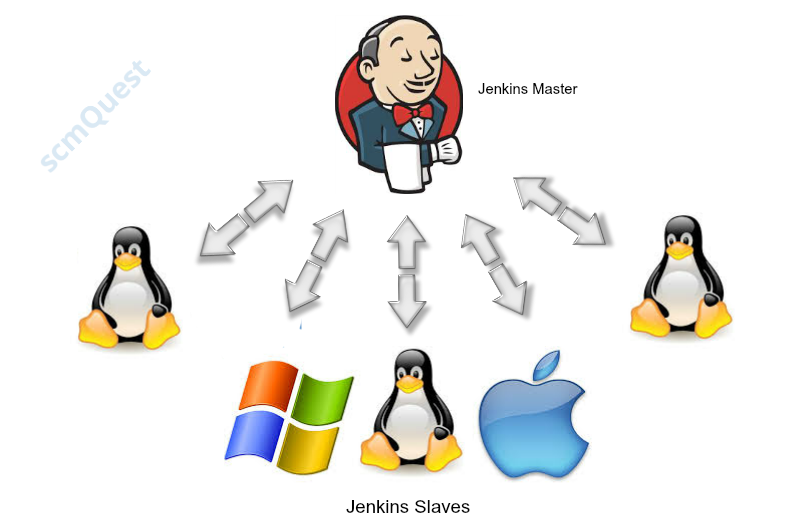
We have already learnt about [Jenkins Installation](http://scmquest.com/jenkins-installation-and-configuration-on-redhat-distributions-with-screenshots/) and basic Configuration along with setting up our first project and running a build for the same.

Jenkins supports the master-slaves architecture, i.e. many slaves works for a master. It is also known as Jenkins Distributed Builds.



A Jenkins master comes with the basic installation of Jenkins and in this configuration the master handles all tasks for your build system.

Now, someone might thinks whether they really need this Jenkins slave or they can run their project without it ?

True, its not mandatory to have Jenkins client, Your master Jenkins can also create the build

But, that will sustain just for small projects where they have very less jobs (say 5-10 job in a day and that too not very frequent)

When, you are working with numbers of projects and wants to run multiple jobs – You need to configure slaves/aagents to serve this purpose.

Jenkins slaves/agents are nothing just a small Java “Client” processes that connects to the “Master” Jenkins instance over the Java Network Launch Protocol ([JNLP](http://en.wikipedia.org/wiki/Java_Web_Start#Java_Network_Launching_Protocol_.28JNLP.29)).

Jenkins Master role is to schedule the build jobs, assign slaves/agents and send builds to the slaves for the actual execution.

It also monitor the slaves (taking them online and offline as and when required), getting back the response of the build results from slaves then showing the build results on console.

All of the job results are collated on the master node for easy viewing.

With this approach of Jenkins master-slaves, the actual workload of building projects are delegated to multiple “slave” nodes, which allows to run numerous projects and their jobs.

It will also allow you to run jobs on different environments, like Linux, Windows, MacOS, etc (as shown in above image).

You might need to run same test case on different environments in parallel, that is where this distributed approach helps you to achieve the desired results quickly.

Once you configures few Jenkins slaves/agents, you might remove the executors on the Jenkins master in order to free up Jenkins master resources, but this isn’t necessary.

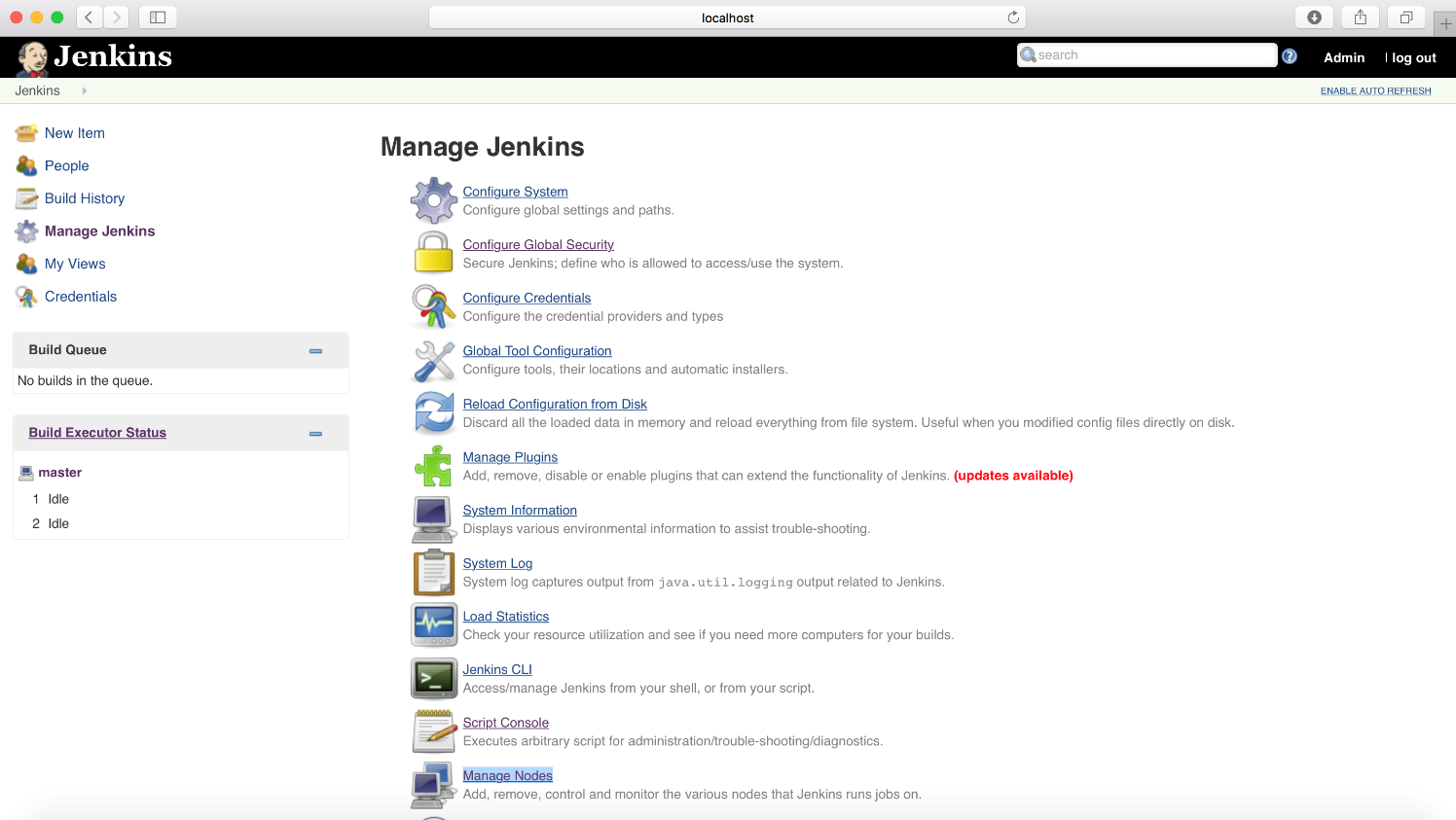
You can install and configure several Jenkins Slaves/Agents on 1 machine considering its input/output, cpu, disk & memory states.

