Integrate with Jenkins

This integration provides a free, easy, secure, and reliable way to connect your Jenkins server, running behind the firewall, with either Jira Software Cloud or Jira Service Management Cloud.

This gives your team more visibility and context on every issue in Jira, showing the latest build status or if your work has been successfully deployed to an environment.

You can also use this information to [search across issues using the Jira Query Language (JQL)](https://support.atlassian.com/jira-software-cloud/docs/advanced-search-reference-jql-developer-status/), answering questions like *“Which issues in the current sprint have been deployed to production?”* You can even add these as quick filters on your boards!

**Check the following things before you connect Jenkins to Jira Software Cloud:**

* You're a site administrator of your Jira Cloud site.
* You're an admin in Jenkins and you can install, update, and delete plugins.
* You can edit the *Jenkinsfile*in your source code/repo for the pipelines you want to use this integration with.
* Your team is including issue keys (e.g. FUSE-123) in their commit messages (for deployment information) and branch names (for build information). If your team isn't already doing this, [learn more about referencing issues in your development work](https://support.atlassian.com/jira-software-cloud/docs/reference-issues-in-your-development-work/).

Link Jira Software Cloud with Jenkins

**Install the Jira Cloud plugin for Jenkins**

1. Log in to your Jenkins server and navigate to the Plugin Manager.
2. Select the “Available” tab and search for “Atlassian Jira Software Cloud”.
3. Install the plugin.

The open-source plugin is hosted at GitHub. [You can check it out here](https://github.com/jenkinsci/atlassian-jira-software-cloud-plugin).

**Install the “Jenkins for Jira” app in Jira**

You can install the "Jenkins for Jira" via the Jira Marketplace:

1. Log in to your Jira site and go to **Apps > Explore more apps**.
2. Type "Jenkins for Jira" into the search box.
3. Click on the "Jenkins for Jira by Atlassian" app (if it doesn't show up, you might have to remove the "Top trending" filter).
4. Click on "Get app".

**Set up a webhook in Jira**

After installing the "Jenkins for Jira" app on your Jira site, you’ll need to set up a webhook:

1. In Jira, go to **Apps > Manage your apps**.
2. In the left sidebar, under "Apps", click the link to the "Jenkins for Jira" app.
3. Click on "Connect a Jenkins server".
4. The app will remind you to install the Jenkins plugin. Click on "Next".
5. Enter a name for your Jenkins server.
6. Copy the webhook URL for later use.
7. Copy the secret for later use.

**Connect Jenkins to the webhook**

With the webhook URL and secret, you can now create a connection between Jenkins and Jira.

1. In Jenkins, go to **Manage Jenkins > Configure System** and scroll to the Jira Software Cloud Integration section.
2. Select **Add Jira Cloud Site > Jira Cloud Site**. New fields will appear for your site name, webhook URL, and secret.
3. Enter the following details:
   * Site name: The URL for your Jira Cloud site, for example [yourcompany.atlassian.net](http://yourcompany.atlassian.net/).
   * Webhook URL: The webhook URL you copied from the Jenkins app in Jira earlier.
   * Secret:
     + Select Add > Jenkins
       - For *Kind*, select **Secret text**.
       - For *Secret*, paste the secret you copied from the Jenkins app in Jira earlier.
       - For *Description*, provide a description that helps you identify the secret.
     + The secret should now show in the dropdown menu. Select your newly created secret.
4. Select **Test connection** to make sure your credentials are valid for your Jira site.
5. Click **Save**.

Use the Jira cloud plugin for Jenkins

This plugin will send build and deployment events to Jira so that they’re visible in your Jira issues, on the deployments timeline, and in the Releases feature.

To send information from Jenkins to Jira, your team must include Jira issue keys (e.g. FUSE-123) in their commit messages (for deployment information) and branch names (for build information). Whenever a pipeline runs in Jenkins, the plugin will look for Jira issue keys in the branch name and commit messages. If it finds issue keys, it will send build and deployment information to Jira. If it doesn’t find issue keys, the Jenkins plugin won’t send anything to Jira.

**Sending builds automatically**

To automatically send build events without having to add anything to your Jenkinsfiles, go to Manage **Jenkins > Configure System** and enable the checkbox “Sends builds automatically”.

When you enable this, the plugin will send an "in progress" build event to Jira once a pipeline run has started and a "success" or "failure" build event once the pipeline has finished successfully or stopped due to an error.

If you also specify a regular expression for builds, the plugin will only send a build event to Jira once a build step with a matching name has been finished.

