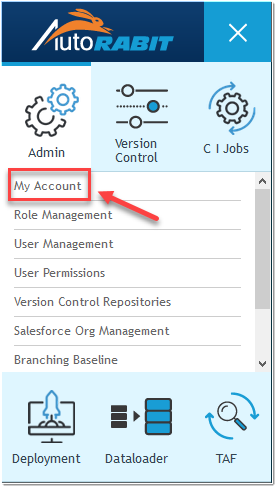
Managing Users Account

If you are the Administrator of AutoRABIT, you can create, edit, and view details about user accounts.

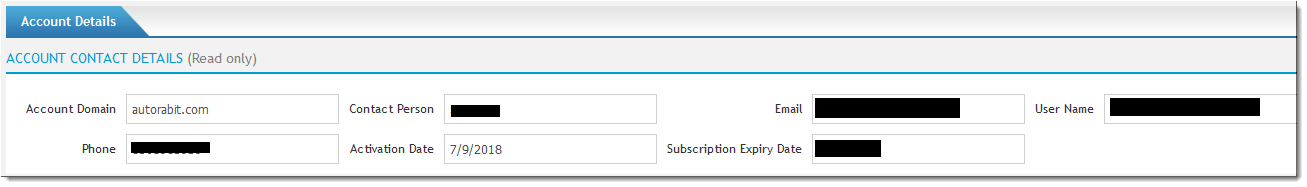
1. In the AutoRABIT homepage, click on **Admin > My Account**.



1. You’ll then be presented with a screen divided into ***Account contact Details****,* ***Subscription Type****,* ***SSO Configuration****,* ***Mail Extensions****,* ***ALM Management****,* ***Plugins****,* ***My Salesforce Settings****,* ***Profile / PermissionSets Settings****,* ***Commit Validation - Approval Settings****,* ***Validation Criteria- Static Code Analysis*** *and* ***Merge Settings*** sections.

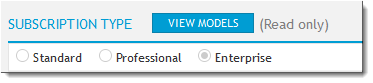
**Account Contact Details**

The account details section contains your accounts main information along with your subscription period with AutoRABIT.

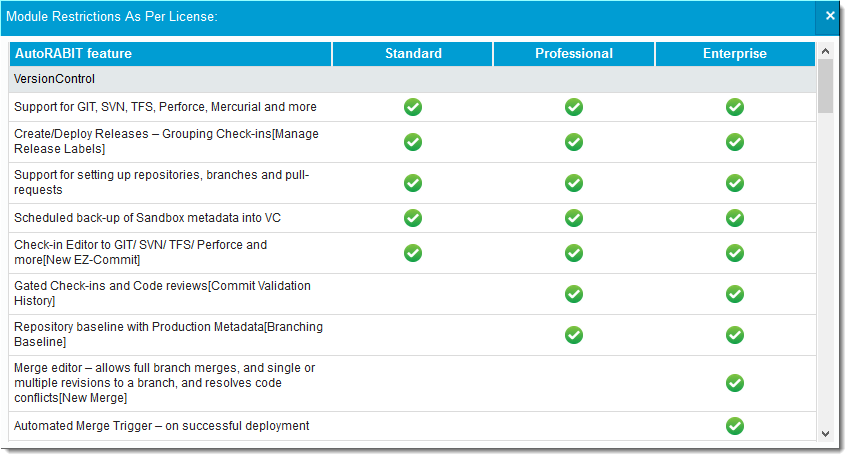


**Subscription Type**

Subscription Types are set up to match your organization requirements. View the current subscription type and list of modules granted by AutoRABIT based on your subscription.

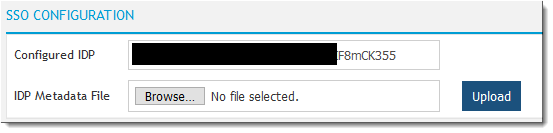


Click on **View Models** to view the complete subscription model.



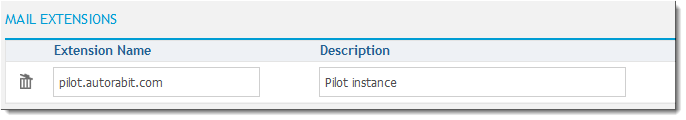
**SSO Configuration**

AutoRABIT uses the secure and widely adopted industry standard Security Assertion Markup Language 2.0 (SAML 2.0), which means our implementation of SSO integrates easily with any large identity provider that supports SAML 2.0. Here, the user can configure and validate the identity provider settings to enable single sign-on.



**Mail Extension**

The admin has the option to add new mail extensions based on the organization's requirement.



**ALM Management**

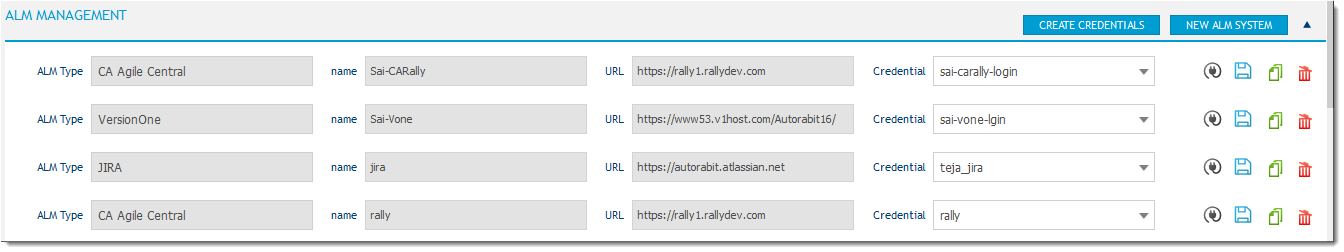
The admin can configure a new ALM system in this section. AutoRABIT currently supports integration with ALM tools like **Jira, IBMRTC, Version One, CA Agile Central** and **VSTFS**.

**To configure an ALM, follow the below steps:**

1. Click on **New ALM system**.
2. Select the **ALM type**.
3. Enter the **name** and the **URL** associated with the ALM type.
4. Choose the **Credentials**. Click on **Create Credentials** if the credential is not created.
5. Click to **Save**.



1. Click to check if the connection has been authenticated or not. A success message is displayed, after the authentication is completed.



1. Additional options:
   1. **Clone the ALM system**: Click on icon to configure a new ALM system using the existing ALM details.