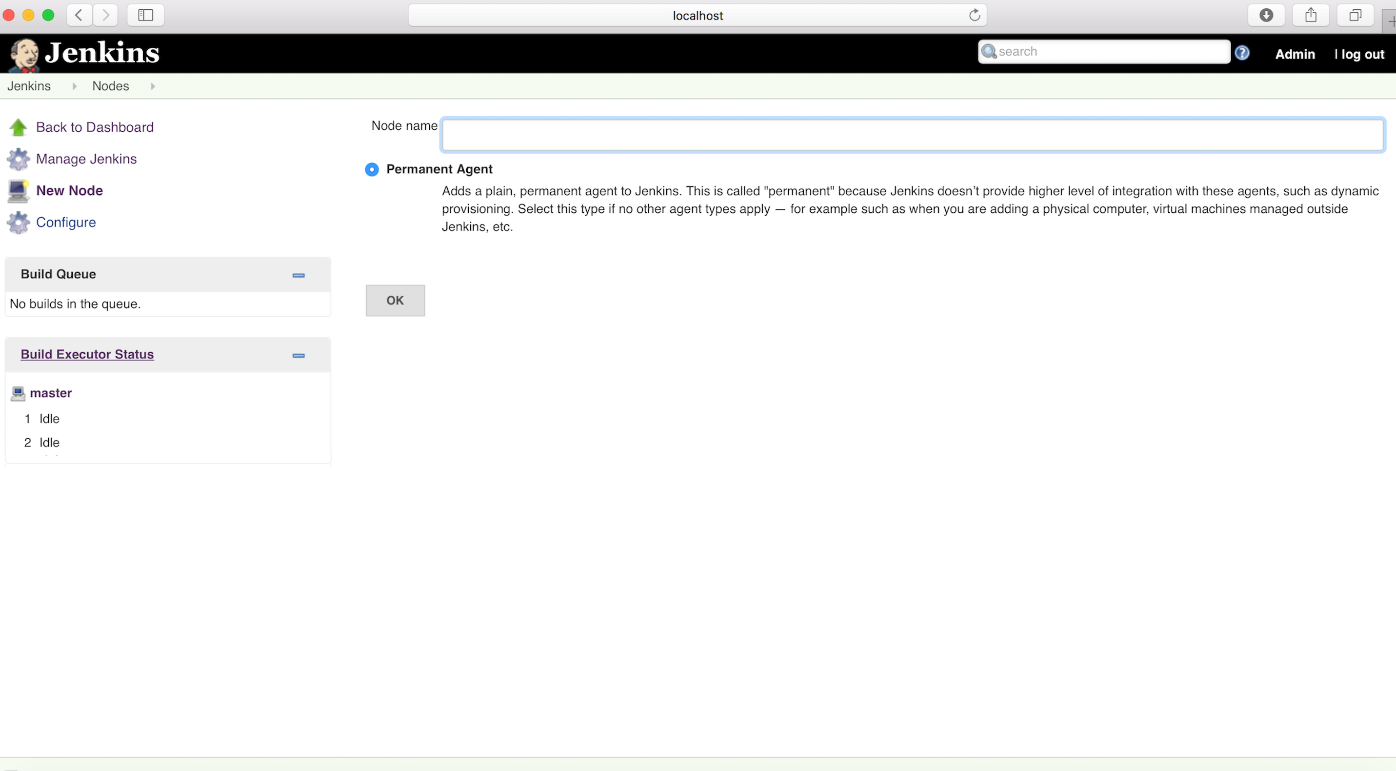
Having multiple slaves will allow you to work more quickly without wasting time on waiting for build executors.

So, lets proceed with setting up a slave/agent/client for our Jenkins Master.

First, check whether you have **Java** installed or not. (If not then install it in similar fashion how we have installed during [Jenkins installation](http://scmquest.com/jenkins-installation-and-configuration-on-redhat-distributions-with-screenshots/))

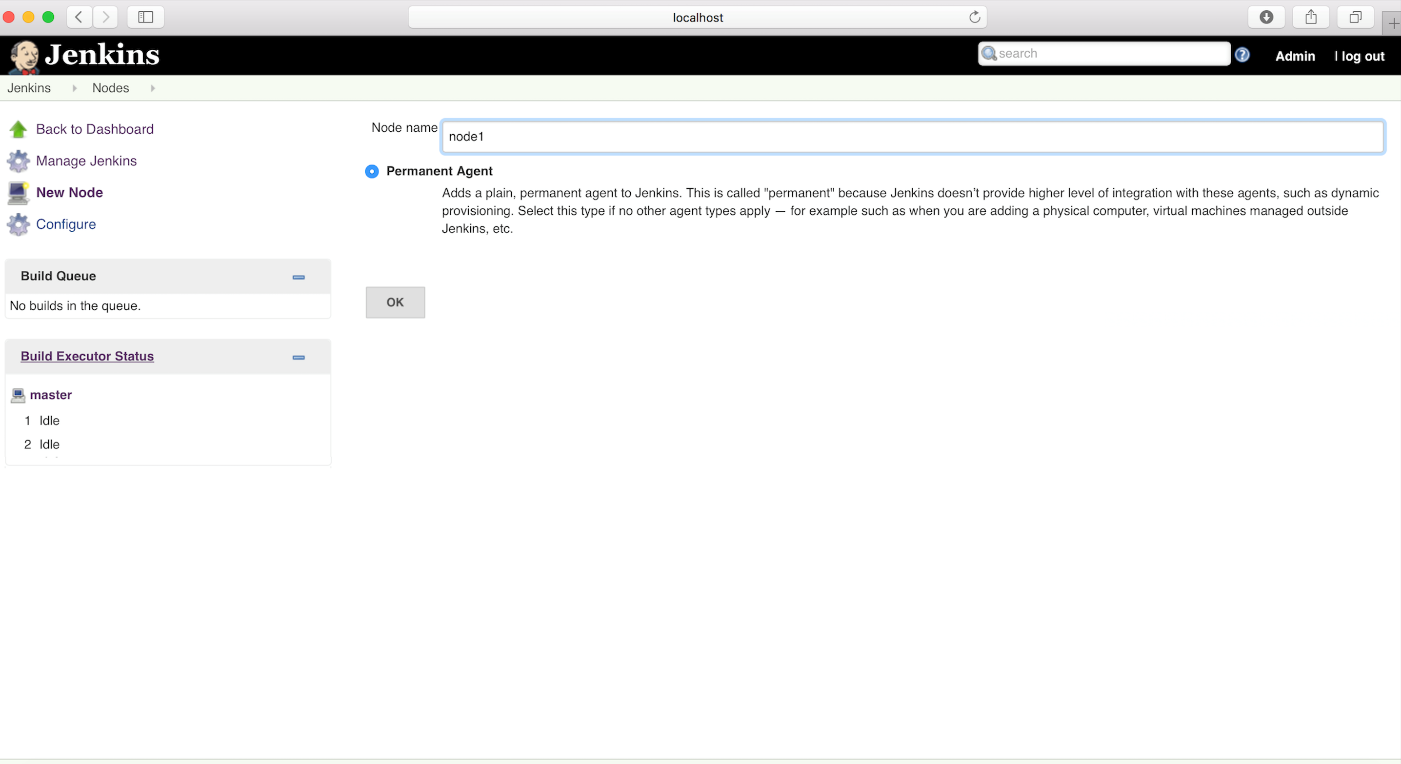
Go to **Jenkins dashboard** -> **Manage Jenkins** -> **Manage Nodes.**



