

**Introduction to CAFA-SIT Jenkins**

***How to execute test cases in Jenkins and check for errors in ATF Reporter***

Danny M. Byers, Top Secret Stuff Doer

Time Warner Cable— 9710 Crescent Executive Dr., Suite 4141, Charlotte NC 28217

**T:** 704.206.5270 **E:** danny.byers@twc-contractor.com

TABLE OF Contents

[I. Executive Summary 3](#_Toc418087429)

[II. Jenkins Home Page 4](#_Toc418087430)

[III. Running Jenkins Jobs 6](#_Toc418087431)

[IV. Build With Parameters 7](#_Toc418087432)

[V. Jenkins’s Build Queue and Build Executor Status 9](#_Toc418087433)

[VI. Cancelling Jenkins Jobs 9](#_Toc418087434)

[VII. Configure Jenkins Jobs 11](#_Toc418087435)

[VIII. Changing How and When Jenkins Jobs Run Automatically 12](#_Toc418087436)

[IX. Console Output 14](#_Toc418087437)

[X. Viewing Build Results 19](#_Toc418087438)

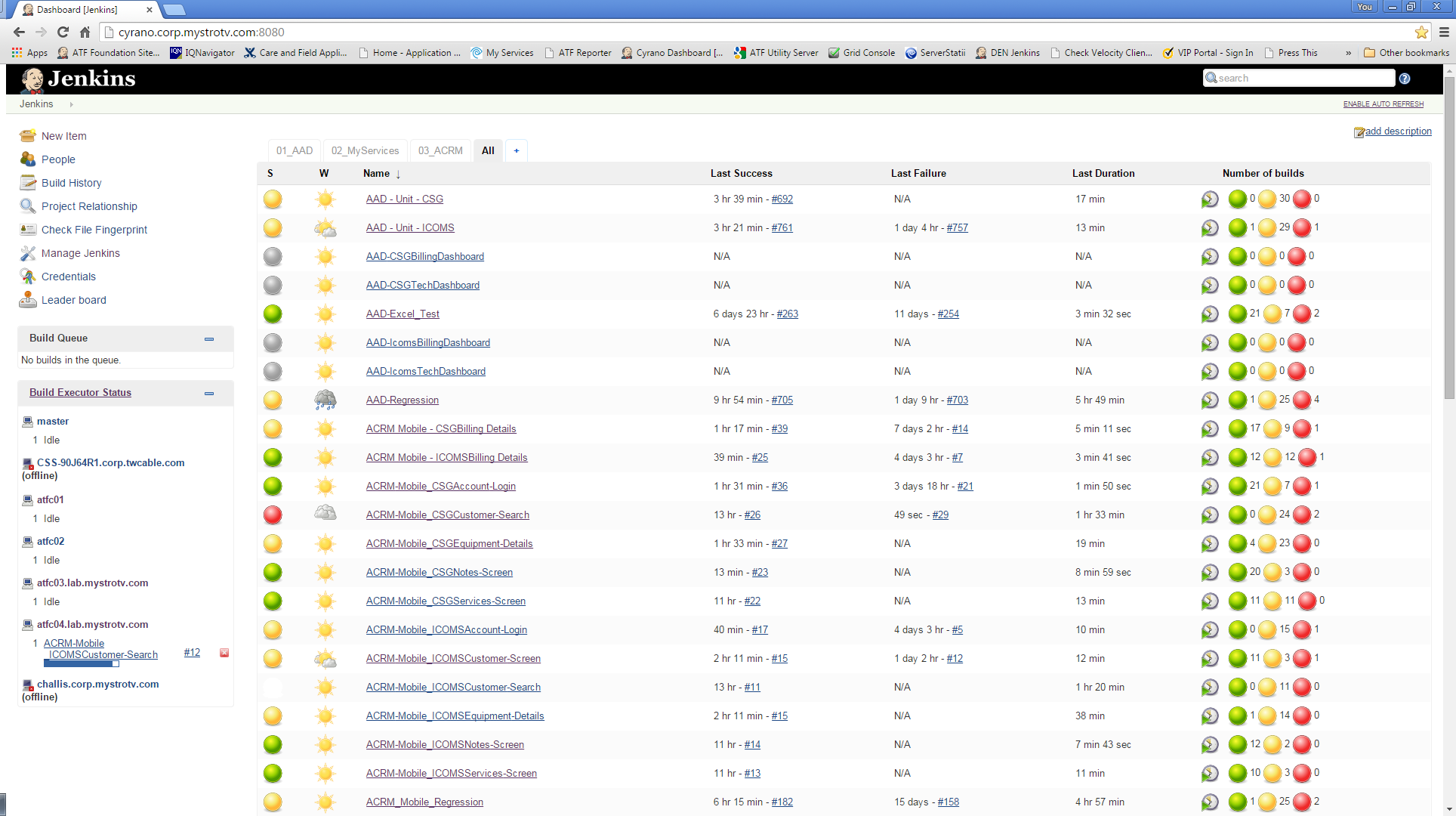
[XI. Viewing Browser Test Executions on Remote Machines 22](#_Toc418087439)

## Executive Summary

What is Jenkins and how do we use it here at Time Warner Cable CAFA-SIT?

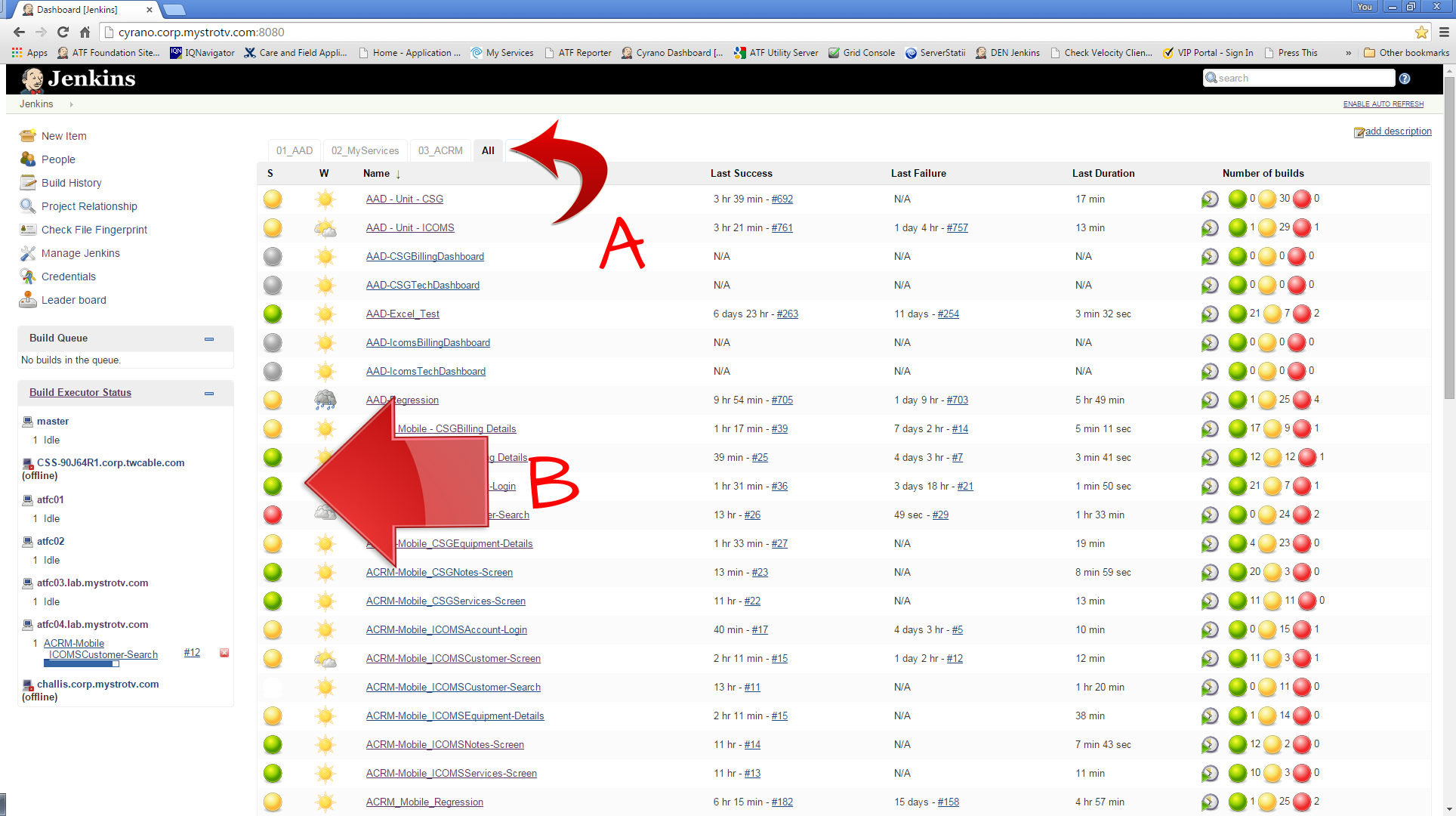
Jenkins is a Continuous Integration server. In a nutshell, Continuous Integration is the practice of automatically running tests on an independent, non-developer owned machine at scheduled periodic intervals or on-the-spot demand builds.  
  
