**Aim**:

To understand Jenkins Master-Slave Architecture and scale your Jenkins standalone implementation by implementing slave nodes.

**Introduction**:

Jenkins offers a simple way to set up a continuous integration or continuous delivery (CI/CD) environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks. While Jenkins doesn’t eliminate the need to create scripts for individual steps, it does give you a faster and more robust way to integrate your entire chain of build, test, and deployment tools than you can easily build yourself.

“Don’t break the nightly build!” is a cardinal rule in software development shops that post a freshly built daily product version every morning for their testers. Before Jenkins, the best a developer could do to avoid breaking the nightly build was to build and test carefully and successfully on a local machine before committing the code. But that meant testing one’s changes in isolation, without everyone else’s daily commits. There was no firm guarantee that the nightly build would survive one’s commitment.

Jenkins automation

Today Jenkins is the leading open-source automation server with some 1,600 plug-ins to support the automation of all kinds of development tasks. The problem Kawaguchi was originally trying to solve, continuous integration and continuous delivery of Java code (i.e. building projects, running tests, doing static code analysis, and deploying) is only one of many processes that people automate with Jenkins. Those 1,600 plug-ins span five areas: platforms, UI, administration, source code management, and, most frequently, build management.

How Jenkins works

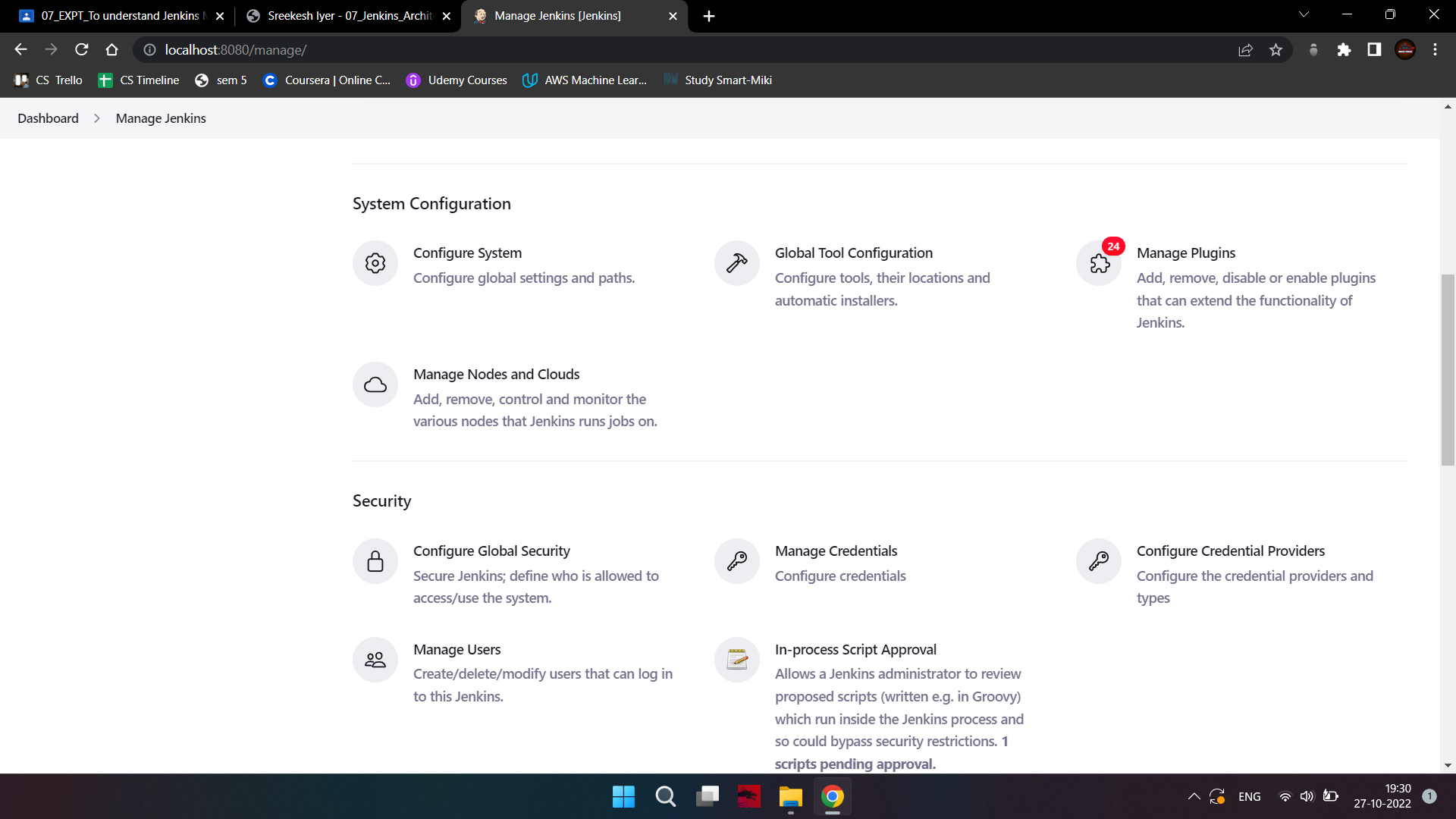
Jenkins is distributed as a WAR archive and as installer packages for the major operating systems, as a Homebrew package, as a Docker image, and as source code. The source code is mostly Java, with a few Groovy, Ruby, and Antlr files.

You can run the Jenkins WAR standalone or as a servlet in a Java application server such as Tomcat. In either case, it produces a web user interface and accepts calls to its REST API.