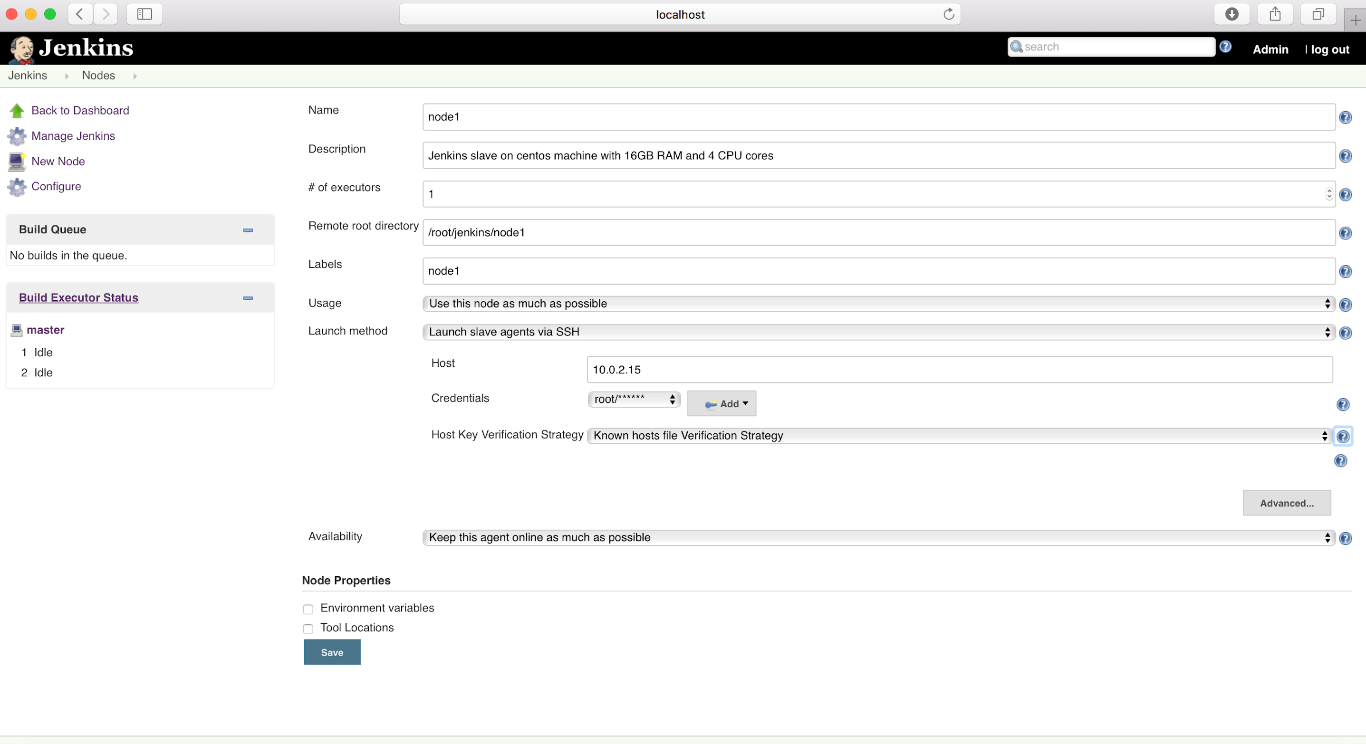


**Name** the node and click on **OK**.

**\*** Initially you will get only one option “**Permanent Agent**” or “**Dumb Slave**“(in earlier jenkins versions), but once you have one slave then you will get another option to copy the node from another node which named as “**Copy Existing Node**”



You will get the node configuration page where you need to enter required details. Not all the details are mandatory but it would be good practice to enter as much information as you can so that another team member can get all the idea without troubling you.



Lets go through with all the parameters in the node configuration page as displayed above :

**Name** : Name of the Slave which should be unique.

**Description** : Optional , Description for this slave, but as I said earlier, it would be really helpful for other team members

**# of executors** : The maximum number of concurrent builds that Jenkins may perform on this agent. I have used just 1 executor for testing purpose, but a good value to start with would be the number of CPU cores on the machine. You can check the server stat and then define the number of executors.

**Remote root directory** : An agent needs to have a directory dedicated to Jenkins. Specify the path to this directory on the agent. It is best to use an absolute path, such as /var/jenkins/node1 or c:\jenkins\node1. This should be a path local to the agent machine. There is no need for this path to be visible from the master.

**Labels** : Labels (or tags) are used to group multiple agents into one logical group. Multiple labels must be separated by a space. For example, linux docker would assign two labels to the agent: linux and docker.

**Usage** : Controls how Jenkins schedules builds on this node. **Utilize this node as much as possible-**This is the default and normal setting. In this mode, Jenkins uses this node freely

**Launch method** : It Controls how Jenkins starts this agent.

**– Launch agent via Java Web Start**:It allows slave to be launched using [Java Web Start](https://en.wikipedia.org/wiki/Java_Web_Start). In this case, a JNLP file must be opened on the agent machine, which will establish a TCP connection to the Jenkins master. This means that the agent need not be reachable from the master; the agent just needs to be able to reach the master. If you have enabled security via the *Configure Global Security* page, you can customize the port on which the Jenkins master will listen for incoming JNLP agent connections.  
By default, the JNLP agent will launch a GUI, but its also possible to run a JNLP agent without a GUI, e.g. as a Window service.

**– Launch agent via execution of command on the master**:It starts slave by having Jenkins execute a command from the master. Use this when the master is capable of remotely executing a process on another machine, e.g. via SSH or RSH.

**– Launch slave agents via SSH**:It starts a slave by sending commands over a secure SSH connection. The slave needs to be reachable from the master, and you will have to supply an account that can log in on the target machine. No root privileges are required. This is the one which i am using for my slave configuration.

— **Host Key Verification Strategy** : Controls how Jenkins verifies the SSH key presented by the remote host whilst connecting.

— **Known Host file verification strategy** : Checks the known\_hosts file (~/.ssh/known\_hosts) for the user Jenkins is executing under, to see if an entry exists that matches the current connection.

{ If you are getting a SSH Host Key Verification error as below:

|  |  |
| --- | --- |
| 1  2  3 | No entry currently exists in the Known Hosts file for this host. Connections will be denied until this new host and its associated key is added to the Known Hosts file.  Key exchange was not finished, connection is closed.  java.io.IOException: There was a problem while connecting to node2.scmquest.com |

It could be a problem with SSH lib used by Jenkins which does not support newer ciphers like ecdsa-sha2-nistp256. Just delete the known\_hosts entry and create a new one through below command

|  |  |
| --- | --- |
| 1 | ssh -o HostKeyAlgorithms=ssh-rsa node2.scmquest.com      (where node2.scmquest.com is the hostname of the slave server) |

It will solve your problem.  }

— **Mannually provided key verification strategy :**Checks the key provided by the remote host matches the key set by the user who configured this connection.

— **Mannually trusted key verification strategy :**Require a user with Computer.CONFIGURE permission to authorise the key presented during the first connection to this host before the connection will be allowed to be established.