This gives the advantage of always knowing if unit and regression tests work, while getting immediate feedback on new code changes to our test environments. This allows for our manual test team to concentrate on new application features while leaving the automation tests to handle stable, non-changing functionality.  
  
If you only run your tests occasionally the problem is that a lot of code changes may have happened since the last time and it is rather hard to figure out which change introduced the problem. When it is run automatically every day, then it is usually obvious when code functionality changed and who introduced the problem.

## Jenkins Home Page



The CAFA-SIT Jenkins server is located at <http://cyrano.corp.mystrotv.com:8080> . As you first open this page, the view will be defaulted to the “All” tab. This tab shows a list of all the jobs currently on Jenkins.

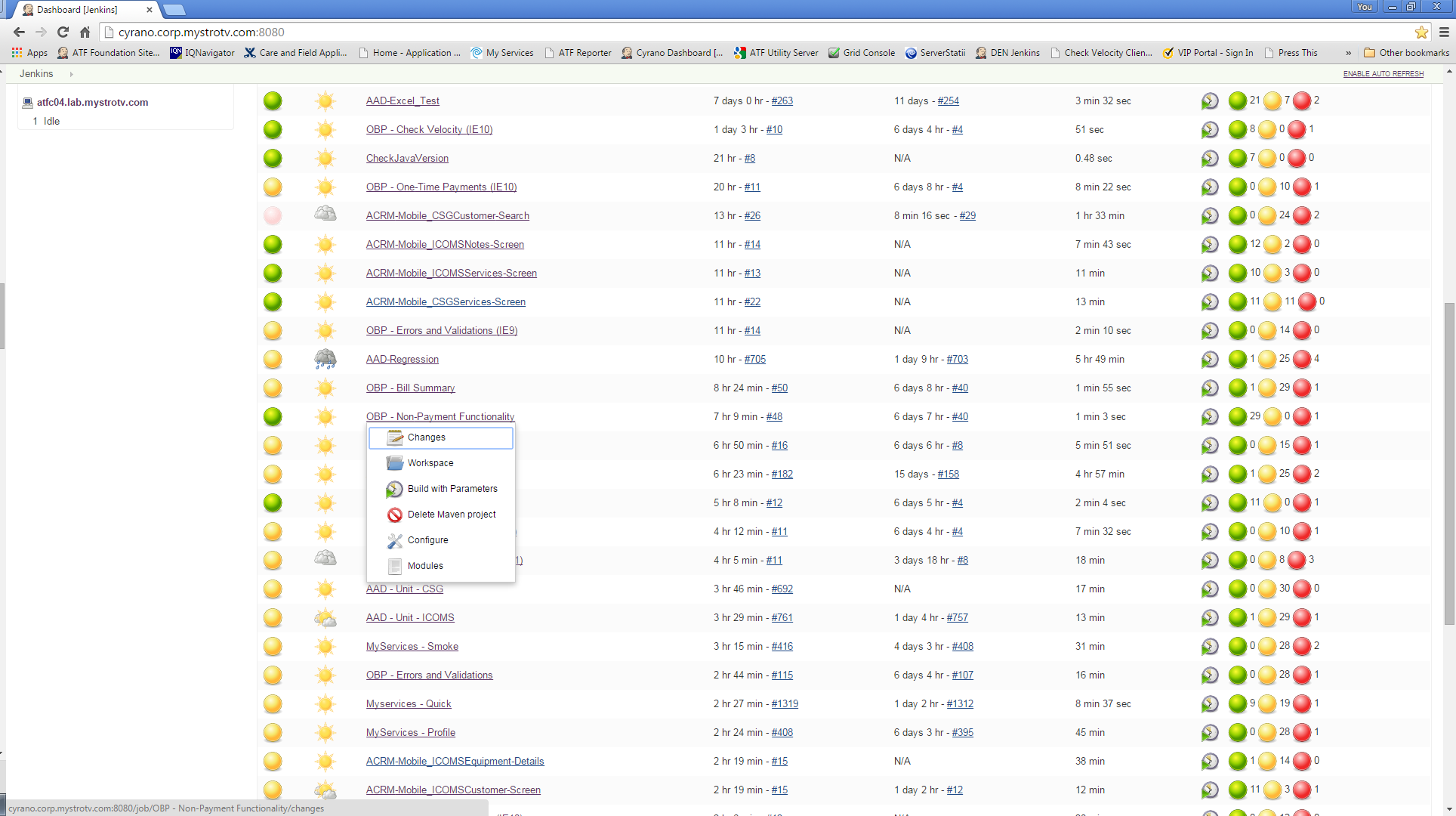
On this page you can see a list of all of jobs that exist on this Jenkins server, the status of the last completed builds, an indicator of which jobs are currently in process and/or queued up to run, durations of last runs, last build failures and successes, and tabs to give you smaller sets of grouped tests.



1. As stated earlier, “All” jobs are the default view on this page, however, you can also narrow down the list to a specific group of jobs by clicking one of the tabs at the top, which will be sorted out by project name. Only the Jenkins jobs associated with that project name will be listed under the specific tab.
2. The first column, S, stands for “Status” and will show the outcome of the last build which ran for that particular job. There are 4 colors which are used as visual indicators to get a quick glance of the health of the job. **Gray** balls indicate that the job has never run. **Yellow** indicates that the build was stable, but had test case failures during execution. **Green** indicates that the build was stable and there were no failures during test execution. **Red** indicates that the build was unstable and no test passed or ran properly.