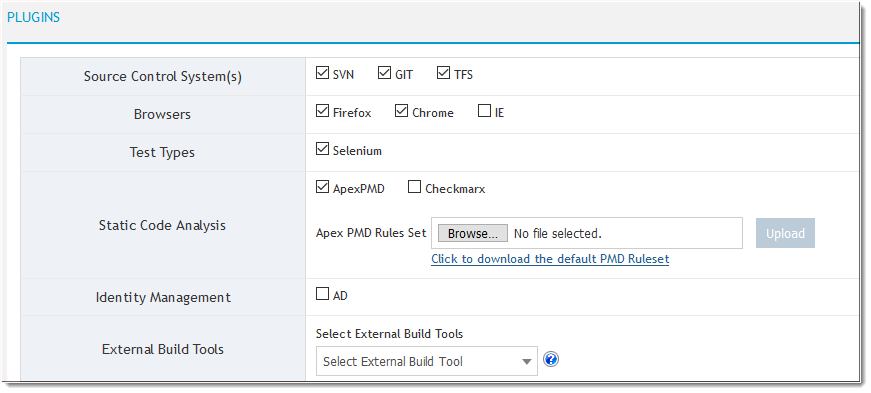


* 1. **Delete a ALM system**: Click on icon to delete the ALM system.



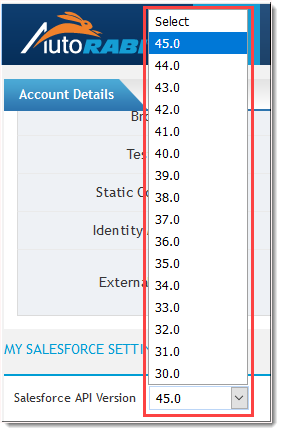
**Plugins**

This section lists various plugins that are configured in AutoRABIT. Based on the organization requirement, the admin can select the desired Plugins to be used and register the same by giving the correct credentials. ([Learn More](#c0b98f5b-1392-44f5-87f3-d100218f0a29))



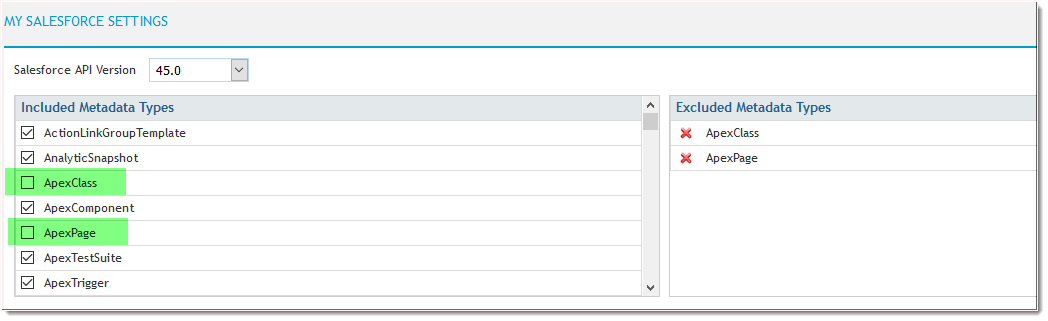
**My Salesforce Settings**

AutoRABIT supports all the metadata types, based on the **Salesforce API** version. AutoRABIT supports APIversion **30.0** to **45.0** as of now.



**Note:** Whenever the API Version is modified in the mid of a task, re-save the ongoing task to get new configuration reflected.

Select the API version to see the supported metadata types. To avoid unnecessary fetching of metadata types while executing a CI job, ensure that you have excluded them.



**Additional options**:

There are several options that can be configured in this section:



1. **Exclude Baseline Managed Package Changes**: Upon selection, it will consider only modified/changed metadata members and not base components of the managed packages.
2. **Include Default Apex Tests For Run Tests Based On Changes**: When selected, the [default configured tests](#c6203975-41bf-4bd4-bc21-24573b195704) will be added to the set even if Test classes or Apex Class, Apex Triggers are not available. Apex Test Level will execute as **''RunSpecifiedTests''**.

However, if the checkbox is left unchecked, then no default tests will be added, and no Apex Test Level will be set. Salesforce default behavior will be expected in such case.

1. **Enable Delta on PermissionSets:** As per the Salesforce behaviour, for **Salesforce API 40 or later**, the entire PermissionSets will be replaced with the latest changes. However, when **"Enable Delta on PermissionSets**" checkbox is selected, the PermissionSets will get retrieved from the Source org and will append with the latest changes in the deployment package.

To understand more, let’s take the below scenario:

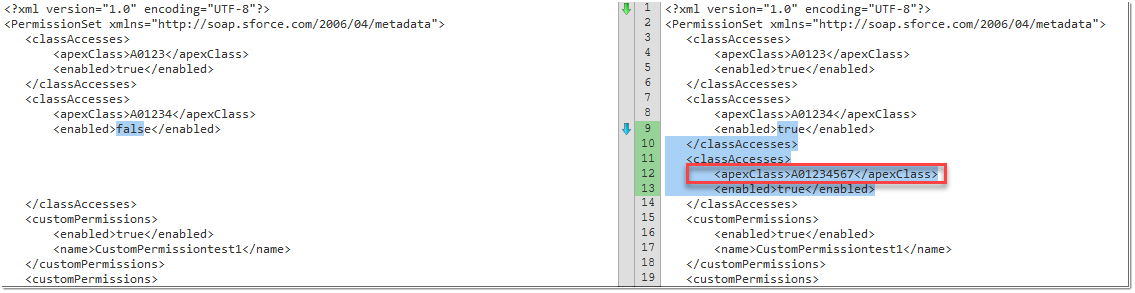
**Scenario 1**:

* + **In Source org**: Deploying Permissionsets with one Apex class permission
  + **In Destination Org**: Same permissionsets with different apex classes permission already exists

As per the Salesforce behaviour, the destination org permissionsets will be overriden with newly deployed permissionset.

However, if the user selects **"Enable Delta on PermissionSets"** checkbox, the destination org permissionset permissions will be appended with new class permission as shown below.

