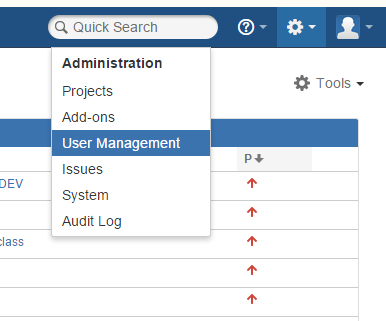
JIRA

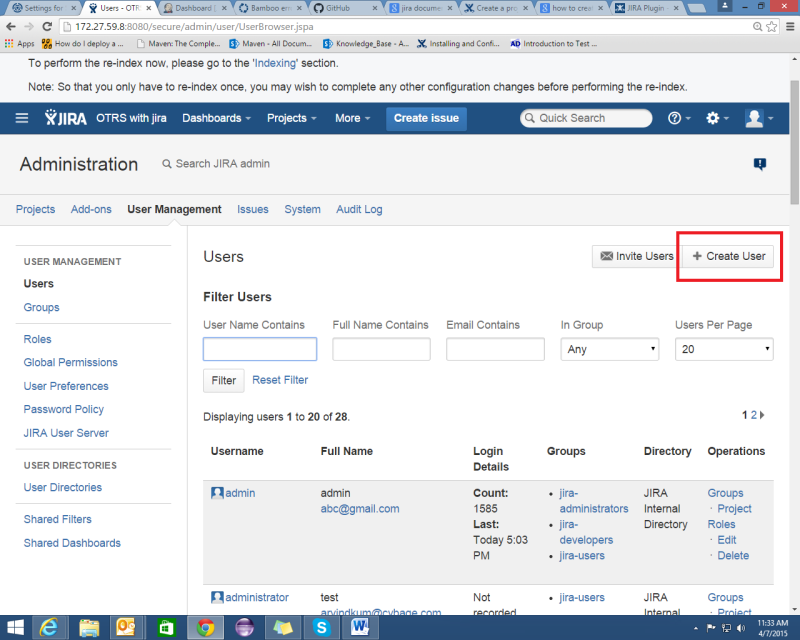
Add a few users.

You will be adding two users to JIRA in this step: Sakshi\_dev and Sakshi\_TL. You should be logged in to JIRA as an administrator after setting up your site.

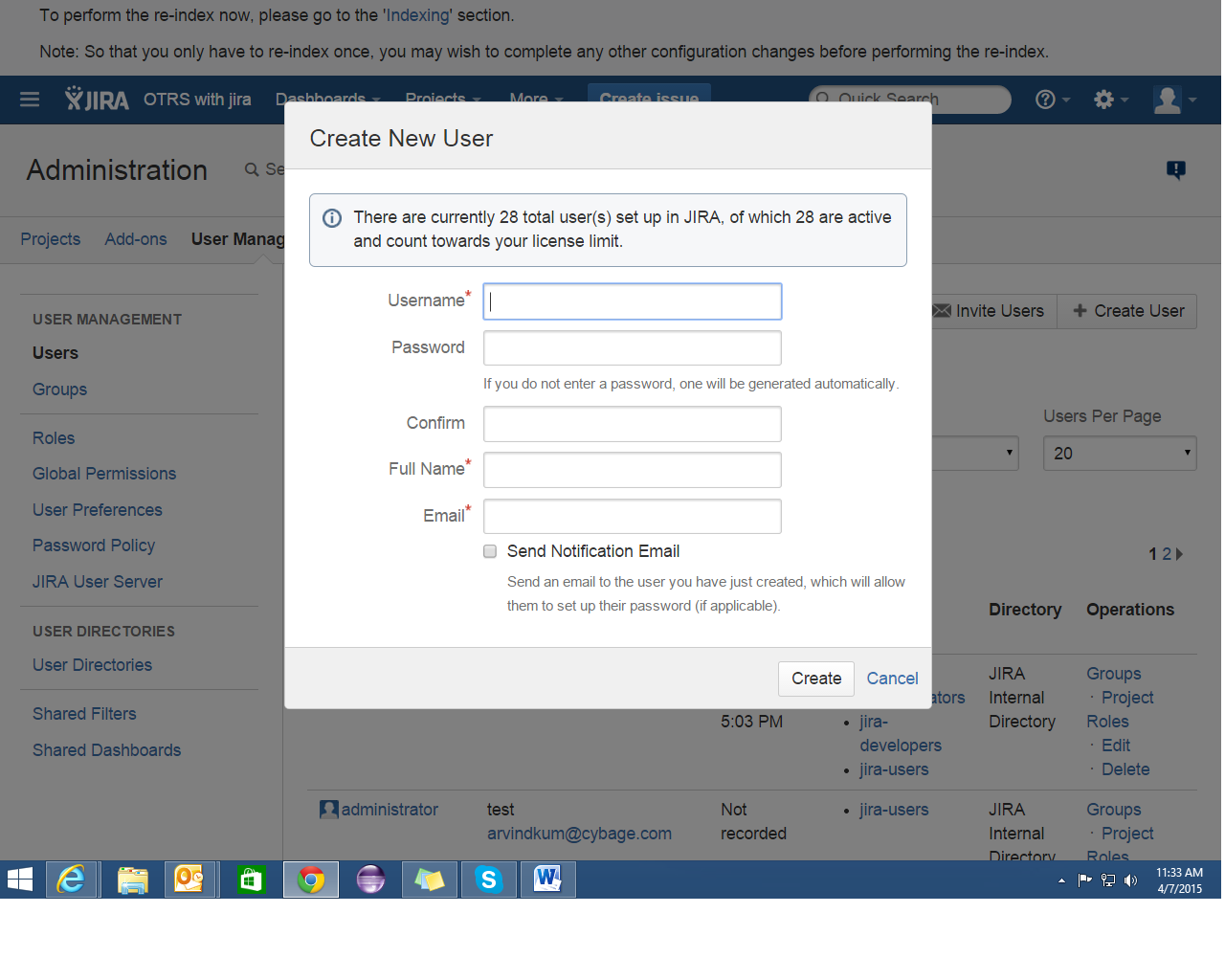
Go to Settings 🡪 User Management.



Click on Create user tab.



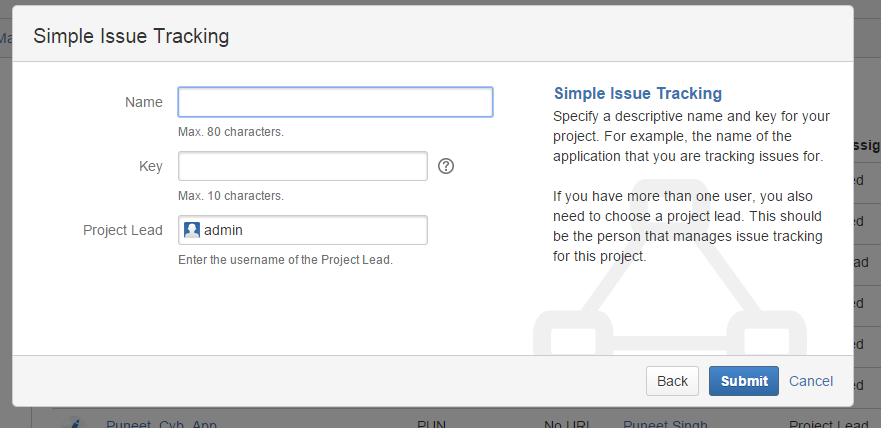
Enter the details and click on create. User will get created.



Create a Project

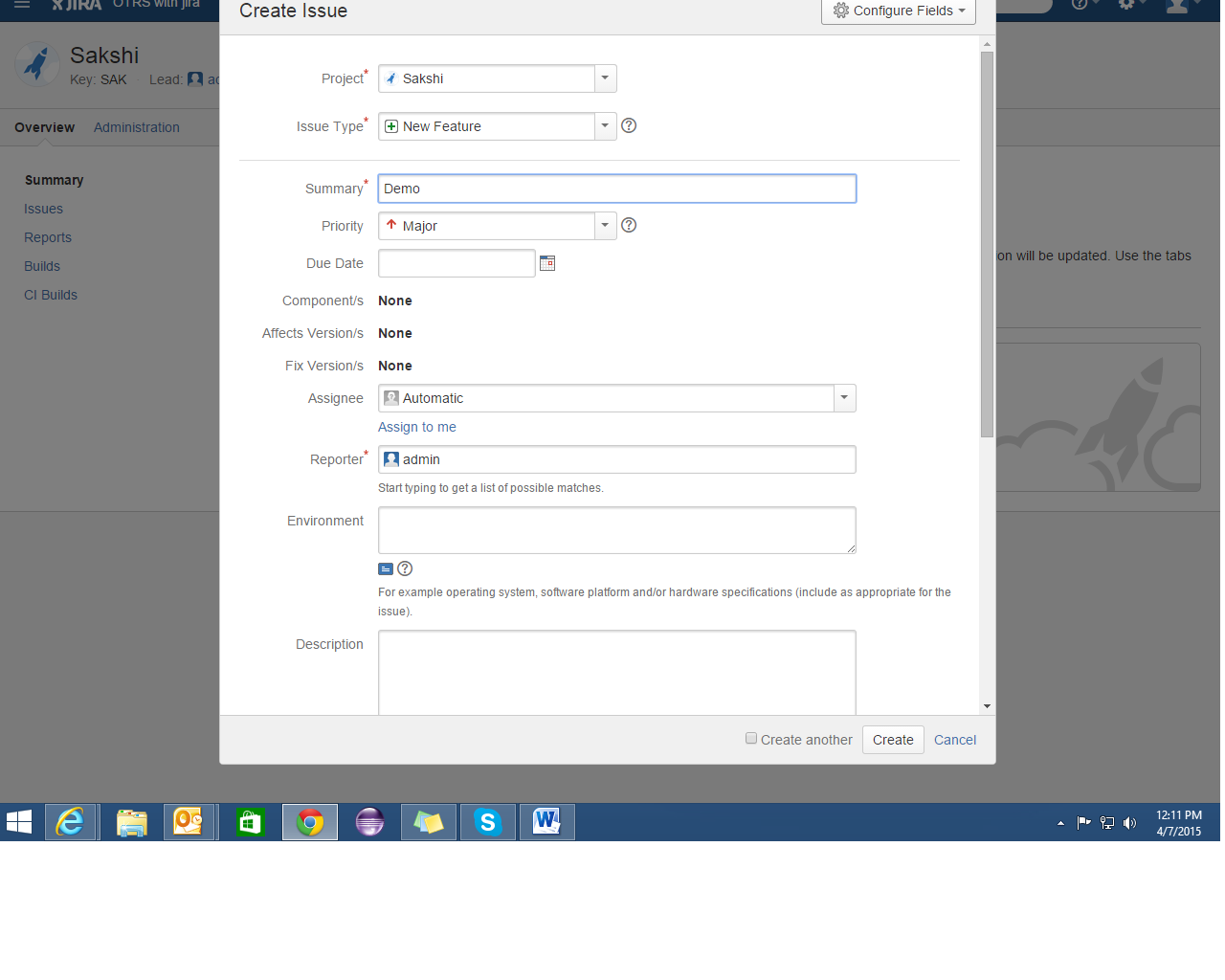
In this stage, you will create a **project.**A JIRA project is a collection of issues. Your team could use a JIRA project to coordinate the development of a product, track a project, manage a help desk, and more, depending on your requirements.

Choose **Projects** > **Create Project** and choose type of the project depending on the requirement. Specify the name and key of the project.



Create an Issue

1. Choose **Create** in the JIRA header to open the 'Create Issue' dialog.
2. Fill out the fields using the sample data is shown below. Only the fields with \* are mandatory.



Configure Permissions

You won't want every user in your team to have the same level of access to JIRA. For example, you may want to restrict who can administer JIRA, or prevent users from viewing a project. In this stage, you will learn about the different permissions in JIRA and try setting permissions for a project.