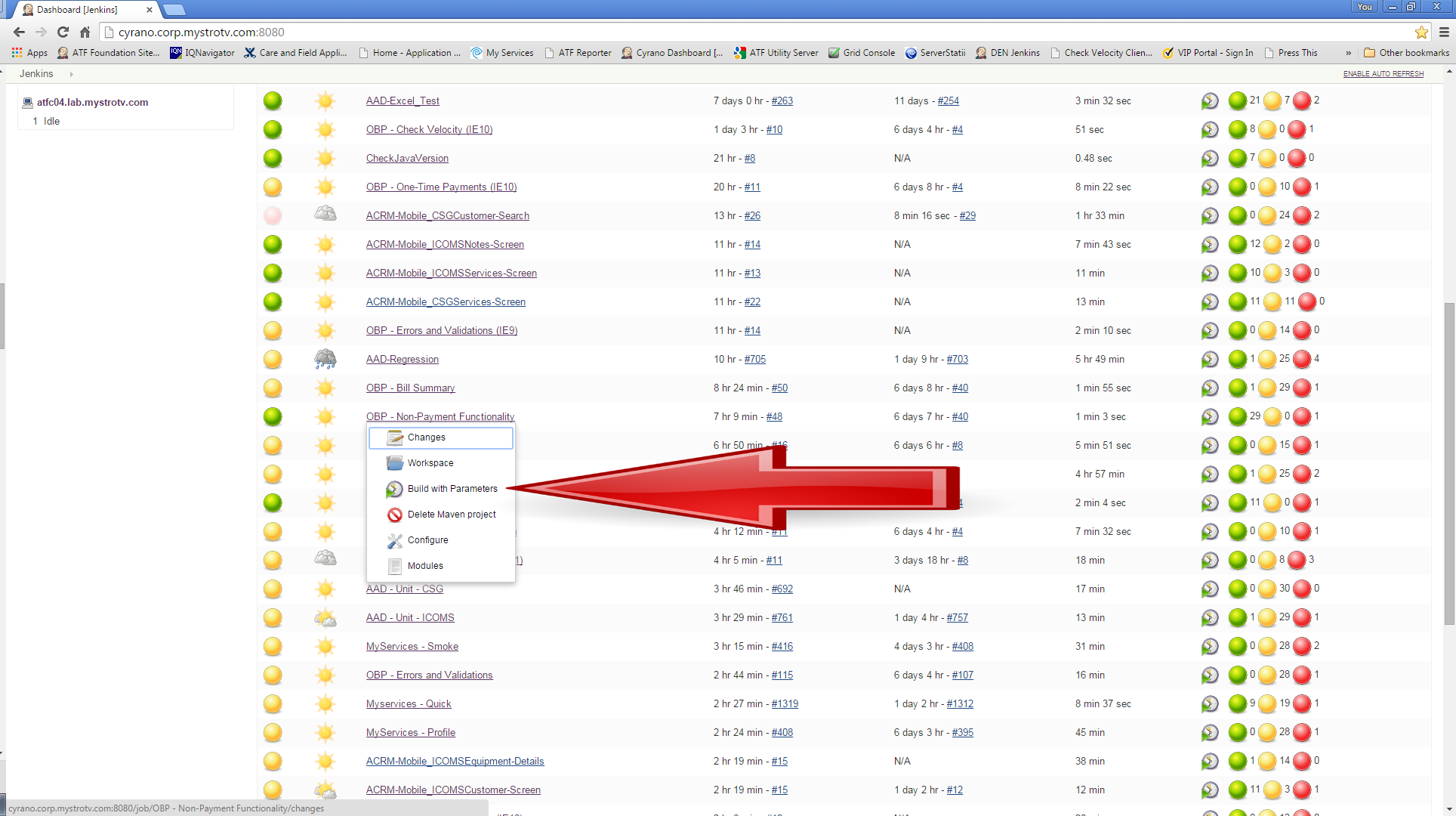
Also, if you are watching the page and see a “blinking” ball, this indicates that the Jenkins job is currently in progress.

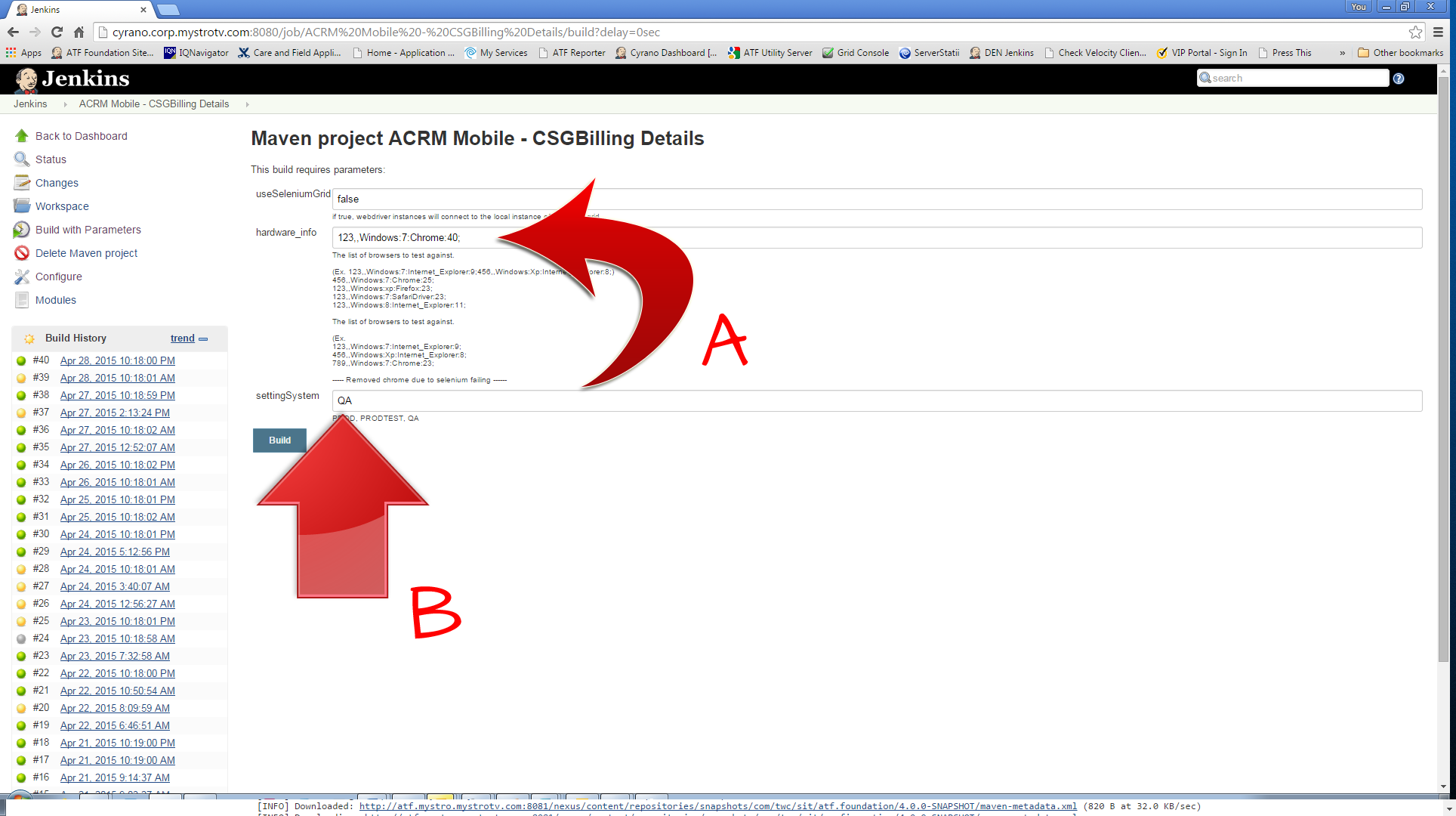
## Running Jenkins Jobs



To start a job running on Jenkins, first find your job which is to be run on the Jenkins page on the “All” tab, or under its individual product tab page. After finding the job name, Right Click and hold the mouse button down on the job name until a drop-down menu appears with a list of options. There are only two job options in which you should ever have a need to use: (1) Build with Parameters and (2) Configure.