The regular expression ^build$ would match the build stage in the following Jenkinsfile, for example:

1

2

3

4

5

6

7

8

9

10

pipeline {

agent any

stages {

stage('build') {

steps {

echo 'build done'

}

}

}

}

Whenever the pipeline in this Jenkinsfile runs, it will send build events to all configured Jira Cloud sites on start and finish of the build stage.

**Sending deployments automatically**

To automatically send deployment events without having to add anything to your Jenkinsfiles, go to **Manage Jenkins > Configure System** and enable the checkbox “Sends deployments automatically”.

When you enable this, the plugin will send an "in progress" deployment event to Jira once a build step with a name matching the specified regular expression has started, and a "success" or "failure" deployment event once that build step has finished.

For this to work, the deployment steps in your Jenkinsfile must contain the environment name in their name. The regular expression must contain the fragment (?<envName>.\*) to match the environment name so that the plugin can extract the environment name from the build step names.

Let's look at an example Jenkinsfile:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

pipeline {

agent any

stages {

stage('deployments') {

parallel {

stage('deploy to stg') {

steps {

echo 'stg deployment done'

}

}

stage('deploy to prod') {

steps {

echo 'prod deployment done'

}

}

}

}

}

}

 If the checkbox "Send deployments automatically" is enabled and the regular expression is set to ^deploy to (?<envName>.\*)$, a run of the above Jenkinsfile will send "in progress" deployment events for the stg and prod environments to all configured Jira Cloud sites, followed by respective "success" deployment events once the build steps are finished.

**Sending builds explicitly**

If you want more control over when to send build events, you can use the jiraSendBuildInfo build step:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

post {

always {

// previous to version 2.0.0 you must provide parameters to this command (see below)!

jiraSendBuildInfo()

}

}

}

}

}

This will send a "success" or "failure" build event to all configured Jira Cloud sites after the Build stage has finished successfully or with an error. The Jenkins plugin will automatically extract Jira issue keys from the branch name.

You can also specify a Jira site URL and instruct the plugin to only send build events to this Jira site (rather than to all configured Jira sites).

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

post {

always {

jiraSendBuildInfo site: 'example.atlassian.net', branch: 'TEST-123-awesome-feature'

}

}

}

}

}

**Sending deployments explicitly**

If you want more control over when to send deployment events, you can use the jiraSendDeploymentInfo build step:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

pipeline {

agent any

stages {

stage('Deploy - Staging') {

when {

branch 'master'

}

steps {

echo 'Deploying to Staging from master...'

}

post {

always {

jiraSendDeploymentInfo environmentId: 'us-stg-1', environmentName: 'us-stg-1', environmentType: 'staging'

}

}

}

stage('Deploy - Production') {

when {

branch 'master'

}

steps {

echo 'Deploying to Production from master...'

}

post {

always {

jiraSendDeploymentInfo environmentId: 'us-prod-1', environmentName: 'us-prod-1', environmentType: 'production'

}

}

}

}

This will send a "success" or "failure" deployment event to all configured Jira sites at the end of the stages Deploy - Staging and Deploy - Production.

You **must** provide the parameters environmentId, environmentName, and environmentType. The environmentType must be one of the following: unmapped, development, testing, staging, production.

You can provide the parameter site to specify to send the deployment events to a single Jira site instead of all configured Jira sites.

When multiple Jira sites are connected to a Jenkins server, the site parameter is required for jiraSendDeploymentInfo with enableGating:true. More details about Deployment Gating can be found [here](https://support.atlassian.com/jira-service-management-cloud/docs/use-deployment-gating-with-jenkins/).

You can also specify a branch with the branch parameter to define the branch from which to extract Jira issue keys to connect the deployments with.

**Example of a complete Jenkinsfile**

You can mix build and deployments as in the Jenkinsfile below:

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

pipeline {

agent any

stages {

stage('Build') {

steps {

echo 'Building...'

}

post {

always {

jiraSendBuildInfo site: 'example.atlassian.net'

}

}

}

stage('Deploy - Staging') {

when {

branch 'master'

}

steps {

echo 'Deploying to Staging from master...'

}

post {

always {

jiraSendDeploymentInfo environmentId: 'us-stg-1', environmentName: 'us-stg-1', environmentType: 'staging'

}

}

}

stage('Deploy - Production') {

when {

branch 'master'

}

steps {

echo 'Deploying to Production from master...'

