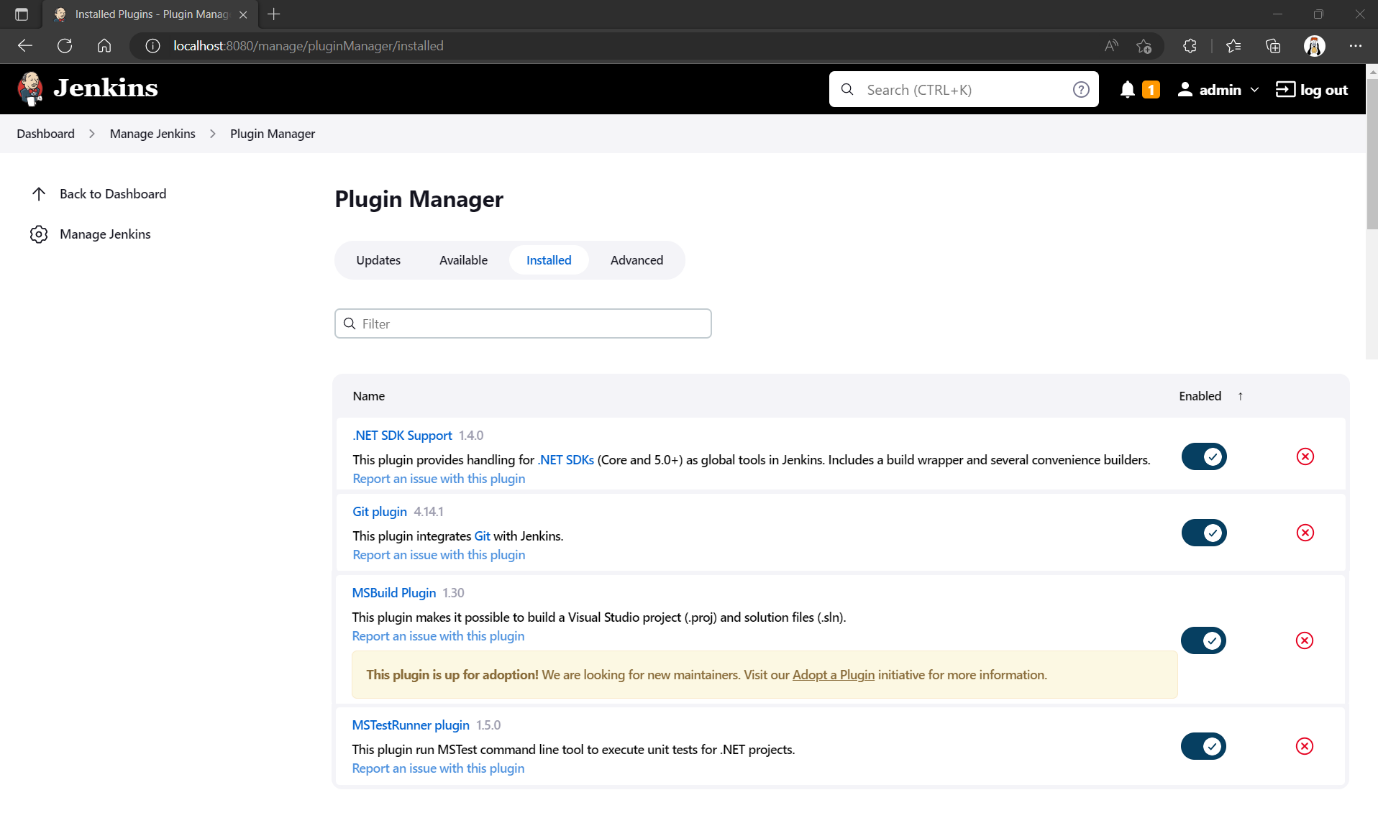


Figure . Add additional plugins

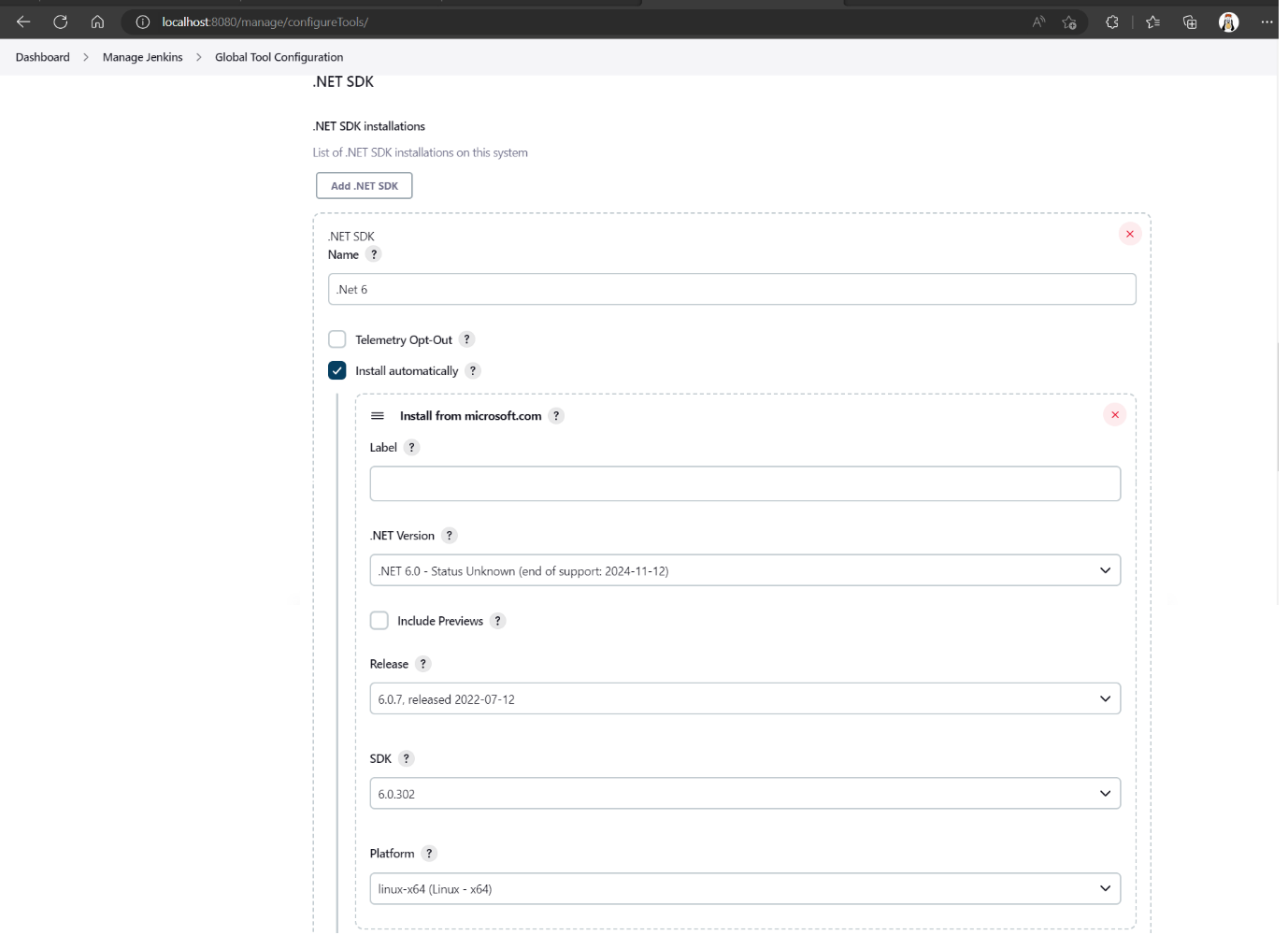