— **Non Verifying verification strategy :**Does not perform any verification of the SSH key presented by the remote host, allowing all connections regardless of the key they present. Its not advisable to select as it may open the path for attackers.

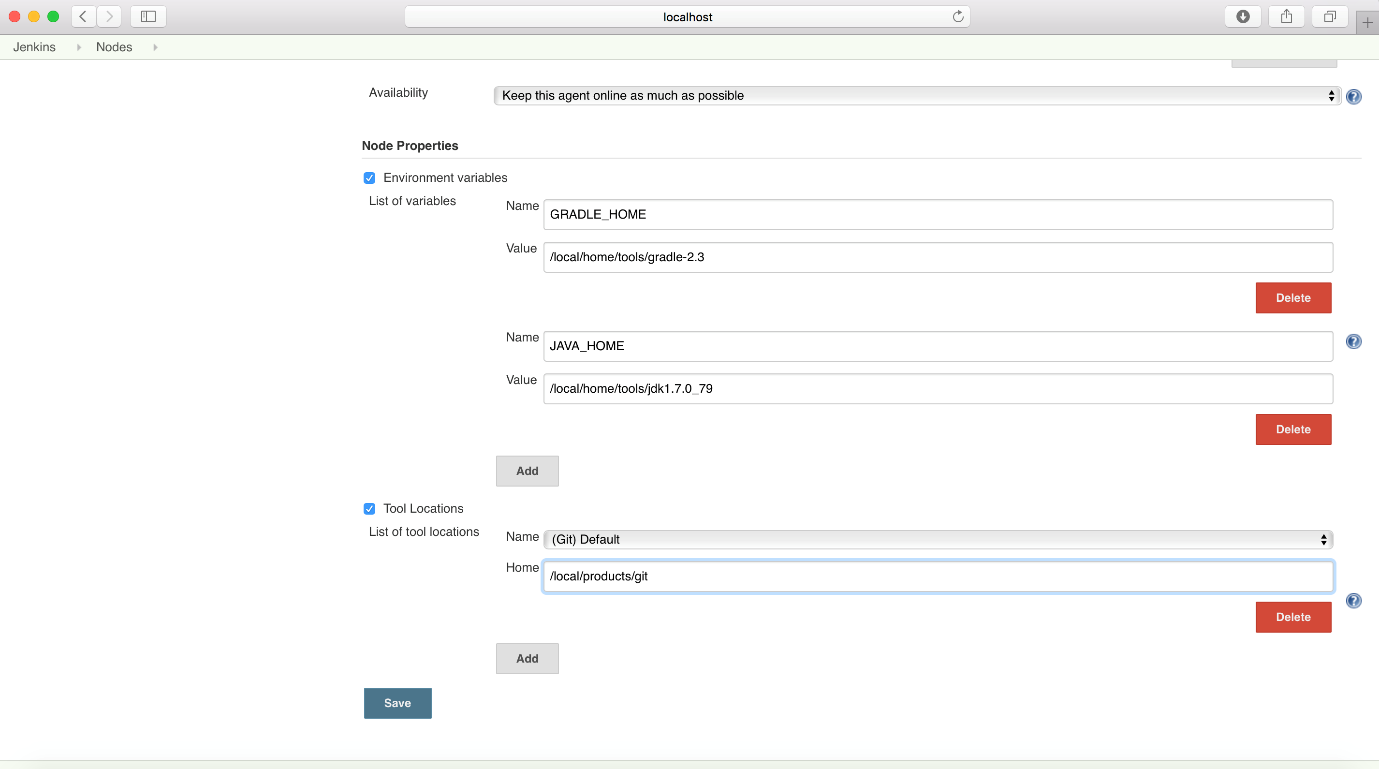
**– Let Jenkins control this Windows slave as a Windows service**: It starts a Windows slave by [a remote management facility](http://en.wikipedia.org/wiki/Windows_Management_Instrumentation) built into Windows. Suitable for managing Windows slaves. Slaves need to be IP reachable from the master.

**Availability** : Controls when Jenkins starts and stops this agent.

**Keep this slave on-line as much as possible :**This one is the default and normal setting. In this mode, Jenkins tries to keep the slave on-line as much as possible. If Jenkins can start the slave without user assistance, it will periodically attempt to restart the slave if it is unavailable. Jenkins will not take the slave off-line.

**Take this slave on-line when in demand and off-line when idle :**In this mode, if Jenkins can launch the slave without user assistance, it will periodically attempt to launch the slave while there are unexecuted jobs else the slave will be taken off-line by Jenkins

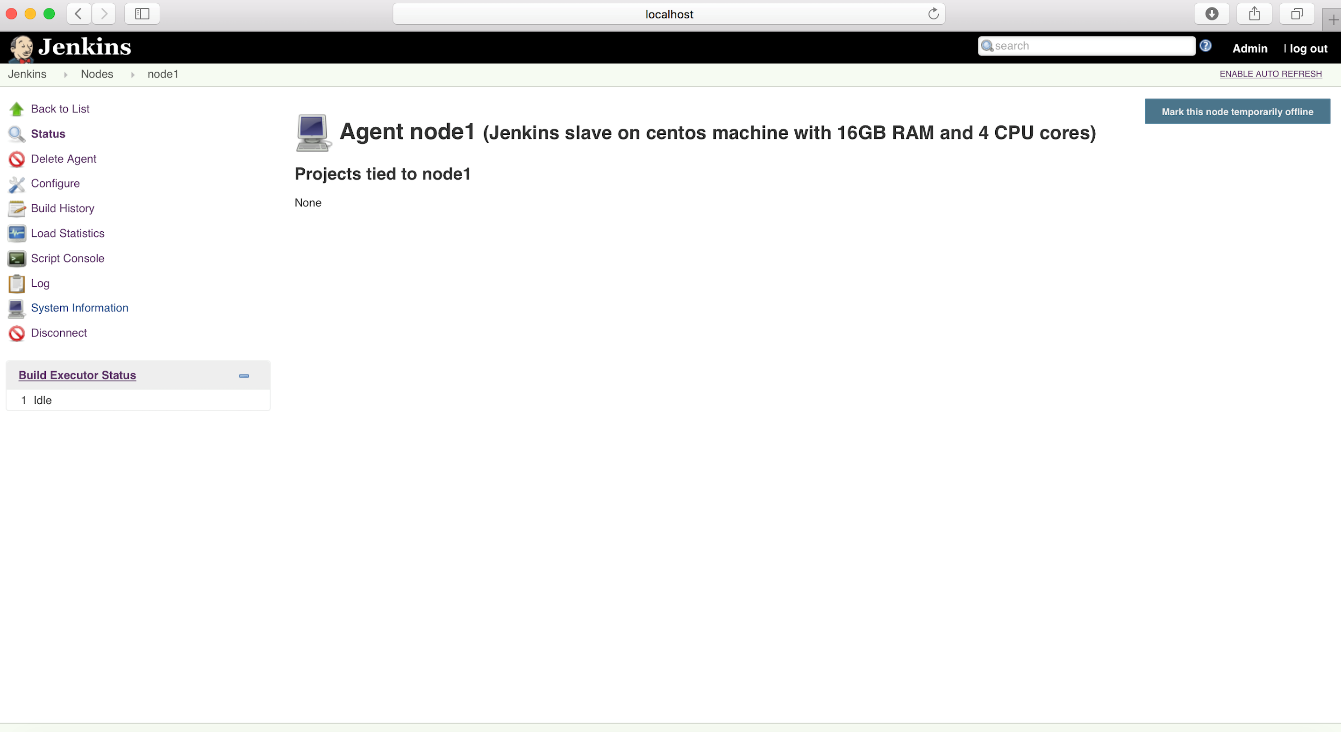
You can also define the node properties, like environment variables, such as PATH, JAVA\_HOME, etc and Tools locations such as, ANT, GIT, etc.