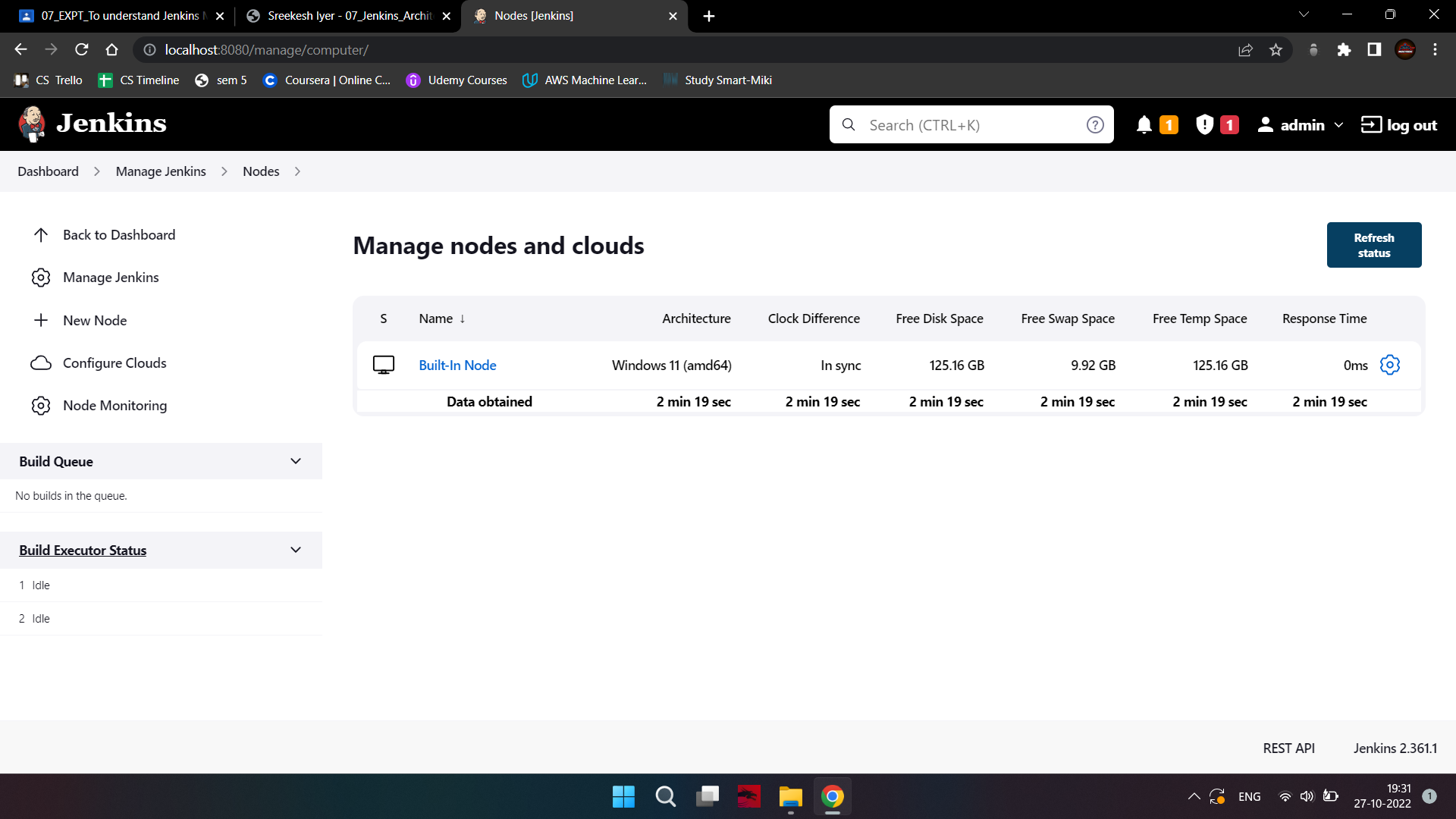
When you run Jenkins for the first time, it creates an administrative user with a long random password, which you can paste into its initial webpage to unlock the installation.

**Steps:**

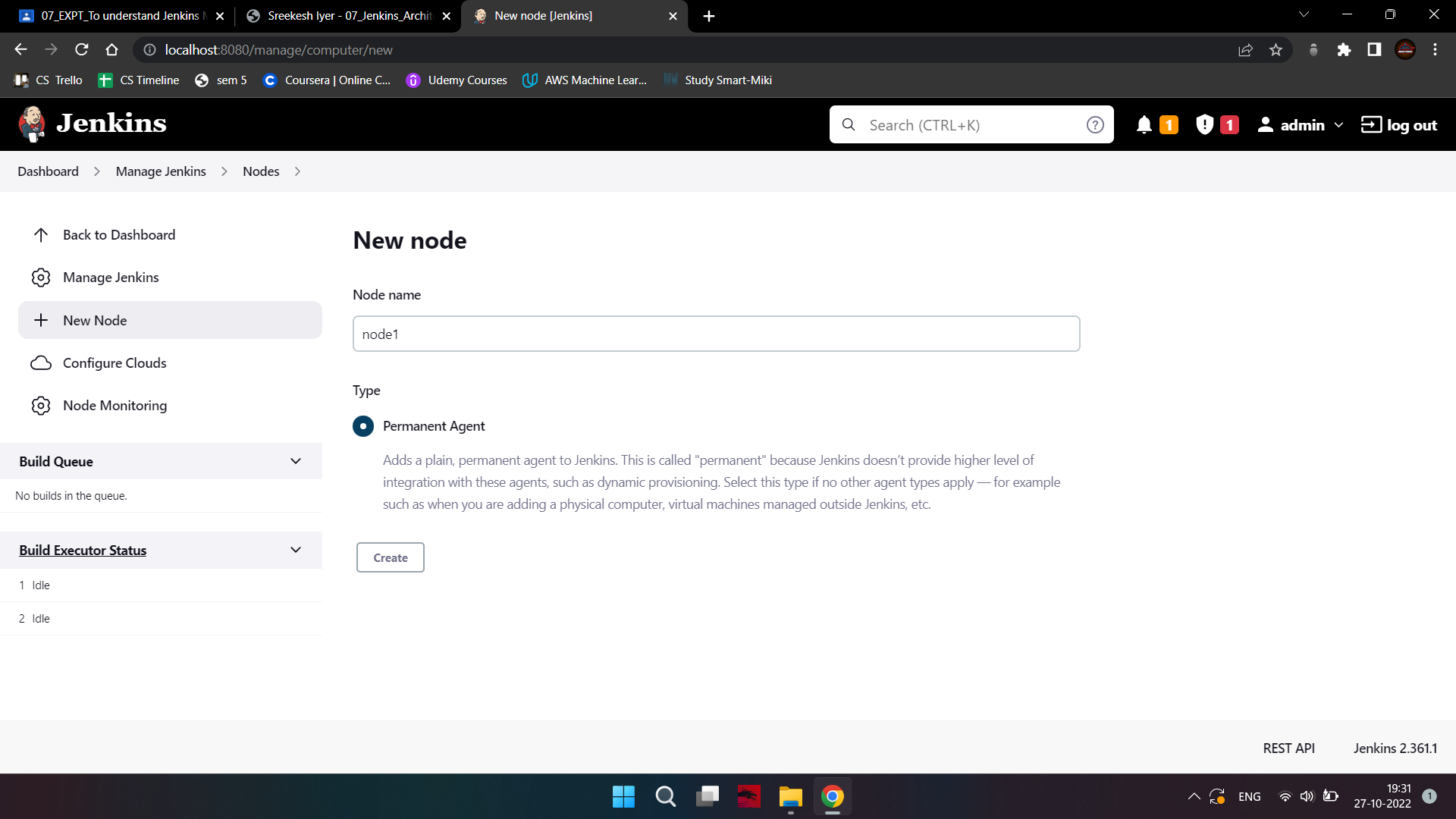
1. Go to the Manage Jenkins section and scroll down to the section of Manage Nodes.