Figure . Configure .Net Sdk

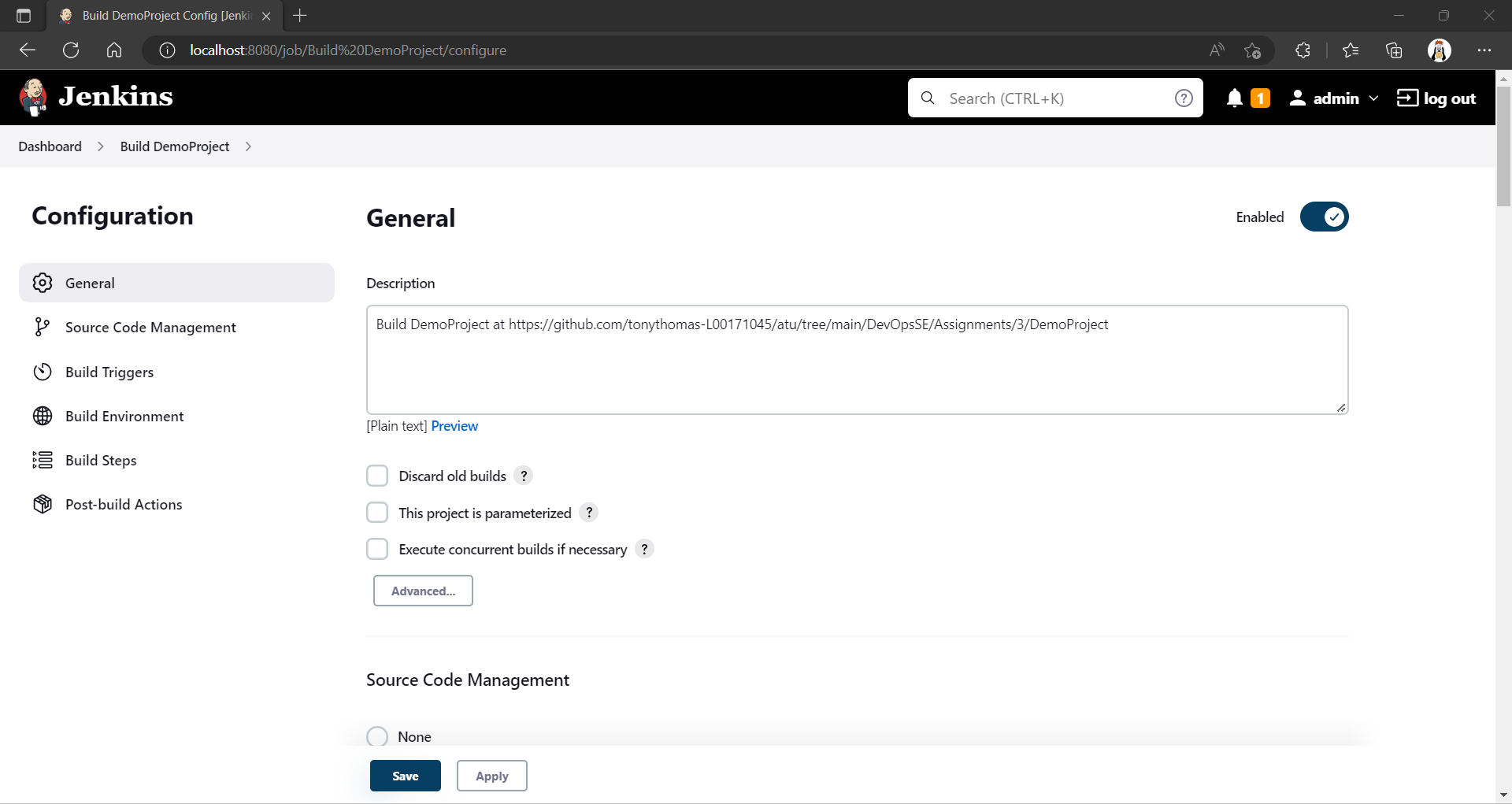