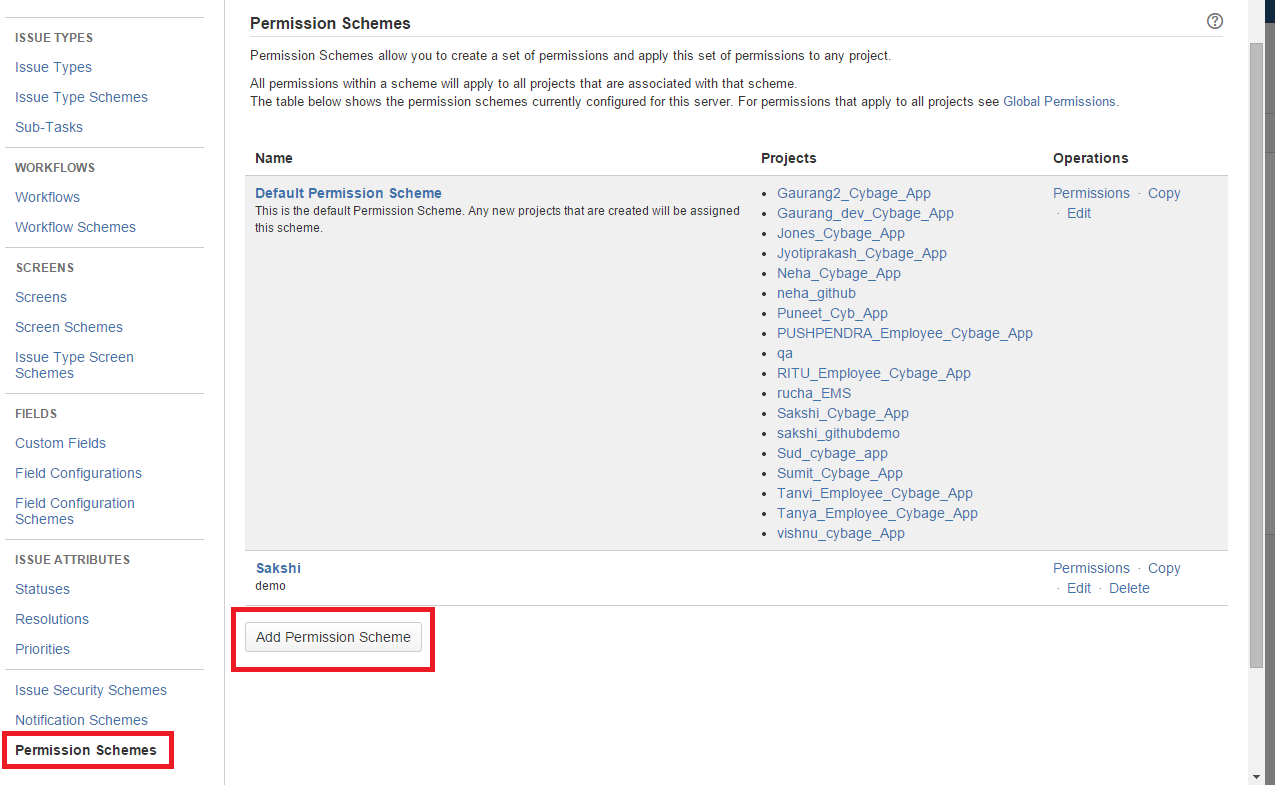
There are two levels of permissions in JIRA: **global permissions and project permissions**.

**Global permissions** cover a small set of functions that affect all projects in JIRA. For example, permission to administer JIRA. They can only be assigned to groups.

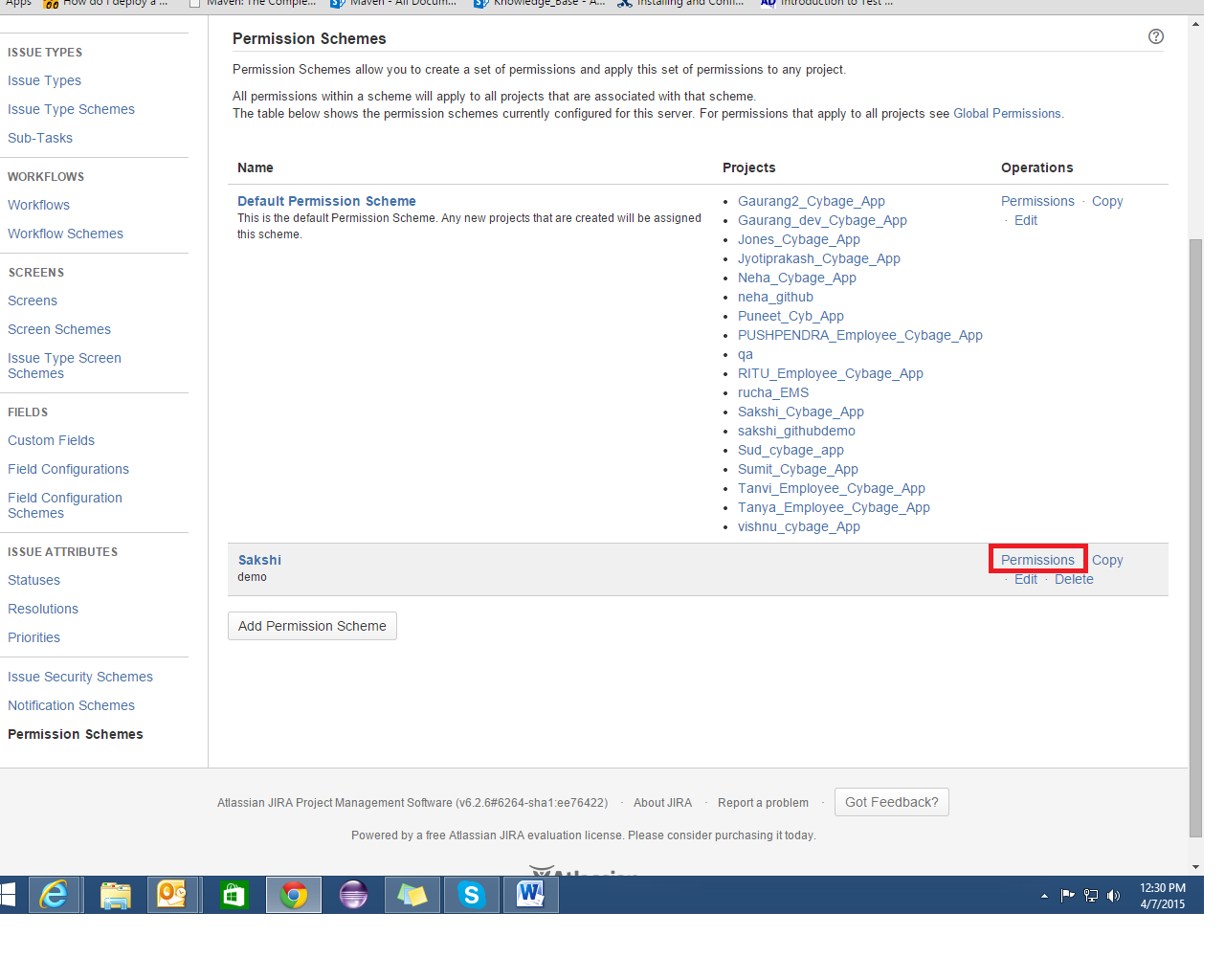
**Project permissions** cover a set of more granular functions that affect a single project in JIRA. For example, permission to create issues in a project. They can be assigned to groups, users and roles.

If you wish to set permissions on a project-by-project basis you can set them up in the [Permission Schemes](http://172.27.59.8:8080/secure/admin/ViewPermissionSchemes.jspa).

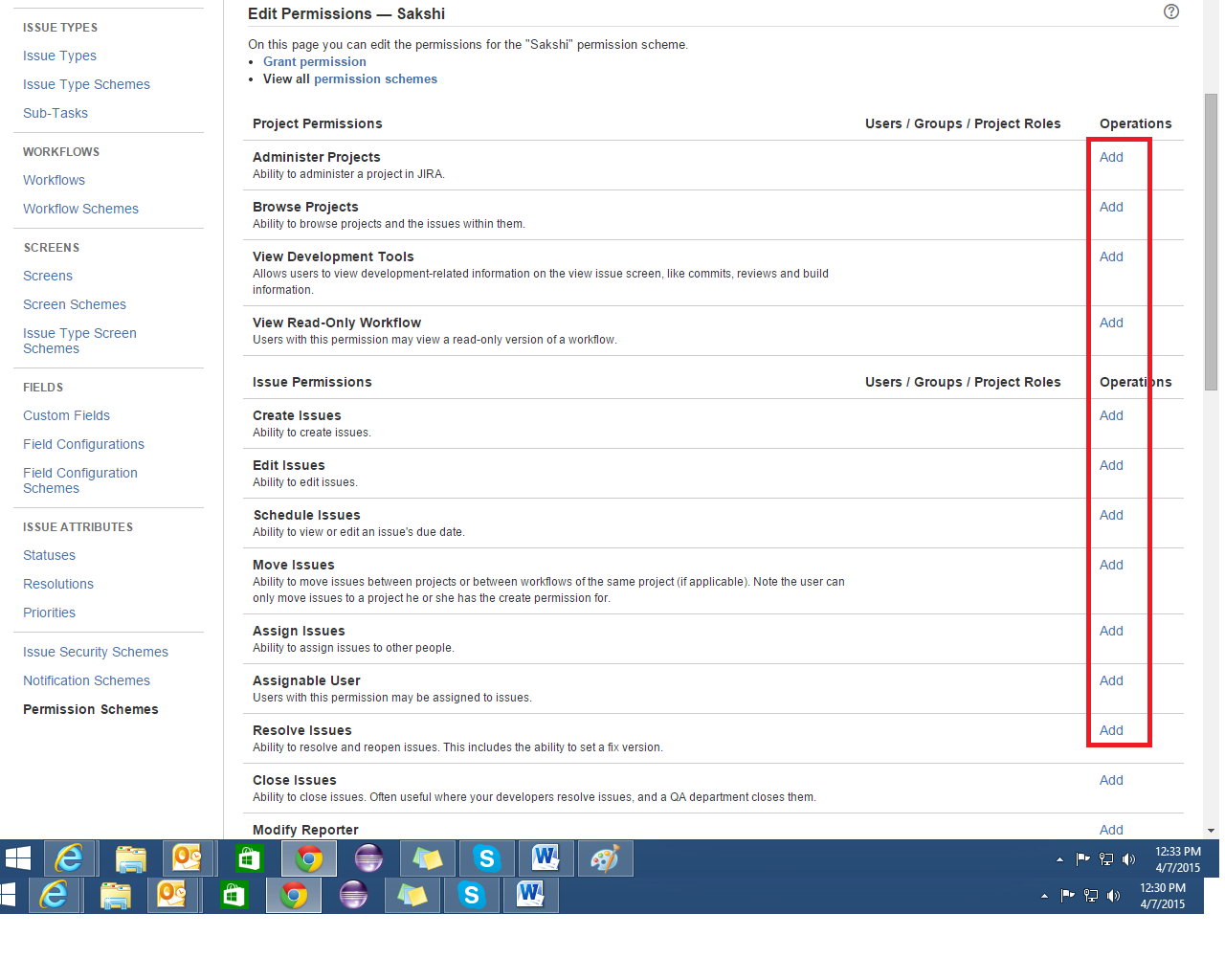
Click on Add Permission Scheme. Give name and description and click on add.



Click on Permissions to add permissions.

