https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s\_v1=39098&elqem=3295803\_EM\_CN\_21-07\_NEWSLETTER\_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360

[Skip to content](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "content_container)

切换主导航

[](https://ww2.mathworks.cn/?s_tid=gn_logo)

# Technical Articles and Newsletters

Toggle Sub Navigation

窗体顶端

技术文章

技术文章 Search

窗体底端

# Continuous Integration for Verification of Simulink Models

By David Boissy, Paul Urban, Krishna Balasubramanian, Pablo Romero Cumbreras, and Colin Branch, MathWorks

Continuous integration (CI) is gaining in popularity and becoming an integral part of Model-Based Design. But what is CI? What are its benefits, and what problems does it attempt to solve? How does Simulink® fit into the CI ecosystem? And how can you best leverage CI for your projects?

If you are familiar with Model-Based Design but new to CI, you may be asking yourself these questions. In this technical article, we explore a common CI workflow and apply it to Model-Based Design. Then, we walk through an example of that workflow using Jenkins™, GitLab®, and Simulink Test™.

The project used in this example is [available for download](https://ww2.mathworks.cn/matlabcentral/fileexchange/77557-continuous-integration-for-verification-of-simulink-models).

## What Is CI?

CI is an agile methodology best practice in which developers regularly submit and merge their source code changes into a central repository. These “change sets” are then automatically built, qualified, and released. Figure 1 illustrates this basic CI workflow together with the development workflow.

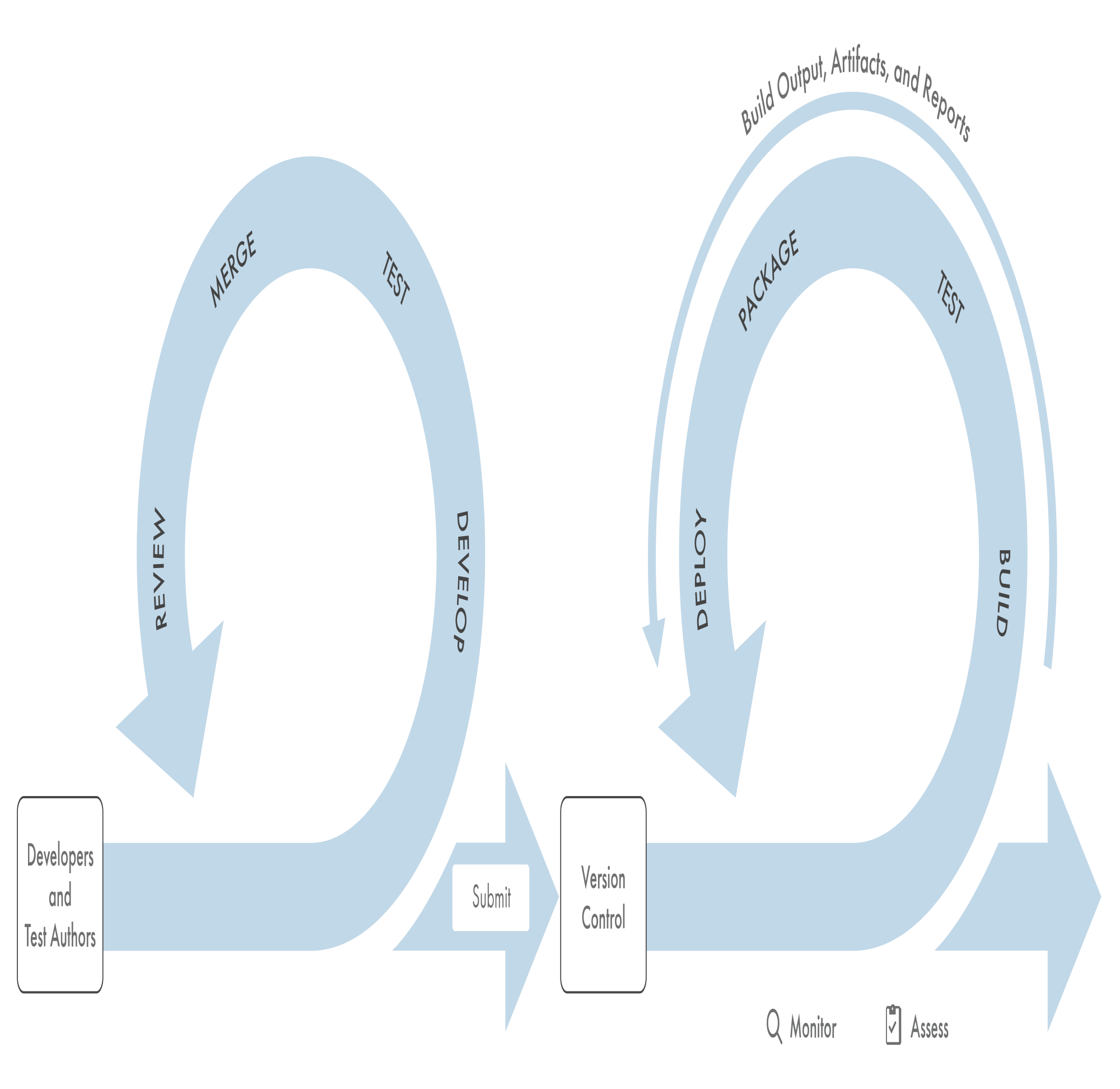


Figure 1. CI workflow.

In the development part of the workflow, models and tests are developed, verified, merged, reviewed, and submitted to a version control system on the developer desktop. The version control system then triggers the automated CI portion of the workflow. The key parts of the CI workflow are:

**Build:** Source code and models become object files and executables.

**Test:** Testing is performed as a quality gate.

**Package:** Executables, documentation, artifacts, and other deliverables are bundled for delivery to end users.

**Deploy:** Packages are deployed to the production environment.

Together, these four steps are known as the CI “pipeline.” The pipeline is typically automated, and it can take anywhere from minutes to days to complete, depending on the system. It is worth noting that throughout these steps, numerous artifacts are created, such as bills of materials, test results, and reports.

CI workflows are often paired with developer workflows related to version control systems. In these workflows, developers often keep their changes in local repositories and use a local CI pipeline to qualify their changes prior to deployment.

## What Are the Benefits of CI?

Teams that have implemented CI typically report the following benefits:

* **Repeatability.** The CI pipeline provides a consistent and repeatable automated process for building, testing, packaging, and deployment. Repeatable automation allows developers to focus on necessary work and save time on a project. It is also an important aspect of risk reduction, and is often a requirement for certification.
* **Quality assurance.** Manual testing is effective, but it is often based on days-old snapshots and lacks repeatability. With CI, changes are always tested against the most recent code base.
* **Reduced development time.** Repeatable processes with built-in quality assurance lead to faster delivery of high-quality products. Automated deployment means that your code is always production-ready.
* **Improved collaboration.** With CI, developers have a defined process for managing change sets and merging their code into the production line. Consistent processes make managing large teams possible and reduce the cost of ramping up new developers.
* **Audit-ready code.**The CI workflow provides an extensive audit trail. For every change making its way through the CI pipeline, it’s possible to identify who made the change, who reviewed it, and the nature of the change, as well as dependencies, tests and their results, and any number of related reports and artifacts generated along the way.

## How Does Model-Based Design Fit into CI?

By design, the CI workflow and tools are language- and domain-neutral（区域中立）. This means that the challenge is to *teach* CI tools, systems, and processes to *speak* Model-Based Design—in other words, to make Simulink® and related tools the *lingua franca* of the CI workflow.

This can be done by integrating three key components of Model-Based Design into the CI workflow: verification, code generation, and testing (Figure 2). Model-Based Design emphasizes early verification, which maps to the CI pipeline with a Verify phase before the Build phase. Code generation takes place in the Build phase. Dynamic testing through simulation and static analysis of generated code can be done in the Test phase.

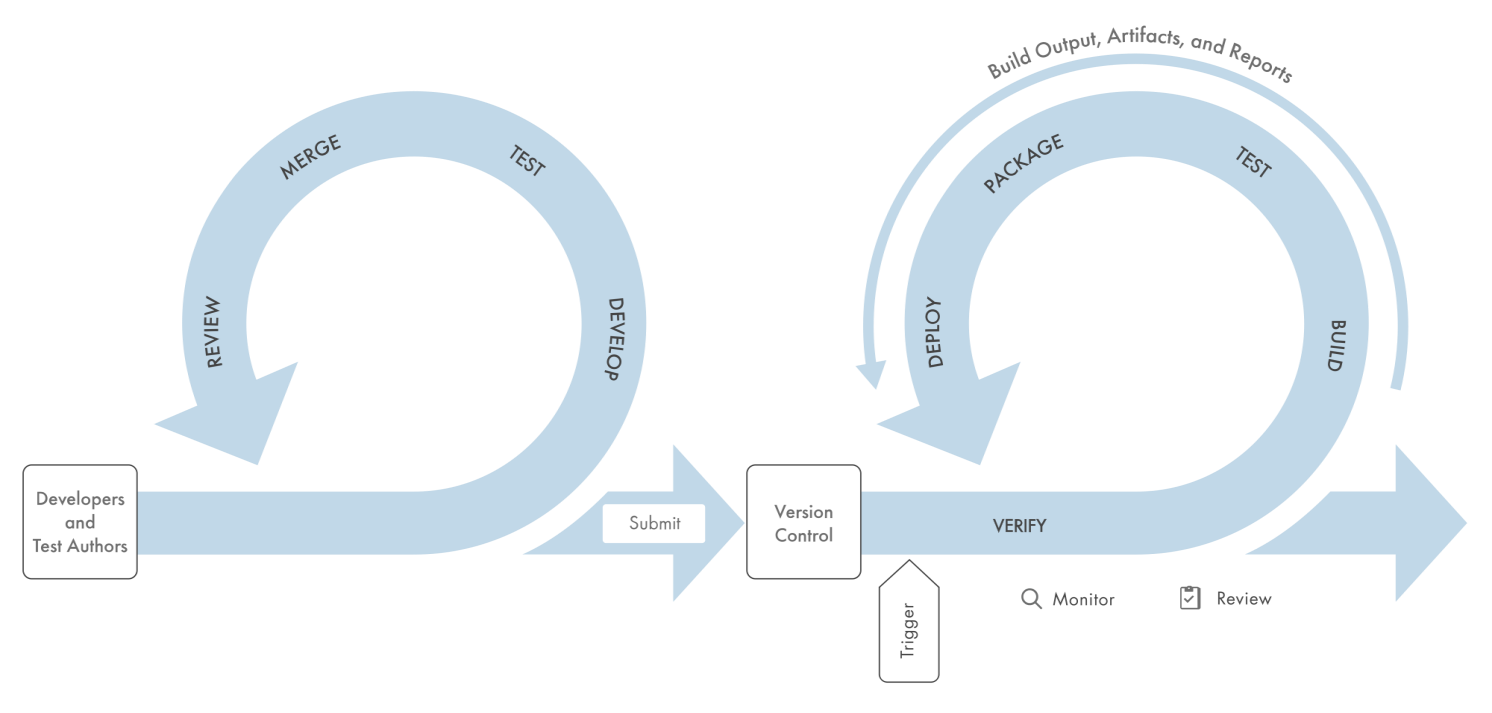
[](https://ww2.mathworks.cn/content/mathworks/cn/zh/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models/_jcr_content/mainParsys/image_0_copy_copy_co_1934420530.adapt.full.high.svg/1621386462846.svg)

Figure 2. Model-Based Design mapped to CI pipeline.