**Source org**  **Destination org (before Deployment)**

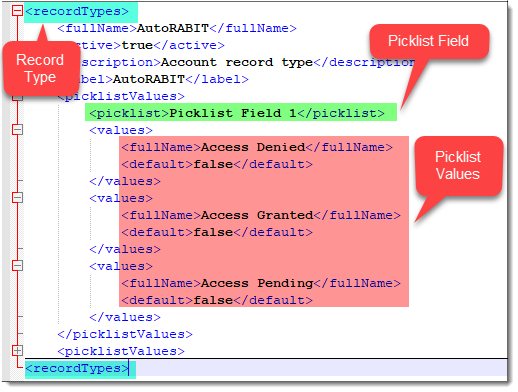


**Destination org (after Deployment)**



1. **Configuration for recordTypes picklistValues**: Here you can choose the configuration for the **recordTypes picklistValues** i.e., either *replace*, *replace all* or *append*.

Record Type node contains certain Picklist fields and its corresponding Picklist values that are assigned to Record Type. Refer the screenshot attached.



As per the behaviour of Salesforce, if a user likes to retrieve certain picklist field along with Record type, only those picklist values in Record Type Node gets retrieved. In case of selection of Configuration for recordTypes picklistValues option as '**Replace**', for every ez-commit operation, if the Record Type has no picklist values, it will over-ride the Record Type Node in Version control even it has more than one picklist field values. Same applies to '**Replace All**' option. The only difference is '**Replace All**' option will replace the entire existing picklist values.

For '**Append**' option, instead of over-riding the entire record type picklist values, it keeps on adding to the existing picklist values.

The below table highlights the expected behavior on **Picklist Fields** and **Picklist Values** of Record Type node for *Append, Replace and Replace All configuration.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **ITEM** | **OPERATION** | | **ACTION**  **(Performed in your CustomObject file in Version Control)**    ***For added/ modified***  ***For deleted*** | | | |
|  | | Append (No Overwrite) | a. **Modified/Added :** Reads the CustomObject file and appends the CHANGE (PicklistField/ PicklistValue) to the RecordType Assignment irrespective of the existing data. | No action | |
| **Picklist Field/ Value Assignement** | | Replace (Partial Overwrite) | a. **Modified:** Reads the existing Picklist Field in the CustomObject file and overwrites it with the newly introduced CHANGE in the RecordType Assignment.  b. **Added:** Reads the existing Picklist Field in the CustomObject file and adds the CHANGE into the RecordType Assignment. | Deleted Picklist Fields/Values will still remain in the Version Control system. | |
|  | | Replace All (Complete Overwrite) | a. **Modified/ Added:** Overrides the entire RecordType in the CustomObject file with the CHANGE | Deletes the entire Picklist fields and PicklistValue as the action will overwrite the entire RecordType Picklist Assignment  **Note:** *To delete a Picklist field assignment from Record Type node in your Version Control System, select all the Picklist fields and its corresponding Record Type in your EZ- commit operation.* | |

**Added\* -** A new Picklist Field/value has been introduced in the Salesforce Org

**Modified\* -** An existing Picklist Field/value has been changed in the Salesforce Org

**Deleted\* -** A Picklist Field/Value has been deleted in the Salesforce Org

**Change\* -** Difference pulled from the Salesforce Org

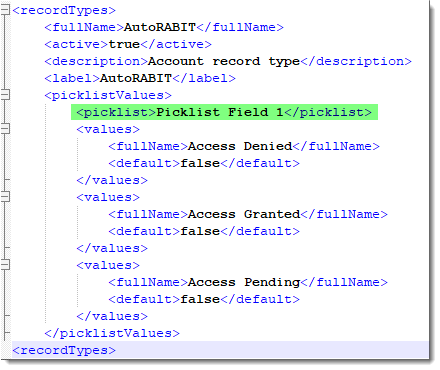
**Detailed Explanation (with examples)**

1. For **Configuration for recordTypes picklistValues** as **"Replace"**, there can be two possible scenarios:

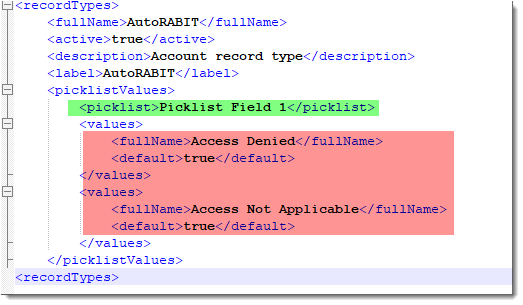
**Scenario 1:** Picklist field exists in the Record Type

If the Picklist field exists in the Record Type, it overrides its corresponding Picklist values.

**Before:** Search Picklist field 1



**After:** Picklist field 1 being available, it overrides the existing values for Picklist field 1.

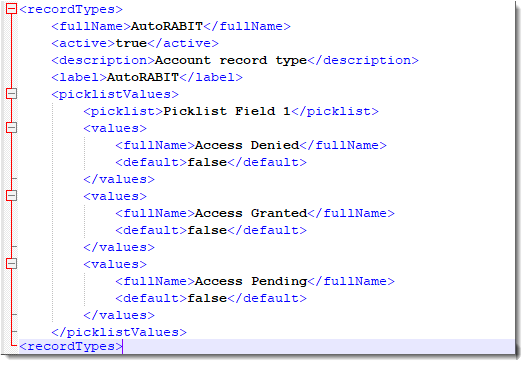


**Scenario 2:** Picklist field do not exist in the Record Type

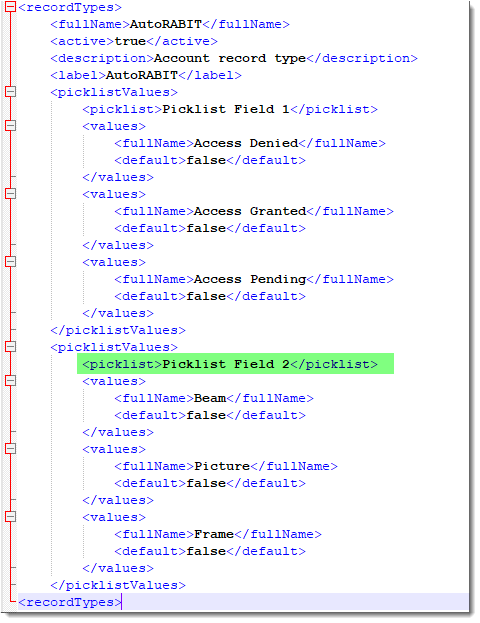
It search for the Picklist field availability in the Record Type and if the Picklist field is not present, it gets added to the Record Type.

