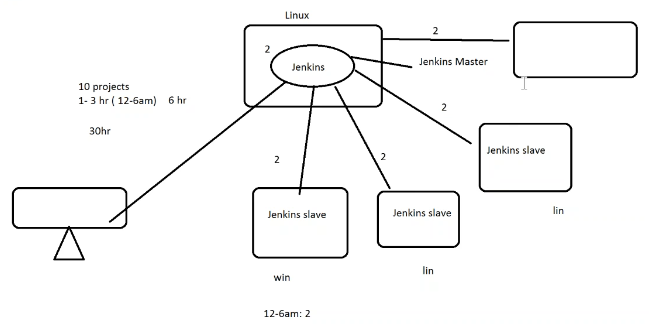
* We can install Jenkins slaves to split the work



* We can add n no of slaves

**Linux Slave Configuration:**

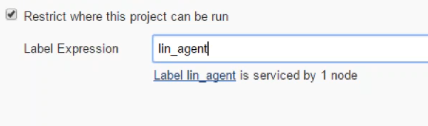
* Set a HTTP security group for master
* Create a pair of authentication keys from the ~/.ssh directory
* **$ ssh-keygen -t rsa**
* And then copy the key in **id\_rsa.pub** and paste it in **“~/.ssh/authorized keys”** on slave server. Do the same if we have more than 1 slave
* **$ cat ~/.ssh/id\_rsa.pub**
* Return to the main server and check if we can able to connect to the slave server
* **$ ssh** [**ec2-user@IP**](mailto:ec2-user@111.222.333.444)
* If we create key again by overwriting the current one. We need to copy the new key to the nodes. Then only it will work
* User needs to have sudo permissions to connect without password
* We need to install java in slave server also
* Now go to manage Jenkins to create a slave by giving “**remote root directory**” as the folder which we created in slave server. We need to give permissions to the folder. So that Jenkins can use that folder to store
* **Sudo chmod -R 777 /Jenkins**
* Under “**launch method**”, select ‘**launch slave agent via SSH’** and set the ‘**host’** to the private IP of slave server
* Then add **credentials** by selecting the **kind** as “**SSH username with private key”**
* Then enter username as ec2-user and password to enter manually
* Copy the key from **“~/.ssh/id\_rsa”** in master server and paste it. Then add it
* After that, select the credentials which we have created just now
* Then save it and launch the agent
* After you have configured your worker nodes, it’s important to change the number of executors to 0 on the master. Select the master from the nodes table and configure the “# of executors” settings to 0. This ensures that project builds will now happen on the worker nodes, and not on the master
* And then for slave2, we need to select **“copy existing node”** option while creating the slave

**Creating job:**

* While creating job we could use “**Restrict where this project can be run”** to allocate slaves
* We can give GitHub remote repo URL and credentials under **“source code management”.** so that Jenkins can pull code from GitHub
* Under build settings, we can select the type of build, that may be shell or gradle or ant
* We can use junit test results report for post build actions. The results will be stored in target location

**Detailed options in job creation:**

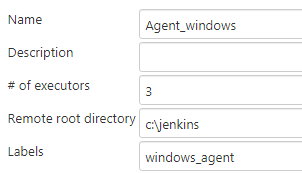
* As mentioned below, with this we can actually run with the groups also, agent in Jenkins referred as label



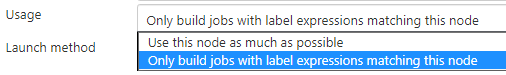
* Suppose we have jobs scheduled on slave1, slave2, slave3 and if suddenly slave1 went down, then the job won’t run. To overcome this problem, we can create a group with all the three slaves and assign them to a project
* So, if slave1 fails it will move to slave2. If all are online, it randomly chooses anyone of them
* To create a group, we need to go to the agent and select configure settings. Then enter a name at labels section. Do the same on another slave also with the same, then both the slaves come under that group which we have given
* And while configuring the job, we need to give the group name in restrict section. So that it can make use of it
* This is called grouping or labelling of slaves

**More options on agent configuration:**

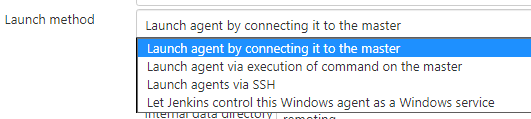
* We can give a name, executors & label to the agent as below.



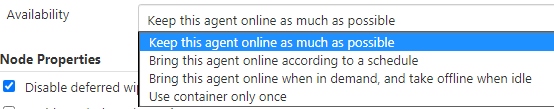
* Then usage, we can use below two options.



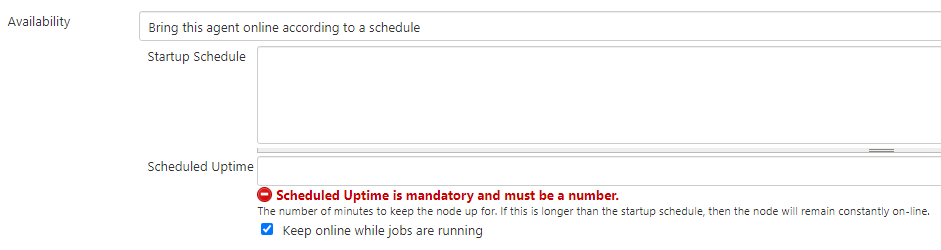
* And launching agent also depends on the option we give as below.



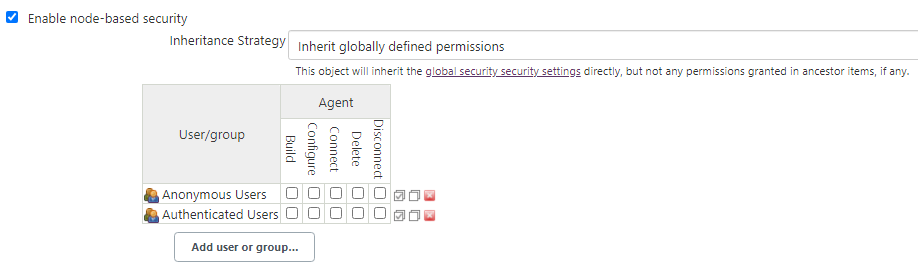
* We can also select the options to keep the agent available.



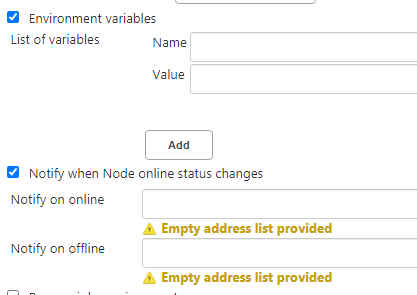
* We can schedule the node to get up just like cronjob and also we can set how long it should be up as below.



* And then at the end in **“node properties”.** We can set below things.
* We can enable node security as below.



* Environment variables & email notification when node status changes as below.



* And then we also need to give the tool configuration for node as below.