Environment variables defined here will be made available to every build executed by this agent, and will override any environment variables that have the same *Name* as those defined on the *Configure System* page.

For Tool Locations, you can specify the location of certain tools on this node, overriding the global configuration (You may prefer to use automatic tool installers instead, removing the need to configure each node separately)

Once you are done click on **SAVE a**nd that’s it. Your new **Jenkins Slave is up and running** and ready to take the builds



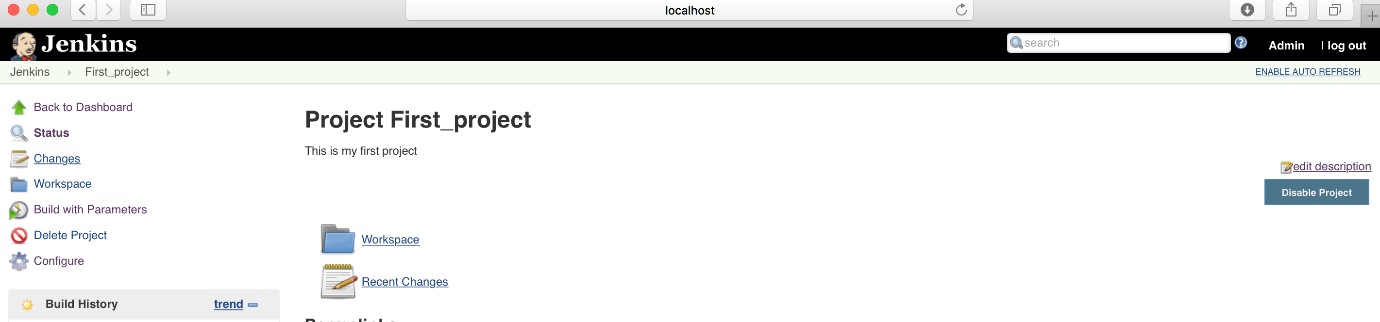
Now, lets go ahead and run our first build on this slave.

As I have mentioned and you might have noted that Jenkins master do have their build executors and until we turn them off, Jenkins can use those executors to perform builds based on the CPU stats, memory and other factors. So to avoid this issue we need to do some changes in our project configuration to force the project’s job to run on newly created Jenkins Slave.

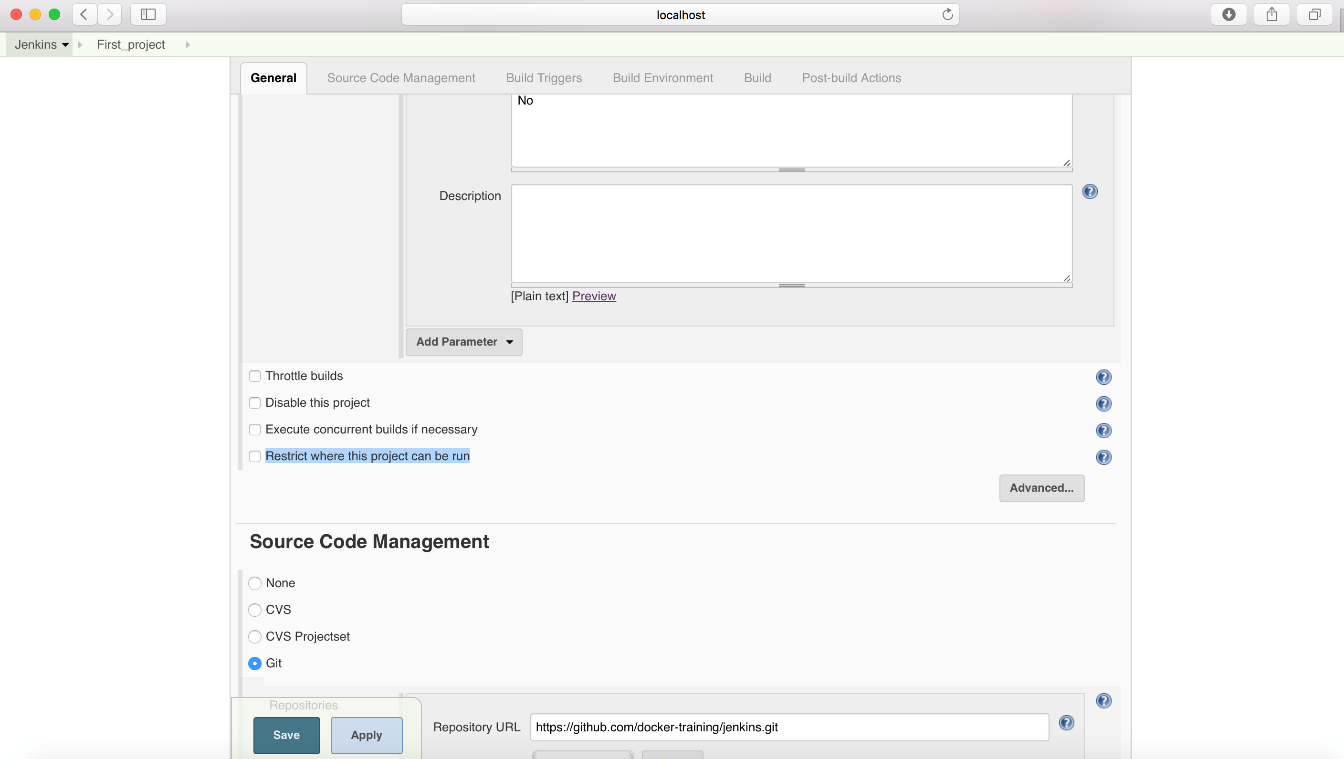
This technique is also helpful when you have some jobs to be run on different platforms, with specific configurations, etc (like we discussed in the beginning of this article)