1. Click on New Node.



1. Give a name for the node, choose the Permanent Agent option, and click on Ok.



1. Enter the details of the node slave machine.

1. Here no. of executors is nothing but no. of jobs that this slave can run parallelly. Here we have kept it to 2.

2. The Labels for which the name is entered as "Slave1" is what can be used to configure

jobs to use this slave machine.

3. Select Usage to Use this node as much as possible.

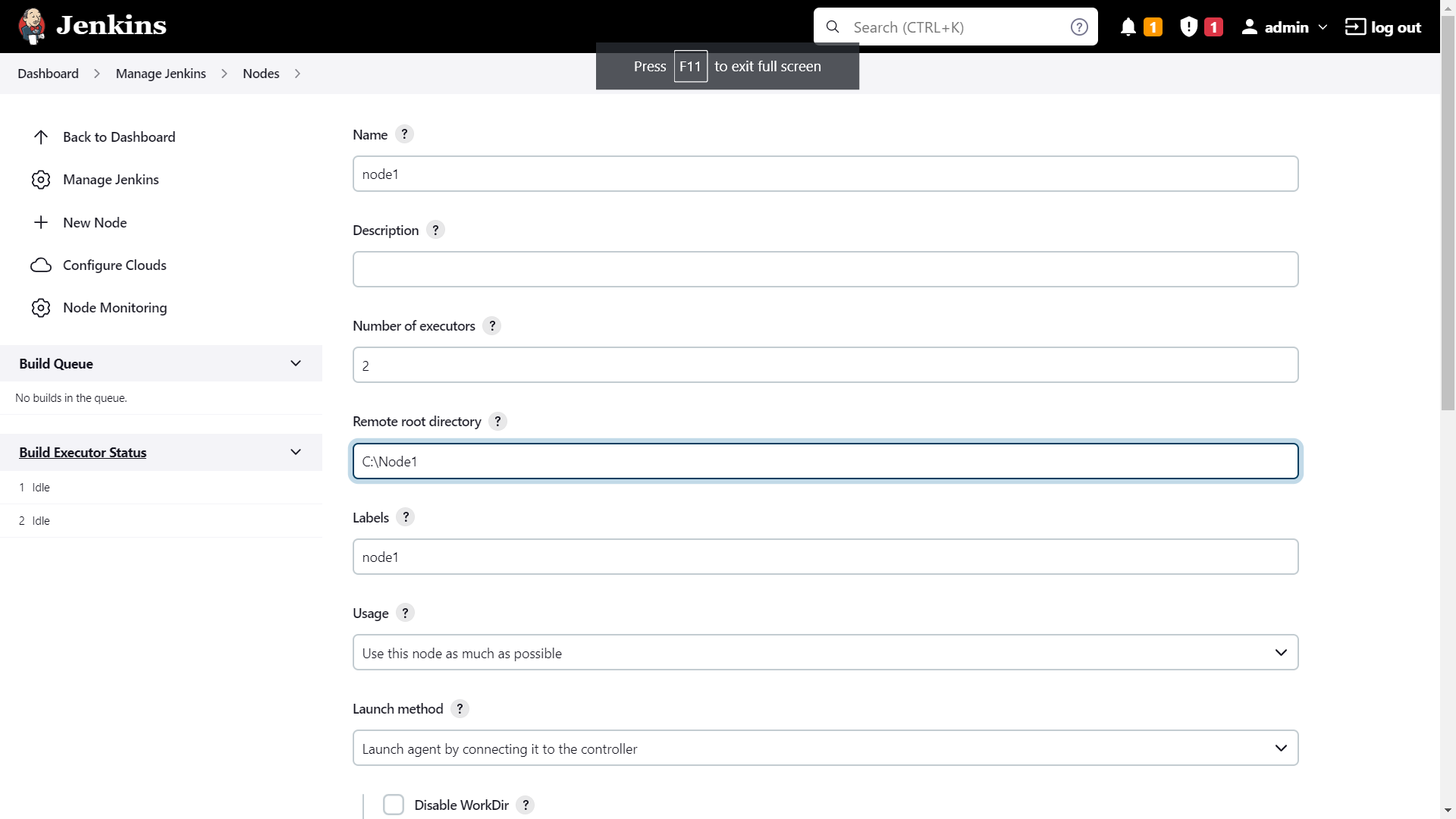
4. For the launch method we select the option of “Launch agent by connecting it to the master”.