Figure . Create new build pipeline

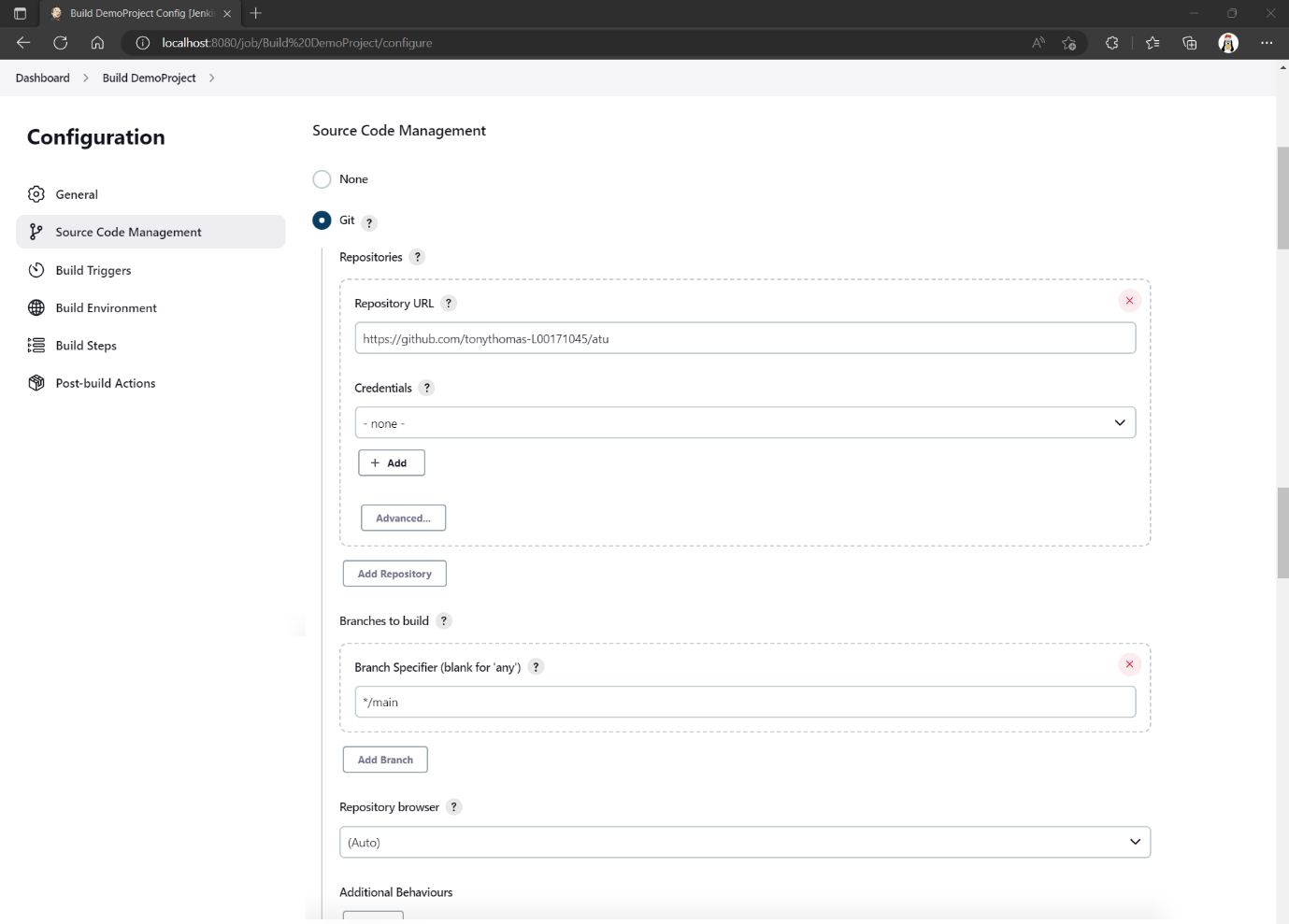


Figure . Configure source code repository