Here is an overview of how we *teach* the CI workflow to *speak* Model-Based Design:

**Develop.** MATLAB®, Simulink, coders, and toolboxes are used for development activities. MATLAB Projects are used to organize work, collaborate, and interface with version control systems.

**Test.**Simulink Check™ is used to perform model quality checks before simulation and code generation. Simulink Test™ is used to develop, manage, and execute simulation-based tests. Simulink Coverage™ is used to measure coverage and assess test effectiveness. The quality checks, test results, and coverage metrics can then be used as a quality gate for developers to qualify their work.

**Merge.**The Compare Files and Folders feature of MATLAB is used to compare and merge MATLAB files. The Model Comparison tool is used to compare and merge Simulink models.

**Review.** Review is the final step in the quality process before changes are submitted to the version control system. Changes to MATLAB scripts and Simulink models are reviewed here. Test results from prequalification are also reviewed as a final quality gate prior to submission.

**Submit.** MATLAB Projects provide an interface to version control systems.

**Verify.**Simulink Check, the same tool used for local verification, is used for automated verification within the CI system.

**Build.** MATLAB Coder™, Simulink Coder™, and Embedder Coder® are used to generate code for software-in-the-loop (SIL) testing.

**Test.** Simulink Test, the same tool used for local testing, is used for automated testing within the CI system.

**Package and deploy.** Packaging is where executables, documentation, artifacts, and other deliverables are bundled up for delivery to end users. Deployment is the release of the packaged software. In workflows for Model-Based Design, these phases vary widely among organizations and groups, and often involve bundling different builds and certification artifacts into a product ready for delivery to other teams.

Modern development tools and practices enable developers to create more robust systems and test functionality early and often.  When a CI system is integrated into the workflow, unit-level testing and system-level testing are automated. This means that the developer can focus on developing new features, not on verifying that features have been integrated correctly.

The following case study describes a workflow that incorporates CI and Model-Based Design.

## Case Study: A Simulink Model Verified, Built, and Tested Within a CI System

In this example we use Model-Based Design with CI to perform requirements-based testing on an automotive lane-following system (Figure 3).

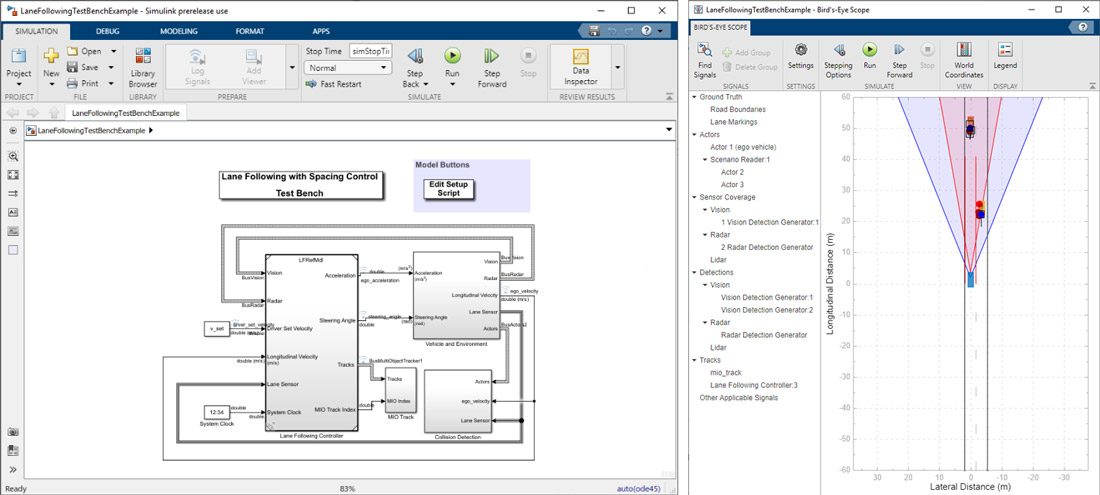


Figure 3. Lane-following system model.

The pipeline we’ll be using (Figure 4) is executed with each Jenkins build.

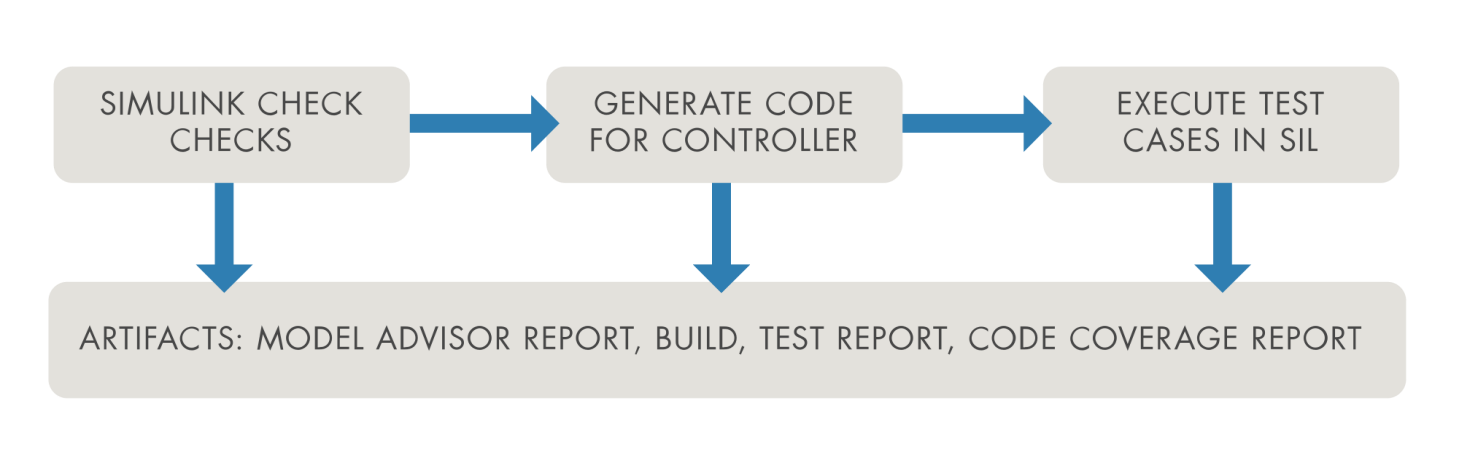


Figure 4. Pipeline for lane-following example.

The phases in the pipeline are as follows:

1. Verify standards compliance: A MATLAB Unit script runs a simple Model Advisor check. The assessment criteria ensure that the model does not have unconnected lines.
2. Build model: A MATLAB Unit test file builds production SIL code for our model. The assessment criteria passes if the build succeeds without warning.
3. Execute test cases: A test suite in Simulink Test uses several driving scenarios(情景) to test the lane-following controller. Three assessment criteria are used to verify satisfactory operation of the controller:
   * Collision avoidance: The ego car does not collide with the lead car at any point during the driving scenario.
   * Safe distance maintenance: The time gap between the ego car and the lead car is above 1.5 seconds. The time gap between the two cars is defined as the ratio of the calculated headway and the ego car velocity.
   * Lane following: The lateral deviation from the centerline of the lane is within 0.2 m.
4. Package artifacts: Each of the previous stages produces artifacts, including a Model Advisor report, a generated executable, and a set of test results that can be archived for future use or reference.

## Workflow Steps

The workflow consists of the following steps (Figure 5):

1. **Trigger** a build in Jenkins and observe that the Verify and Build stages pass.
2. **Detect** a test case failure in Jenkins.
3. **Reproduce** the issue on our desktop MATLAB.
4. **Fix the issue** in the model by relaxing the assessment criteria.
5. **Test locally** to ensure the test case passes.
6. **Merge and review** the changes on the testing branch.
7. **Commit** the change to Git and trigger a build in Jenkins.
8. **Verify, build, and test** in Jenkins.

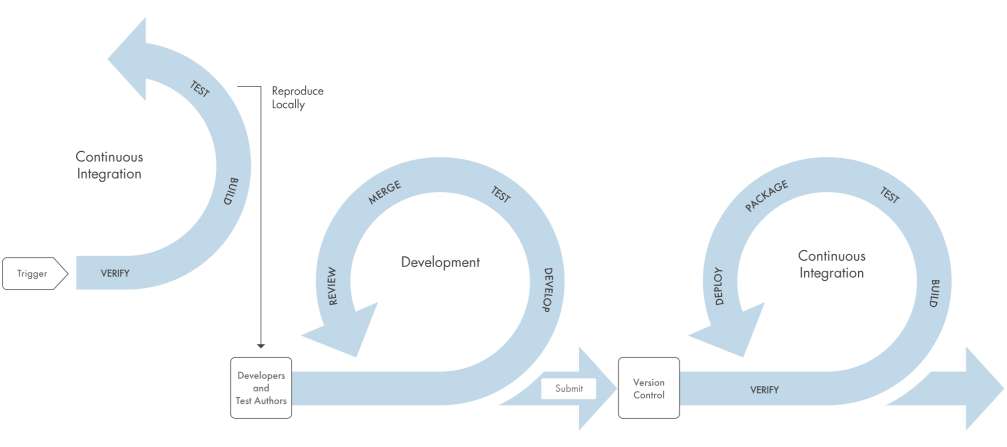
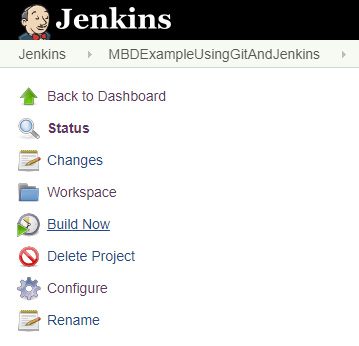
[](https://ww2.mathworks.cn/content/mathworks/cn/zh/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models/_jcr_content/mainParsys/image_0_copy_copy_co_1850942158.adapt.full.high.svg/1621386462929.svg)

Figure 5. Example workflow.

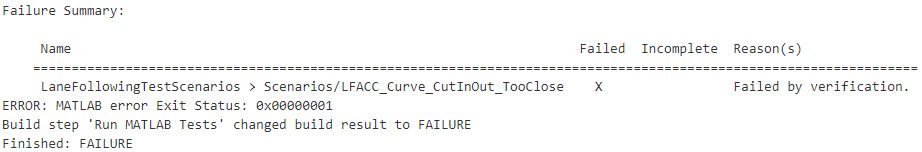
Our first failed pass through the CI loop is illustrated at the top left. It shows the CI test failure, local reproduction, criteria relaxing, and successful completion of the CI workflow.

## Workflow Details

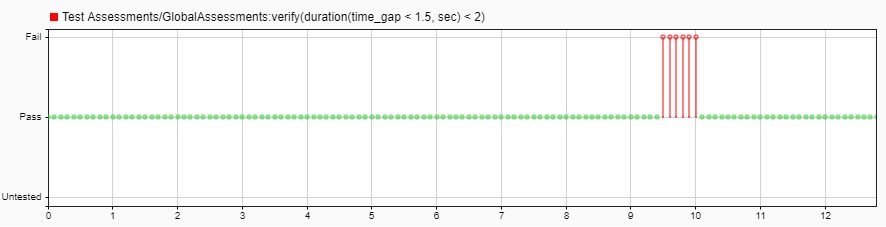
1. We begin by **triggering** a build in Jenkins by selecting **Build Now**. The Simulink Check checks and code generation pass.