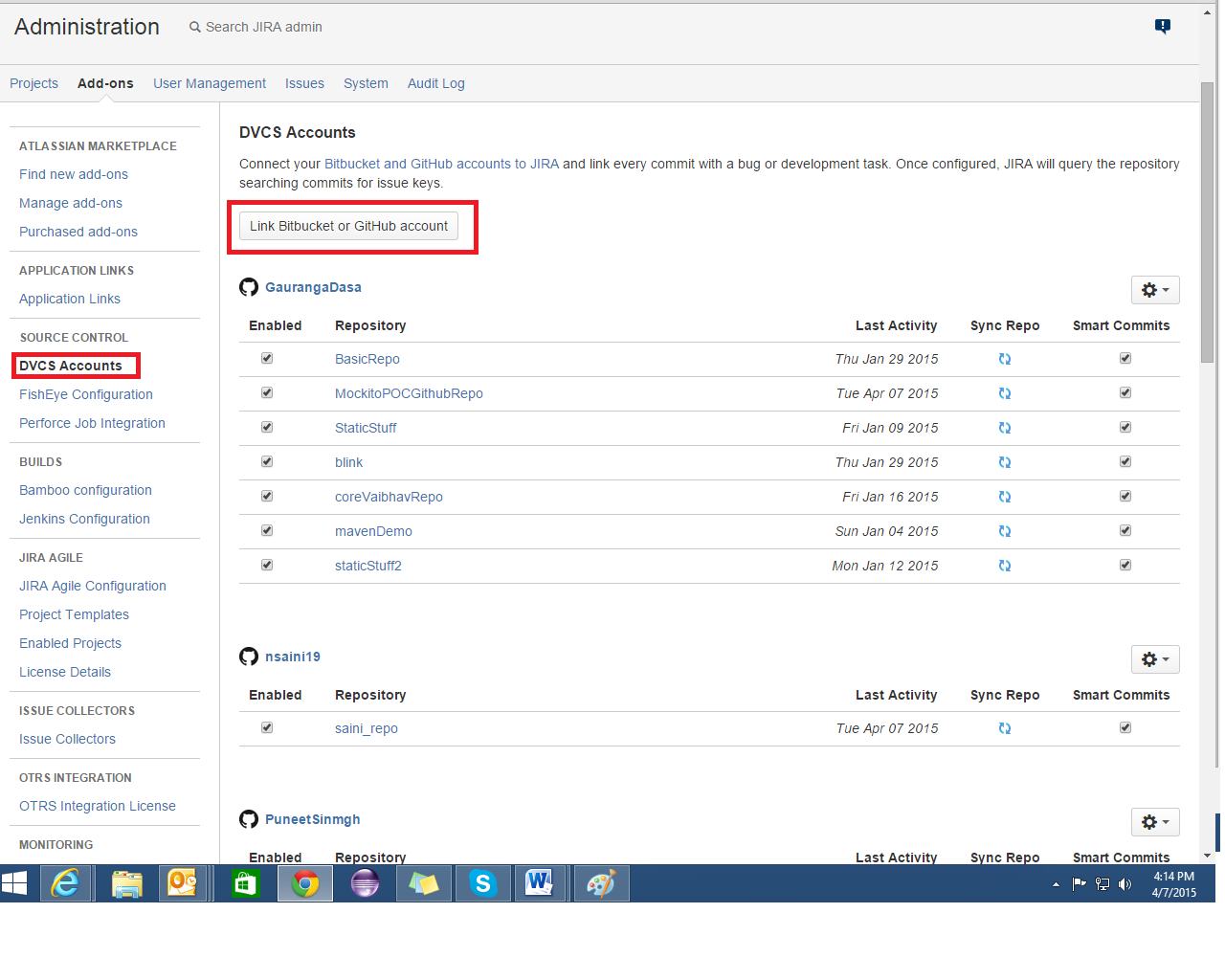


You can add the various permissions depending on the requirement.



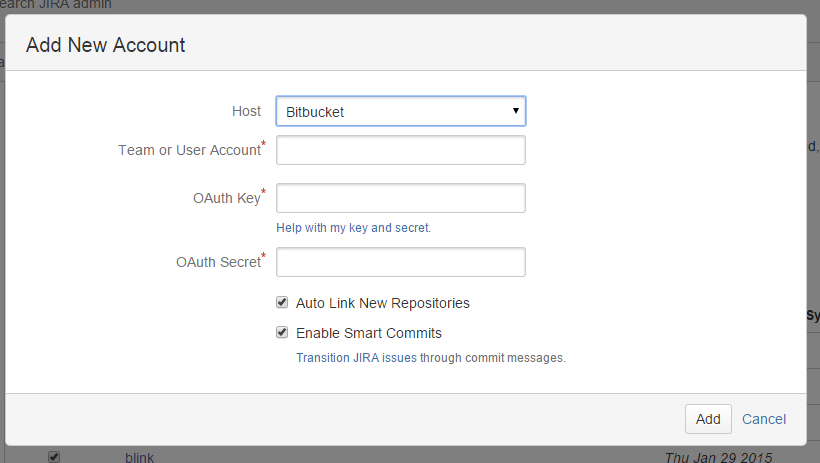
For Integration with GitHub:

* Go to Settings 🡪 Add-ons 🡪 Choose DVCS Accounts under source control.
* Click on Link Bitbucket or GitHub account.



* Under **Host**, choose **GitHub** from the dropdown menu.
* Under **Team or User Account**, type the name of your GitHub account
* Under **OAuth Key**, type the "Client ID" shown for the developer application you created on GitHub. (refer GitHub configuration below)
* Under **OAuth Secret**, type the "Client Secret" shown for the developer application you created on GitHub. (refer GitHub configuration below)
* Review the permissions that you are granting to your GitHub and click **Authorize application**.
* If necessary, type your password to continue.

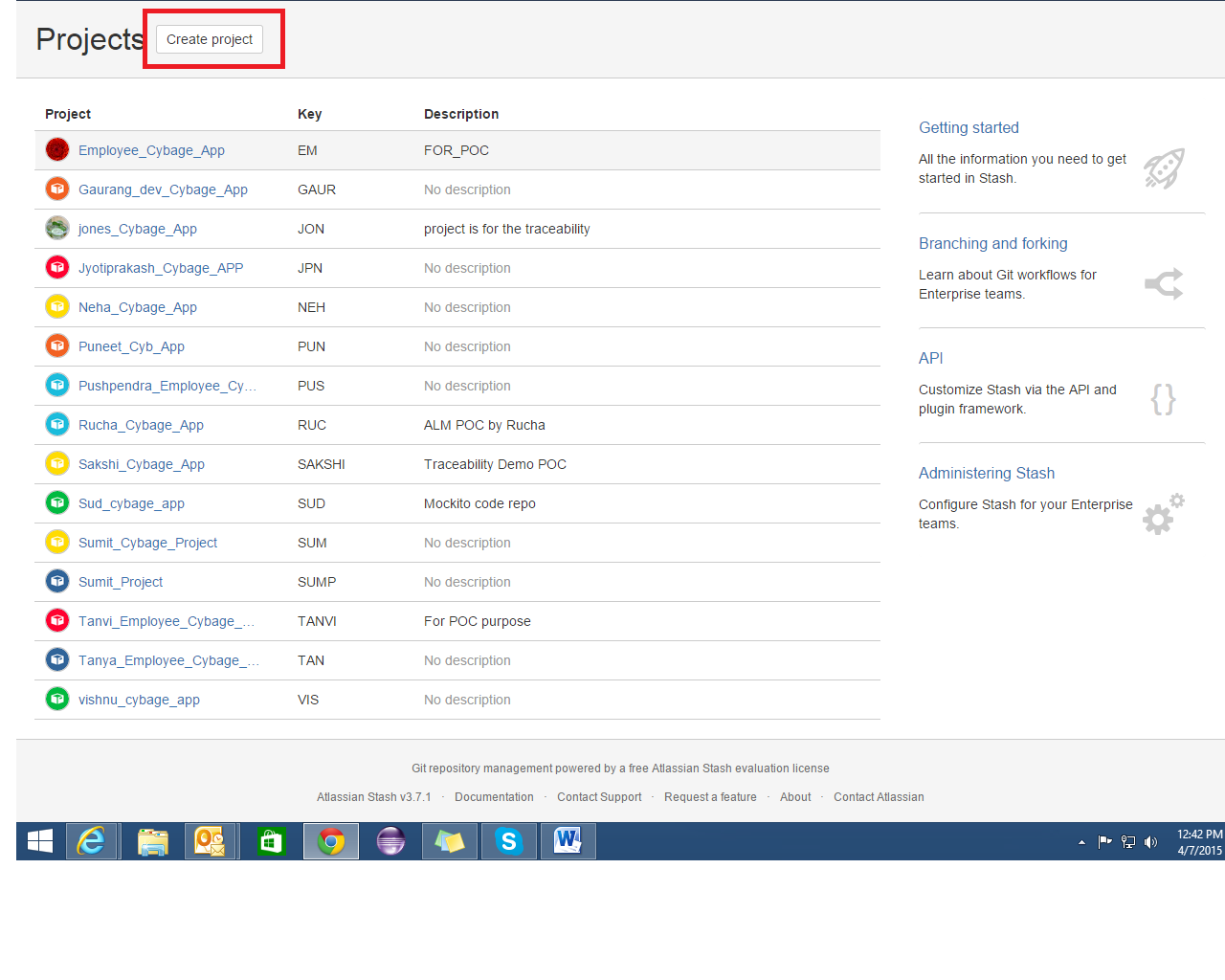
:



STASH

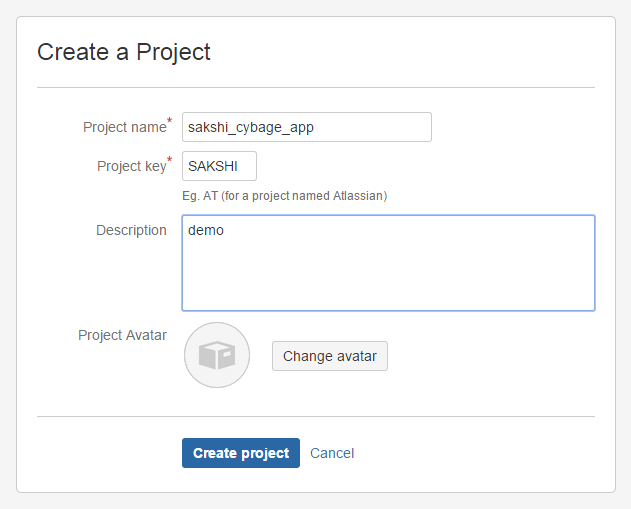
Creating Projects:

To create a project, click on **Create project**:

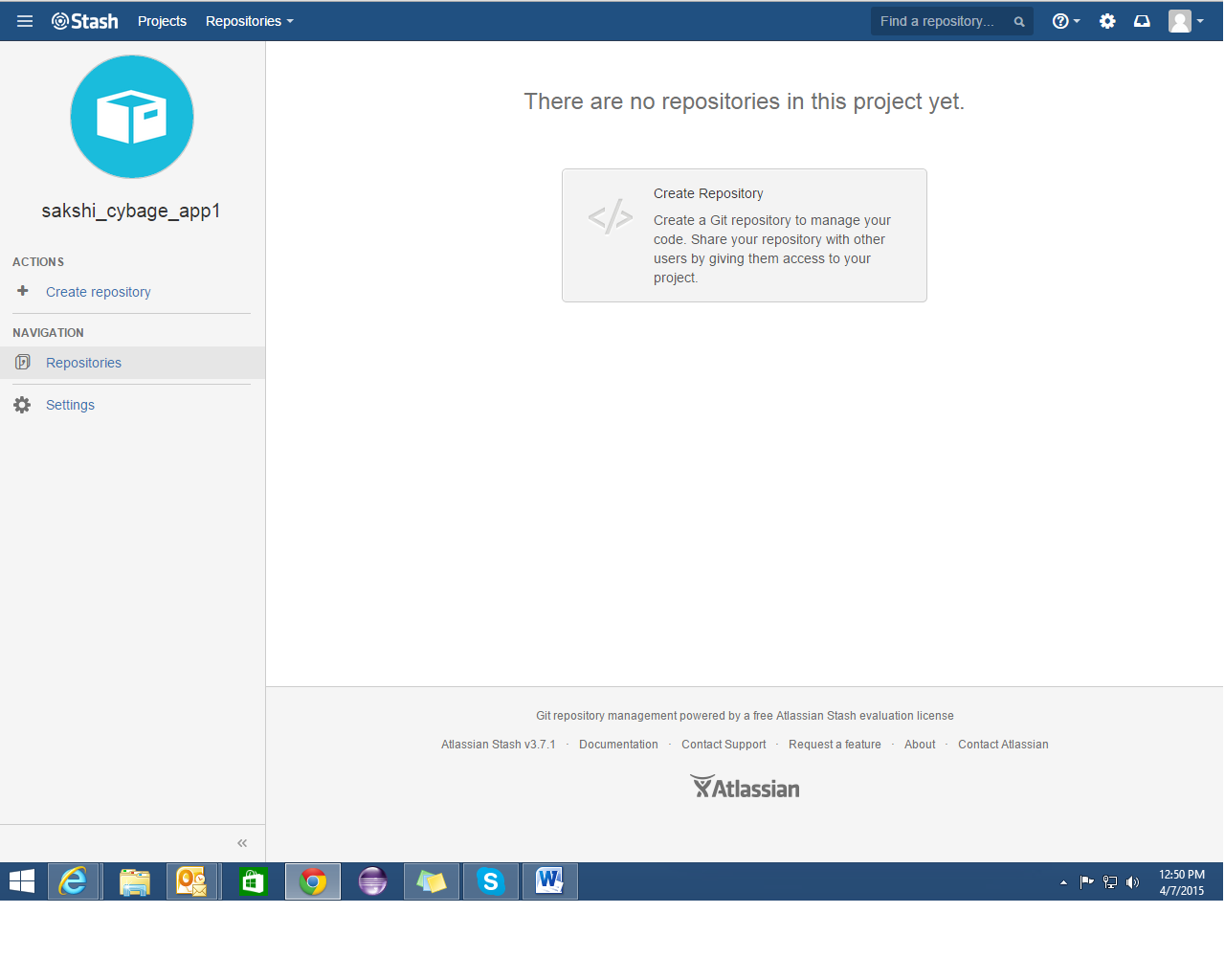


Fill out the form. We recommend that you use a short project key. It will be used as an identifier for your project and will appear in the URLs.