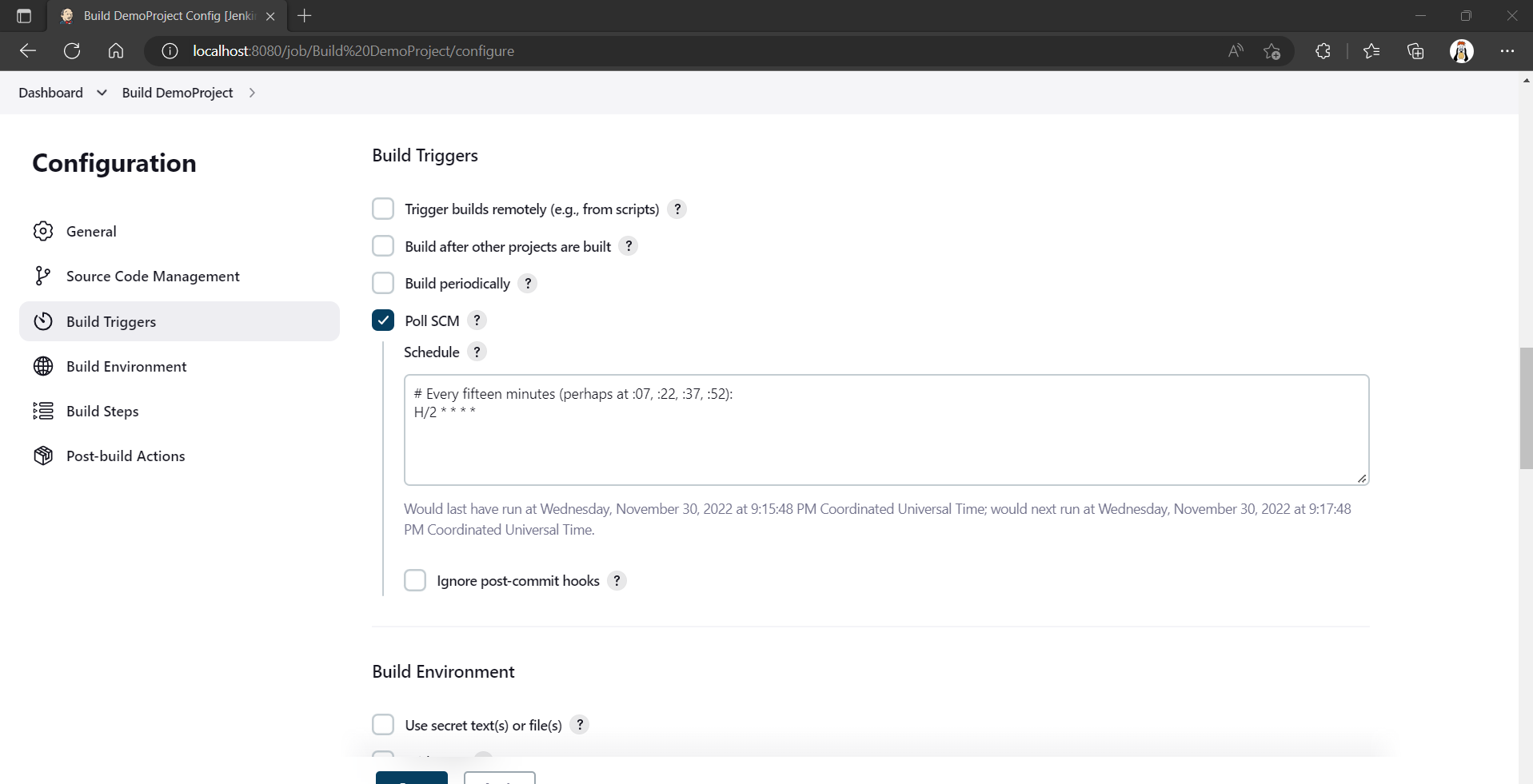


Figure . Poll source control repository (every minute)