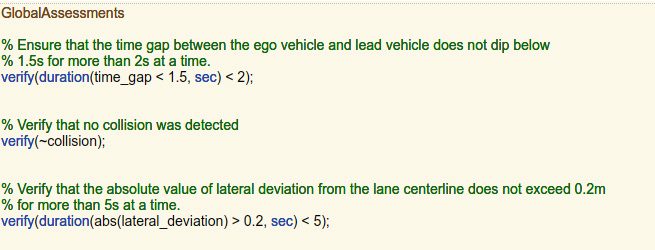
1. Next, we **detect** a test case failure in the second verify phase. The test case LFACC\_Curve\_CutInOut\_TooClose in test suite LaneFollowingTestScenarios fails the assessment criteria.



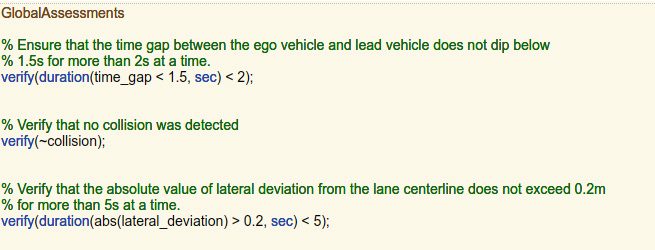
1. To understand the failure better, we **reproduce**the failure locally using Simulink Test. We open test file LaneFollowingTestScenarios.mldatx and run test case LFACC\_Curve\_CutInOut\_TooClose. Note that it fails the Safe Distance assessment criteria. More flexibility in establishing the time gap between the lead car and ego car is required.



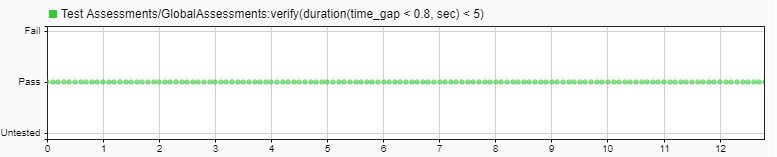
1. With an understanding of the problem, we now **fix the issue**. We open the LaneFollowingTestBenchExample.slx model and navigate to the Collision Detection/Test Assessments Test Sequence block. The first assessment asserts that the time gap between the ego and the lead car should not dip below 1.5 seconds for more than two seconds at a time.



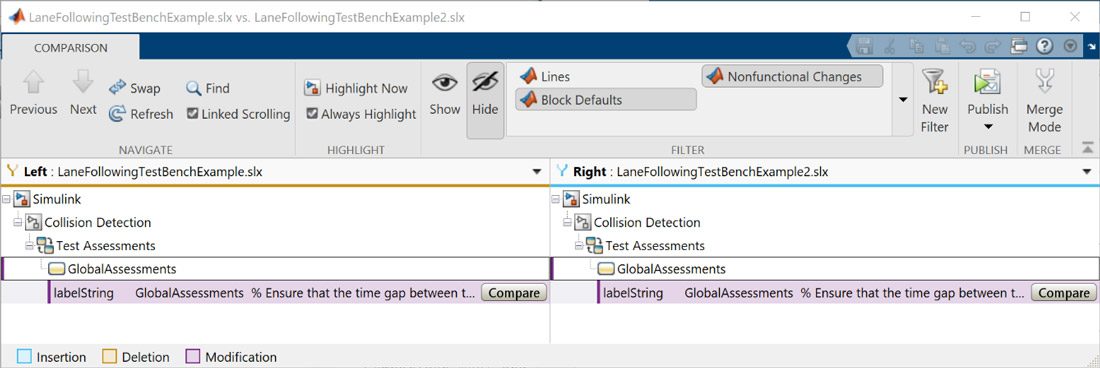
This assessment is too restrictive for the aggressive driving maneuver being tested. For the purposes of this example, we relax the assessment criteria to ensure that the time gap does not dip below 0.8 seconds for more than five seconds at a time.



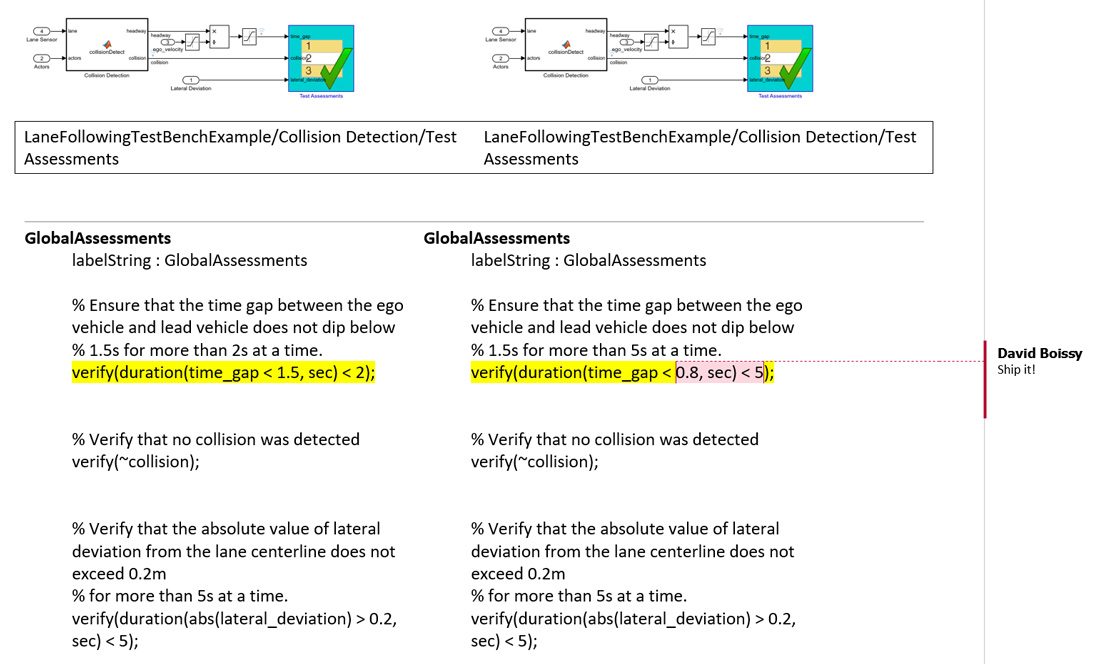
1. The issue appears fixed in our simulation. To confirm, we **test locally,**saving the model and rerunning the test in the test manager. Notice that it passes with the new assessment criteria.



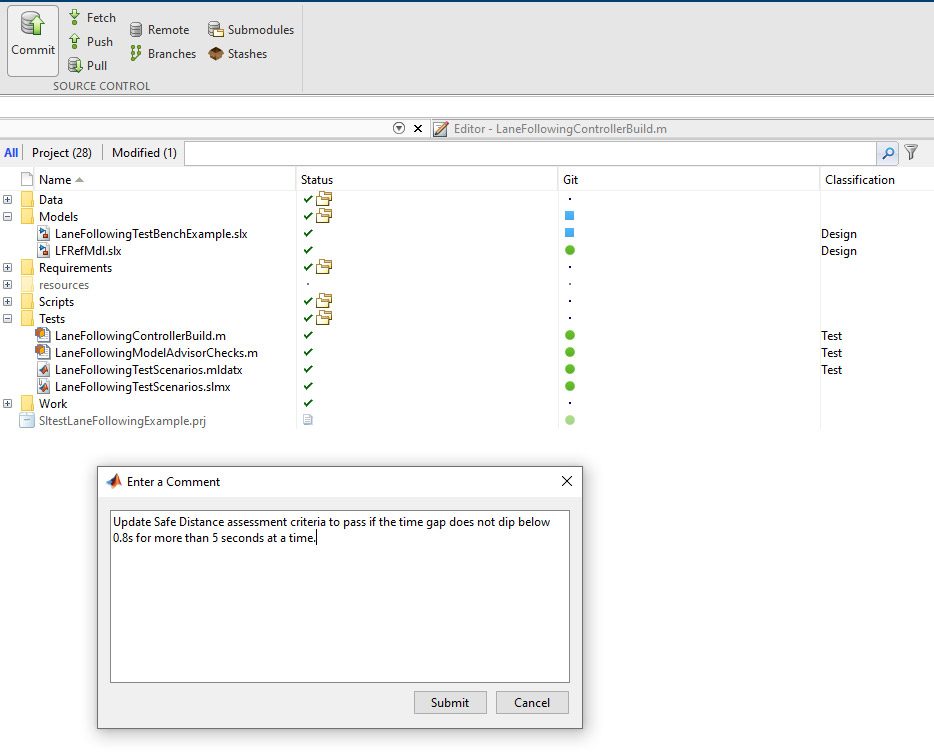
1. We have fixed the issue and verified locally. We now use the Model Comparison tool to **review**the changes before committing them to version control.



We could also use the Publish feature of the Model Comparison tool to review the code.



1. With the bug fixed, we push these changes to GitLab with MATLAB projects, adding a commit message to note the change to the assessment criteria.



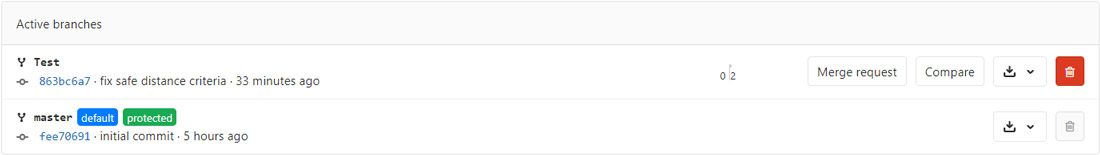
We then note the latest commit in GitLab.

GitLab automatically triggers a build in Jenkins. The Jenkins Project dashboard shows the build status and progress.

IMG_271

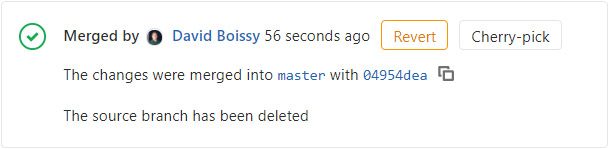
1. The Jenkins build runs. We see that the **verify, build, and test** pipeline phases now pass.

We can now start a merge request to merge the changes in the test branch into the master branch. In GitLab, under Repository we select Branches then click **Merge request** next to the latest commit on the test branch.

[](https://ww2.mathworks.cn/content/mathworks/cn/zh/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models/_jcr_content/mainParsys/image_0_copy_copy_co_1080874460.adapt.full.high.jpg/1621386463232.jpg)

We complete the form and submit the merge request.

As the owners of the branch, we can accept the merge request by clicking the Merge button. All changes are now captured on the main branch.



## Using the Example: Tools, Resources, and Requirements

The following sections outline resources to help you get started, the tools you will require, and ~~how~~ what they should be configured.

### Configuring Systems

Jenkins is leveraged as our CI system and GitLab as our version control system. MATLAB, Jenkins, and GitLab must be configured to work together. The following tutorials will help with setup.

[Configure Our MATLAB Project](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "configure_project)

[Configure Jenkins](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "configure_jenkins)

[Configure GitLab to Trigger Jenkins](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "configure_gitlab)

The tutorials are specific to GitLab and Jenkins, but the concepts can apply to other version control and CI systems.

### Tools Required

The following tools are required for this example:

* [Jenkins](https://www.jenkins.io/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank) installation version 2.7.3 or later. Jenkins is used for continuous integration.
* [MATLAB Plugin for Jenkins](https://plugins.jenkins.io/matlab/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank) version 1.0.3 or later. MATLAB, Simulink, and Simulink Test all leverage this plugin to communicate with Jenkins. [Learn more on GitHub](https://github.com/jenkinsci/matlab-plugin" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank).
* Additional plugins required:
  + [GitLab](https://wiki.jenkins.io/display/JENKINS/GitLab+Plugin" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank)
  + [TAP Plugin](https://wiki.jenkins.io/display/JENKINS/TAP+Plugin" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank)
* [GitLab](https://about.gitlab.com/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank) account. GitLab is used for source control and is available as a cloud service. MATLAB Projects include a Git interface for communication with GitLab.