So, to configure the project specific to this new node,

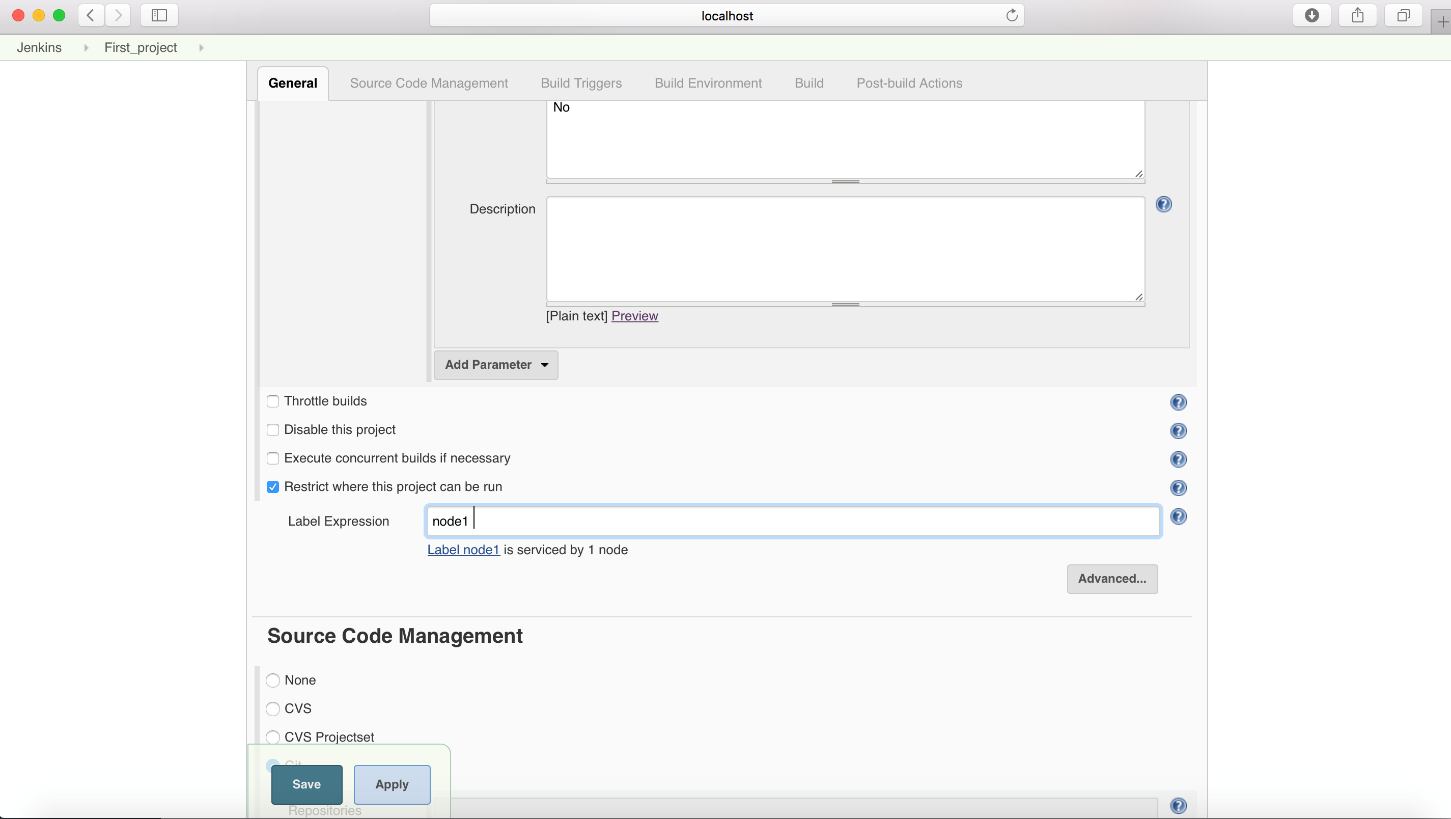
Open the **Project** and click on **Configure**



Find the check-box in General section “**Restrict where this project can be run**” and click on it

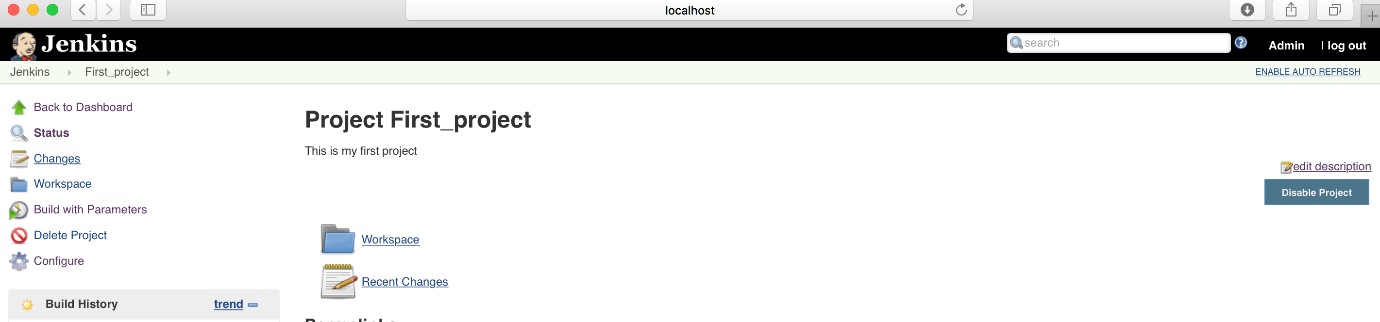


Just type the **Label** name of your slave, in this case its “node1”

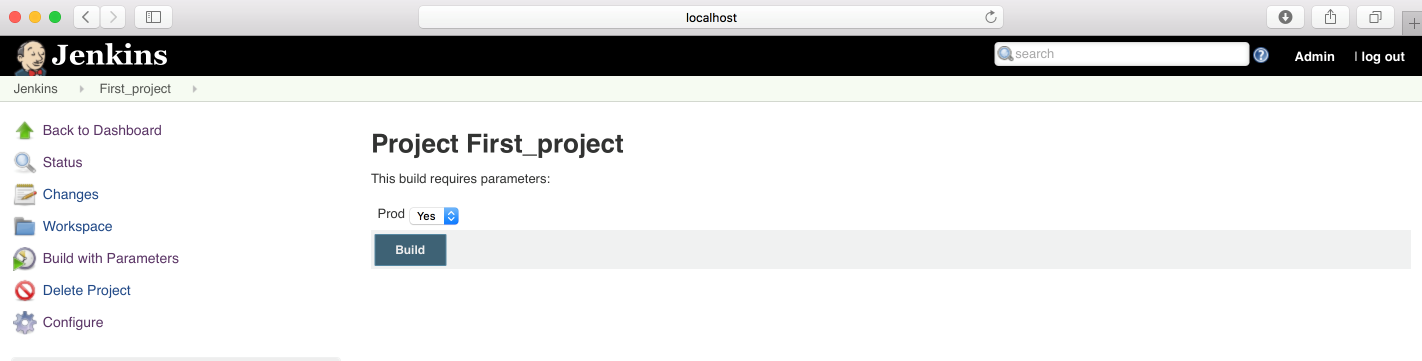


And then **Save** the changes.