Refer the screenshot attached below:

**Before:** Search Picklist field 2

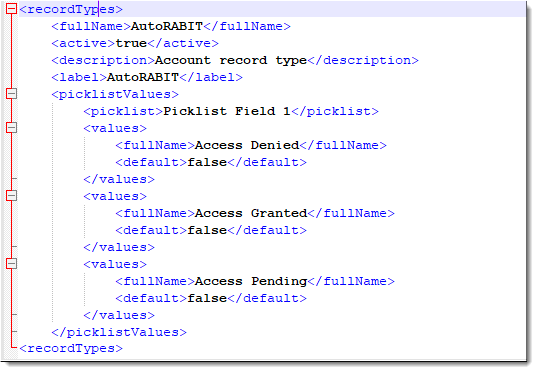


**After:** Picklist field 2 not being available, it gets added to the Record Type thereby acting as an append operation.

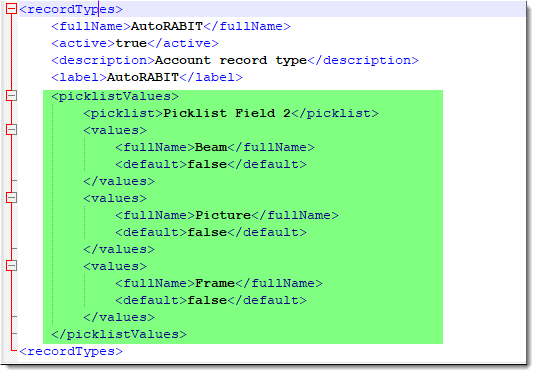


1. For configuration for recordTypes picklistValues as **"Replace ALL"**, it will replace the entire Picklist fields and its corresponding Picklist values for the Record type.

**Before**



**After**

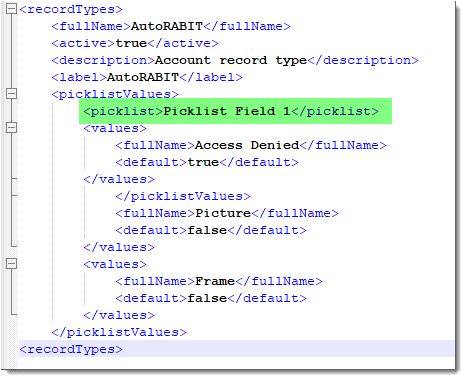


1. For Configuration for recordTypes picklistValues as **"Append"**, there can also be two possible scenarios:

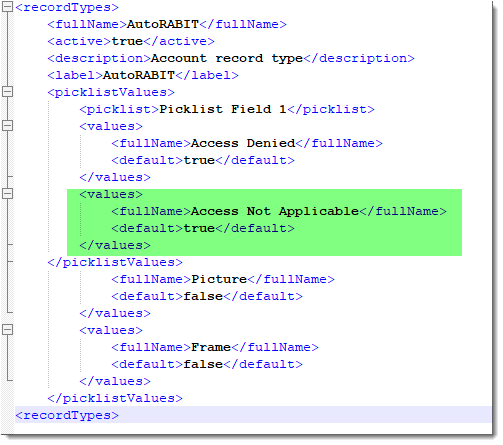
**Scenario 1:** Picklist field exists in the Record Type

If the Picklist field exists in the Record Type, it updates the existing Picklist values and add the new ones.

**Before:** Search Picklist Field 1



**After:** Picklist field 1 being available, new Picklist Values gets added.

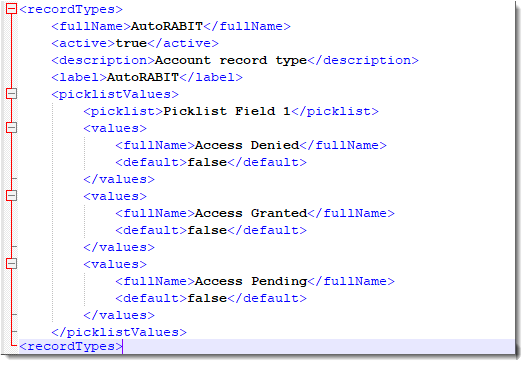


**Scenario 2:** Picklist Field do not exist in the Record Type

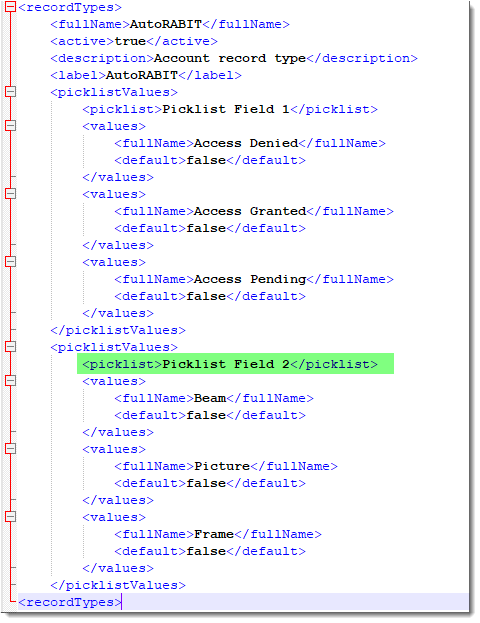
It searches for the Picklist Field availability in the Record Type and if the Picklist Field is not present, it gets added to the Record Type.

Refer the screenshot attached below:

**Before:** Search Picklist Field 2



**After:** Picklist Field 2 not being available, it gets added to the Record Type.



3. **Configuration for LabelTranslations**: Options to choose the configuration for the LabelTranslations i.e., either *replace* or *append*.

In case of selection of Configuration for LabelTranslations option as '**Replace**' , for every ez-commit operation, if the Label Translation has no custom label metadata type, it will over-ride the LabelTranslations in Version control even it has more than one custom label metadata type values.

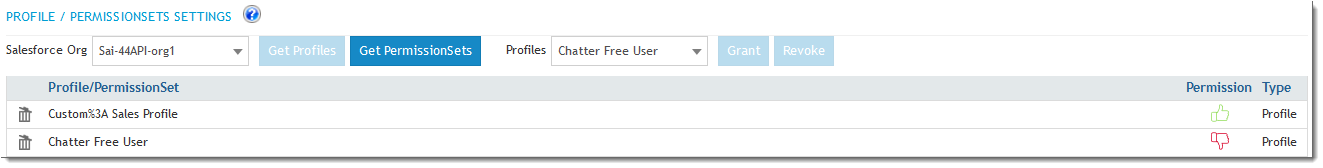
For '**Append**' option, instead of over-riding the custom label metadata types, it keeps on adding to the existing one.