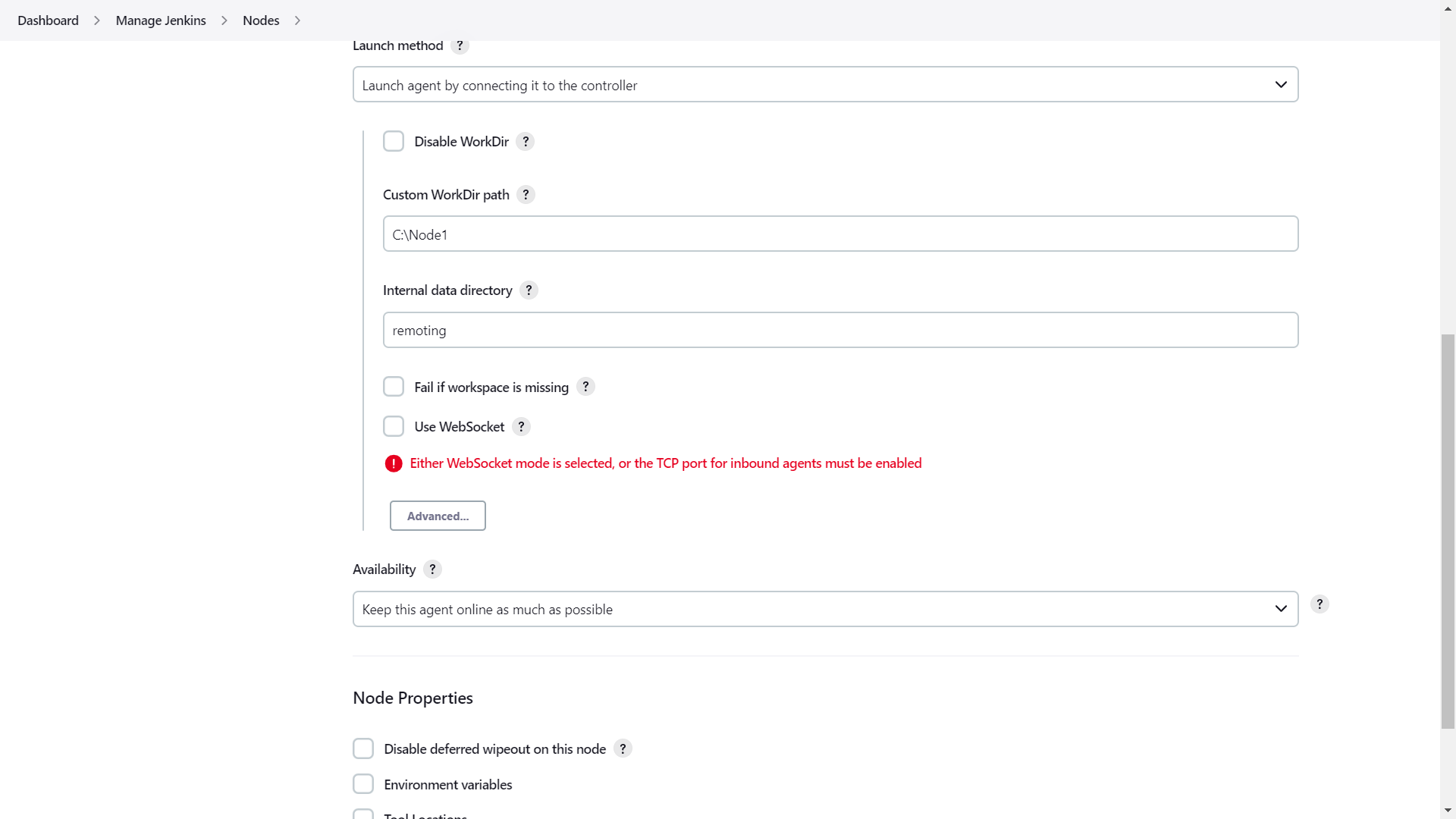
5. Here in the Agents section click on Random and Save it. Now you will find the required option.