## License Considerations for CI

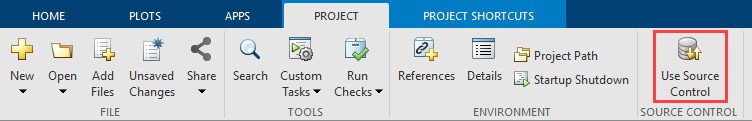
If you plan to perform CI on many hosts or on the cloud, contact [continuous-integration@mathworks.com](mailto:continuous-integration@mathworks.com) for help. *Note: Transformational products such as MathWorks coder and compiler products may require Client Access Licenses (CAL).*

## Appendix: Configuring MATLAB, GitLab, and Jenkins

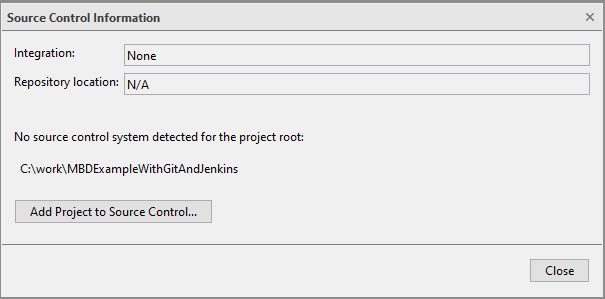
### Step 1. Configure MATLAB project to use source control

The first step in our example is to configure our project to use source control with GitLab.

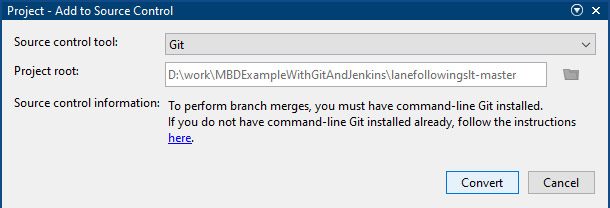
1. Create a new directory named MBDExampleWithGitAndJenkins, load the example into it, and open the MATLAB Project MBDExampleWithGitAndJenkins.prj.
2. In GitLab, create a new project that will be the remote repository. Name it MBDExampleWithGitAndJenkins and record the URL where it is hosted.
3. In MATLAB, convert the project to use source control. On the Project tab, click **Use Source Control**.



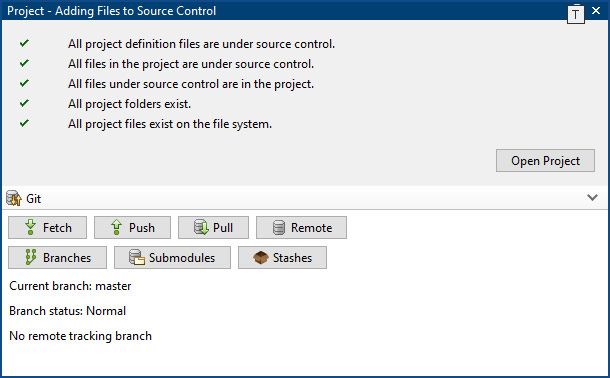
Click **Add Project to Source Control**.



1. Click **Convert**.



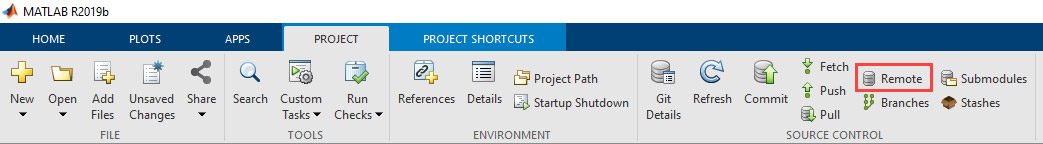
1. Click **Open Project** when done.



The project is now under local Git source control.

### Step 2. Commit changes and push local repository to GitLab

1. On the Project tab, click **Remote**.

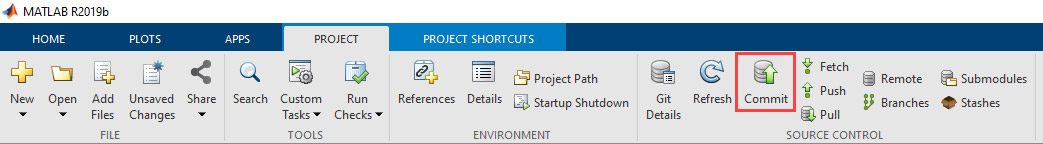


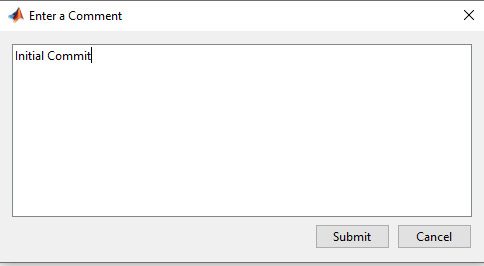
1. Specify the URL of the remote origin in GitLab.



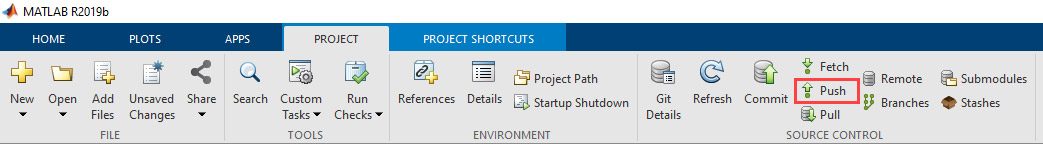
Click **Validate** to ensure the connection to the remote repository is successful and click **OK**. The project is now configured to push and pull changes with GitLab.

1. Click **Commit** to perform an initial commit.

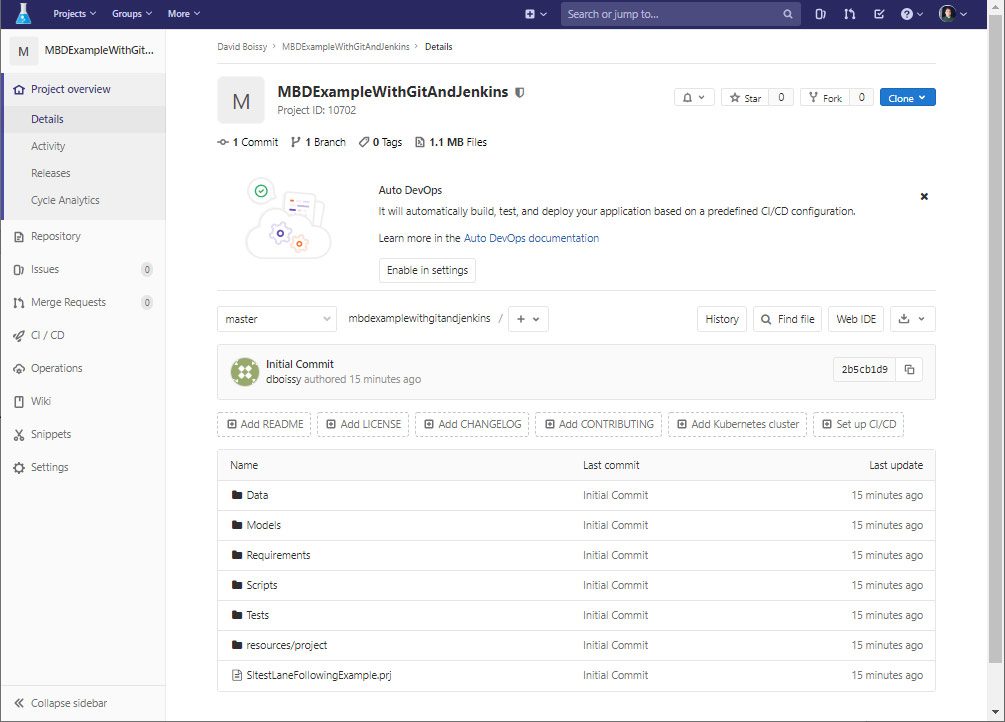




1. Click **Push** to push all changes from the local repository to the remote GitLab repository.



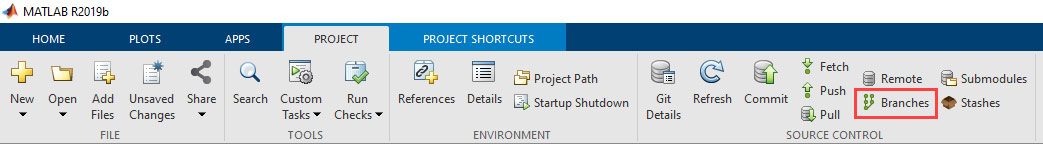
1. Refresh the GitLab dashboard and observe the contents of the MATLAB project.



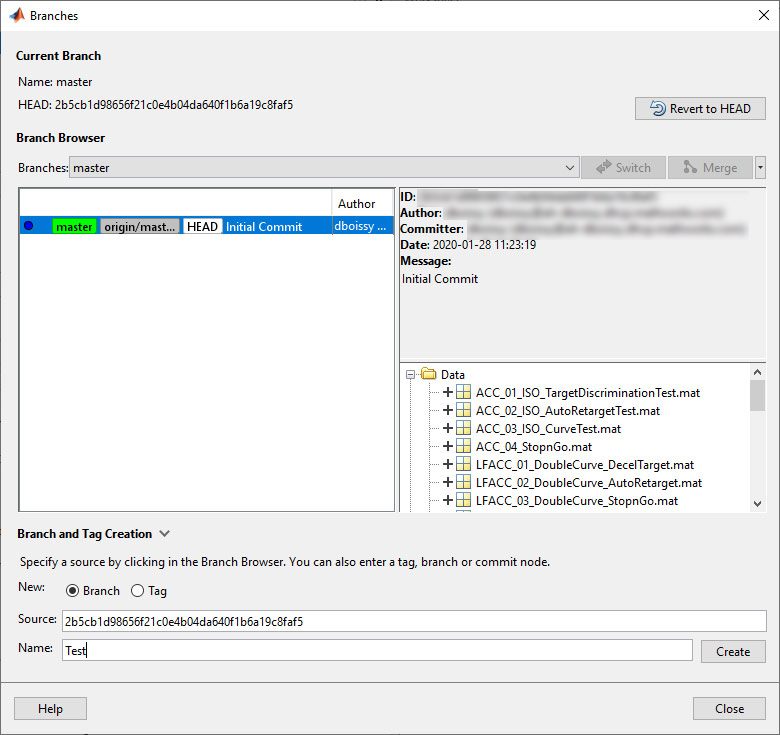
### Step 3: Create testing branch

In this step we create a testing branch for testing and verifying changes before merging with the main branch.