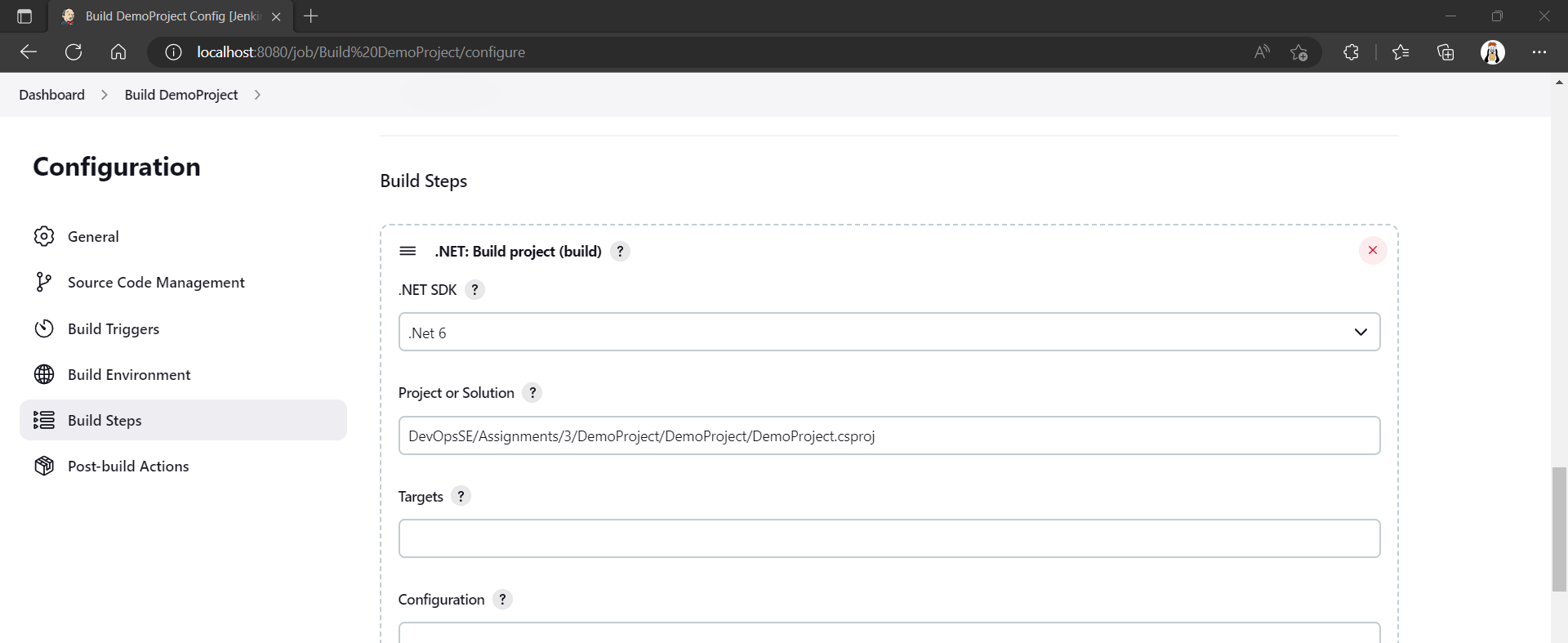


Figure . Build step for project

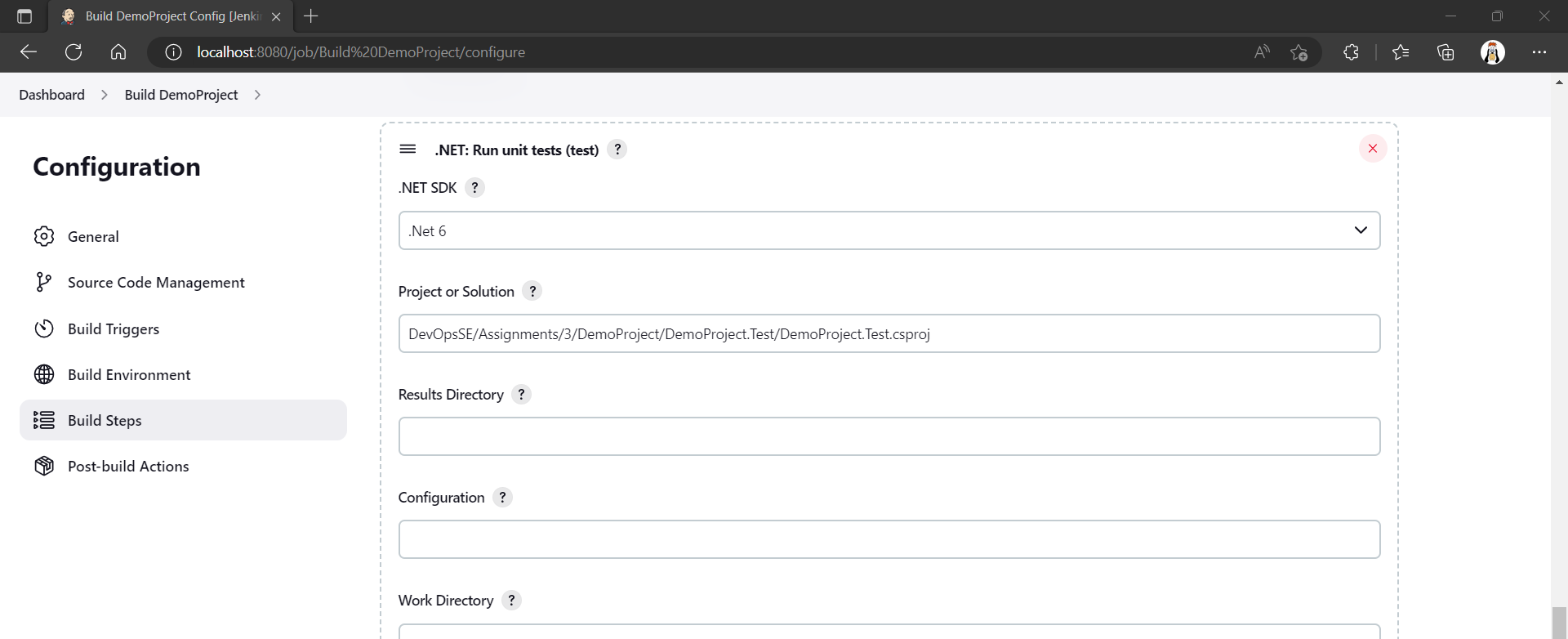


Figure . Unit Test project