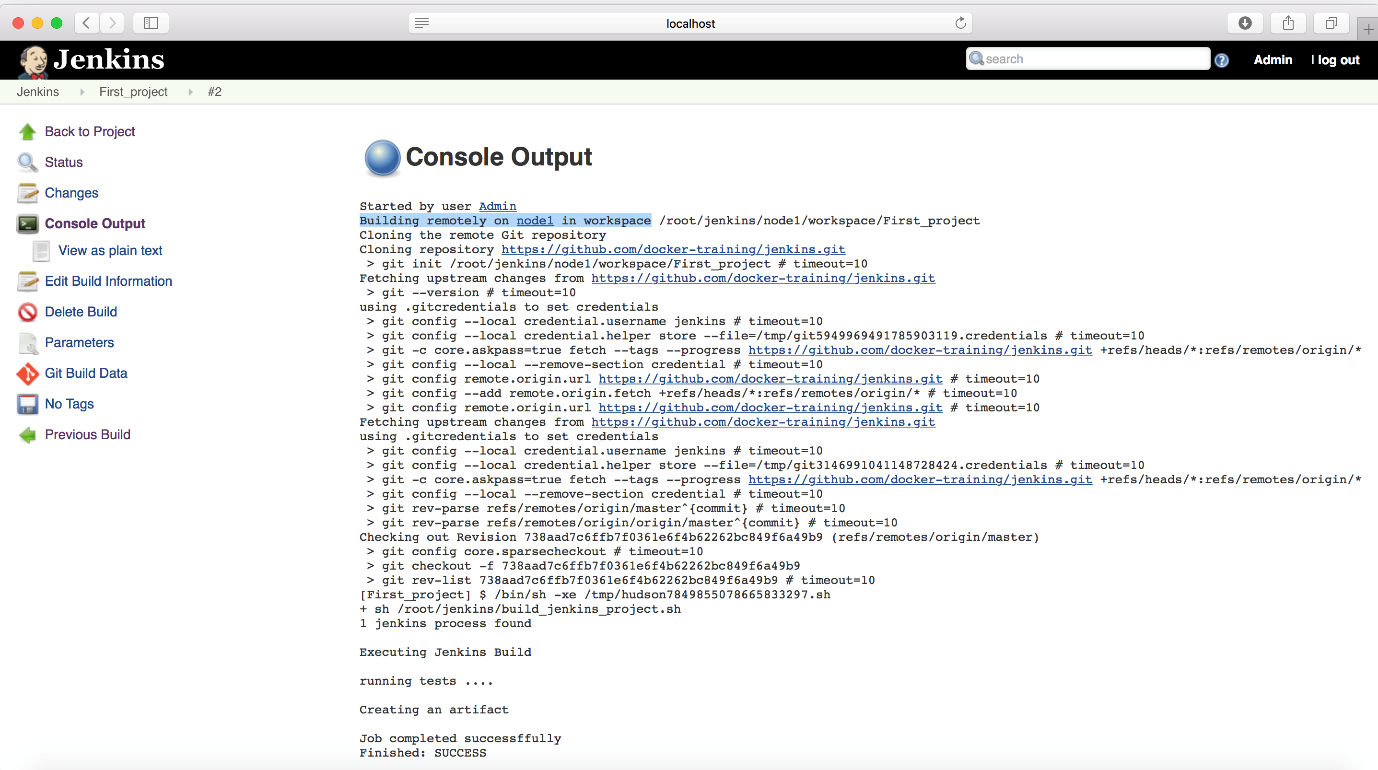
Go back again to Your **Project** and click on “**Build with Parameters**”



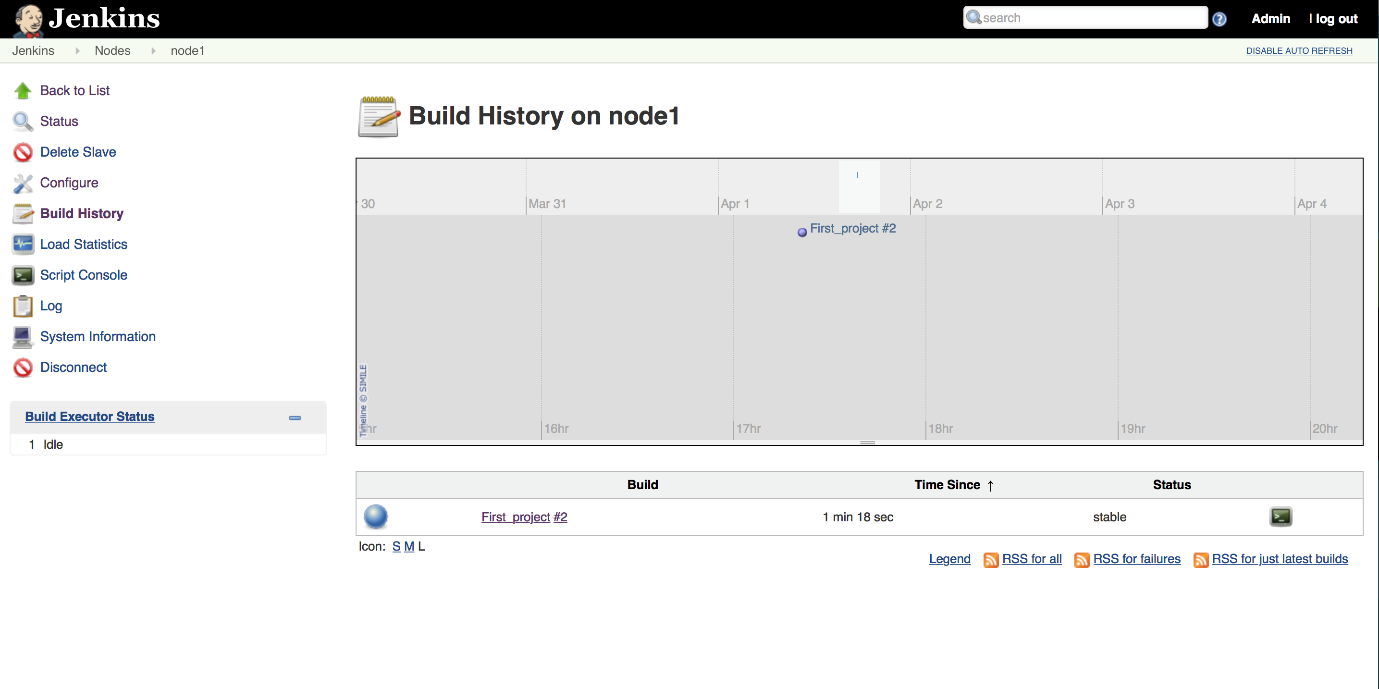
Choose the appropriate parameter(s) [Based on your project configurations] and click on **Build**



Then click on Build number and check the**Console Output**, where you will find that build is running on newly created node and is **successful**