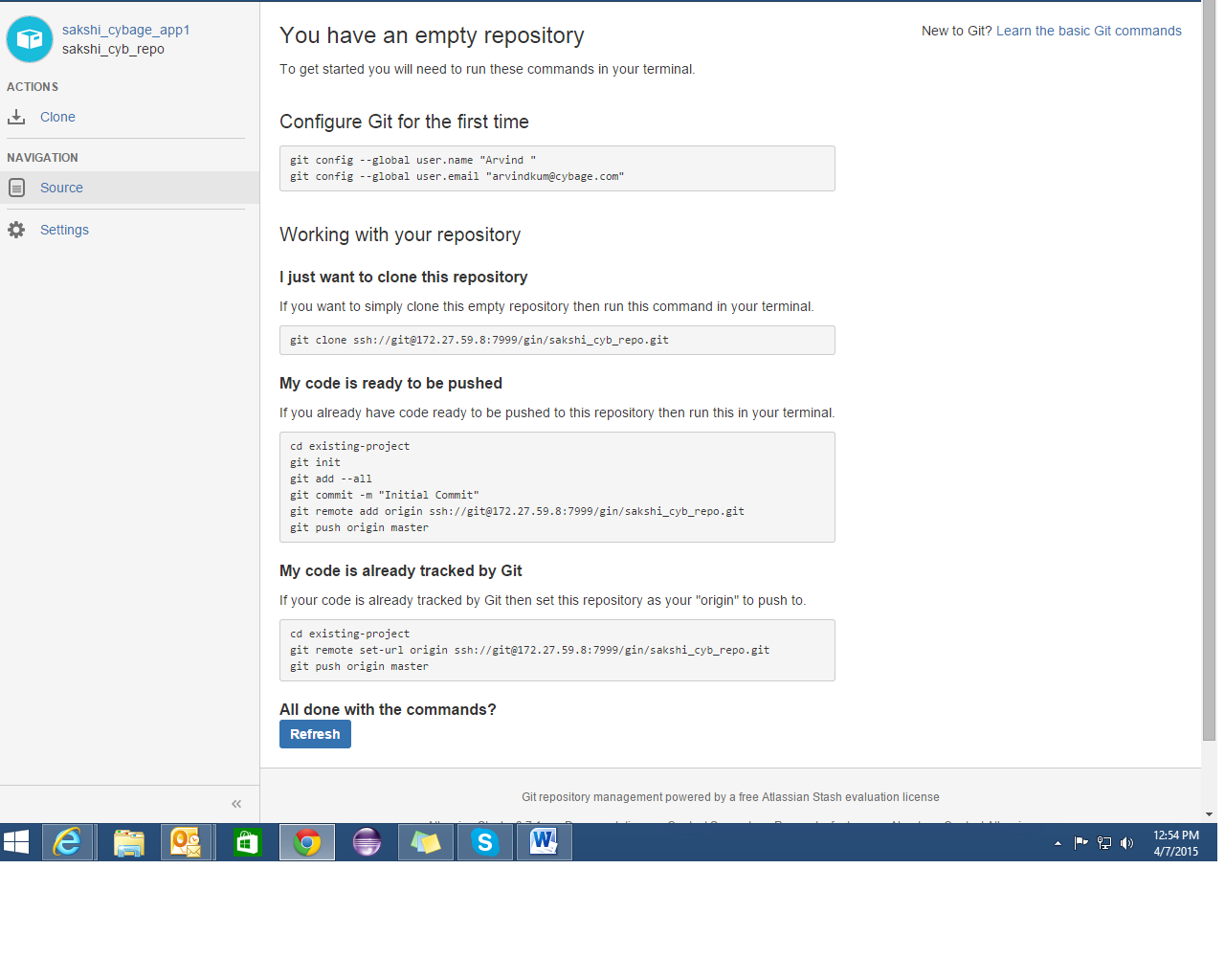
Click **Create project** when you're done.



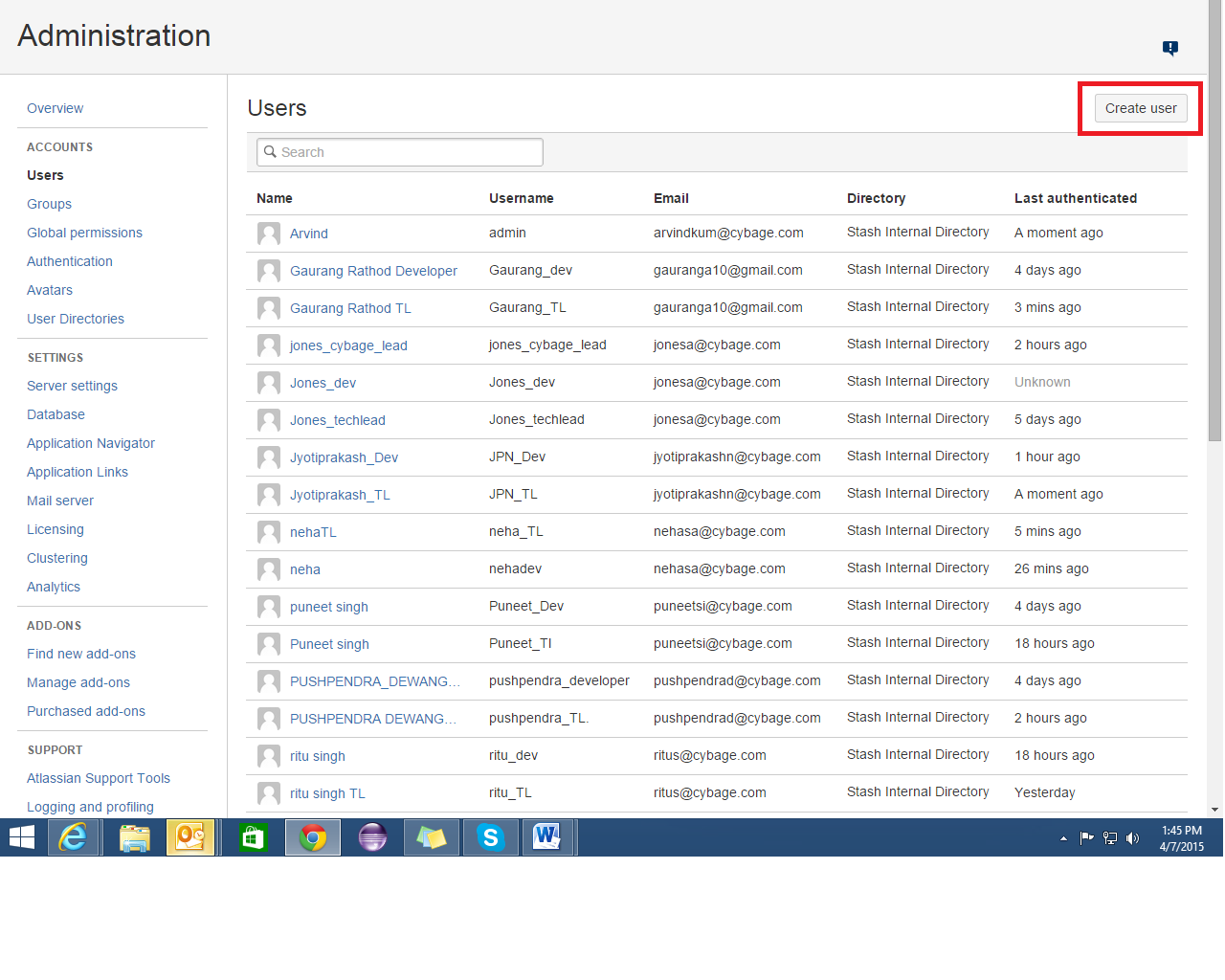
Add Repository to the project.

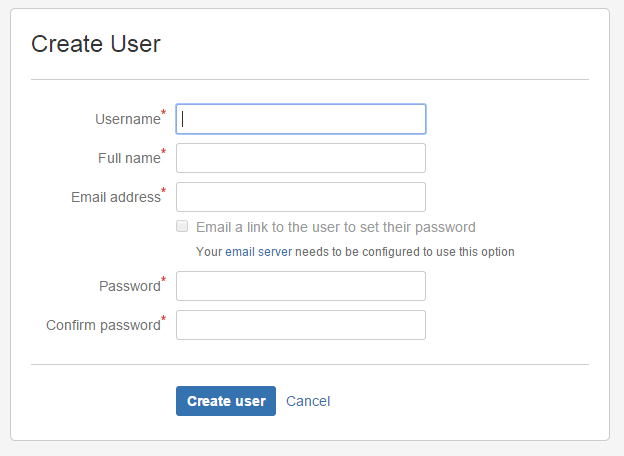


Once submitted you will be taken directly to your repository homepage. As there is no content in your repository yet, you'll see some instructions to help you push code to your repository:



Create users in stash. (Sakshi\_dev and Sakshi\_TL).





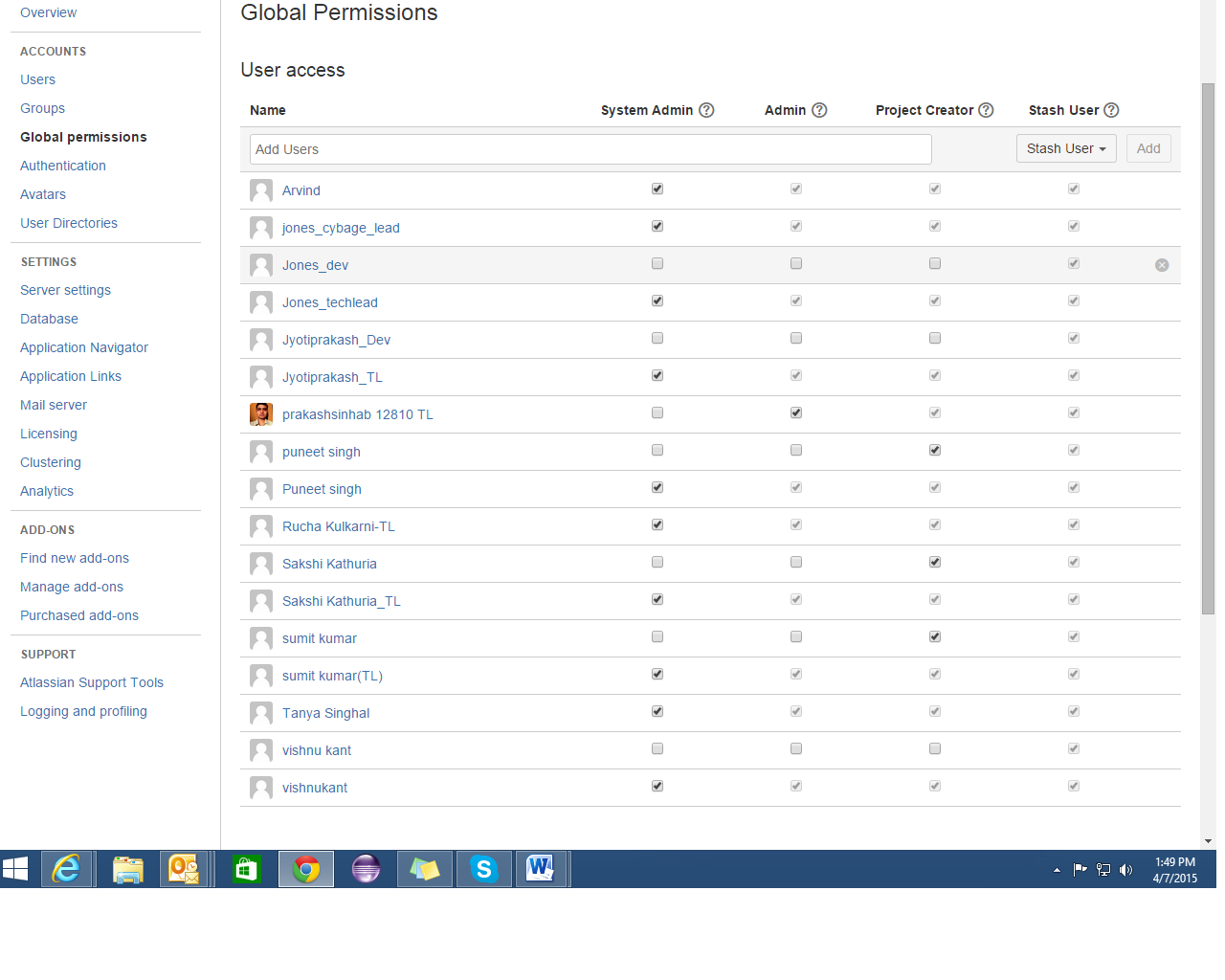
Controlling Access to Code

Stash provides the following types of permissions to allow fully customizable control of access to code.