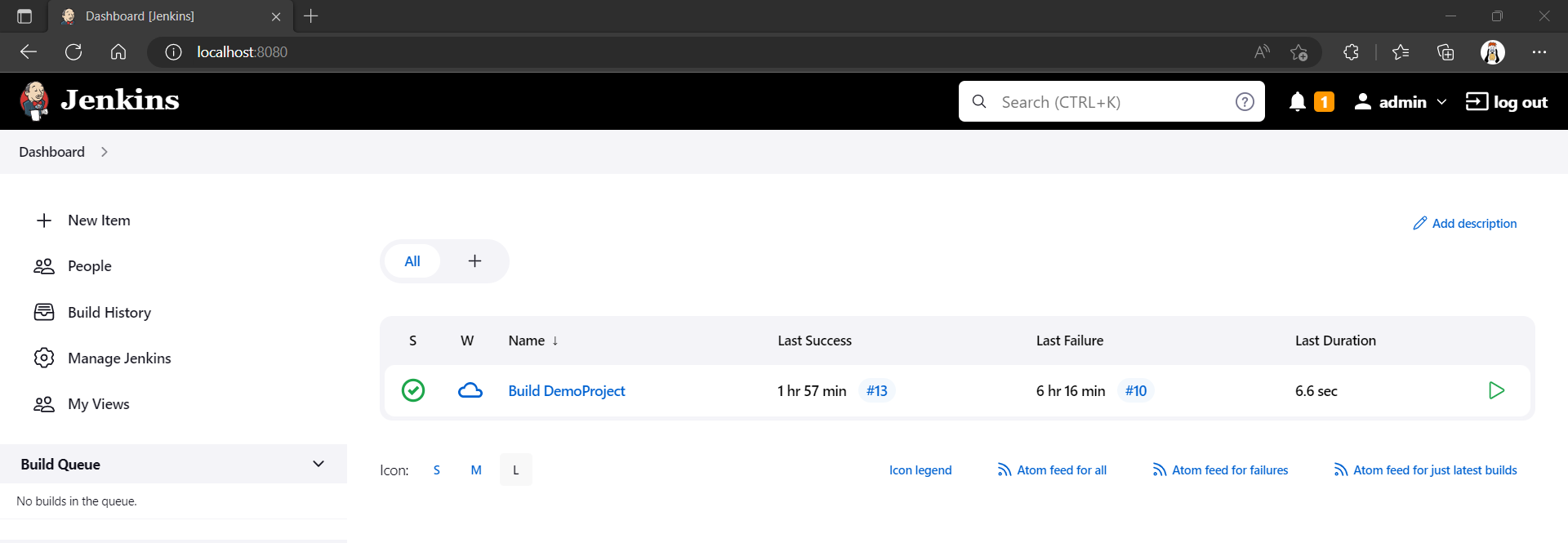


Figure . Dashboard showing build definition after several builds

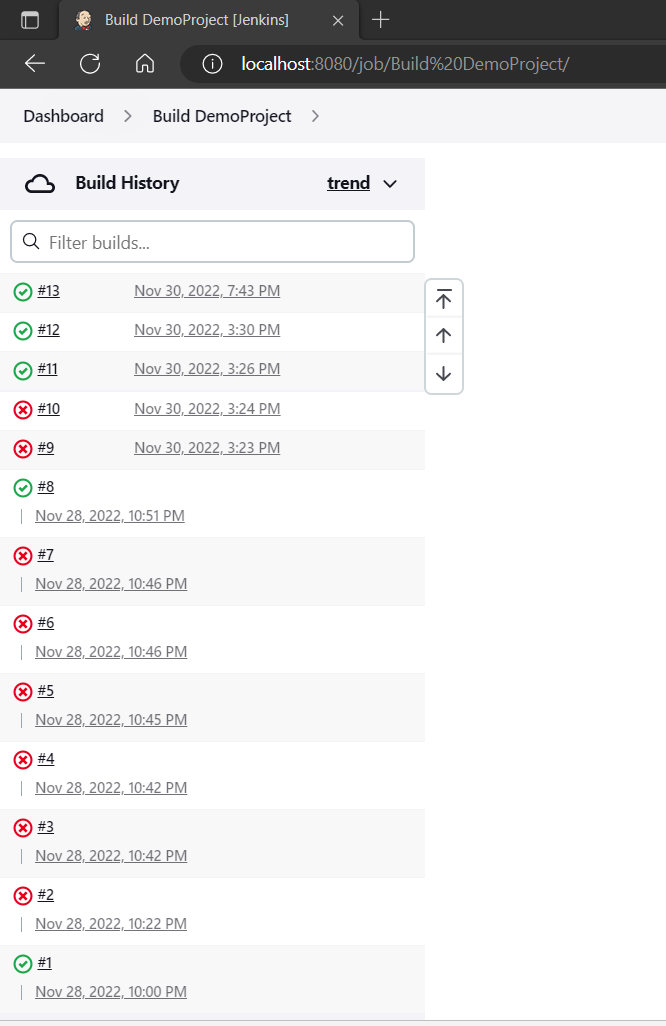