## Build With Parameters





1. When you first select Build with Parameters, a web page will appear with three text boxes: (1) seleniumGrid, (2) hardware\_info, and (3) settingSystem.
2. seleniumGrid defaults to false and **should not** be altered.
3. Hardware\_info is how we tell Jenkins which browser to run the tests on. By default, hardware\_info is set to run as chrome and the syntax looks like this:

**123,,Windows:7:Chrome:29;**

*We can disregard all of the information up-to the browser field, as it pertains to other functionality within the ATF Framework that our automation efforts don’t utilize.*

The list of options that we can use for browsers are:

Chrome, Internet\_Explorer, SafariDriver, Firefox

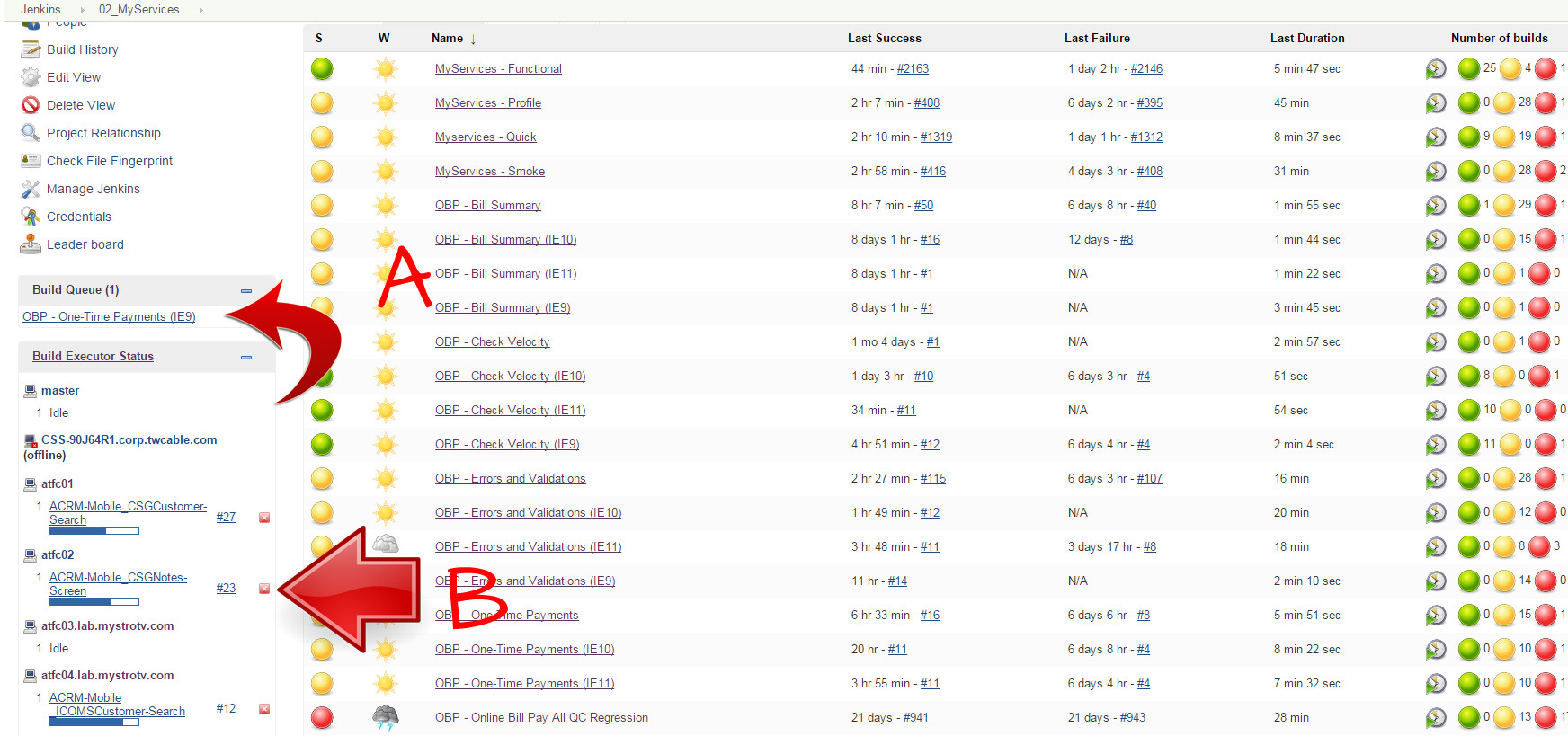
Aside from Internet Explorer, currently all slaves have the same versions of browsers installed. If a particular version of Internet Explorer is necessary, please refer to the slave configuration WIKI chart (<http://mystropedia.corp.mystrotv.com/display/SIT/Automation+Application+Links>) to see where the test needs to run.

1. settingSystem pulls the product’s URL from ATF’s Utility Server. If your application runs on multiple environments, each environment must be setup here: <http://atf.mystro.mystrotv.com:3000/resources/allResources>

The “System” column on the Utility server page is what would be used to pull the default variables necessary to run your test cases for that automation script.

After changing the variables to what is required to run the tests (or keeping default values), simply press the “**Build**” button to start the Jenkins job running.

## Jenkins’s Build Queue and Build Executor Status



1. In the box on the left side of the screen, you will find two important areas: “Build Queue” and “Build Executor Status.”

**Build Queue** is where jobs that have been scheduled to run, but have no free slave to run on, will queue up. Currently, there are only 4 slaves configured to run our Jenkins jobs. So if there are very long jobs running or a problem with some of the slaves being down, oftentimes there could be a job list queued waiting to be run.

**Build Executor Status** shows a list of all of the slaves that are configured with the Jenkins server, and what is currently running on those slaves.

As jobs are in progress, you will notice a blue status bar underneath the name of the job. Clicking on this bar will navigate the page to the “Console Output” page, where you can watch the text output of the test cases executing in real time.