6. Enter the Custom WorkDir path as the workspace of your slave node.

7. In Availability select “Keep this agent online as much as possible”.

8. Click on Save.

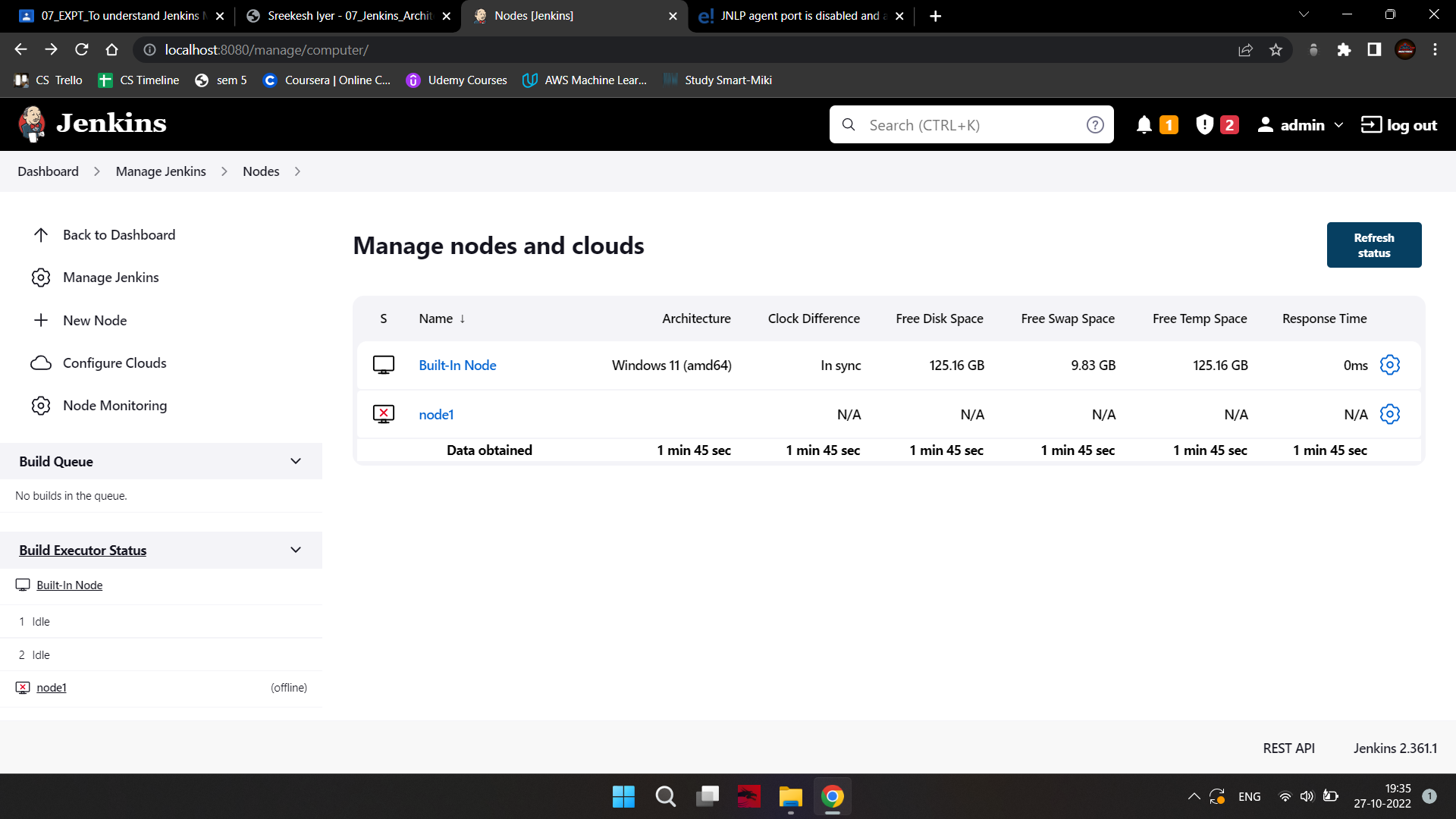