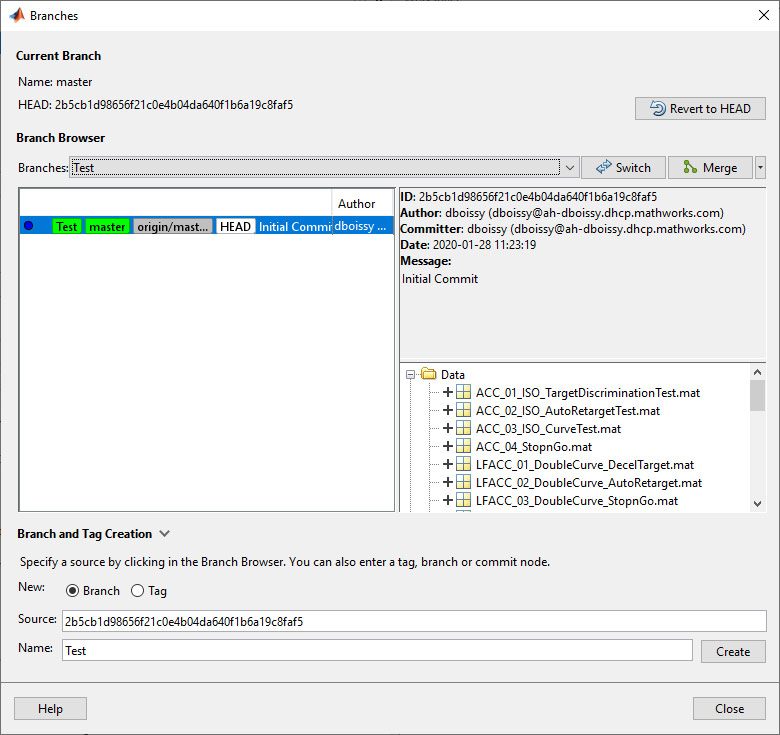
1. Click **Branches**.



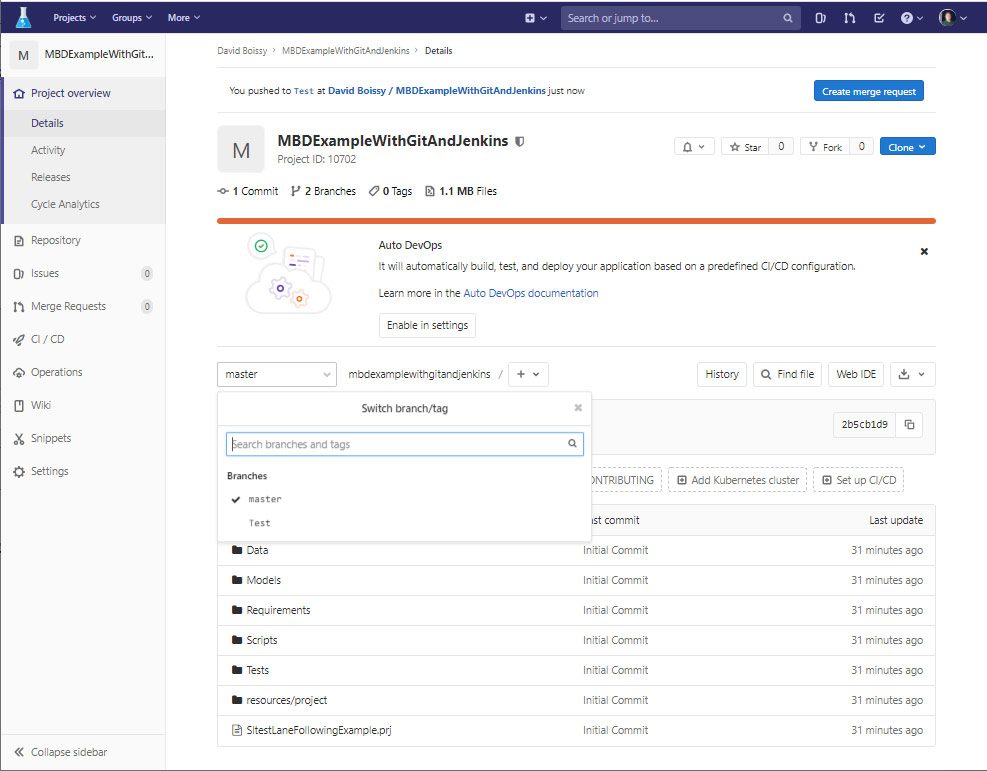
1. Expand the **Branch and Tag Creation**section, name the branch “Test,” and click **Create**.



1. Observe Test in the branch browser. From the Test branch click **Switch** then **Close**.

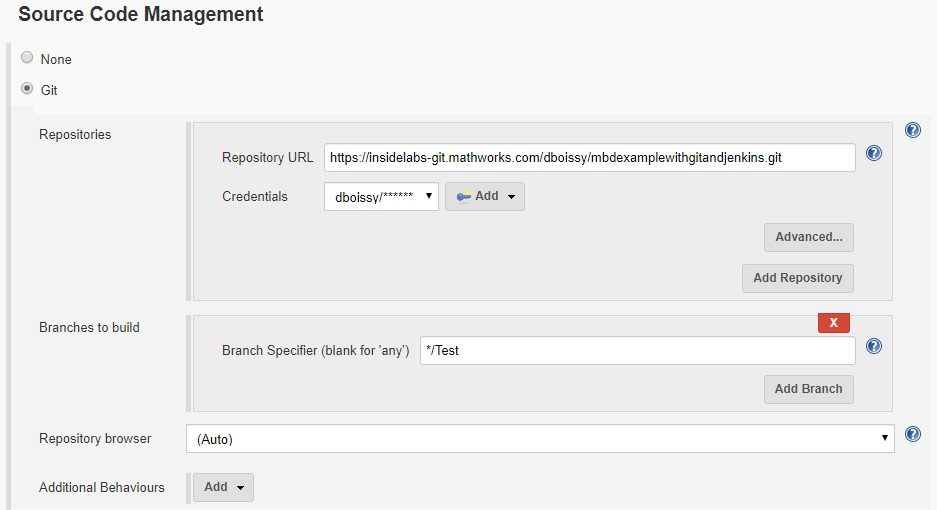


1. In MATLAB select **Push** to push these changes to GitLab and observe the Test branch in GitLab.

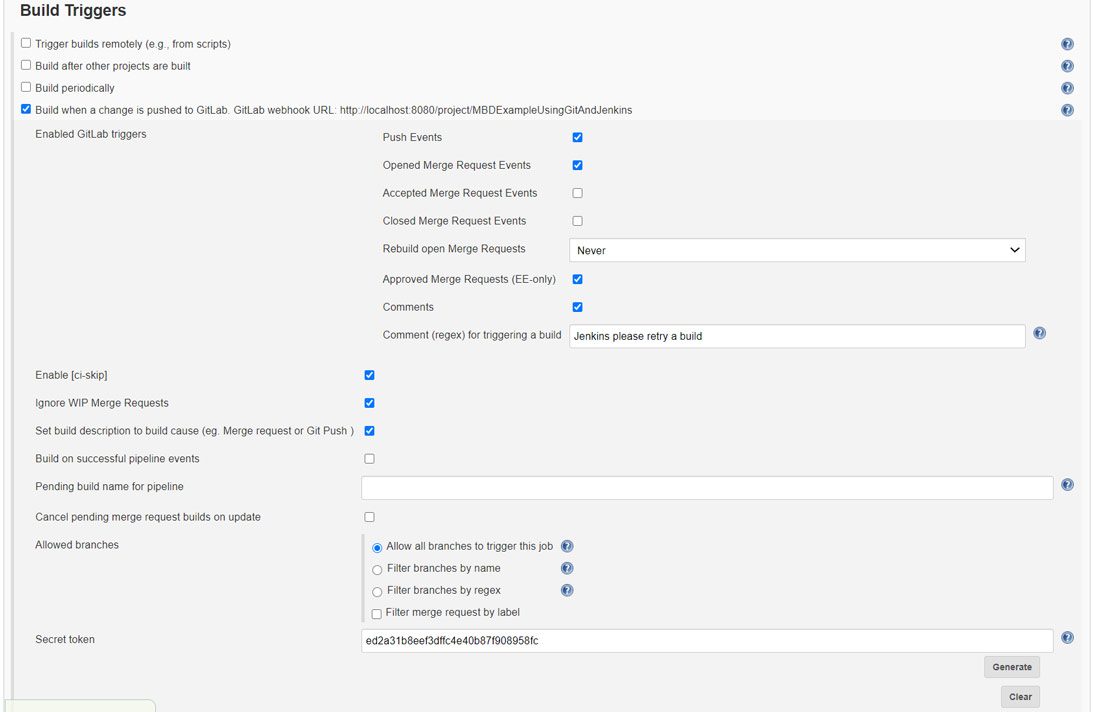


### Step 4: Configure Jenkins to call MATLAB

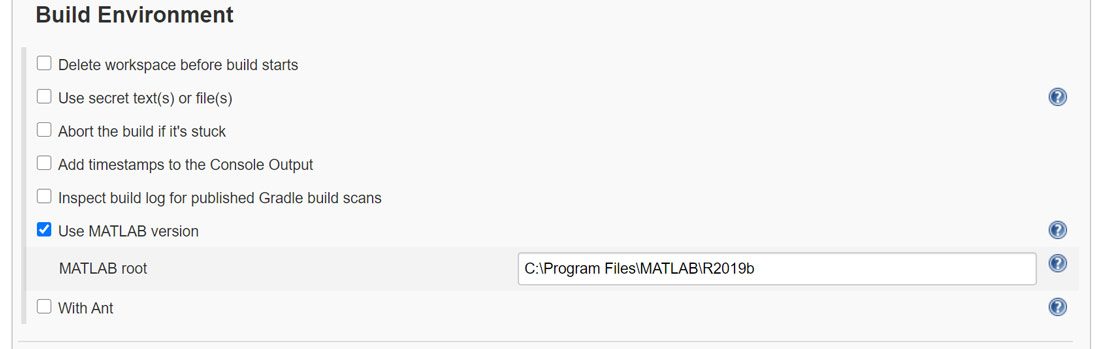
1. Install two required plugins:
   * [GitLab plugin](https://plugins.jenkins.io/gitlab-plugin/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank) – This plugin allows GitLab to trigger Jenkins builds and display their results in the GitLab UI.
   * [MATLAB plugin](https://plugins.jenkins.io/matlab/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank) – This plugin integrates MATLAB with Jenkins and provides the Jenkins interface to call MATLAB and Simulink.
2. Select **New Item** and create a new FreeStyle project named MBDExampleUsingGitAndJenkins.
3. Under Source Code Management, enable Git, point Jenkins to our GitLab repository, and enter the **Test** branch to build. *Note: Login or password and GitLab API token are required.*



1. Configure the build trigger to run a build when a push request is made to the Test branch in GitLab. In the Build Triggers section select **Advanced >** **secret token**. This token is used by GitLab to request a build and authenticate with Jenkins. Make a note of the secret token and the GitLab webhook.