* allow public (anonymous) access to projects and repositories.

Global permissions

* Control user and group access to Stash projects and to the Stash server configuration.
* For example, these can be used to control the number of user accounts that can can access Stash for licensing purposes.

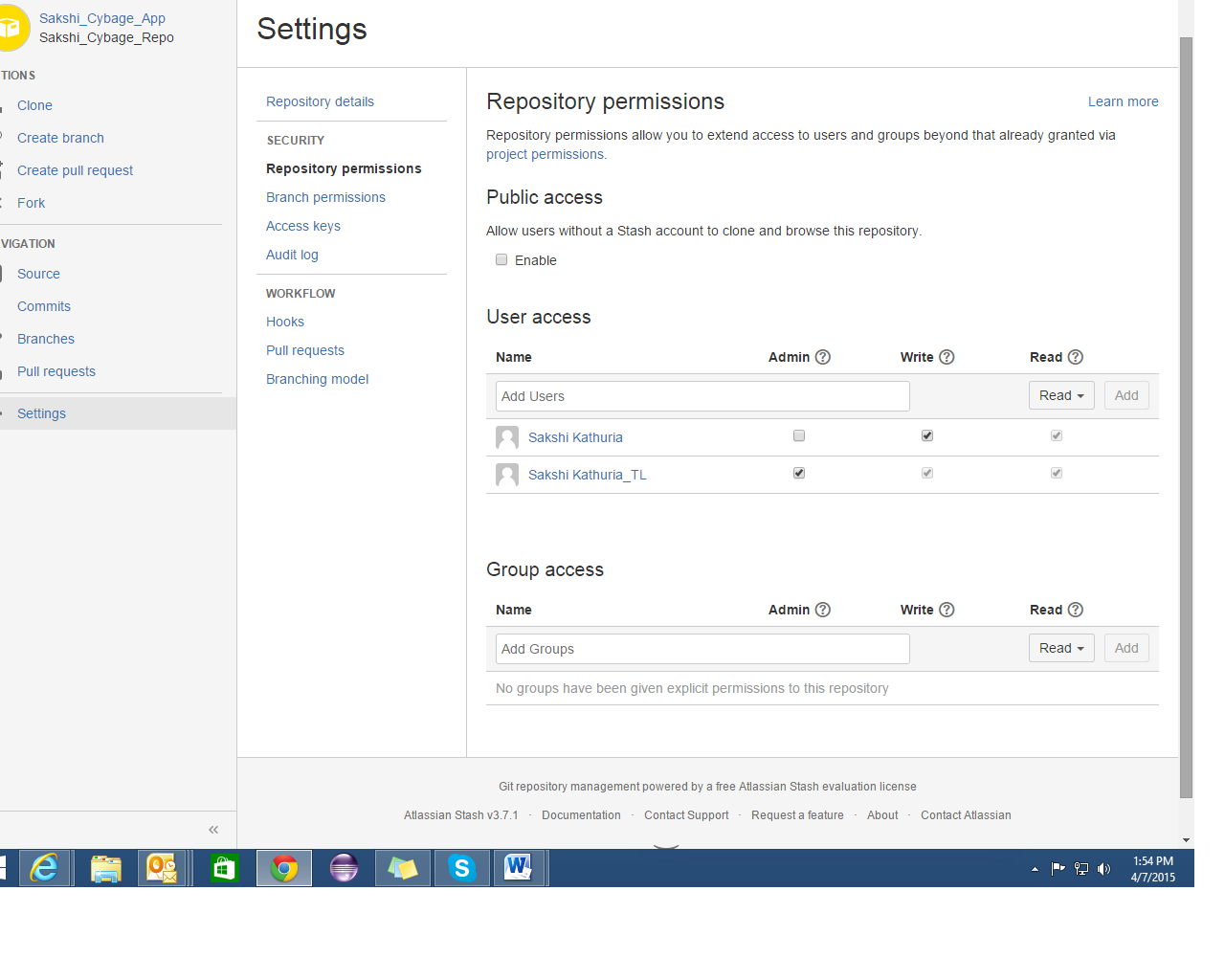


Project permissions

* Apply the same access permissions to all repositories in a project.
* For example, these can be used to define the core development team for a project.

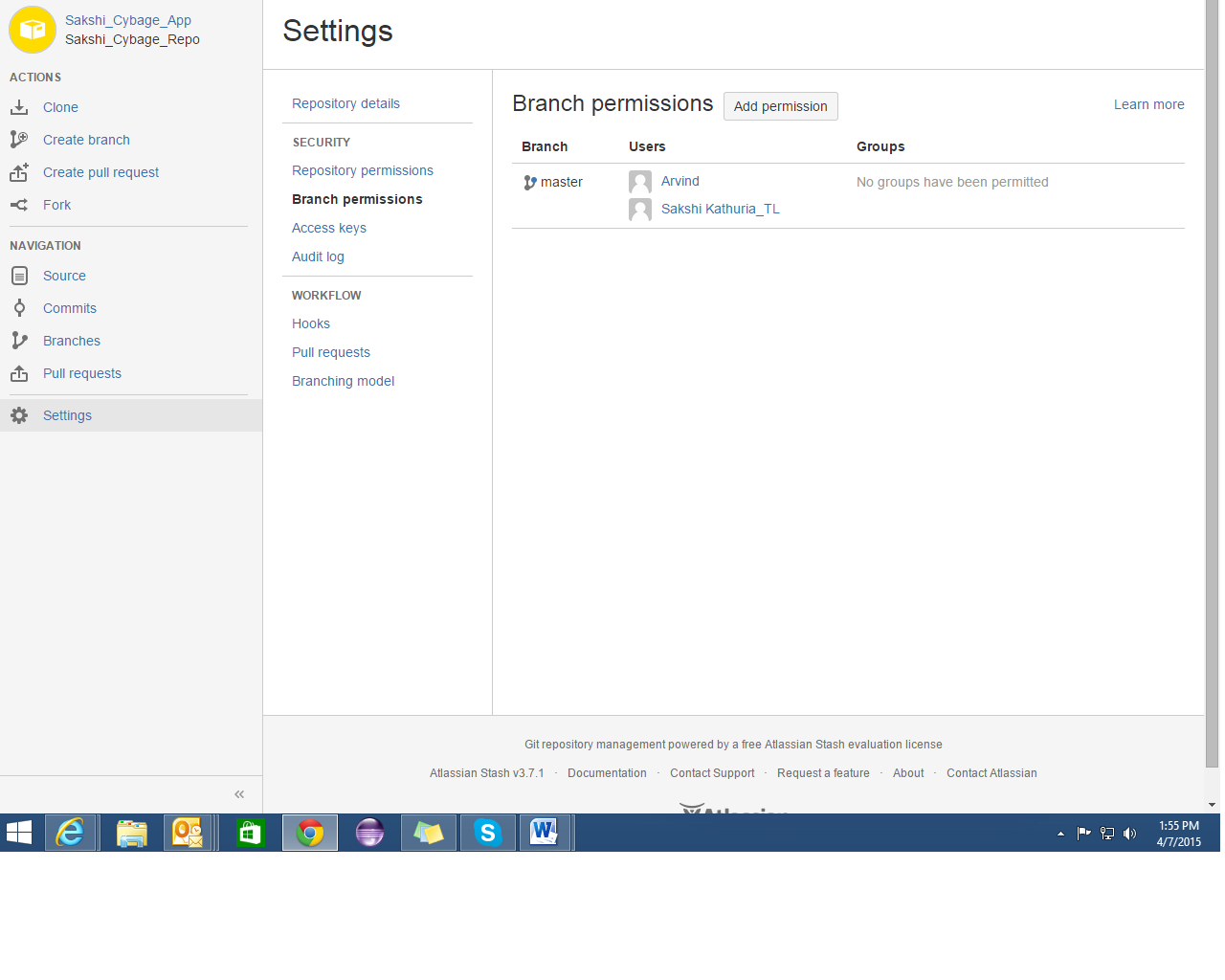
Repository permissions

* Extend access to a particular repository for other, non-core, users.
* For example, these can be used to allow external developers or consultants access to a repository for special tasks or responsibilities.



Branch permissions

* Control commits to specific branches within a repository.
* For example, these can provide a way to enforce workflow roles such as the Release Manager, who needs to control merges to the release branch.



## Making a repository publicly accessible

You can open up a specific repository for public (anonymous) access.

You need admin permission for the repository.

Go to the repository and click **Settings**, then **Repository** (under 'Permissions'). Check **Enable** (under 'Public Access') to allow users without a Stash account to clone and browse the repository.

## Making a project publicly accessible

You can open up a whole project (but not a private project) for public (anonymous) access.

You need admin permission for the project.

Go to the project and choose **Settings**, then **Permissions**. Check **Enable** (under 'Public Access') to allow users without a Stash account to clone and browse any repository in the project.

## Viewing public repositories

Stash displays a list of repositories for which anonymous access has been enabled.

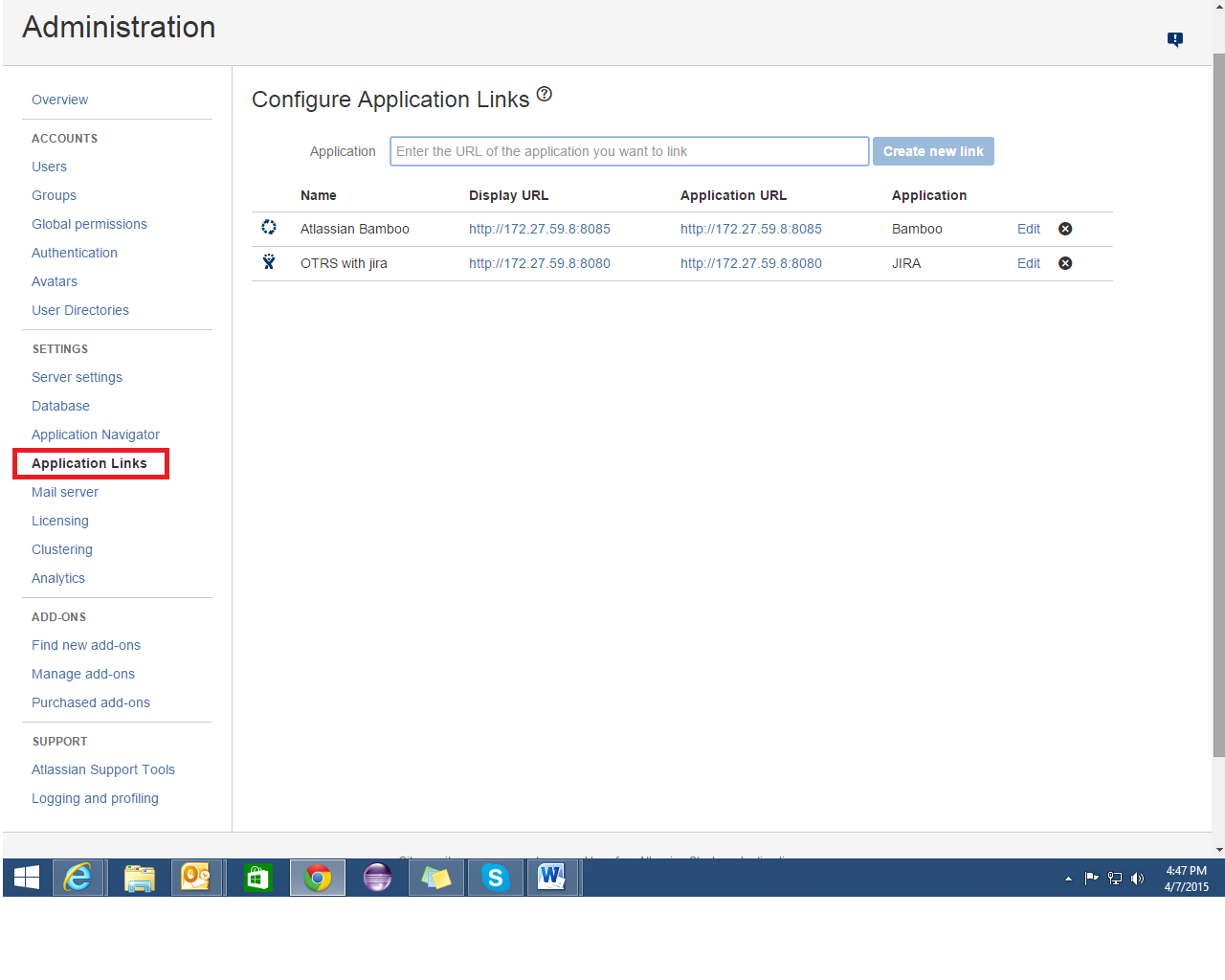
Anonymous and logged-in users can choose **Repositories** > **View all public repositories** to see these.