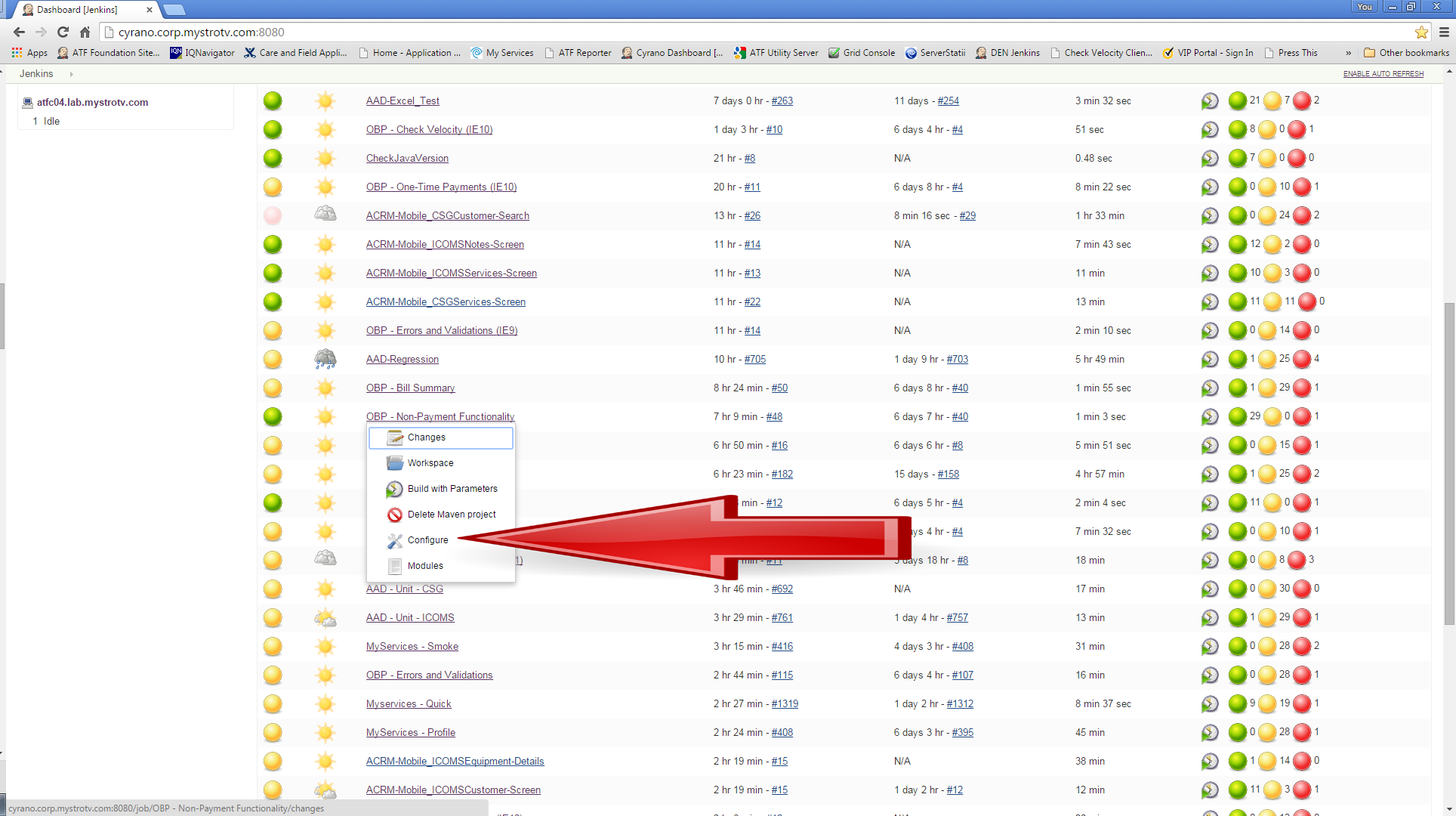
## Cancelling Jenkins Jobs

1. Beside every job, whether in the Build Queue or Build Executor Status, there will be a tiny **red X**. If for any reason you want to cancel a job currently running or one that is queued up, this is how you would stop the job.

Jobs can also be cancelled by using the **red X** located on the “Console Output” page in the upper right corner of the page.

**Be aware, that sometimes stopping a job in the middle of a test execution may leave hanging processes that will require logging into the slave machine and closing opened web browsers and/or killing off zombie webdriver processes. Refer to section “**[**XI. Viewing Browser Test Executions on Remote Machines**](#_Viewing_Browser_Test)**” to see how to login to slave machines.**

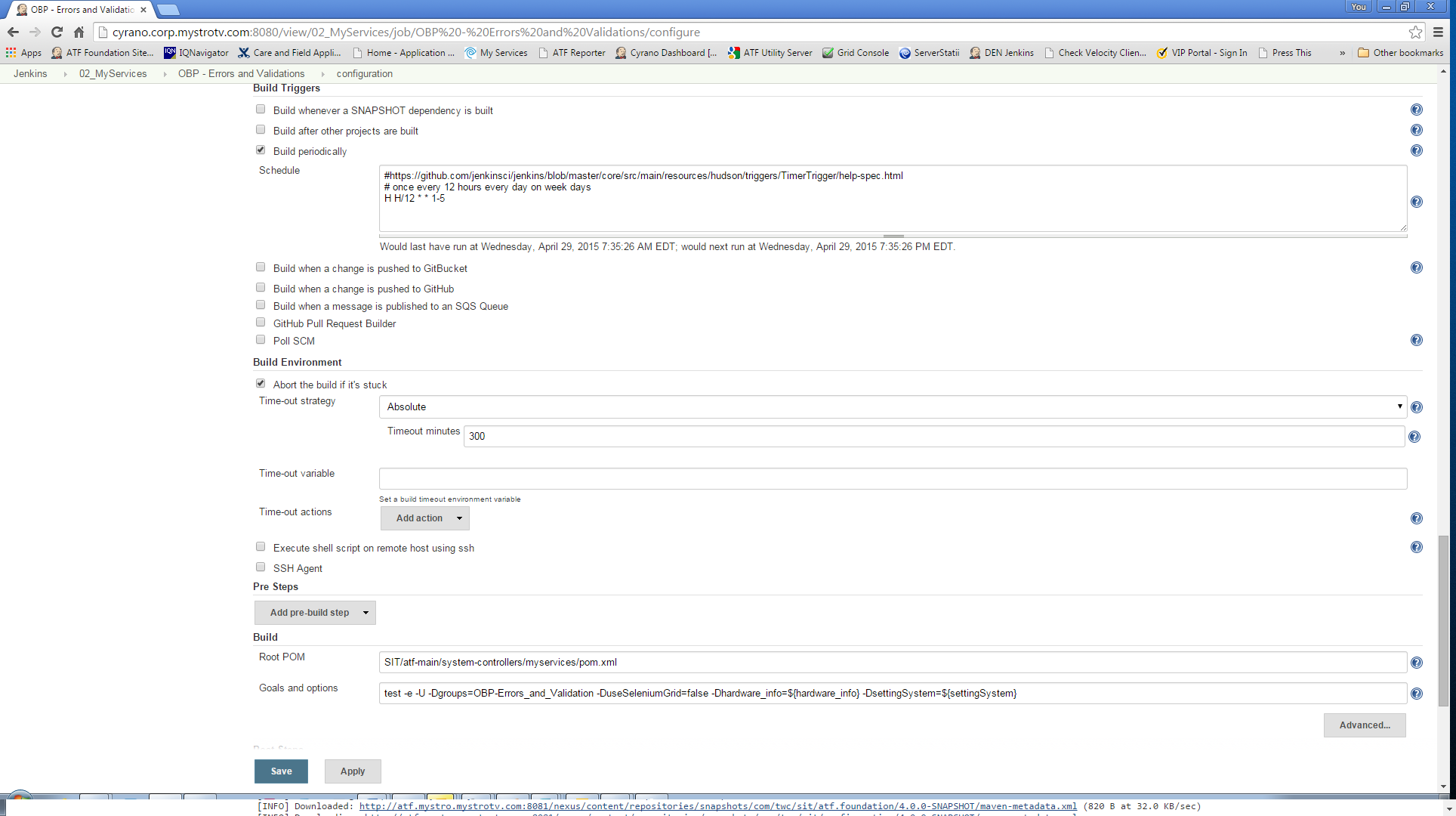
## Configure Jenkins Jobs



Sometimes just running the Jenkins jobs once on demand is not enough. If, for example, you wish to setup the test to run multiple times throughout the day, you can configure a Jenkins job to do just that. Once you Right click on a job name and select “Configure” you will be directed to a web page where there are more advanced options to setup on how the job runs.