[](https://ww2.mathworks.cn/content/mathworks/cn/zh/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models/_jcr_content/mainParsys/image_0_copy_copy_co_797319347.adapt.full.high.jpg/1621386463720.jpg)

1. Configure the build environment. Select **Use MATLAB version** and enter the MATLAB root.

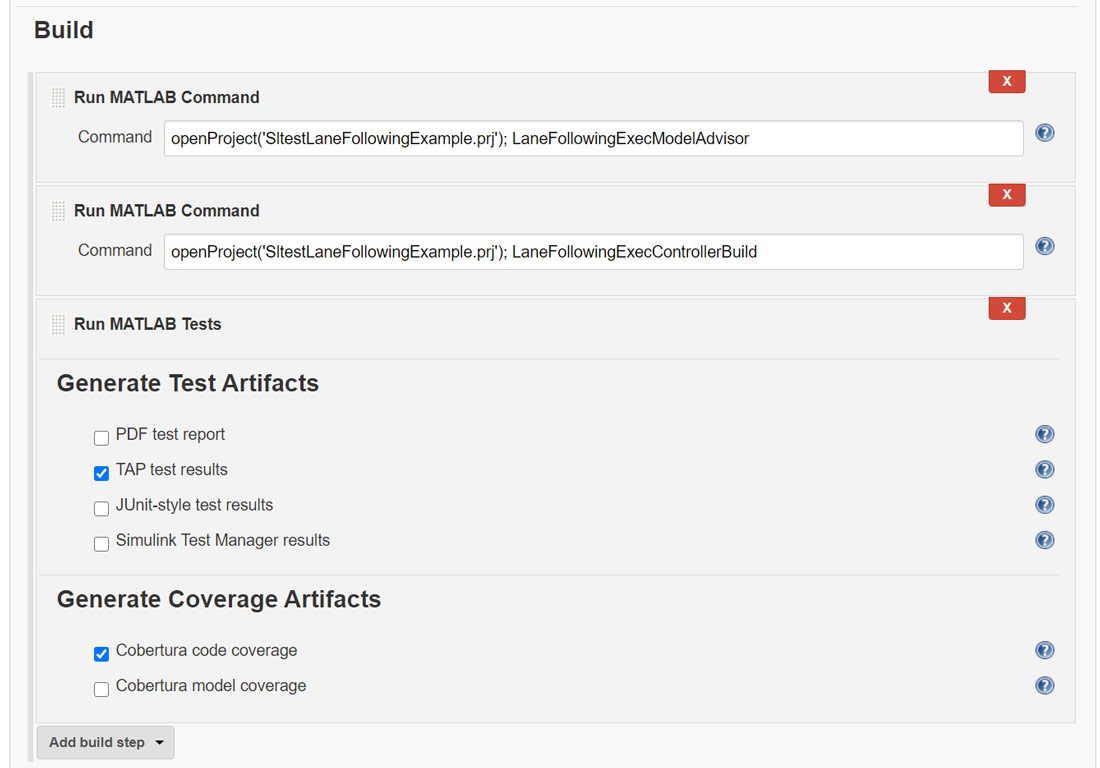


1. Configure the build step.

Click **Add build step** and choose **Run MATLAB Command**. Enter the command openProject('SltestLaneFollowingExample.prj'); LaneFollowingExecModelAdvisor  
to open the project and run model advisor checks.

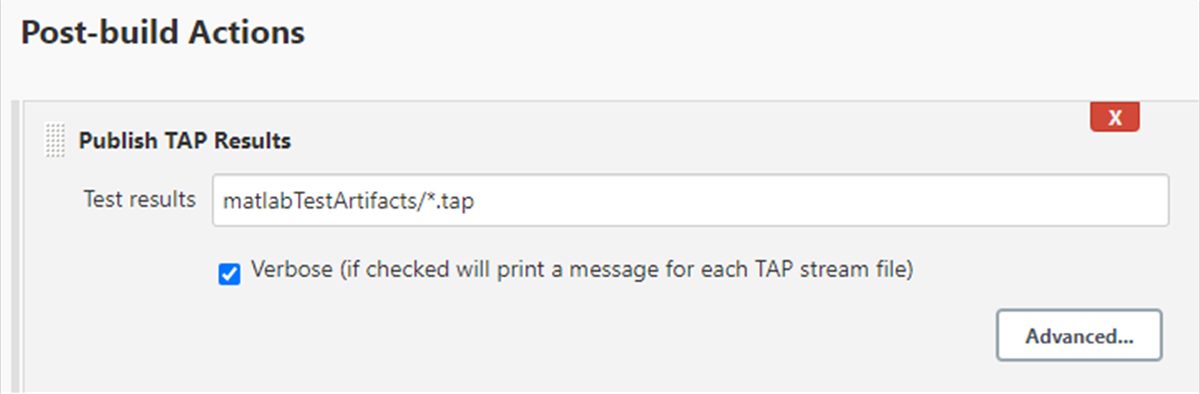
Click **Add build step** and choose **Run MATLAB Command** again. Enter the command: openProject('SltestLaneFollowingExample.prj'); LaneFollowingExecControllerBuild

Click **Add build step** and choose **Run MATLAB Tests**. Select **TAP test results** and **Cobertura code coverage** to complete the build configuration.

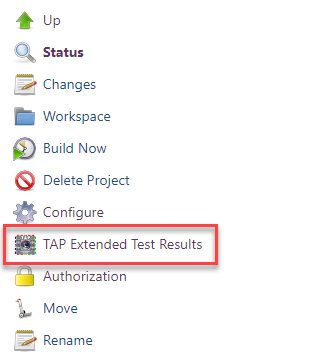


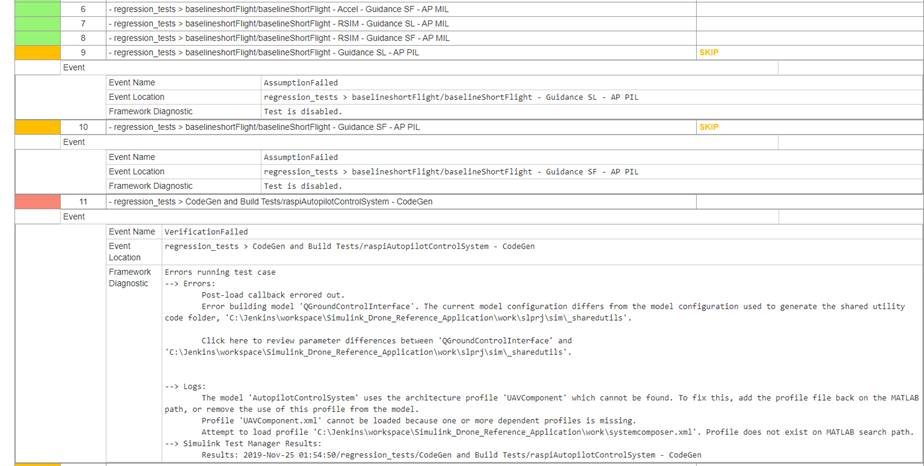
### Step 5. Publishing TAP results

Click **Add post-build action > Publish TAP Results**. Enter the relative path where the TAP test results will be published.

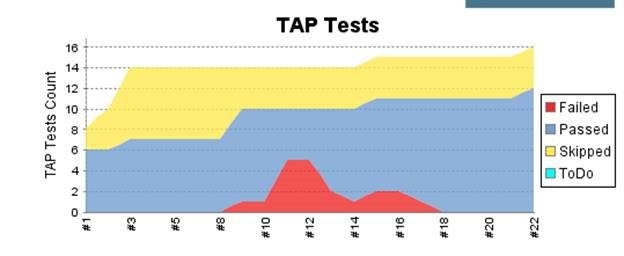


This action parses the TAP test results and makes them viewable when TAP Extended Test Results is selected. The output contains an overview of the executed test cases, the results summary, and logs from the MATLAB console.



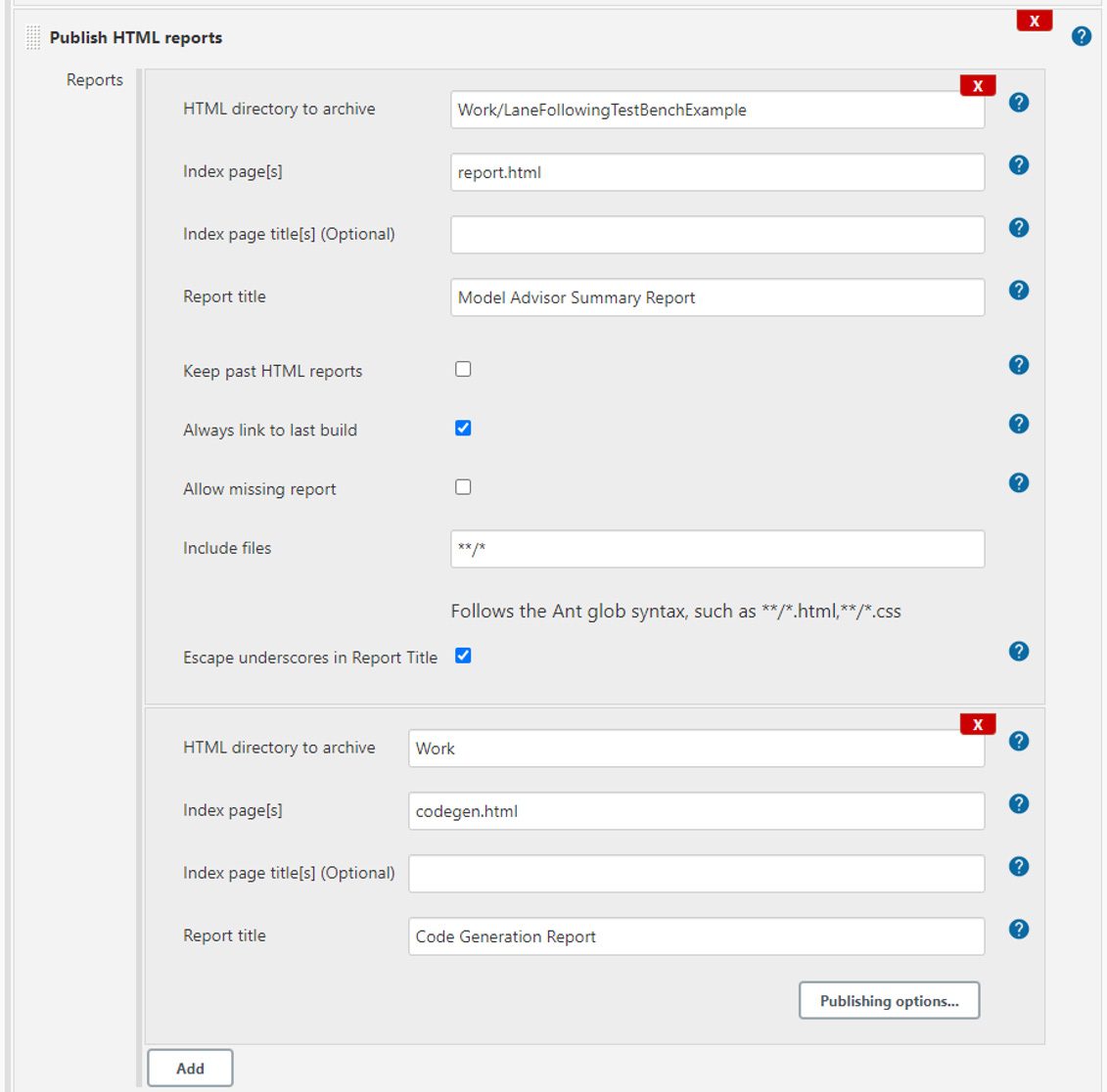


The TAP plugin also collects the results of the latest test executions and displays a health diagram as shown below. You can access any previous build by clicking on the diagram.