* Defined permissions /rights of users (branch, repo)
* Sakshi\_TL: admin, read, write access
* Sakshi\_Dev: read access
* Login from Sakshi\_TL in stash
* Committed some changes in code (master)
* Pushed back to the stash
* Repeated with branch
* Login from Sakshi\_Dev
* Only read access is there so can’t commit on master
* But can commit on branch
* Created a pull request from dev to TL
* Notifications will be shown in TL stash dashboard
* TL having the rights to merge, decline
* But dev can’t do anything he just can pull the request
* Provided the issue id in the comment of commit
* Then that issue will be shown in stash and we can move from stash to Jira with the using of that issue id.

Configure Application Links

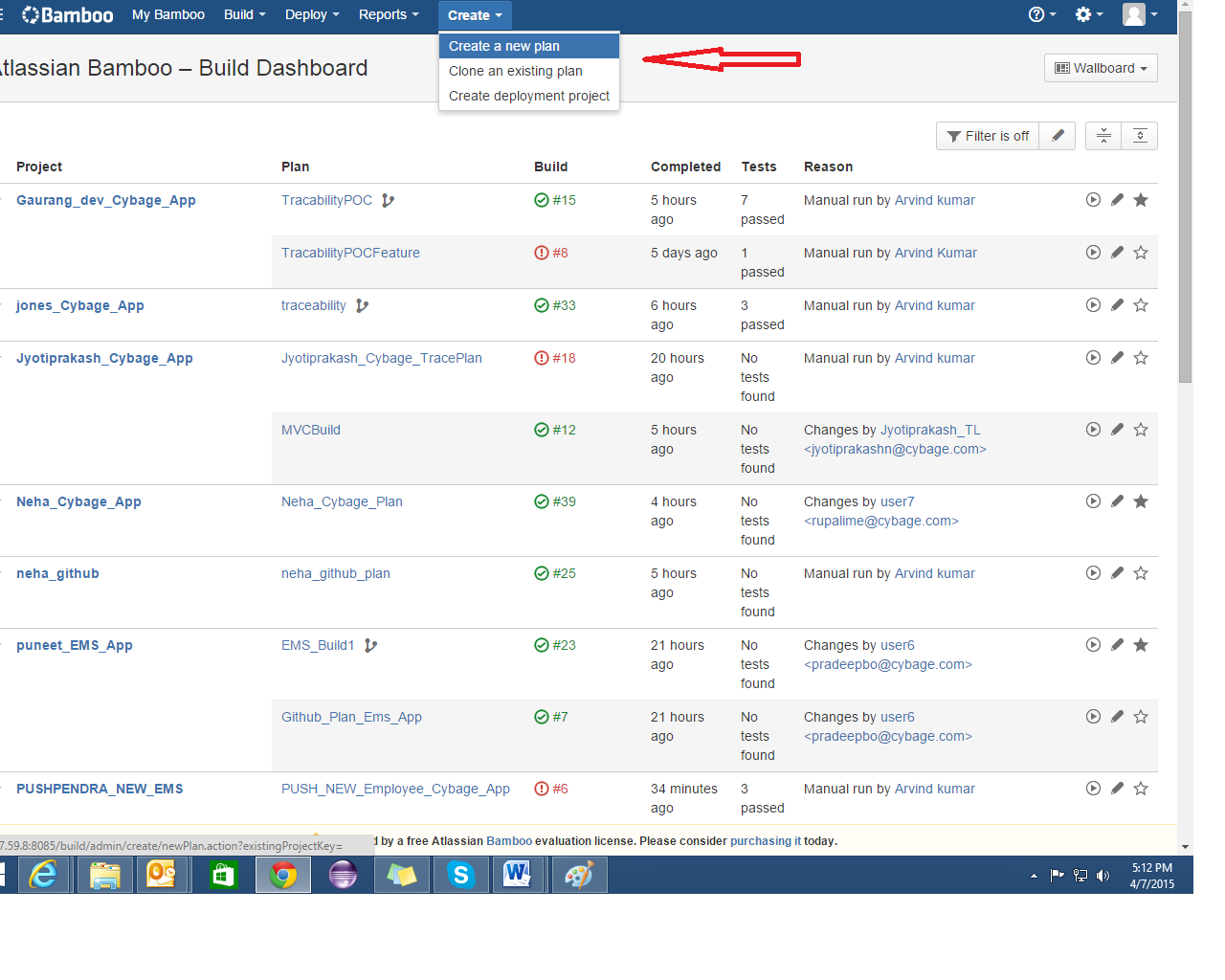
Go to Settings 🡪 Application links.

Enter the URL of the application you want to link.

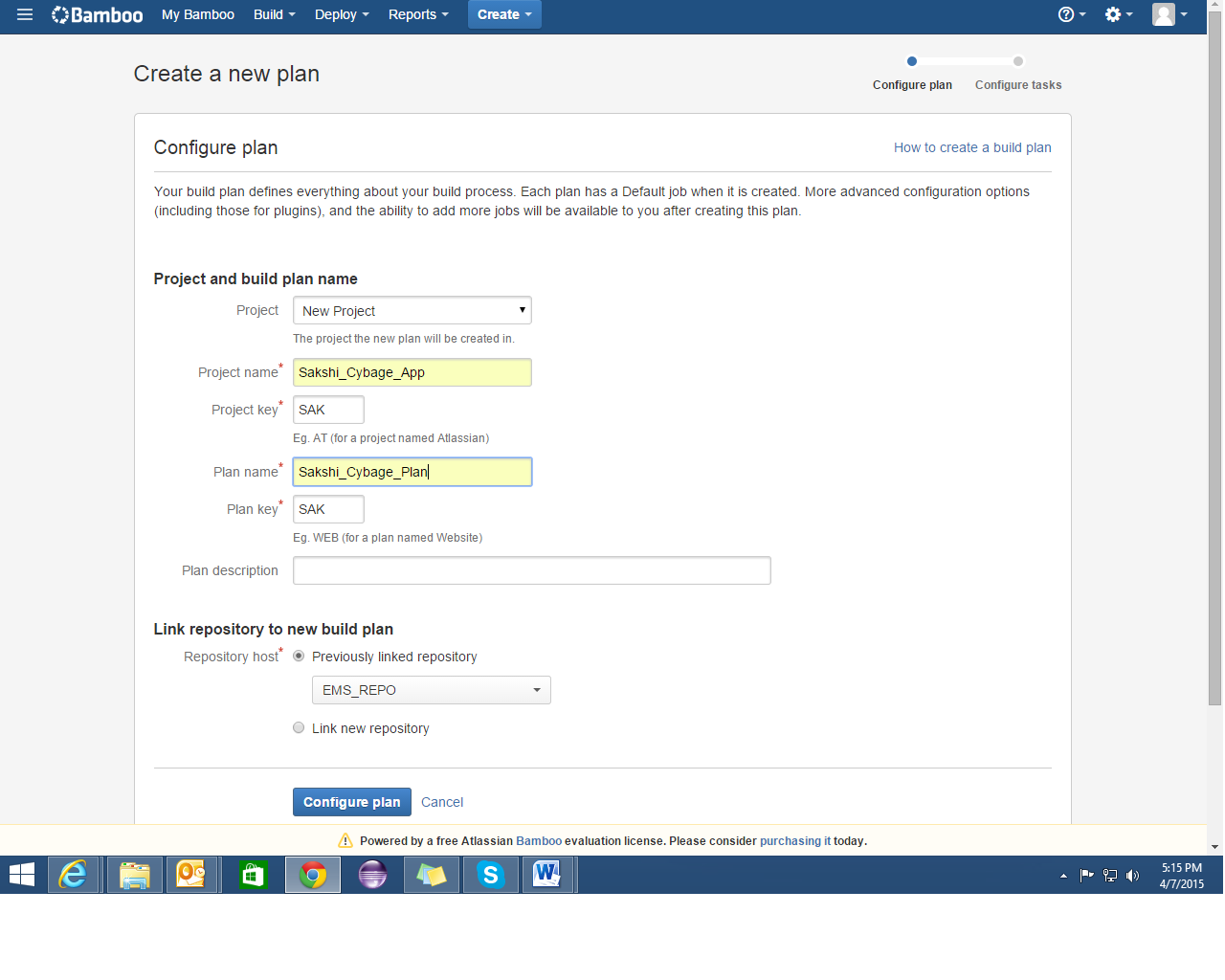


BAMBOO

* 1. Create a new plan.

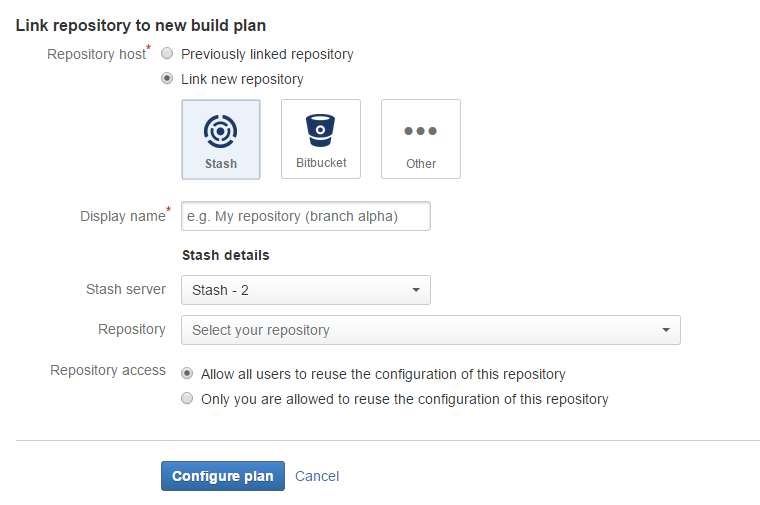


### Fill the details as shown below and Link repository to new build plan.

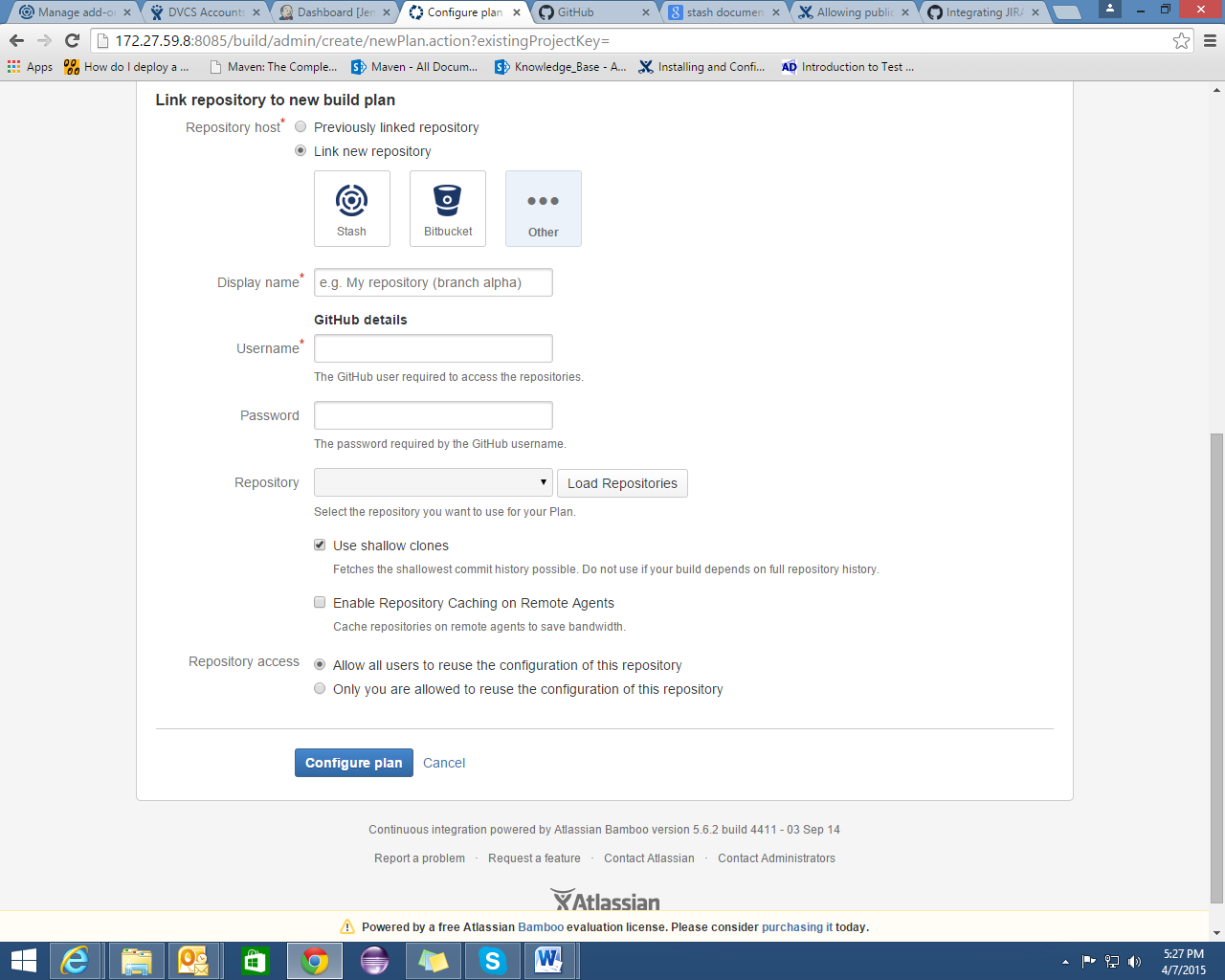


* 1. Link the previous repo if u have or link the new repo that’s all depends on ur requirement.
  2. Link new Repository
  3. Choose stash or github depending on ur requirement.