**Profile / PermissionSets Settings**

This section deals with either granting or revoking permissions to the **Profiles/Permissionsets** of any org. Based on the permission granted or revoked, the same gets affected after committing the custom object in the Version Control.

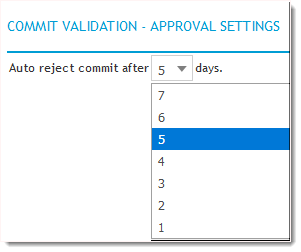
To grant or revoke permissions to Profiles/Permissionsets, follow the below steps:

1. Select the Salesforce Org.
2. Click either **Get Profiles** or **Get PermissionSets**.
3. Based on the above selection, choose either the **profiles** or **permission** from the list.
4. Grant or revoke the permissions for the selected Profiles/Permission Sets.

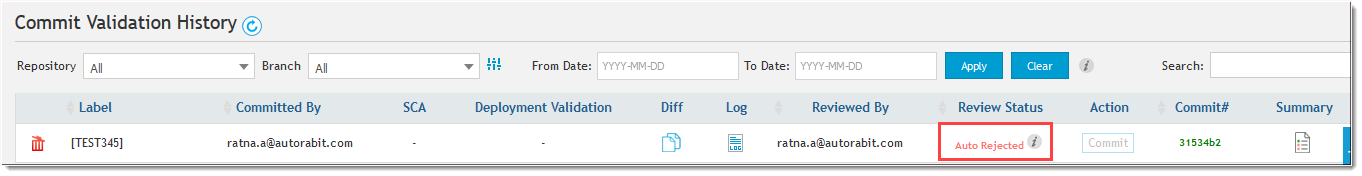


**Commit Validation - Approval Settings**

Auto rejects an approval for pre-validation commit after the days mentioned here.

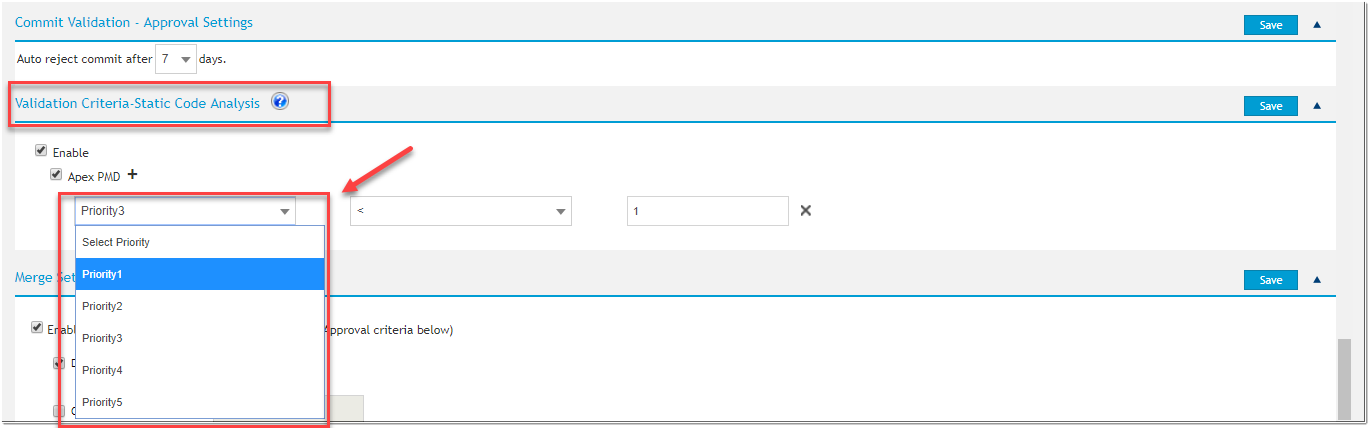


The reviewer has an option to either **approve** or **reject** the pre-validated commit in **Commit Validation History** screen. In case, if the reviewer has failed to approve the commit within the stipulated days above, the review status gets auto rejected. See the image attached.



**Validation Criteria- Static Code Analysis**

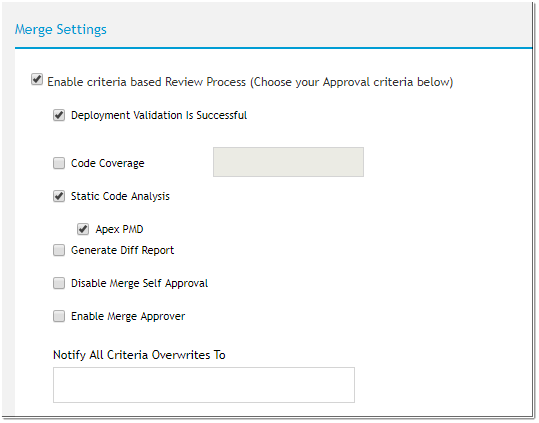
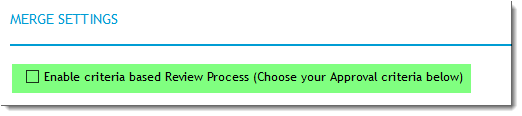
With the current release, the user can set the global criteria to enforce Static code analysis tools across CI Jobs and Merge Jobs. Based on the priority set, the build will be successful only if the criteria are met. The build will get failed if the Apex Classes, Triggers and Visualforce pages in the Build does not the passes the priority set.



**Merge Settings**

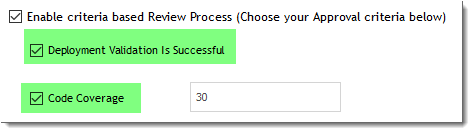
Here the administrator can specify certain **evaluation criteria** on which the merge will get reviewed before getting it committed to Version Control branch. Based on the criteria selected, the same option gets reflected in the [**New Merge**](#bb17517b-61cc-437e-b17a-1199f8ceccec) screen.