Most of the times, this page should only be configured by an automation developer, however, if you understand how cron jobs work, you can feel free to alter the values to change when and how often a particular Jenkins job shall run.

## Changing How and When Jenkins Jobs Run Automatically



To change how and when Jenkins jobs run automatically, go to the configure job page and then scroll to the “Build Triggers” section.

If the job was already set up to run automatically, the “Build periodically” checkbox would be selected, with some syntax like this in the “Schedule” box:

**# once at 9am & 9pm every day**

**H 9, 21 \* \* \***

This special syntax tells Jenkins when and how often to run this particular set of tests. Specifically, each line consists of 5 fields separated by TAB or whitespace:

MINUTE HOUR DAY\_OF\_MONTH MONTH DAY\_OF\_WEEK

MINUTE – Minutes within the hour (0-59 or H).

HOUR – The hour of the day (0-23 or H).

DAY\_OF\_MONTH – The day of the month (1-31).

MONTH – The month (1-12).

DAY\_OF\_WEEK – The day of the week (0-7) where 0 and 7 are Sunday

H – allows periodically scheduled tasks to produce even load on the system. H stands for “hash” and should be used wherever possible. For example, using “0 0 \* \* \*” for a dozen daily jobs will cause a large spike on the Jenkins server at exactly midnight every night. In contrast, using “H H \* \* \*” would still execute a job once a day, but the jobs wont all start at exactly the same minute/second, but would stagger out, better utilizing the limited resources.

**TIPS:**

* To specify multiple values for one field, the following operators are available. In order of precedence,

“\*” – specifies all valid values

“M-N” – specifies a range of values

“M-N/X” or “\*/X” – steps by intervals of X through the specified range or whole valid range

“A,B,…,Z” – enumerates multiple values

* Lines that begin with # or blank lines are ignored
* Examples:

# every fifteen minutes

H/15 \* \* \* \*

# every 10 minutes in the first half of every hour

H(0-29)/10 \* \* \* \*

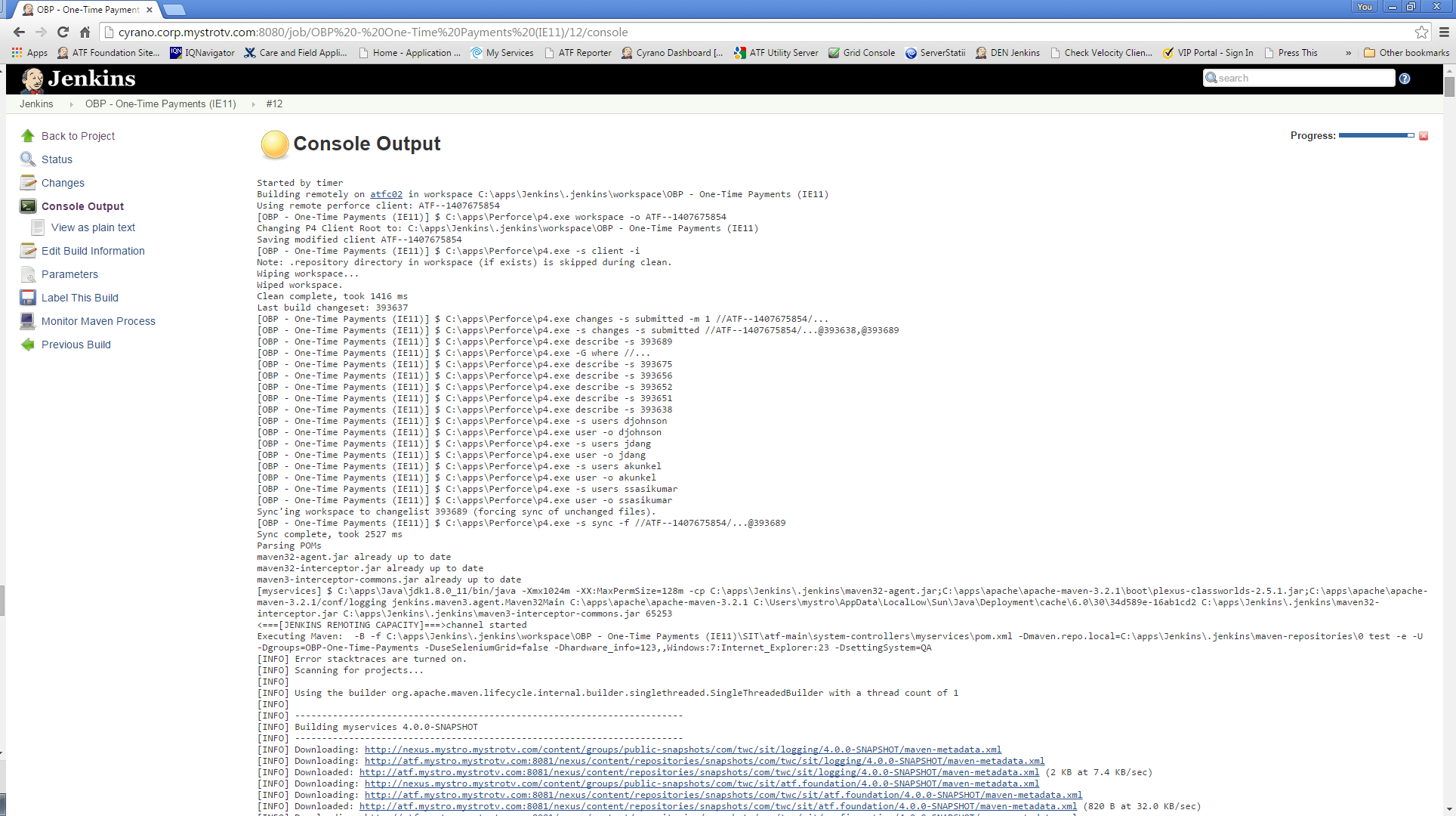
# once every two hours every weekday

H 9-16/2 \* \* 1-5

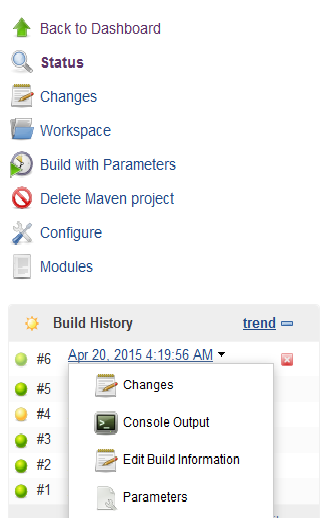
# once a day on the 1st and the 15th of every month except December