For Stash:

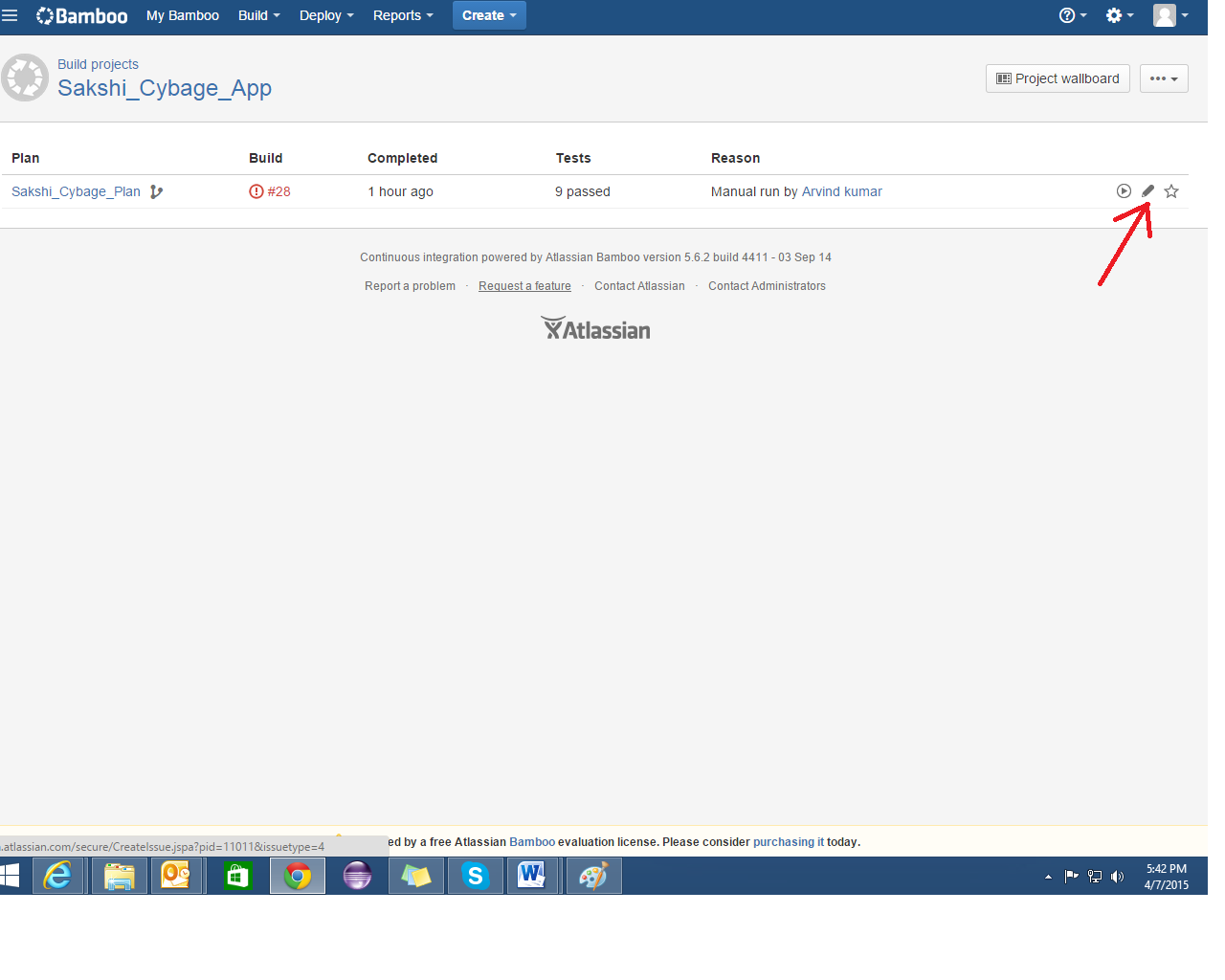


For GitHub:



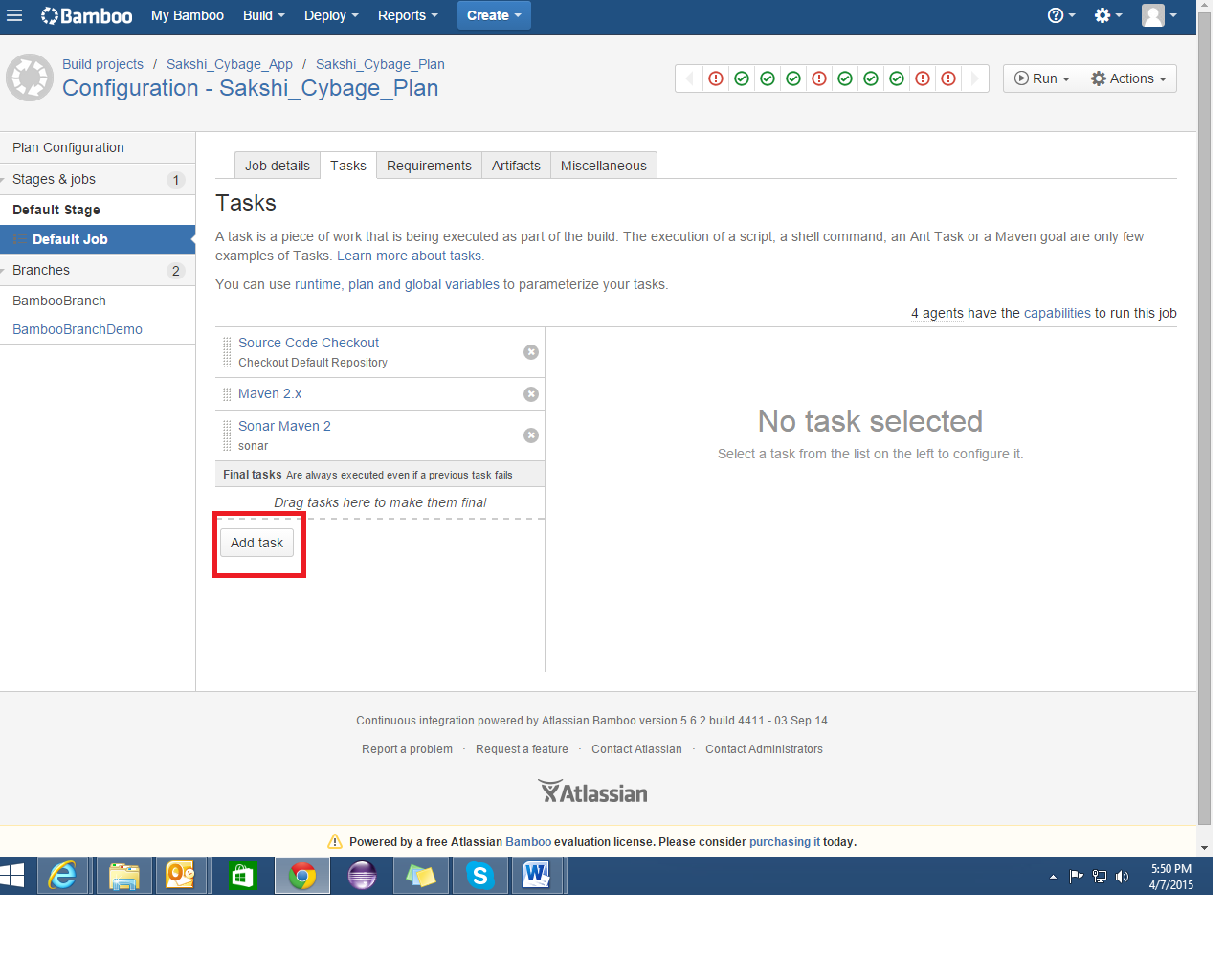
Go to the project that u have made.

Click on edit icon.

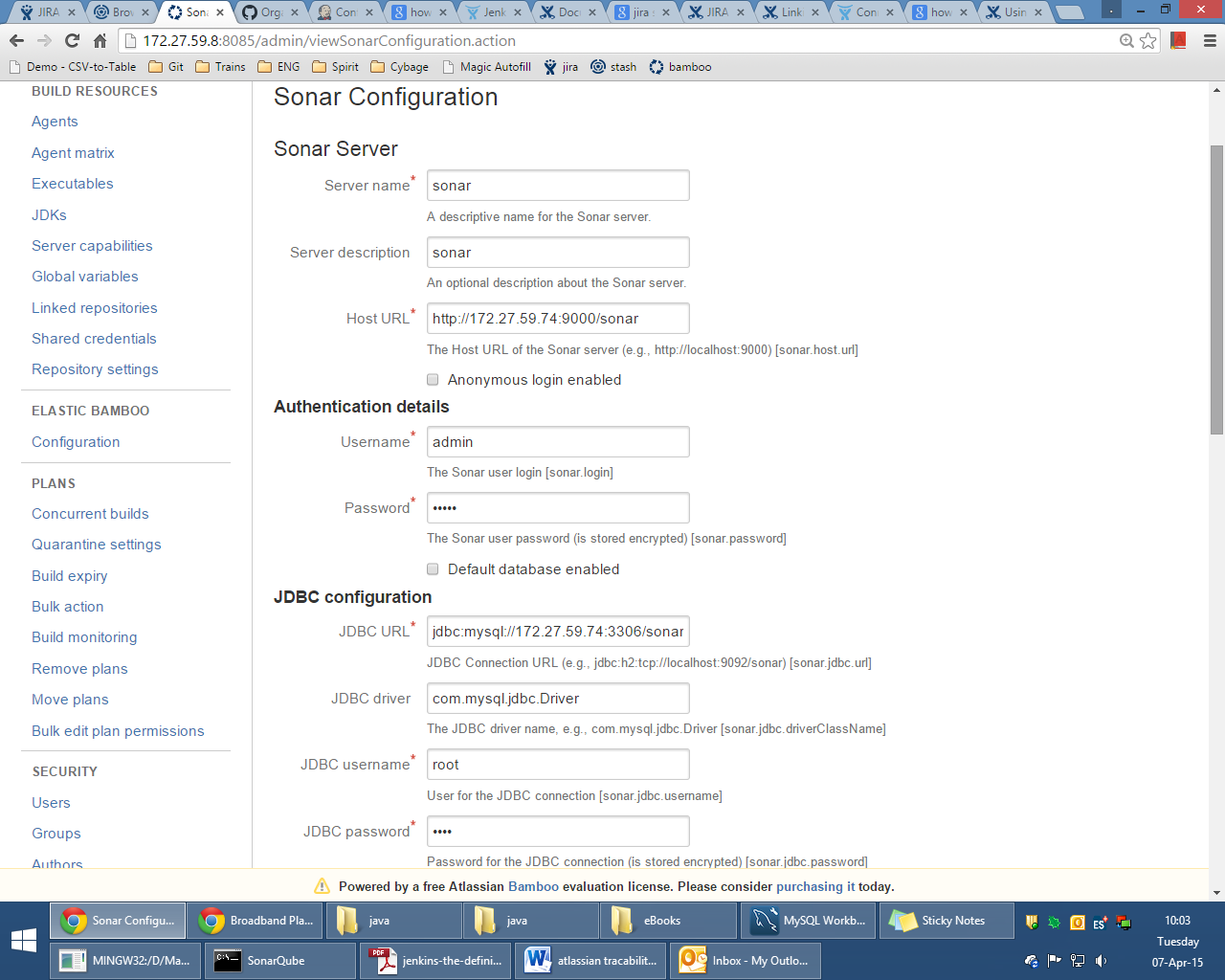


Choose the default job or add a new job.