Figure . Build history

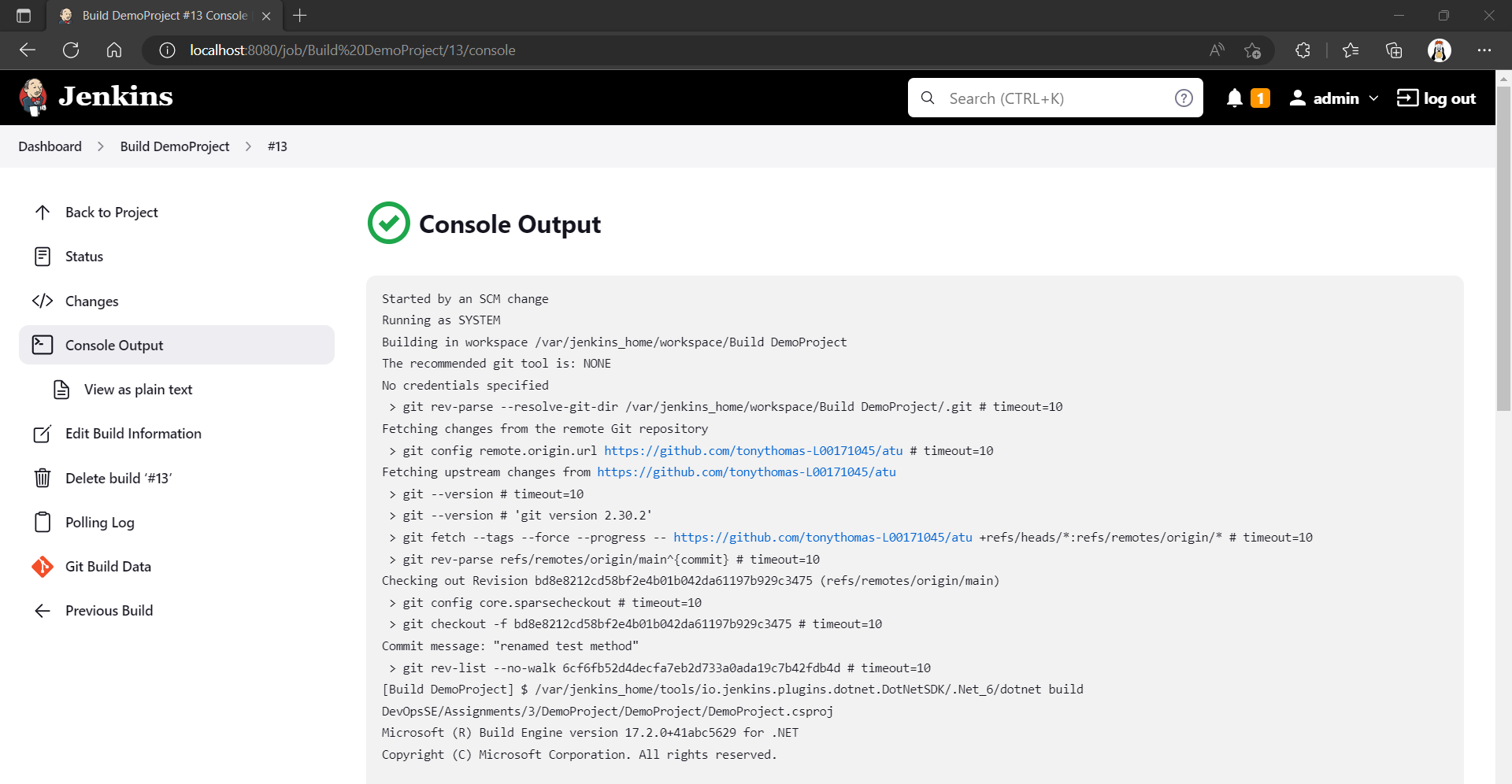


Figure . Console output of build

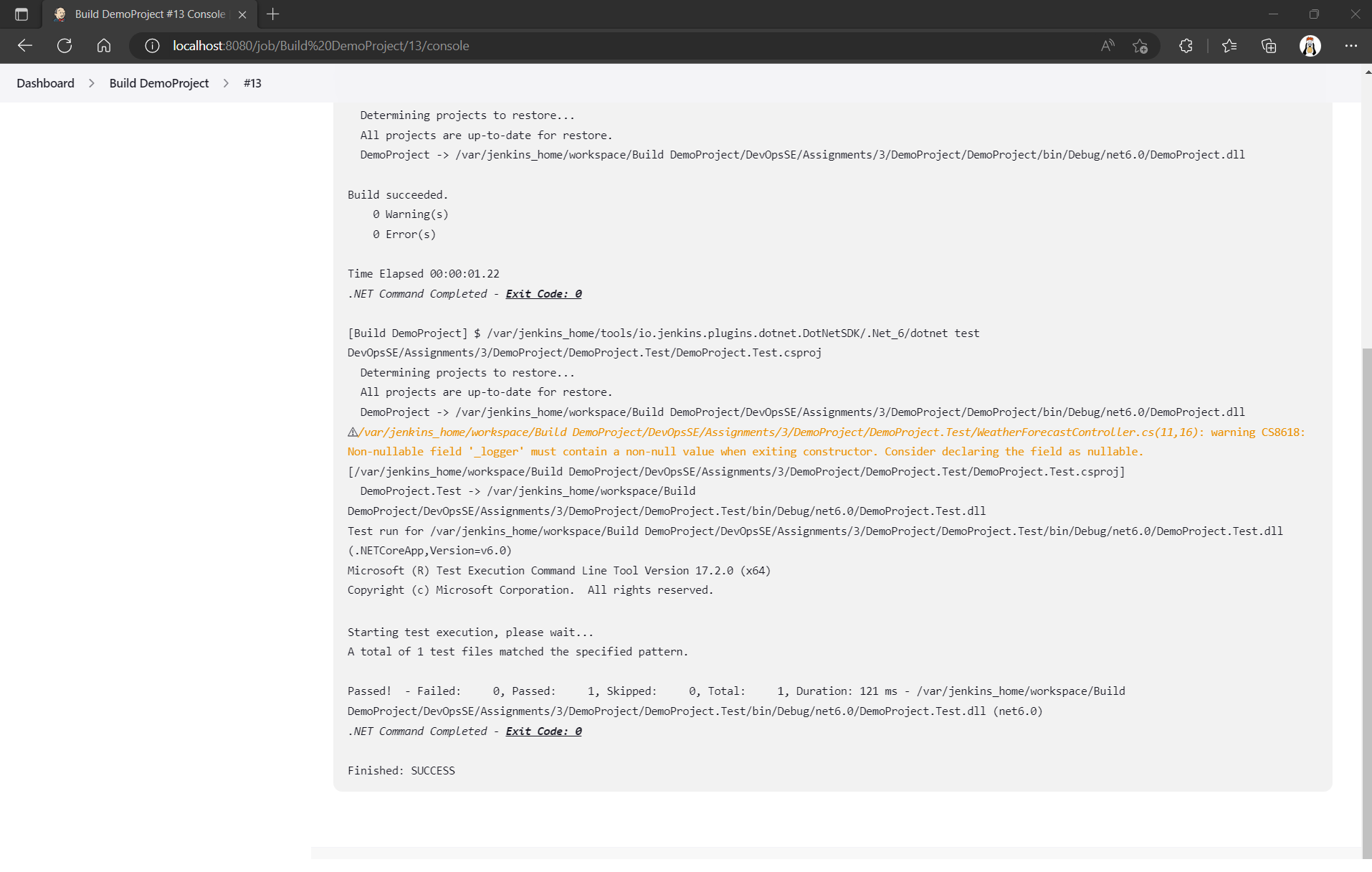