H H 1,15 1-11 \*

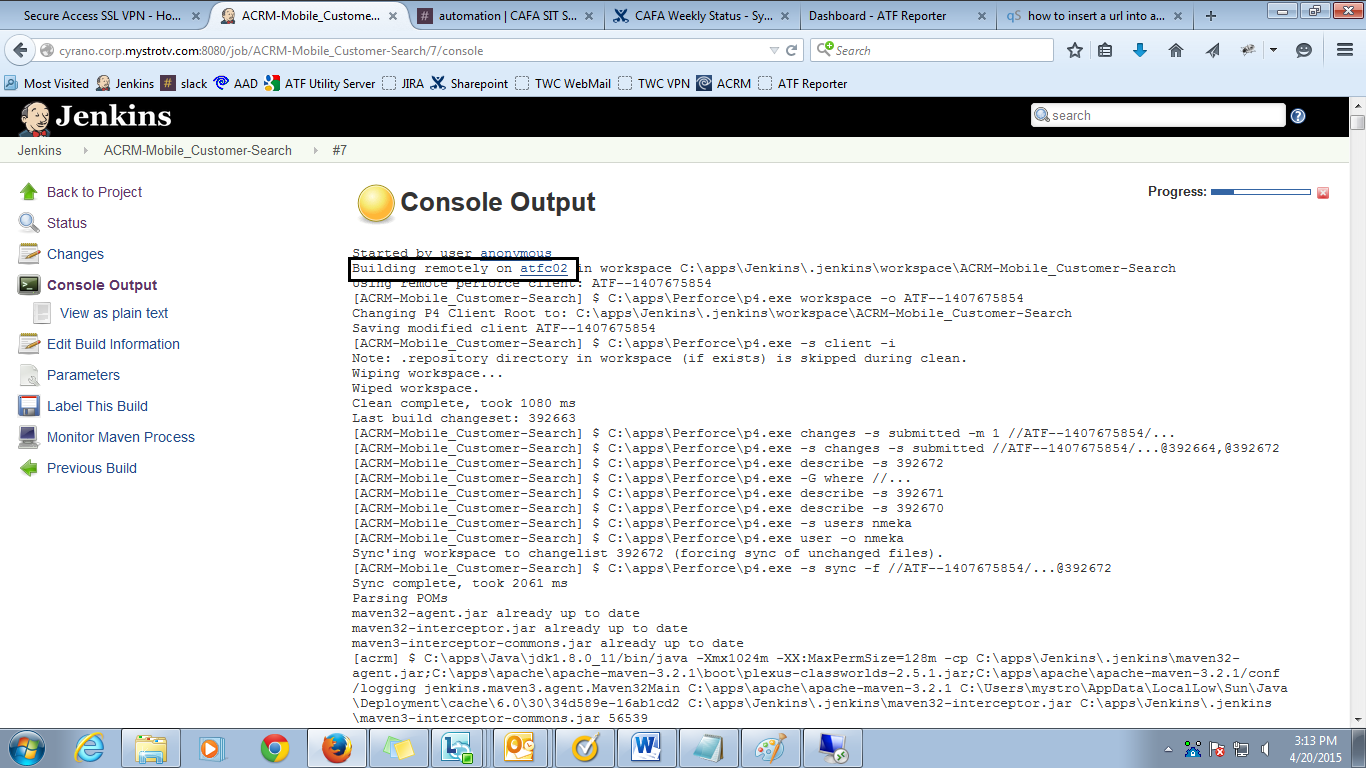
## Console Output



As soon as a build starts running on Jenkins, you are able to view the “console output” or the text version and error messages of the job currently in progress. There are multiple ways to get to this output-- the easiest way is to click on the little blue status bar under the currently running job.

****

You can also view the job output by clicking on the name of the build currently running and select Console Output from the dropdown menu.

****

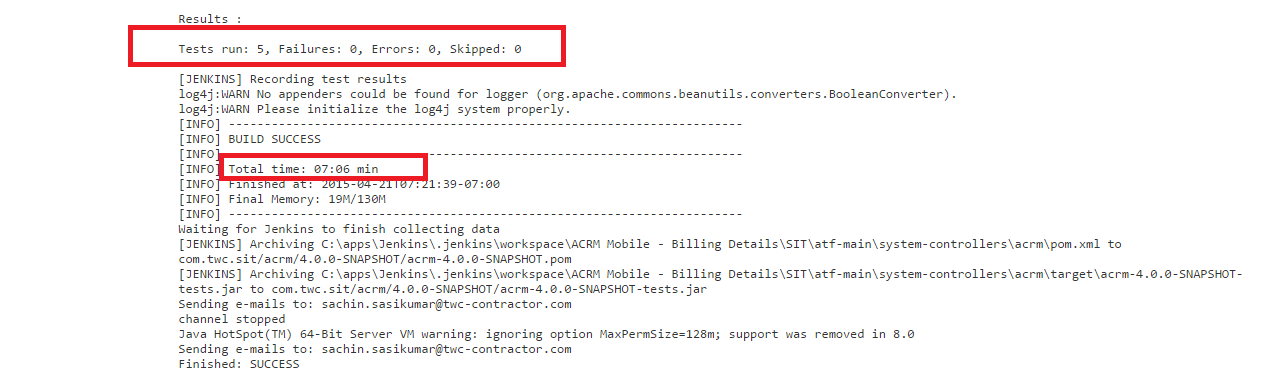
The console output screen is displayed with step-by-step test cases execution. A person watching the output in real-time can follow along with the test execution output messages.

Most of the output that is displayed is debug information for automation developers to view and use to resolve issues. Some helpful information given at the beginning of each run is the remote slave machine which has been selected to run this particular build. (See above screenshot highlighted area.) This information is useful for debugging purposes, especially if the entire build has failed to run. It may be possible that something is affecting this slave machine which is causing the build to be unstable.



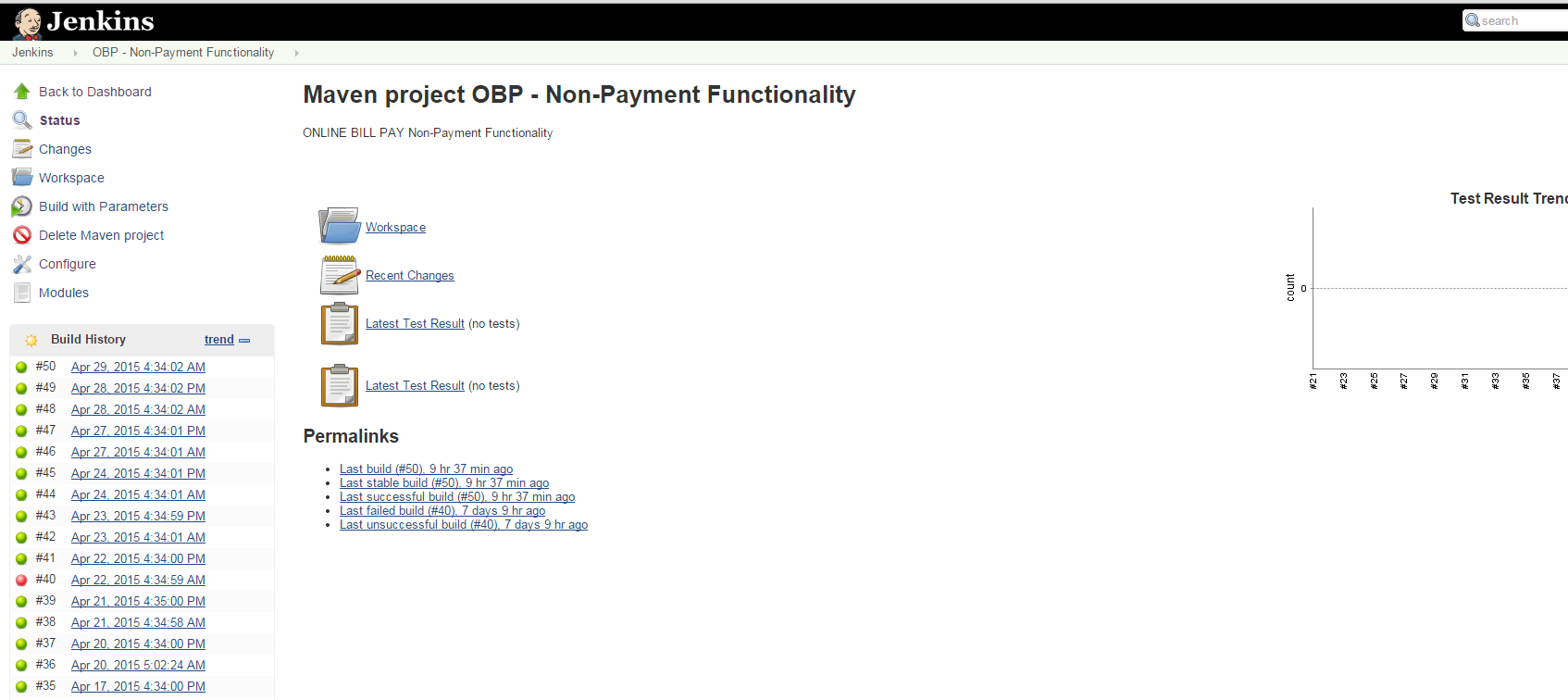
As you scroll down the console output screen, you might encounter errors as shown above in the stack trace screenshot. The highlighted area explains the test case details, the line number in the actual code where it is failing as well as the reason why it is failing. For example, in this screenshot, this individual test case is failing because it cannot find an element (or object on the page) which it was expecting to find. The automation developer would check in the code for ACRMCSGBillingDetailsScreenTest.java at line 146, to figure out what object it is searching for, and why it is failing. A manual tester can copy and paste this stack trace error message in a bug to the automation developer if this error persists for a number of consecutive job executions.