1. To enable the merge setting, you need to select the checkbox: **Enable criteria based Review Process (Choose your Approval criteria below)**

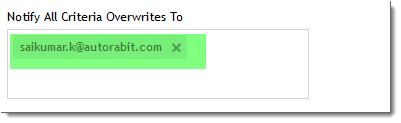


1. Select the required criteria checkboxes:
   1. **Deployment Validation Is Successful:** This will allow the users to perform a validation deployment before committing the changes. If the deployment is successful, the commit is executed. This option will permit the merge operation to proceed further only if the deployment is successfully validated.

**Code Coverage:** This option generates the code coverage report which provides information about the apex tests that are run, the classes that are covered, and the assertions that have failed and provides a percentage of the code that is covered by the test execution.



* 1. **Static Code Analysis:** Select this option if you would like to run static code analysis tool to identifying potential software quality issues before the code moves to production.
  2. **Generate Diff Report**: Select this option to auto generate a code difference report on completion of the commit.
  3. **Disable Merge Self Approval:** This option will allow you to prevent users that have committed to a merge request from approving it.
     1. Tick the checkbox **"Disable Merge Self Approval"**.
     2. Click **Save** to save the changes.
  4. **Enable Merge Approver:** Allowing merge request authors to approve the merge requests.
  5. **Notify All Criteria Overwrites To**: This specified email id(s) will receive an overwrite email notification, every time the user tries to overwrite the evaluation criteria set in **Merge Setting** section and tries to merge the files to a branch.

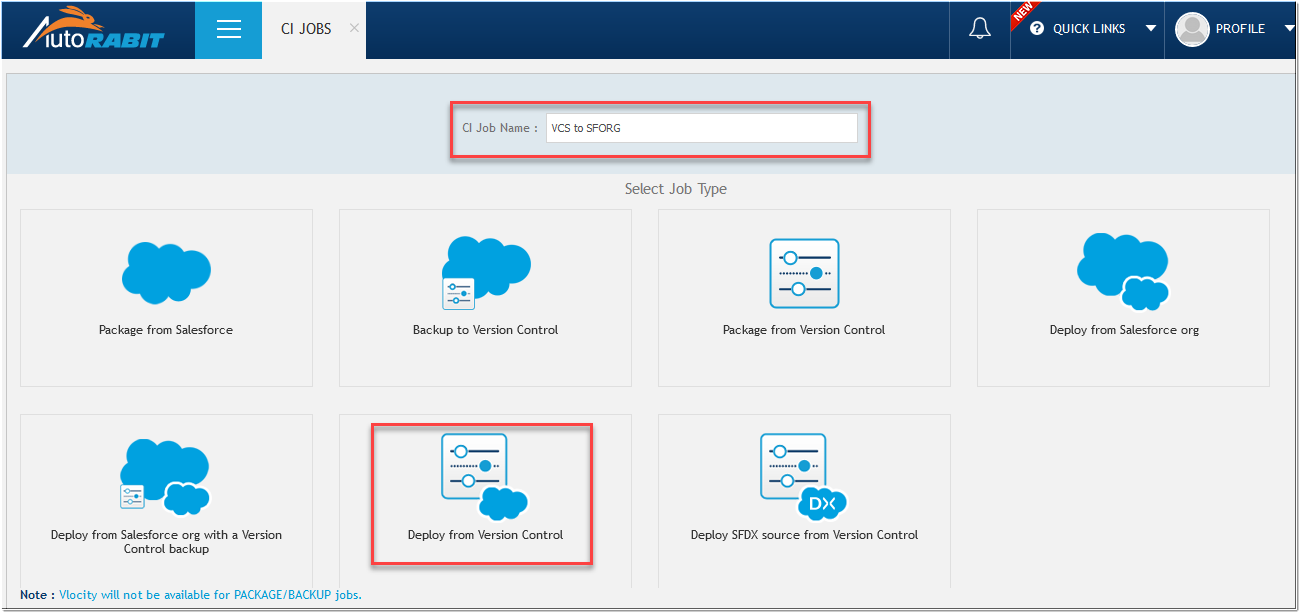


How to deploy from Version Control to a Salesforce Org during CI process?

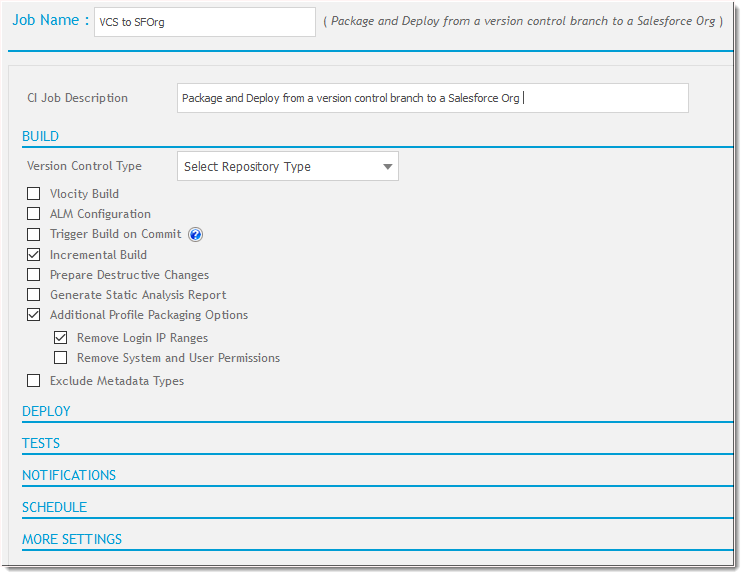
Use AutoRABIT CI intelligence to extract and package from a Version Control branch to deploy into a Salesforce Org. Additionally, you can configure the job to pick up revisions based on your ALM story status and/or run functional test cases on your destination org.

**To deploy from Version Control, follow the below steps:**

1. Navigate to the **Create CI Job** page, found under **CI Jobs.**
2. Give the job a descriptive name in **CI Job Name** field.
3. Click **"Deploy from Version Control"**.



1. You’ll then be presented with a screen divided into **Build**, **Deploy**, **Tests**, **Notifications**, **Schedule**, and **More Settings** sections.