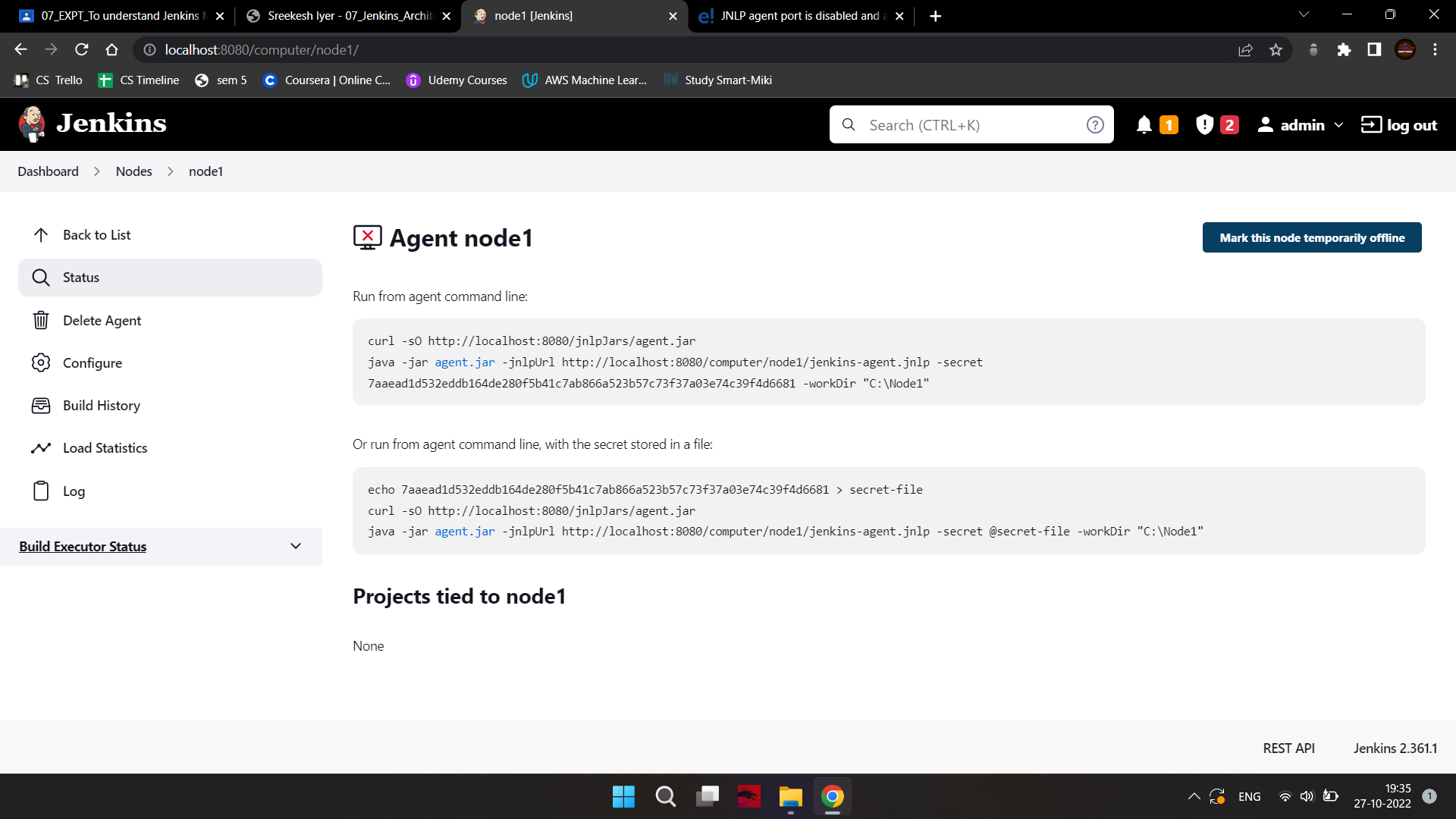


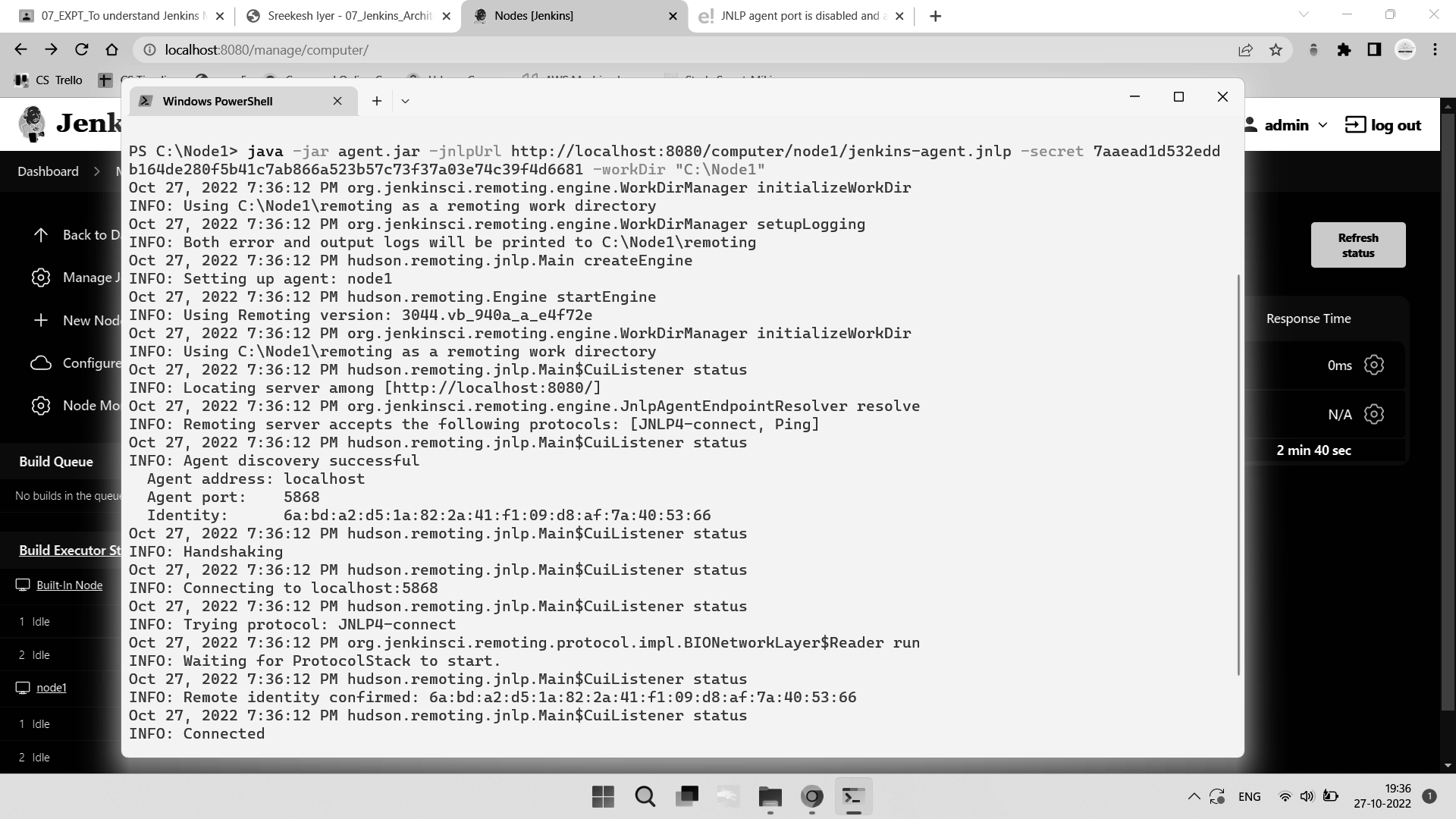
1. Once you complete the above steps, the new node machine will initially be in an offline