When the final test case has completed, there will be an execution summary at the very end, listing the test run summary. The results will show the count of how many test cases were executed in total, how many failed, how many erred out, and how many tests were purposefully skipped. It would also mention the total time to execute all the test cases. A notification email is sent to the Automation team member whose email id is configured for that Jenkins item.

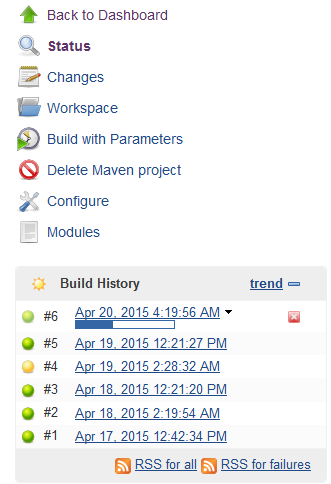


## Viewing Build Results

Once a build has completed, you can review the results of the build by clicking the name of the Jenkins job from the Jenkins Home Page. The page which opens up will give a quick overview of all of the jobs which has run, along with the latest results of the last build.



Clicking on build # in the “Build History” column will open up specific results for that particular run.

****



From the individual build detail page, clicking the ‘Test Result’ link will take you to the “Test Result” screen which identifies which test cases failed to run, and the reasons for the failure.

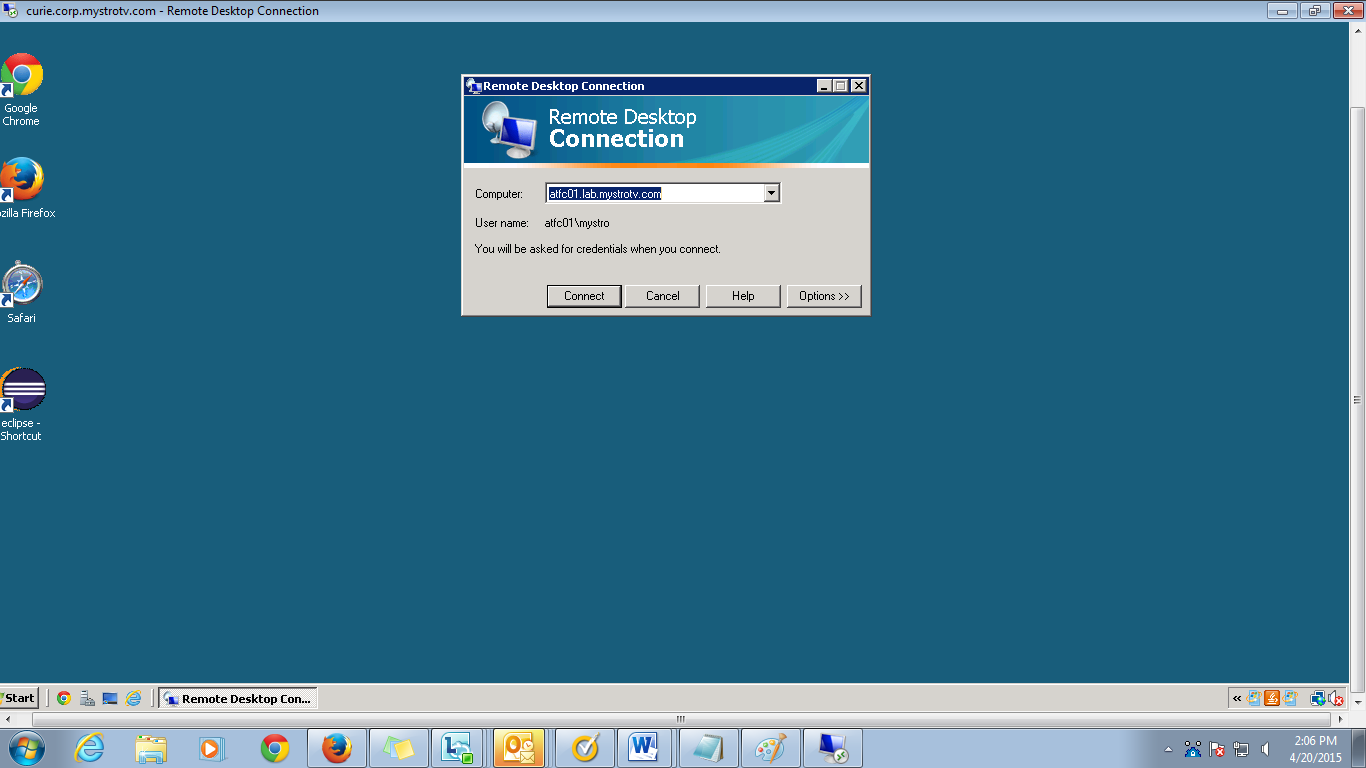
Clicking on the individual test result link will direct you to a page that shows the error that happened, and the resulting stack trace to help debug it. 

## Viewing Browser Test Executions on Remote Machines

Any Jenkins job that is currently running will be executing the actual motions in a browser on one of the slave machines. Any person with remote desktop login access to one of the corp.mystrotv.com VM machines can then remote desktop login into a slave to watch the automation run in the browser. Currently, to view the browser automation running, you must have remote login access to one of these:

1. Cyrano.corp.mystrotv.com,
2. Curie.corp.mystrotv.com,
3. Challis.corp.mystrotv.com,
4. Chaffee.corp.mystrotv.com,
5. Cantor.corp.mystrotv.com, or
6. Carnot.corp.mystrotv.com

Once you have logged in, (using your regular TWC login credentials), to any of the remote desktop machines listed above, open another remote desktop connection. This time you would need to enter in the name of the slave for which the job is running on. (i.e., atfc01.lab.mystrotv.com). The slave names can be found from the Jenkins home page just above the job which you wish to monitor. The slave username/password combinations will always be mystro/mystro.

****