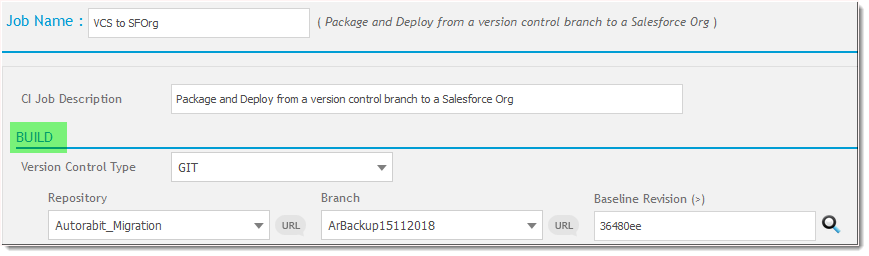


**Build**

1. Select the **Version Control Systems**. Currently, AutoRABIT supports Version Control Systems like GIT, SVN and TFS.

|  |
| --- |
| **Note:**  For CI Jobs with the Version Control as a source, excluding Managed Packages will not be supported in the AutoRABIT since there will no provision to identify the same in the Version Control System. Even the **"Exclude Managed Packages"** is set in **"My Account"** section, AutoRABIT CI Job build will not consider it. |

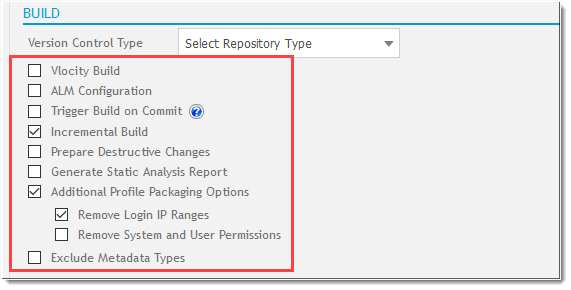
1. Select the **Repository** and the **Branch**.
2. Enter the **Baseline Revision number** manually or click on the search icon () to select the Baseline Revision. A new pop-up window appears; from the list displayed choose the required baseline revision number.



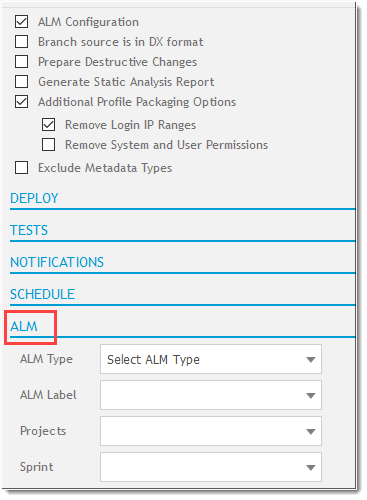
**Note:**

* 1. For the ***first CI Job*** to run, you need to choose the Baseline Revision of your selected Version Control Repository.
  2. When you wish to perform Git Merge-operation from one branch to another, you are requested to select the correct development start revision (from source branch) since it will get all revisions from source branch to destination branch based on the commit date and generate new merged revision.

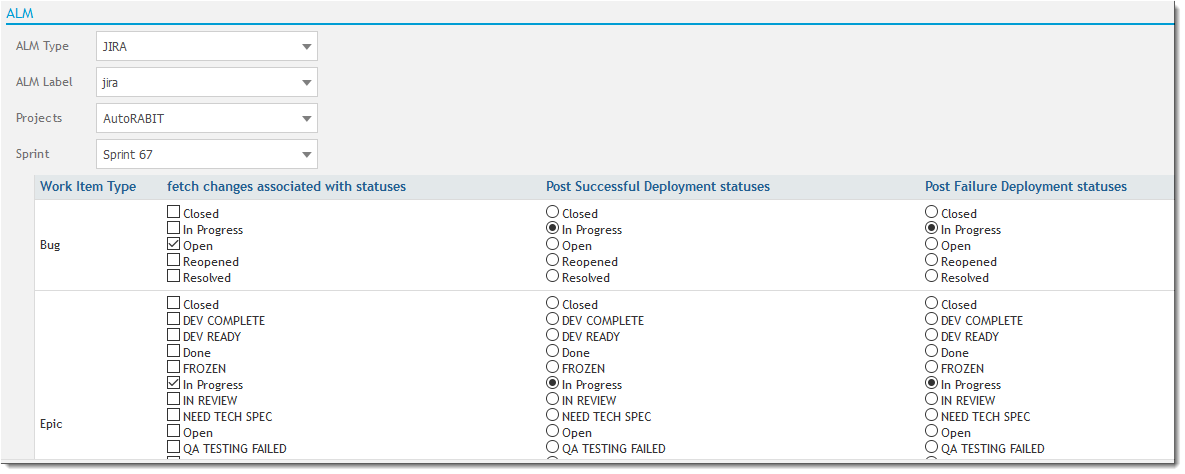
1. There are additional options to choose from:



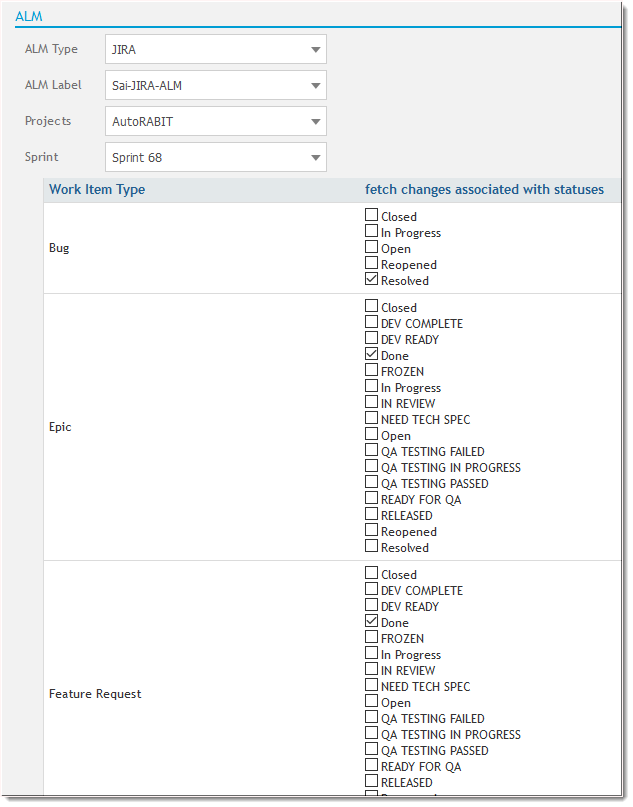
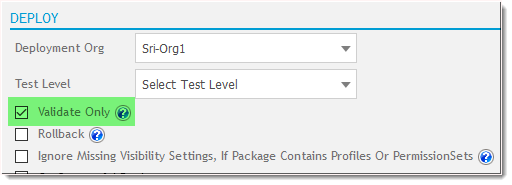
* 1. **Vlocity Build:** This option allows you to deploy the velocity components from metadata folder path to the Sandbox. ([Learn More](#78ef6e93-e954-40ac-af07-f37c4ad0950e))
  2. **ALM Configuration:** Configure work item type status in ALM type to include in the build.