Figure . Console output of build (continued) showing unit test executions

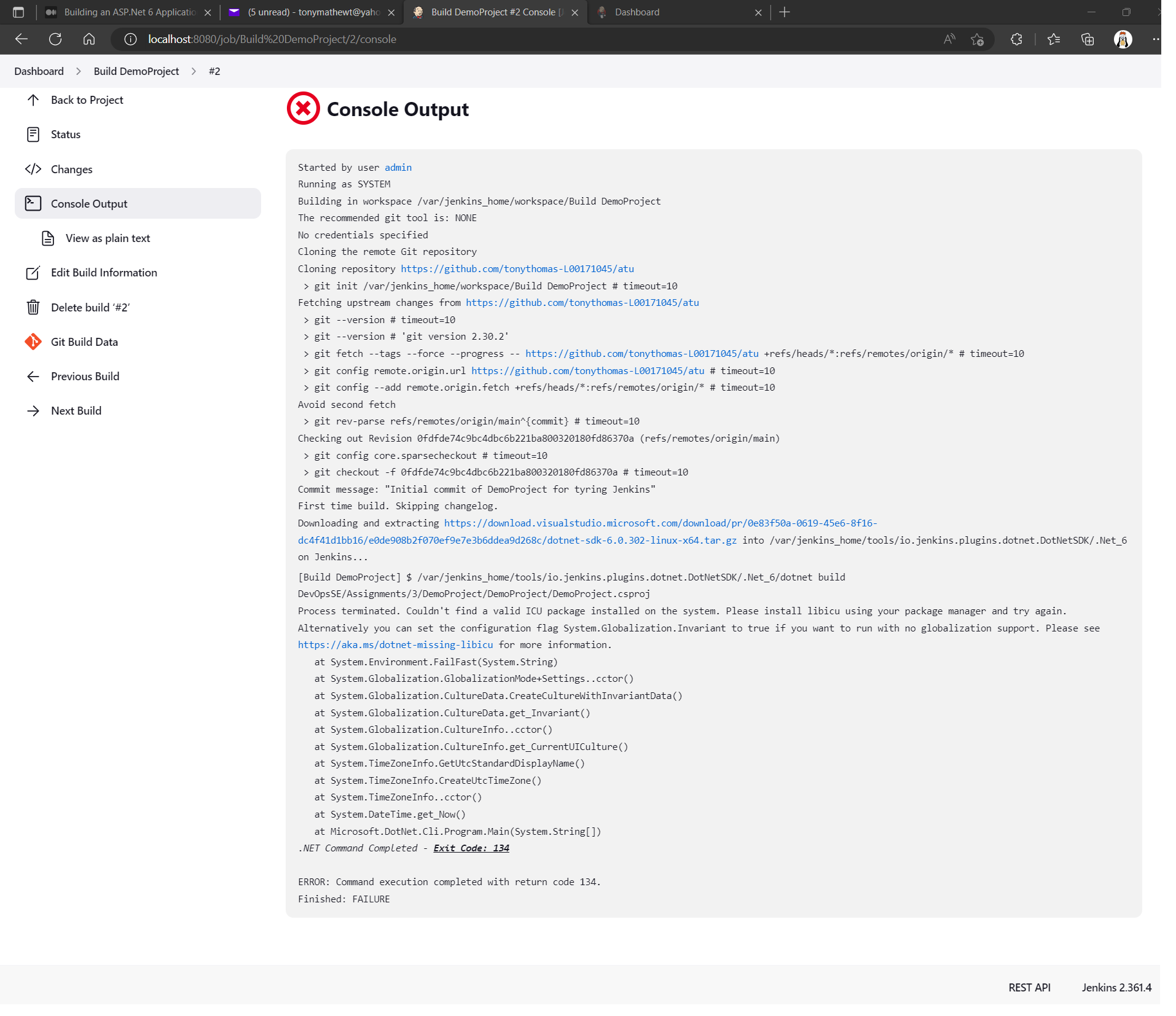


Figure . Console output of a failed build