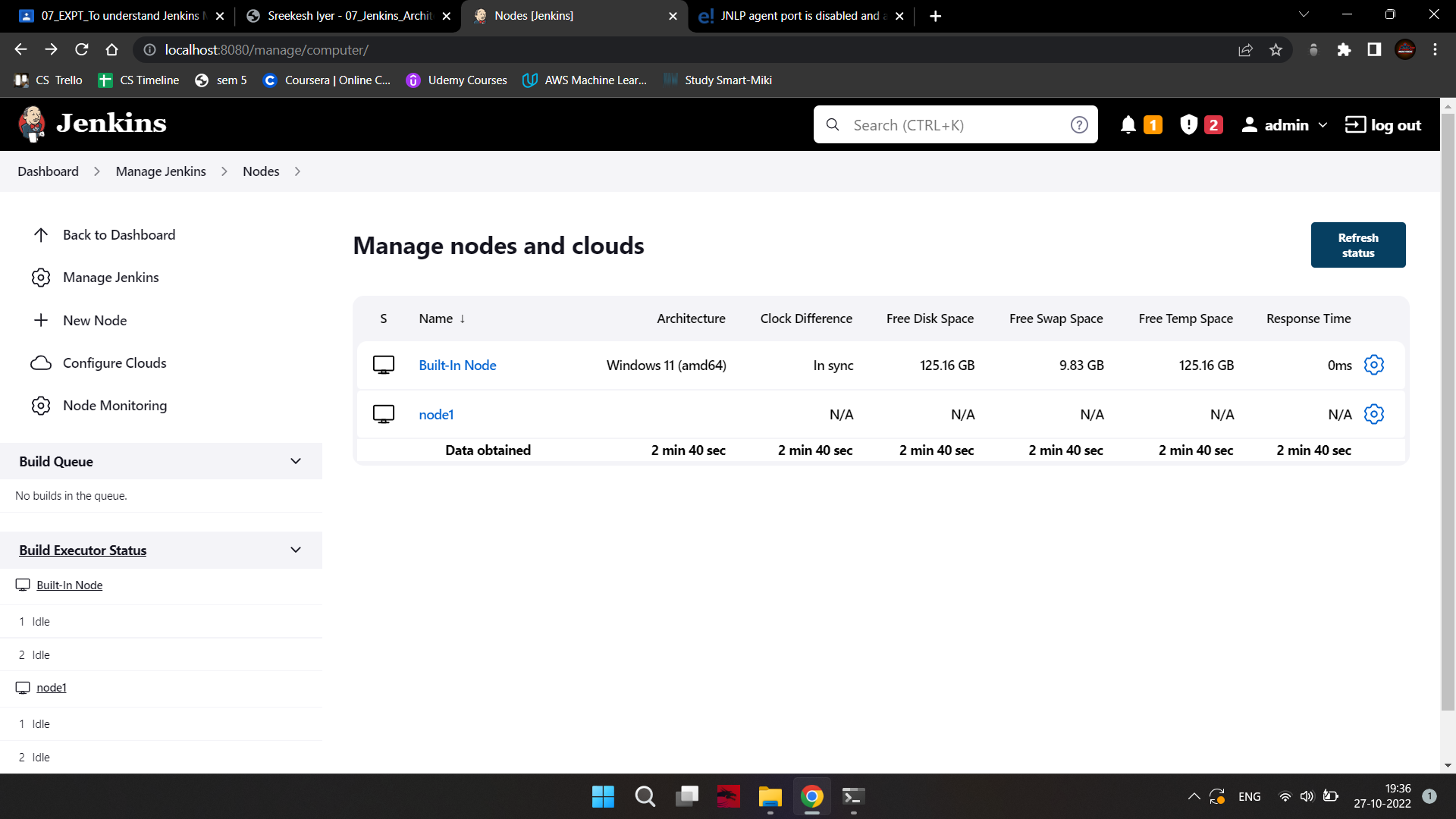
state but will come online if all the settings in the previous screen were entered correctly. One can at any time make the node slave machine offline if required. If the agent does not come online, click on the Agent's name. And use the command given there

to start the agent.









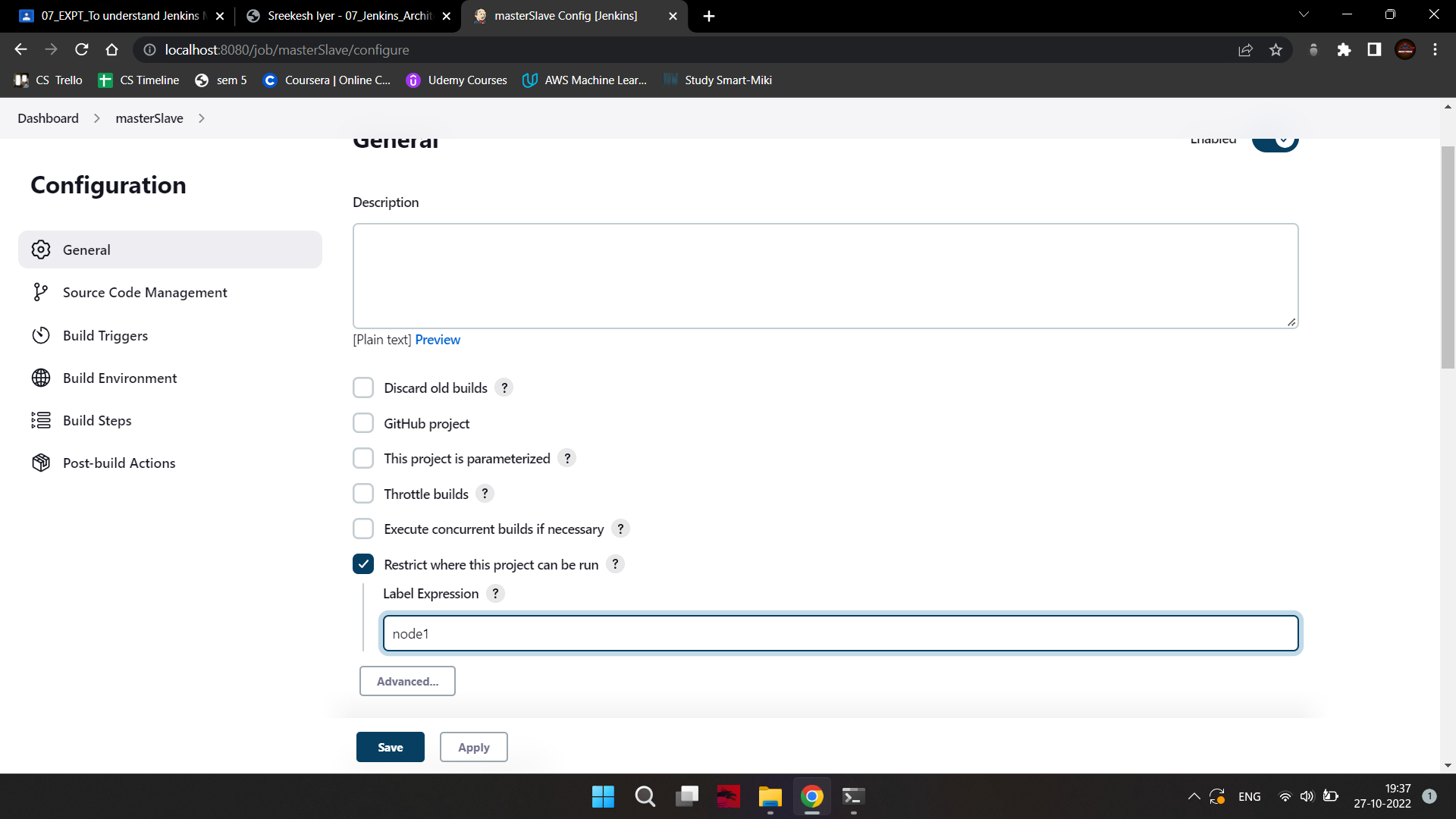
1. Now since your slave agent is up and running, let's execute a job on it.