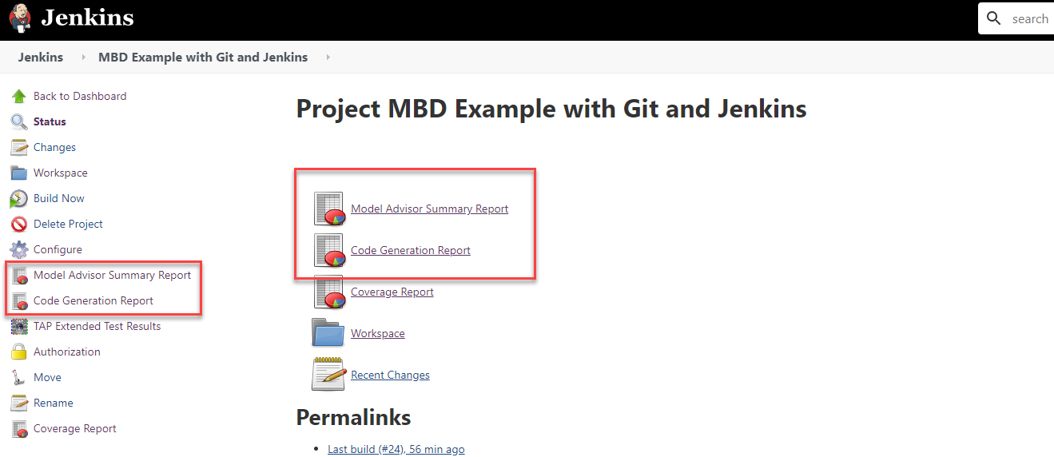
## Step 6. Publishing HTML reports

Click **Add post-build action >** **Publish HTML reports**. Enter the relative root path where the HTML report will be published and the filename of the index page that is in the path.



Add as many entries as there are HTML reports to be published. In this scenario, there are two web reports: The Model Advisor Summary and the Code Generation Report. These are standard reports created with MATLAB built-in functions. You can add custom HTML reports.

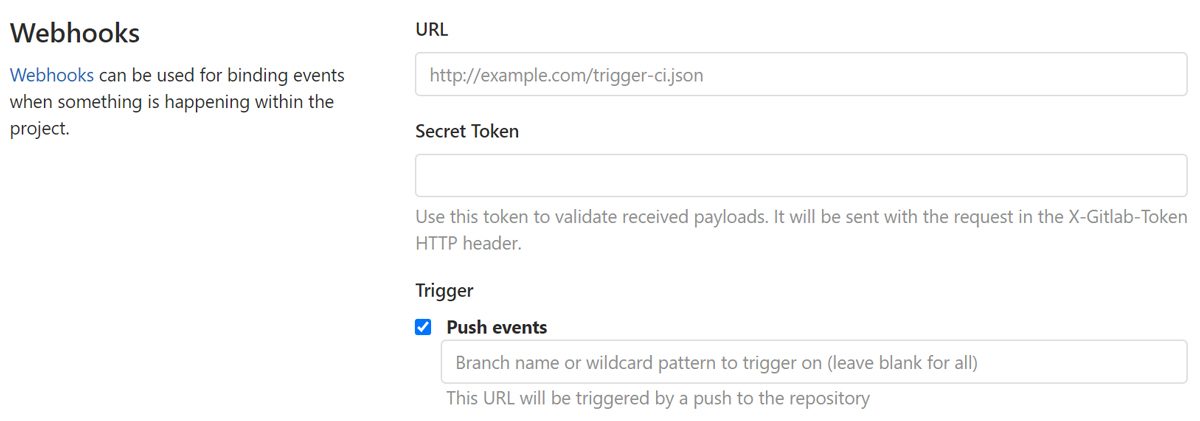
You will find a report link corresponding to the last build for every HTML report on the main Jenkins job page.  If you activate the checkbox “Always link to last build” under publishing options, the plugin will publish reports for the last build regardless of the build status. If this checkbox is not activated, the plugin will only link to the last “successful” build.



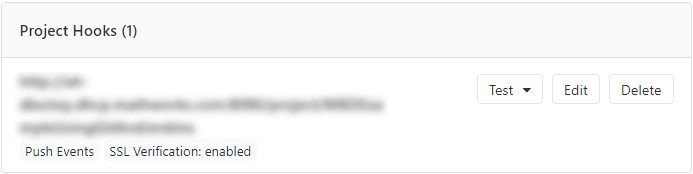
## Step 7. Configuring GitLab to trigger a build in Jenkins

Configure GitLab to trigger an automatic build in Jenkins when a new push occurs on the master branch. To do this, navigate to **Settings >** **Webhooks**. Use the webhook URL and the secret token provided by Jenkins in the Build Trigger configuration and select **Push events**.

*Note: Use a fully qualified domain name in the URL section in place of localhost so that GitLab can find the Jenkins installation.*



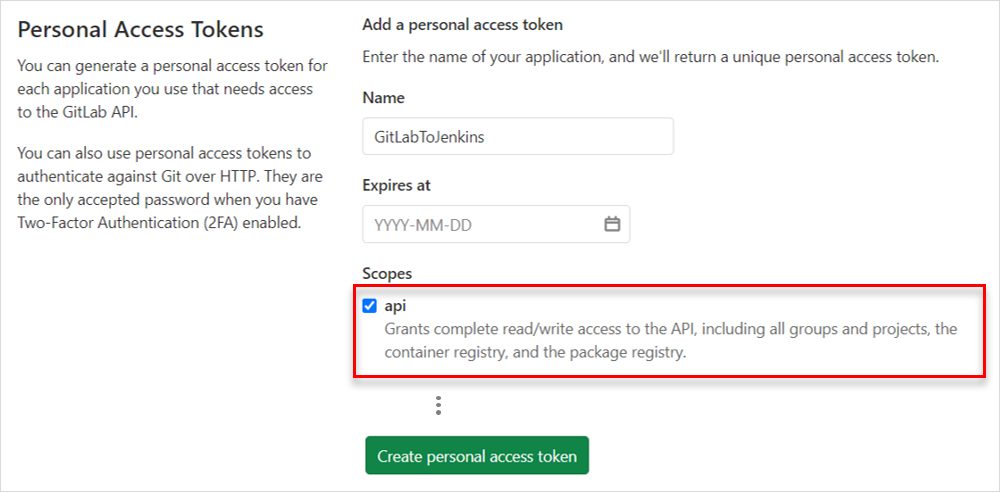
In the **Test** pulldown, select **Push Events** to test the integration. GitLab will show the message “Hook executed successfully: HTTP 200” and Jenkins will start a build.



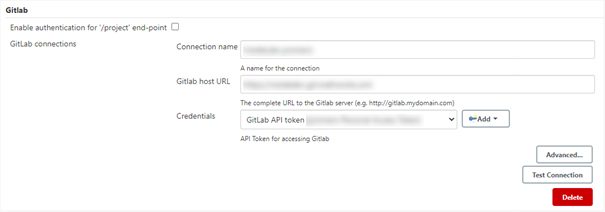
## Step 8. Configuring Jenkins-to-GitLab authentication

To publish a Jenkins build status automatically on GitLab, you must configure the Jenkins-to-GitLab authentication.

1. Create a personal access token on GitLab with API scope selected.



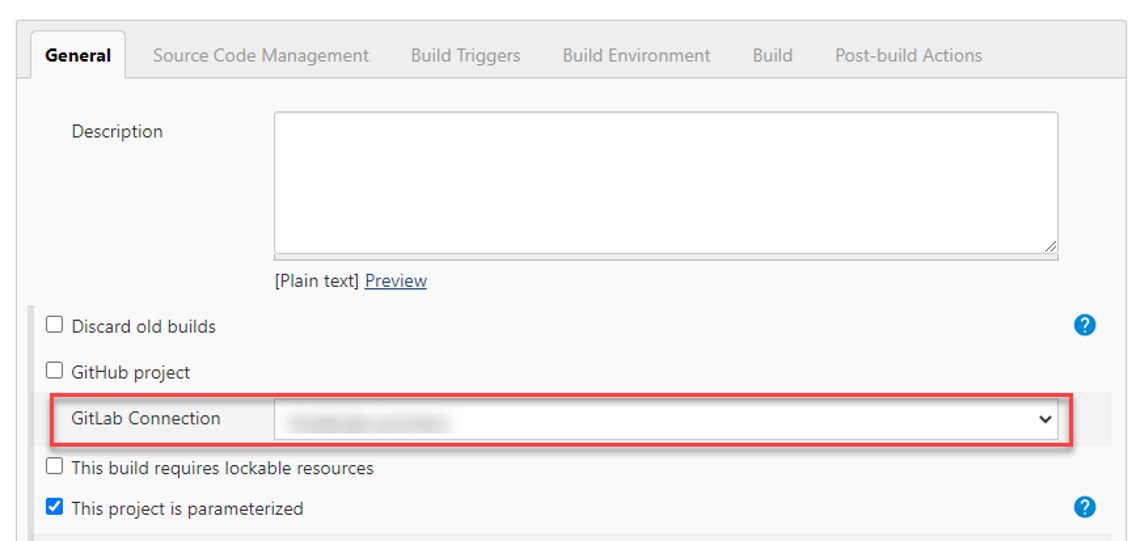
1. Copy the token and create a GitLab connection under Jenkins Configure System.  
   *Note: Connections can be reused over multiple Jenkins jobs and may be configured globally if the user has at least “maintainer” rights.*



## Step 9. Integrating Jenkins into the GitLab pipeline

To integrate Jenkins into the GitLab pipeline you must configure the GitLab connection in Jenkins and publish the job status to GitLab.

1. Select the GitLab Connection in the General section of your Jenkins job.

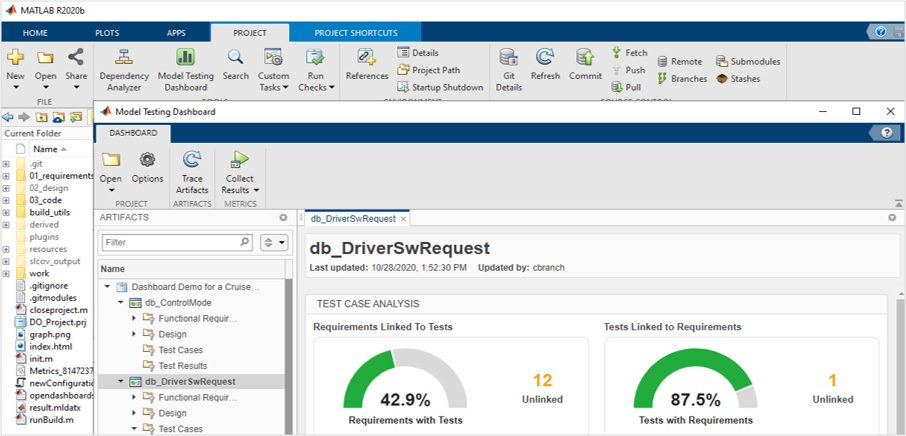


1. Add a post-build action to publish build status to GitLab.  
   *Note: This action has no parameter and will use the existing GitLab connection to publish the build status on GitLab and create bidirectional traceability for every commit and merge request.*



## Step 10: Visualizing requirements-based testing metrics (R2020b)

Requirements-based [testing metrics](https://ww2.mathworks.cn/help/slcheck/collect-model-metric-data-1.html) let you assess the status and quality of your requirements-based testing activities. The metrics results can be visualized using the Model Testing Dashboard.



1. Create a file called collectModelTestingResults.m based on the function shown below. This function will initialize the metrics engine infrastructure and collect all available model metrics.

function collectModelTestingResults()

% metric capability added in R2020a

if exist('metric')

metricIDs = [...

"ConditionCoverageBreakdown" "CoverageDataService"...

"DecisionCoverageBreakdown" "ExecutionCoverageBreakdown"...

"MCDCCoverageBreakdown" "OverallConditionCoverage"...

"OverallDecisionCoverage" "OverallExecutionCoverage"...

"OverallMCDCCoverage" "RequirementWithTestCase"...

"RequirementWithTestCaseDistribution" "RequirementWithTestCasePercentage"...

"RequirementsPerTestCase" "RequirementsPerTestCaseDistribution"...

"TestCaseStatus" "TestCaseStatusDistribution"...

"TestCaseStatusPercentage" "TestCaseTag"...