}

post {

always {

jiraSendDeploymentInfo environmentId: 'us-prod-1', environmentName: 'us-prod-1', environmentType: 'production'

}

}

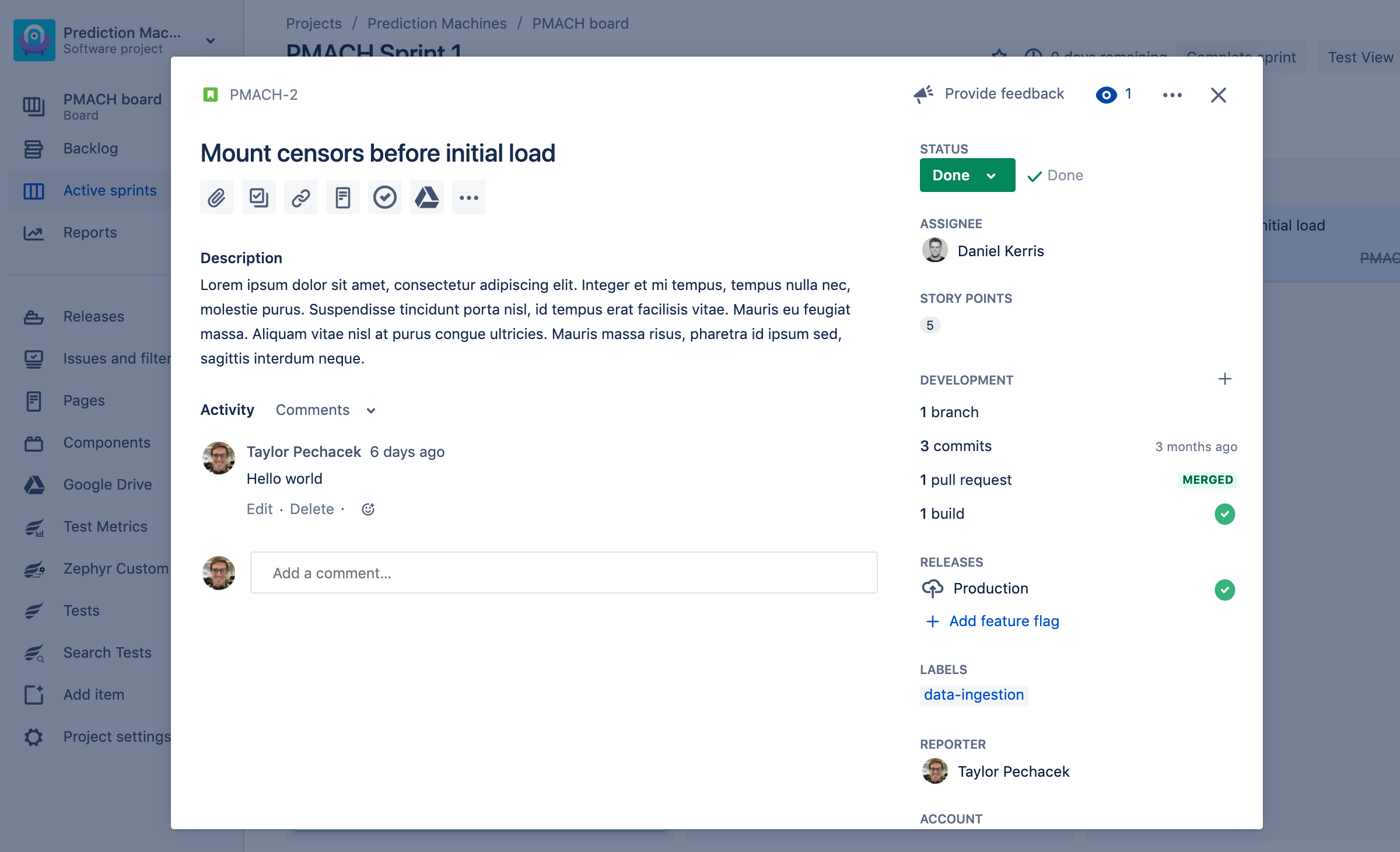
}

}

}

**View development information in Jira**

Whenever you make a commit or merge a pull request in a connected source code management tool, like Bitbucket or GitHub, it should run the Jenkins pipeline you have specified for that repo. The development panel on your Jira issues will update to show any associated build and deployment information, as long as your team is including issue keys in their pull requests, commit messages, and branch names. Learn more about how to [view development information for an issue](https://support.atlassian.com/jira-software-cloud/docs/view-development-information-for-an-issue/).



If you’ve enabled the deployments feature in your Jira project, the Deployments page will show all your Jenkins deployments on a timeline. You can filter or search to view your deployments by environment, assignee, issue type, and more. And if your team is using releases and versions to organize your work, you’ll also find deployment information in the Releases feature.