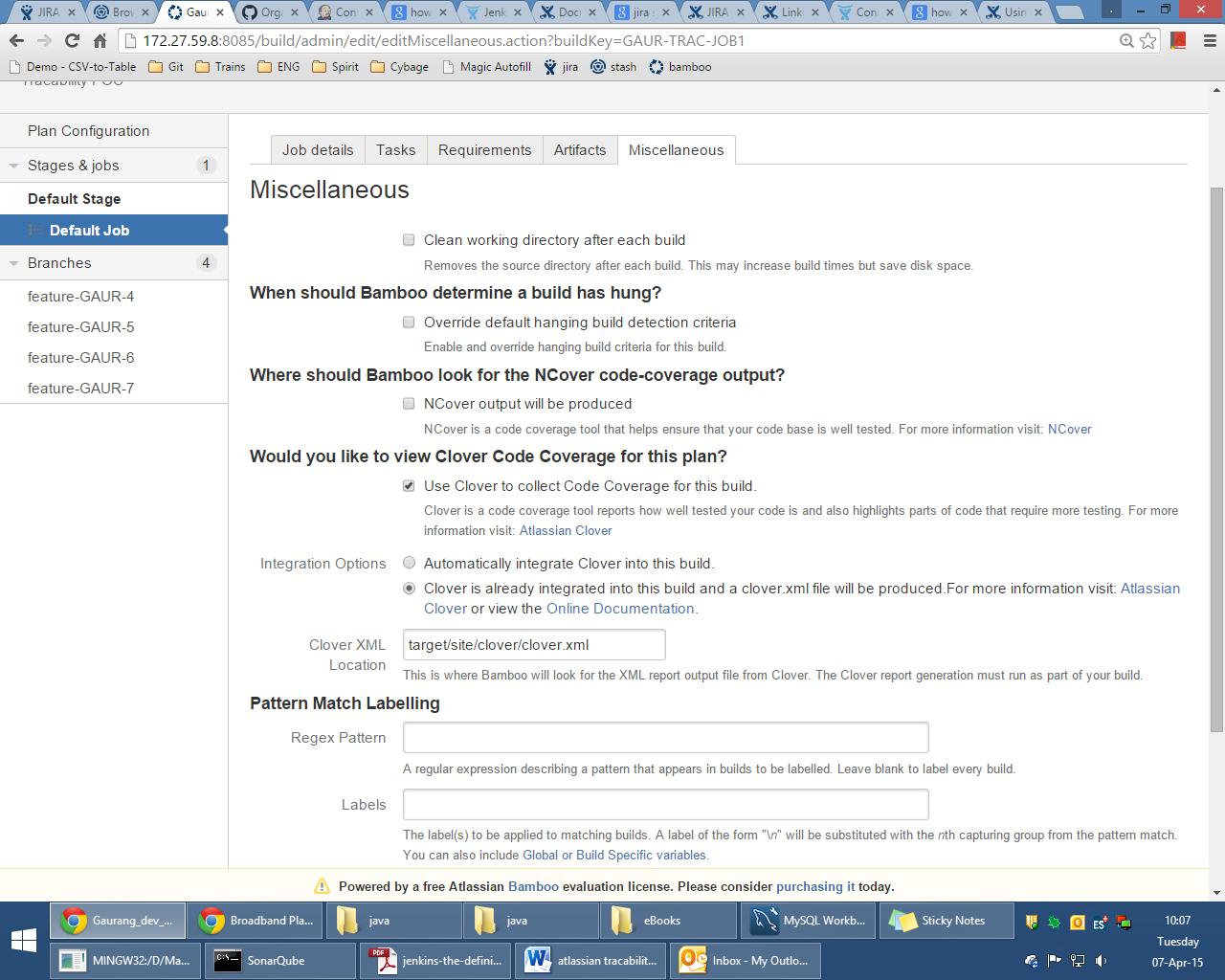
Add task according to your requirement.



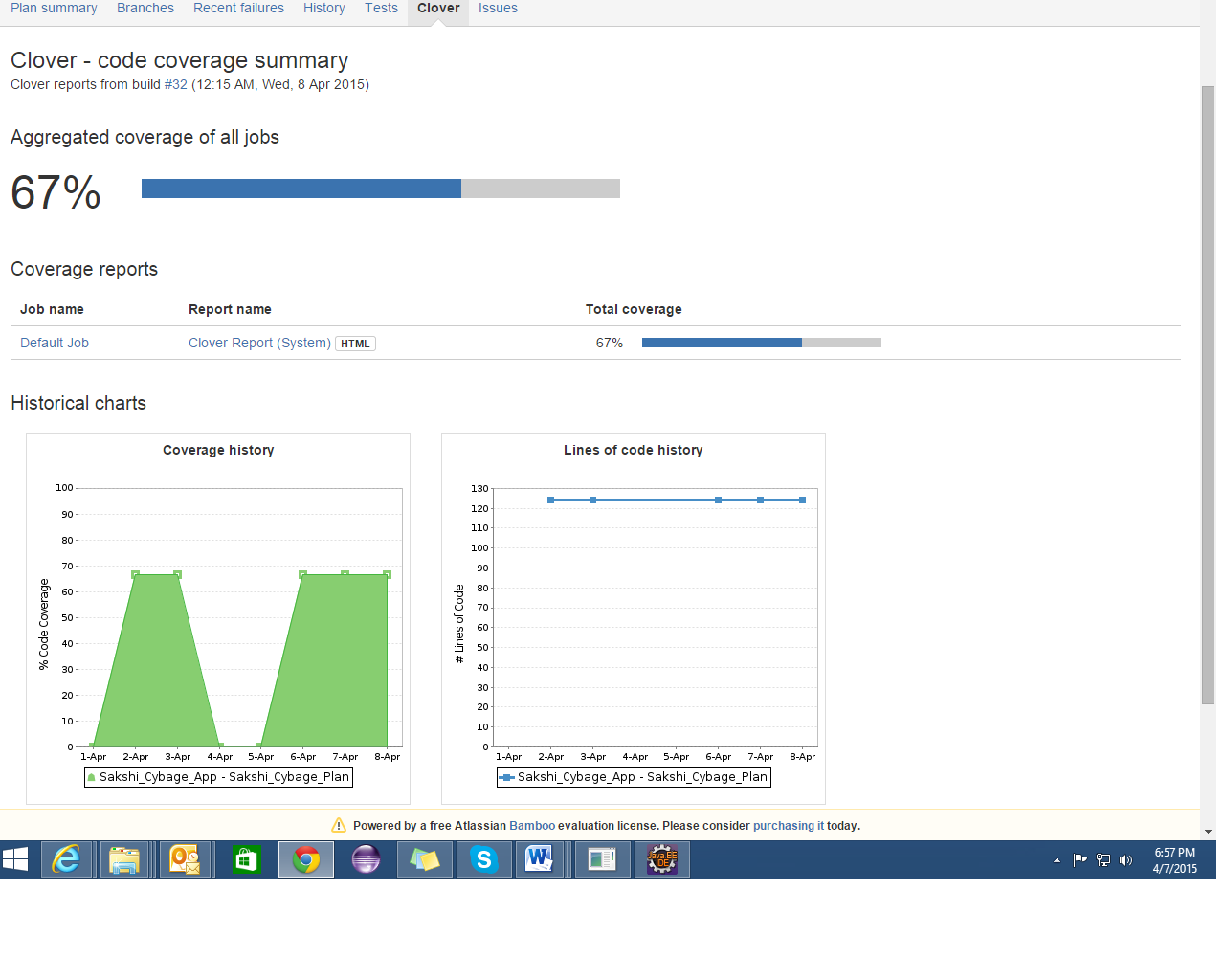
Add add-ons of Sonar from settings->add-ons and configure as follows:-



For clover in bamboo, add on is not required as it is inbuilt. If your POM is creating clover reports, then configure bamboo to render those reports by going to Configuration of default job -> Miscellaneous as follows:-



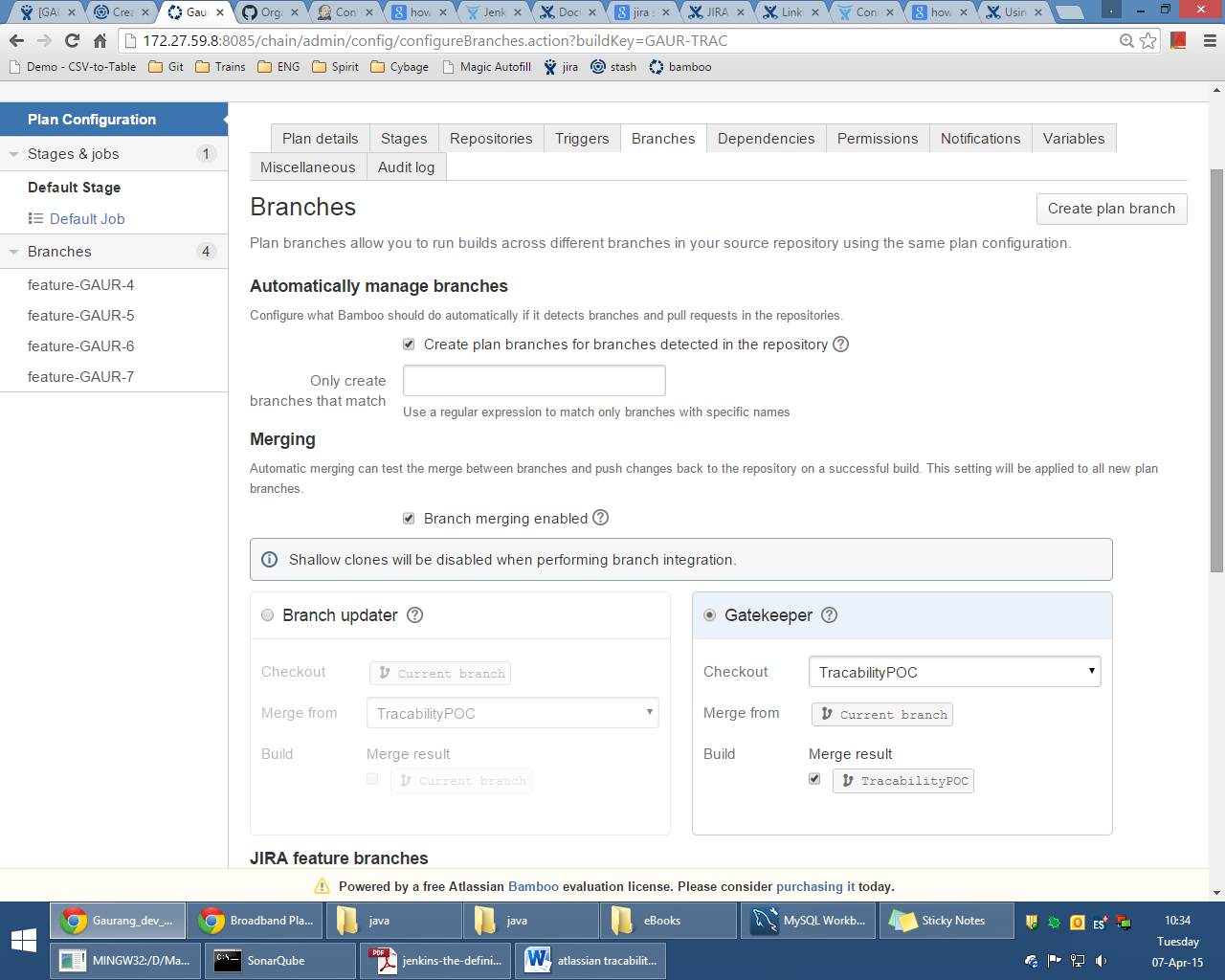
Now you can see Clover as follows:



For auto branch merging and for creation of plan branches, there is a feature in bamboo which detects any new branches in Stash and creates a plan branch whose configuration is inherited from main plan’s configuration. Then, the build is run on the newly detected branch and if the build is successful, then the branch can be automatically merged with the main branch. In this approach, there is no need for pull request.

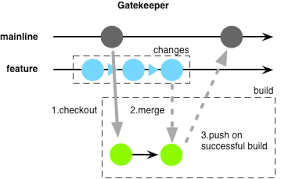
For configuring, go to plan configuration -> branches tab.

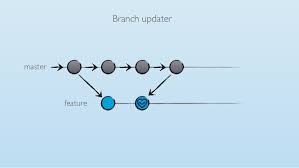
Configure as follows:-



There are 2 strategies:

1. Branch updater.
2. Gatekeeper





GITHUB