"TestCaseTagDistribution" "TestCaseType"...

"TestCaseTypeDistribution" "TestCaseWithRequirement"...

"TestCaseWithRequirementDistribution" "TestCaseWithRequirementPercentage"...

"TestCasesPerRequirement" "TestCasesPerRequirementDistribution"...

];

% collect all metrics for initial reconcile

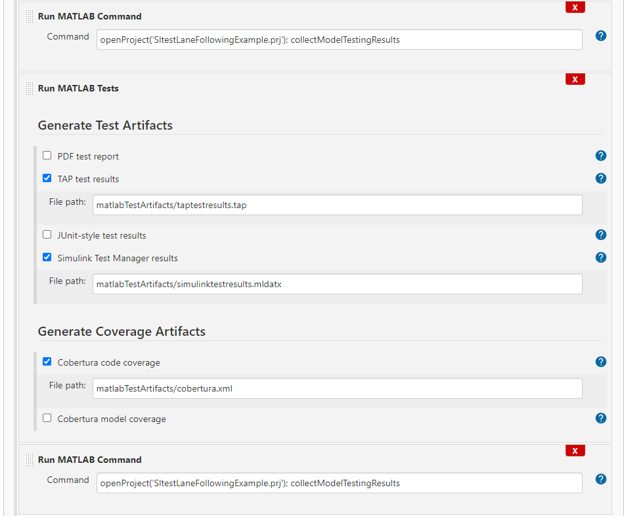
E = metric.Engine();

execute(E, metricIDs);

end

end

1. Add this file to your project and to the path.
2. Configure Jenkins to collect metric results by calling the new collectModelTestingResults function twice. The first call initializes the metrics integration with Simulink Test Manager. The second collects the metrics results using the exported Simulink Test Manager results.
   1. Click **Add build step**and choose **Run MATLAB Command**again. Enter the command: openProject('SltestLaneFollowingExample.prj'); collectModelTestingResults  
      Position this build step before the **Run MATLAB Tests** build step.
   2. Click **Add build step**and choose **Run MATLAB Command**again. Enter the command again: openProject('SltestLaneFollowingExample.prj'); collectModelTestingResults  
      Position this build step after the **Run MATLAB Tests** build step.



iii. Check **Simulink Test Manager results** in the **Run MATLAB Tests** build step.

IMG_306

1. Archive the metrics results in the derived directory. You must also archive the exported test manager results, as they will allow full navigation of the Metric results when loaded back into MATLAB.

Click **Add post-build action** and choose **Archive the artifacts.** Enter path derived/\*\*,matlabTestArtifacts/\*.mldatx to archive all files saved to that directory.



*Note: To view these results in MATLAB on a machine other than the test machine, do the following:*

* *Download the archived artifacts (derived directory and test results .mldatx files).*
* *Extract and copy into a local copy of the same version of the project that was used to run the CI job.*
* *Open the project in MATLAB and launch the Model Testing Dashboard.*

*The results generated by the CI will be displayed on the [dashboard](https://ww2.mathworks.cn/help/slcheck/ug/model-testing-dashboard.html).*

Published 2021

### Products Used

* [MATLAB](https://ww2.mathworks.cn/products/matlab.html)
* [Simulink](https://ww2.mathworks.cn/products/simulink.html)
* [Automated Driving Toolbox](https://ww2.mathworks.cn/products/automated-driving.html)
* [Computer Vision Toolbox](https://ww2.mathworks.cn/products/computer-vision.html)
* [Control System Toolbox](https://ww2.mathworks.cn/products/control.html)
* [Embedded Coder](https://ww2.mathworks.cn/products/embedded-coder.html)
* [Image Processing Toolbox](https://ww2.mathworks.cn/products/image.html)
* [MATLAB Coder](https://ww2.mathworks.cn/products/matlab-coder.html)
* [Model Predictive Control Toolbox](https://ww2.mathworks.cn/products/model-predictive-control.html)
* [Simulink Check](https://ww2.mathworks.cn/products/simulink-check.html)
* [Simulink Coder](https://ww2.mathworks.cn/products/simulink-coder.html)
* [Simulink Control Design](https://ww2.mathworks.cn/products/simcontrol.html)
* [Simulink Coverage](https://ww2.mathworks.cn/products/simulink-coverage.html)
* [Simulink Requirements](https://ww2.mathworks.cn/products/simulink-requirements.html)
* [Simulink Test](https://ww2.mathworks.cn/products/simulink-test.html)

### Learn More

* [Agile and Model-Based Design for Engineering Software Development](https://ww2.mathworks.cn/company/newsletters/articles/agile-and-model-based-design-for-engineering-software-development.html) - Article
* [Agile Model-Based Design: Accelerating Simulink Simulations in Continuous Integration Workflows](https://ww2.mathworks.cn/company/newsletters/articles/agile-model-based-design-accelerating-simulink-simulations-in-continuous-integration-workflows.html) - Article
* [The Other Kind of Continuous Integration](https://blogs.mathworks.com/developer/2015/01/20/the-other-kind-of-continuous-integration/) - Blog
* [Git to the Lab!](https://blogs.mathworks.com/developer/2018/08/23/gitlab-jenkins-workflow/) - Blog
* [Continuous Integration with Simulink Project and Simulink Test](https://blogs.mathworks.com/simulink/2018/09/04/continuous-integration-with-simulink-project-and-simulink-test/) - Blog

### View Articles for Related Capabilities

* [Simulation and Model-Based Design](https://ww2.mathworks.cn/company/newsletters/search.html?fq[]=marketing-capability:model-based-design)
* [System Design and Simulation](https://ww2.mathworks.cn/company/newsletters/search.html?fq[]=marketing-capability:system-design-simulation)
* [Verification, Validation, and Test](https://ww2.mathworks.cn/company/newsletters/search.html?fq[]=marketing-capability:verification-validation)
* [Real-Time Simulation and Testing](https://ww2.mathworks.cn/company/newsletters/search.html?fq[]=marketing-capability:rapid-prototyping)

### View Articles for Related Industries

* [Automotive](https://ww2.mathworks.cn/company/newsletters/search.html?fq[]=marketing-industry:automotive)
* [联系销售](https://ww2.mathworks.cn/company/aboutus/contact_us/contact_sales.html)

MathWorks

*Accelerating the pace of engineering and science*

MathWorks 公司是世界领先的为工程师和科学家提供数学计算软件的开发商。

[发现…](https://ww2.mathworks.cn/discovery.html?s_tid=hp_ff_a_discover)

**[了解产品](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "fatfooter_products)**

* [MATLAB](https://ww2.mathworks.cn/products/matlab.html?s_tid=hp_ff_p_matlab" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [Simulink](https://ww2.mathworks.cn/products/simulink.html?s_tid=hp_ff_p_simulink" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [学生版软件](https://ww2.mathworks.cn/products/matlab/student.html?s_tid=hp_ff_p_student" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [硬件支持](https://ww2.mathworks.cn/hardware-support/home.html?s_tid=hp_ff_p_hwsupport" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [文件交换](https://ww2.mathworks.cn/matlabcentral/fileexchange/?s_tid=hp_ff_p_fx)

**[试用或购买](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "fatfooter_buy)**

* [下载](https://ww2.mathworks.cn/downloads/web_downloads/?s_iid=hp_ff_t_downloads)
* [试用软件](https://ww2.mathworks.cn/campaigns/products/trials.html?s_iid=hp_ff_p_trial)
* [联系销售](https://ww2.mathworks.cn/company/aboutus/contact_us/contact_sales.html?s_iid=hp_ff_t_sales" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [定价和许可](https://ww2.mathworks.cn/pricing-licensing.html?s_iid=hp_ff_t_pricing" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [如何购买](https://ww2.mathworks.cn/store/?s_iid=hp_ff_t_buy" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)

**[如何使用](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "fatfooter_use)**

* [文档](https://ww2.mathworks.cn/help/?s_tid=hp_ff_l_doc)
* [教程](https://ww2.mathworks.cn/support/learn-with-matlab-tutorials.html?s_tid=hp_ff_l_tutorials)
* [示例](https://ww2.mathworks.cn/help/examples.html?s_tid=hp_ff_l_examples)
* [视频与网上研讨会](https://ww2.mathworks.cn/videos.html?s_tid=hp_ff_l_videos" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [培训](https://ww2.mathworks.cn/services/training.html?s_tid=hp_ff_l_training" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)

**[获取支持](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "fatfooter_support)**

* [安装帮助](https://ww2.mathworks.cn/help/install/?s_tid=hp_ff_s_install)
* [MATLAB 问答社区](https://ww2.mathworks.cn/matlabcentral/answers/index?s_tid=hp_ff_s_answers)
* [咨询](https://ww2.mathworks.cn/services/consulting.html?s_tid=hp_ff_s_consulting" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [许可中心](https://ww2.mathworks.cn/licensecenter/?s_tid=hp_ff_s_license)
* [联系支持](https://ww2.mathworks.cn/support/contact_us.html?s_tid=hp_ff_s_support" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)

**[关于 MathWorks](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360" \l "fatfooter_about)**

* [招聘](https://ww2.mathworks.cn/company/jobs/opportunities.html?s_tid=hp_ff_a_careers)
* [新闻室](https://ww2.mathworks.cn/company/newsroom.html?s_tid=hp_ff_a_newsroom" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [社会愿景](https://ww2.mathworks.cn/company/aboutus/soc_mission.html?s_tid=hp_ff_a_socialmission" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [客户案例](https://ww2.mathworks.cn/company/customer-stories.html?s_tid=hp_ff_a_customerstories" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [关于 MathWorks](https://ww2.mathworks.cn/company.html?s_tid=hp_ff_a_company" \t "https://ww2.mathworks.cn/company/newsletters/articles/_self)
* [Select a Web Site中国](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098&elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP&elqTrackId=7f388f6461da4f07ba5b78806028f889&elq=dc566e41a063422f888b7f03185145aa&elqaid=39098&elqat=1&elqCampaignId=14360)
* [专利](https://ww2.mathworks.cn/company/aboutus/policies_statements/patents.html?s_tid=gf_pat)
* [商标](https://ww2.mathworks.cn/company/aboutus/policies_statements/trademarks.html?s_tid=gf_trd)
* [隐私权政策](https://ww2.mathworks.cn/company/aboutus/policies_statements.html?s_tid=gf_priv)
* [防盗版](https://ww2.mathworks.cn/company/aboutus/policies_statements/piracy.html?s_tid=gf_pir)
* [应用状态](https://status.mathworks.com/?s_tid=gf_application)

[京ICP备12052471号](https://beian.miit.gov.cn/" \t "https://ww2.mathworks.cn/company/newsletters/articles/_blank)

© 1994-2021 The MathWorks, Inc.

* [](https://www.weibo.com/mathworks)
* [](https://ww2.mathworks.cn/company/newsletters/articles/continuous-integration-for-verification-of-simulink-models.html?s_v1=39098%26elqem=3295803_EM_CN_21-07_NEWSLETTER_CG-DIGEST-DEFAULT-RESP%26elqTrackId=7f388f6461da4f07ba5b78806028f889%26elq=dc566e41a063422f888b7f03185145aa%26elqaid=39098%26elqat=1%26elqCampaignId=14360)
* [](https://space.bilibili.com/1768836923)
* [](https://i.youku.com/mathworks)
* [](https://www.facebook.com/MATLAB)
* [](https://www.twitter.com/MATLAB)
* [](https://www.linkedin.com/company/the-mathworks_2)
* [](https://ww2.mathworks.cn/company/rss.html)

关注我们

Go to top of page