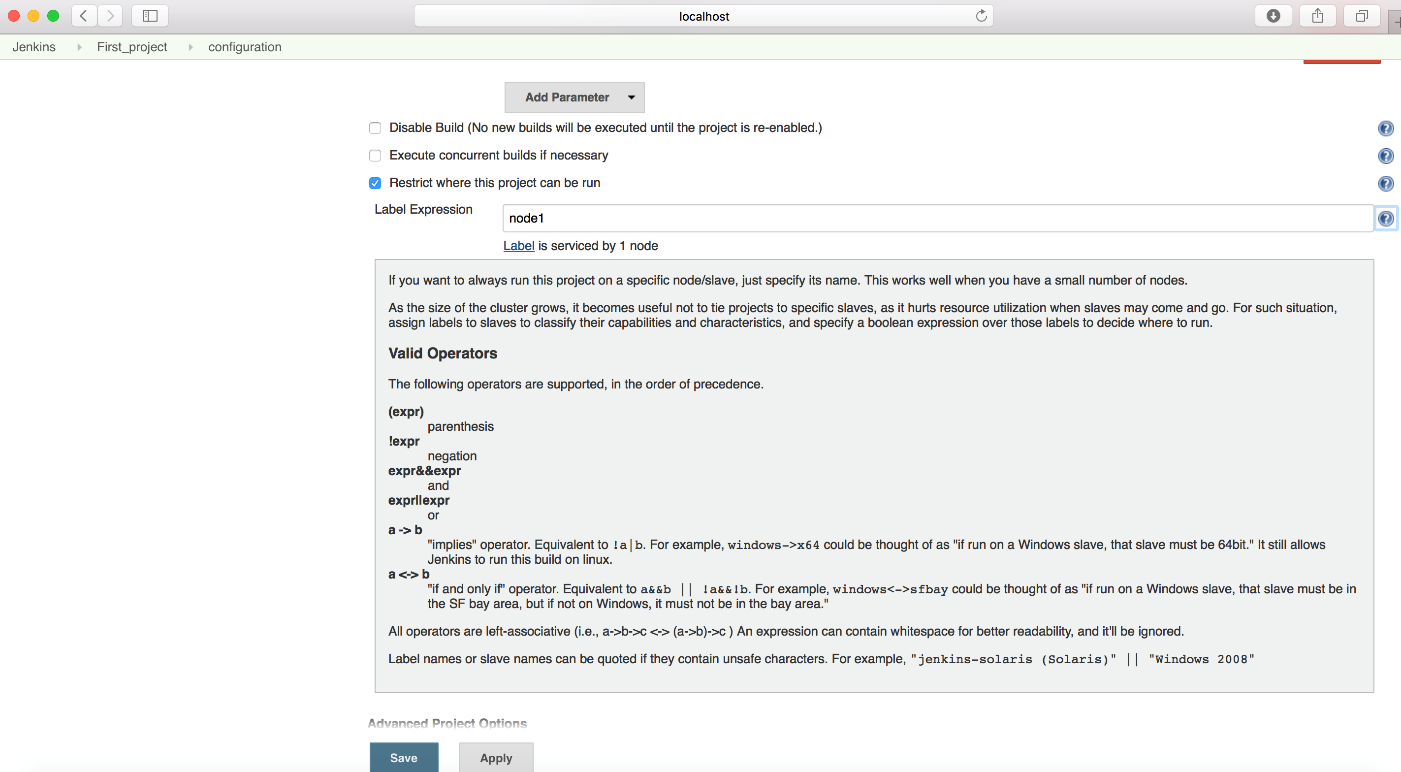


You can also check the Node statistics to get all the idea about the newly created node/slave/agent.

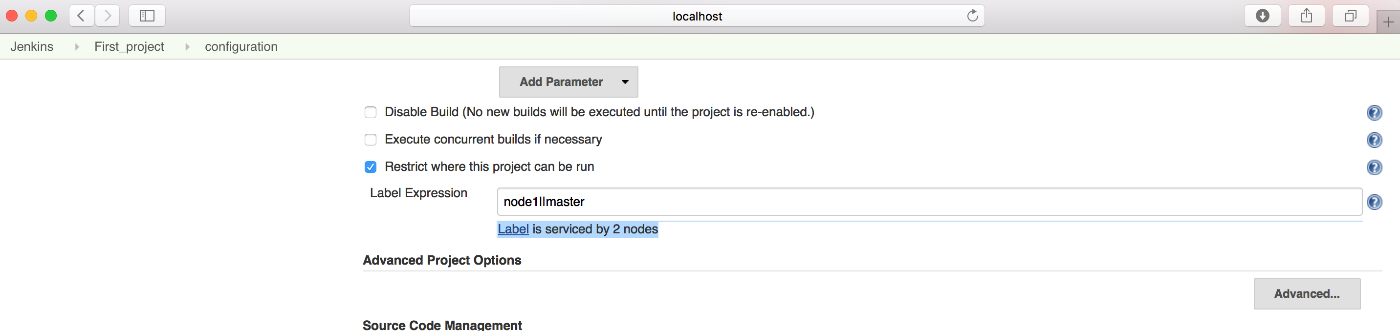


You can also define more than one labels (i.e. more than one node to the project) which will allow you to run the build in parallel and Jenkins will choose the faster slave for your build. To do that,

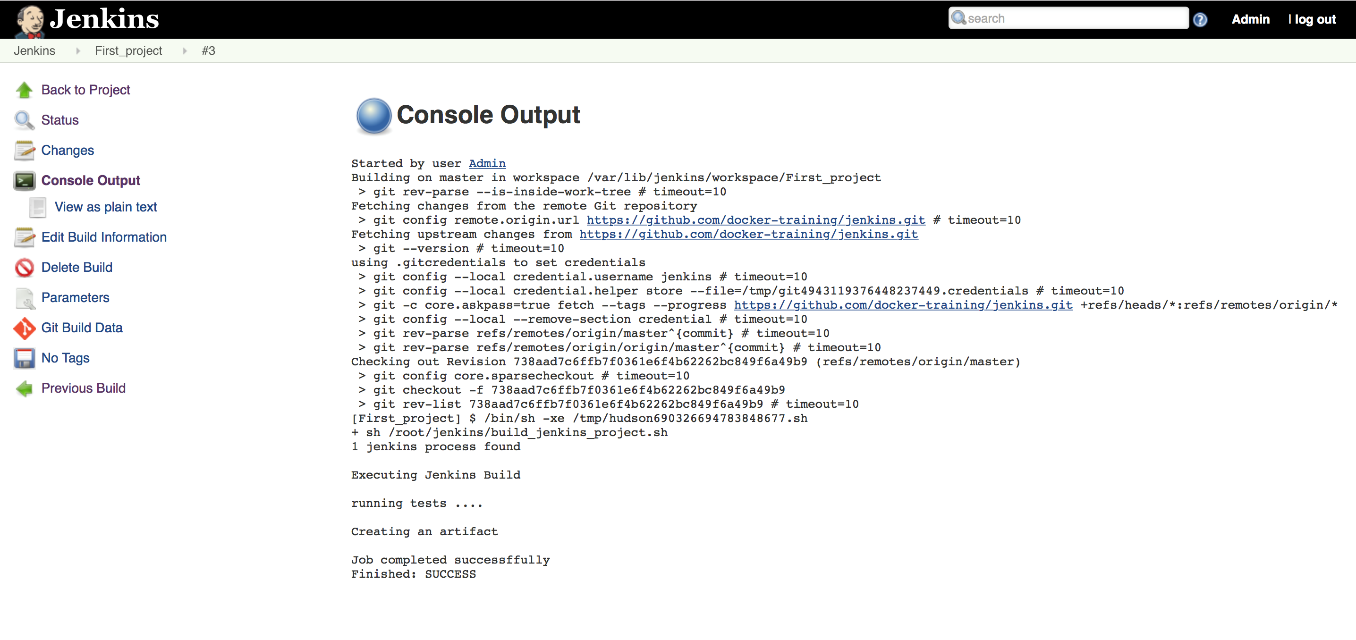
Go back to Project Configuration and find the “**Restrict where this project can be run**” option. Click on  button to know all the expression details



You can use || symbol to tell Jenkins to use any of the slave, like below



Once done, **Save** your changes and again go back to Your **Project** and click on “**Build with Parameters**“. And this time you will find the job **ran** on **master** build executor since we have given a choice to Jenkins to choose any build executor of master or slave (node1)



Congratulations !! You have successfully added the new Jenkins Slave/Agent/Node and configured project accordingly.

In today’s article we have covered:

* Need of Jenkins Slave/Agent
* Concept of Distributed Build
* Jenkins Slave/Agent configuration
* Issue with SSH Host Key Verification and its resolution
* Jenkins Project changes to run job on particular node
* Build on Jenkins Agent