1. For that I already have an existing job and I will run this job on this slave. Open this job and click on configure.

2. Nowhere in the General section, click on “Restrict where this project can be run”.

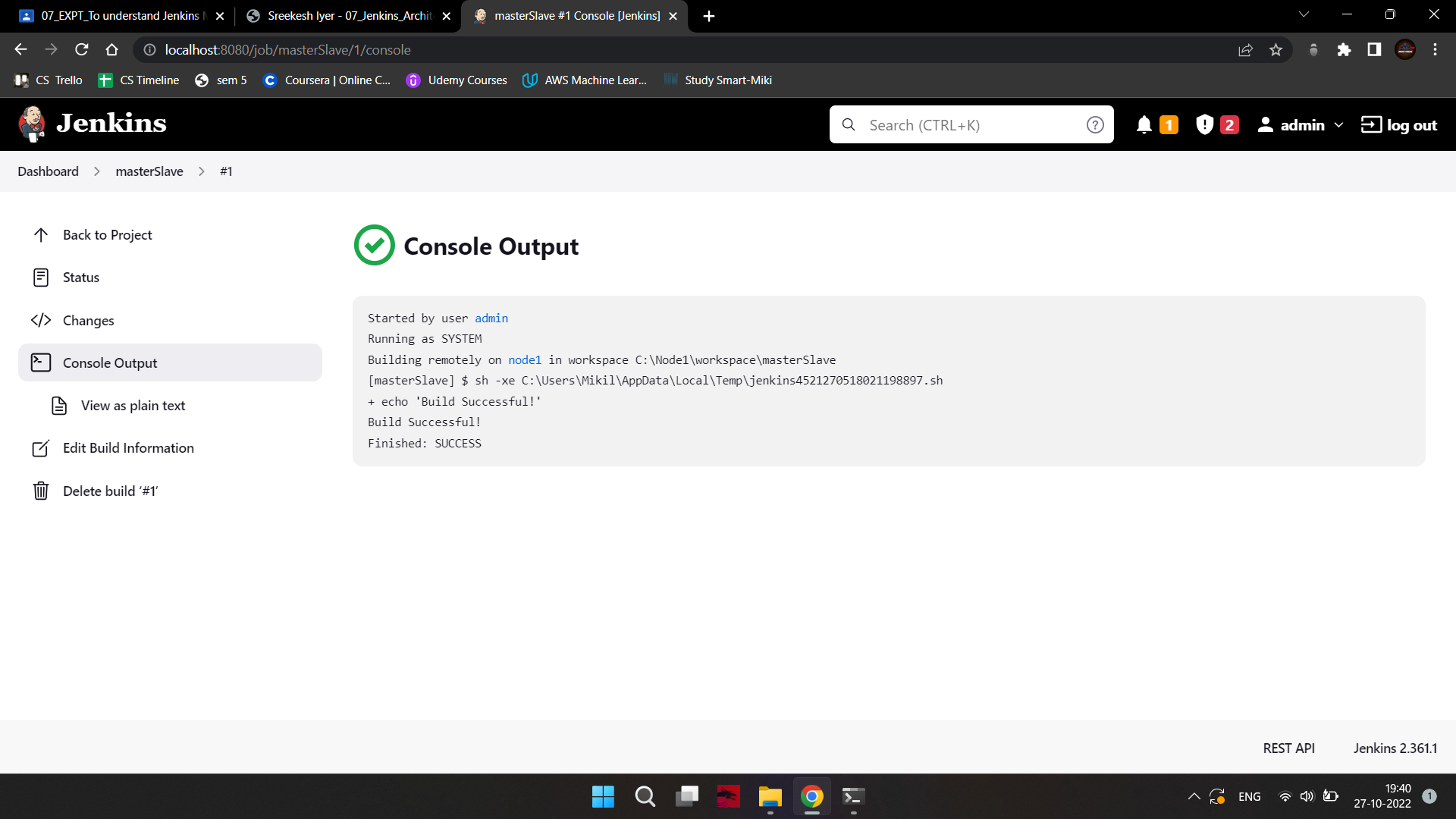
3. Here in Label Expression, enter the name of the slave and save it.

4. Click Save.



1. Now click on Build now and see the output of this job. If everything is correct you will

see the output as Success.



**Conclusion**:

Thus, we understood Jenkins Master-Slave Architecture and scaled Jenkins standalone implementation by implementing slave nodes.