* + 1. Select the **ALM Configuration** checkbox.
    2. Next, under **ALM** section, select the **ALM type**. Currently AutoRABIT supports ALM types such as Jira, IBMRTC, Version One, CA Agile Central and VSTFS.
    3. Next select the **ALM Label**, its related **Projects** and the **Sprint** in their respective field.
    4. A screen is displayed with the mapped project, all the available work item types and its status are displayed as per the workflow defined in the ALM system.



However, for "**Validate Only"** configured CI job, the ALM related screen will not show the post deployment status.



* 1. **Trigger Build on commit**: A new build is triggered, when changes are committed to the mapped version control system.

With **4.7 release** onwards, the users will have another option to choose i.e., **"Process commit revision via hook only"** for Version Control as **GIT** ***(Enterprise BITBUCKET, BITBUCKET, VSGIT, GITLAB, GITHUB)*** type. Upon selection, the build agent will read the commit revision number and generate a package from that revision number to the branch head.

**Let’s take the below scenario:**

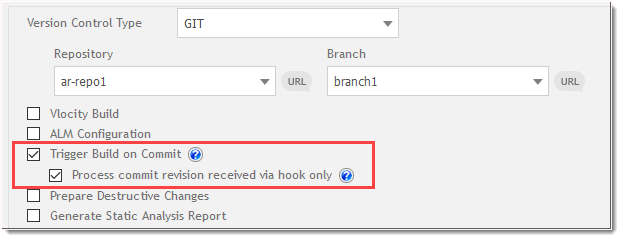
Two developers (**Developer A** and **Developer B**) both working on the same code base.

* + 1. **Developer A** has committed some changes into its local repository at **10 AM** this morning but did not pushed the changes to the remote repository.
    2. **Developer** **B** has also performed some changes in the code, but he pushed the changes into the remote repository by **10:05 AM** this morning.

So, when AutoRABIT builds get triggered by webhook, **Developer B** changes will be packaged and deployed and the codes gets updated with the latest revision. However, **Developer A** changes are ahead of **Developer B**, AutoRABIT will show no modifications for the build since the **Developer B** changes are in the HEAD position.

To overcome this scenario, AutoRABIT has come with an option to **"Process commit revision via hook only"**. This will prepare the build from the revision of **Developer A** to the HEAD revision of the branch, therefore no commits are skipped through AutoRABIT cycle.

|  |
| --- |
| **Additional Info:**  **"Baseline revision"** and **"Incremental Build"** option will not be available for the users if the above option is selected. |



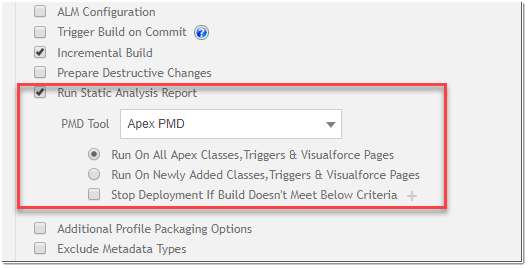
|  |
| --- |
| **Note:**  For **Enterprise BITBUCKET**, follow the below steps to configure ***"Process commit revision received via hook only"***:   * 1. Login to your **Bitbucket** account and select your **Repository**.  * 1. Click on **Repository Settings** and select **Hooks**.  * 1. Click on **Add hook** and keep the **Post-Receive WebHooks** in **Enabled** mode.  * 1. Check for **Web Post Hooks for Bitbucket Server** and click **Install**.  * 1. This will install the **Web Post Hooks for Bitbucket Server** *version 4.0.0*. |

* 1. **Incremental Build**: Incremental builds are important for managing continuous builds for continuous delivery. Incremental Builds substantially decrease build times by avoiding the execution of previous metadata that are not needed. This will fetch all the metadata changes beyond the selected Baseline Revision till the successful deployed revision to the destination org. On the next CI Job run, the previous Baseline Revision automatically gets changed to the successful deployed revision. Hence, there will be a substantial increase in build time performance for large-project incremental builds when a change to a single file or small number of files is performed.

|  |
| --- |
| **Note:** With **4.2.1** release onward, **Incremental Build** also works to validate Deployment Jobs. |

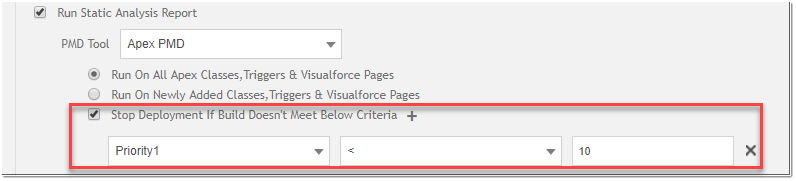
* 1. **Prepare Destructive Changes**: This option will allow the users to delete unwanted fields or metadata components from their destination Salesforce org before the deployments begins.
  2. **Run Static Analysis Report**: Select this option if you would like to run static code analysis tool to identify potential software quality issues before the code moves to production.

Select the desired **PMD tool**. Next choose the **condition** for run the tool i.e., running the static code analysis tool for all the apex classes, Triggers or Visualforce Pages or stating the time period from where it will run.



Additionally, you set the build criteria which will allow you to stop the deployment process if the criteria is not met. Select the **"Stop Deployment If Build Doesn't Meet Below Criteria"**, click on **"+"** icon and set the priority for the build to pass.

**Note:** For globally set Static Code Analysis criteria, the priority list is auto generated. However, you do have a permission to assign another criterion in addition to the globally set criteria.



* 1. Additional Profile Packaging Options:
     1. **Remove login IP Ranges**: If you want to login with a Salesforce org, you have an option to restrict IP ranges. Upon selection, login IP details will be restricted for the destination Salesforce Org or Version Control System.
     2. **Remove System and User Permissions**: You can remove or restrict the user permissions of the metadata types in the destination org or Version Control System.
  2. **Exclude Metadata Types**: Exclude the metadata types that are not required for an AutoRABIT build/deployment.

