# Lab 2 (20 min)

OVERVIEW: In this lab you will use JFrog Curation, configure policies with it and test it with a npm install command.

EXPECTED OUTCOME: Upon successful completion of this lab you will gain knowledge of how to curate open-source dependencies with the JFrog Platform.
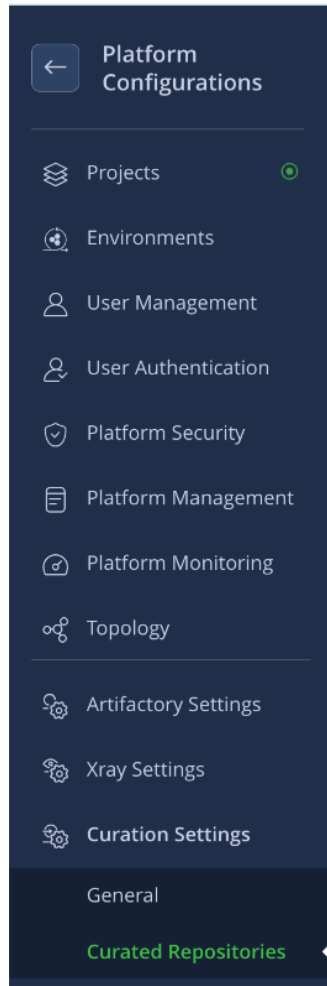
Step by step instructions

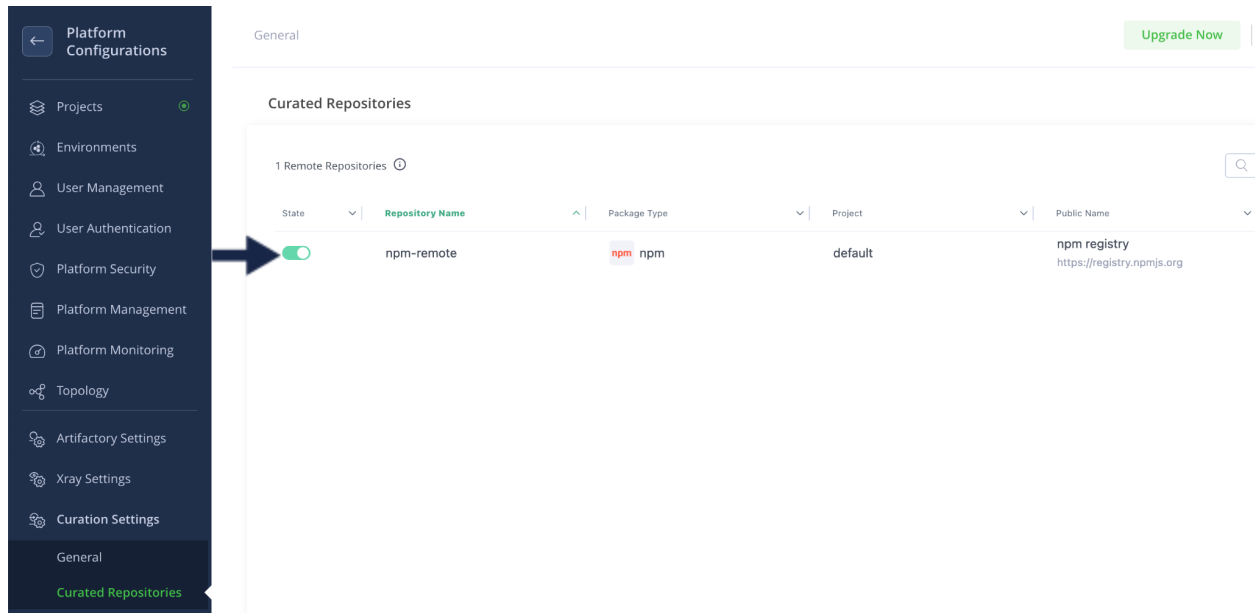*Phase #1 - Selecting a Repository to curate*

1. Click on 'Platform Configuration' -> 'Curation Settings' at the top right
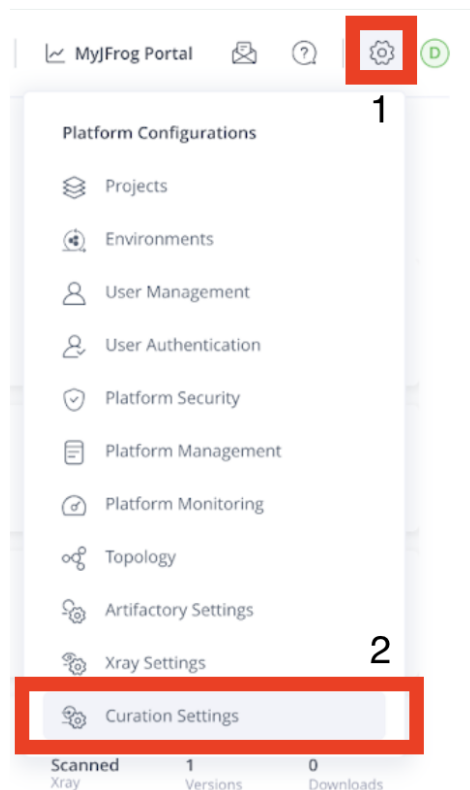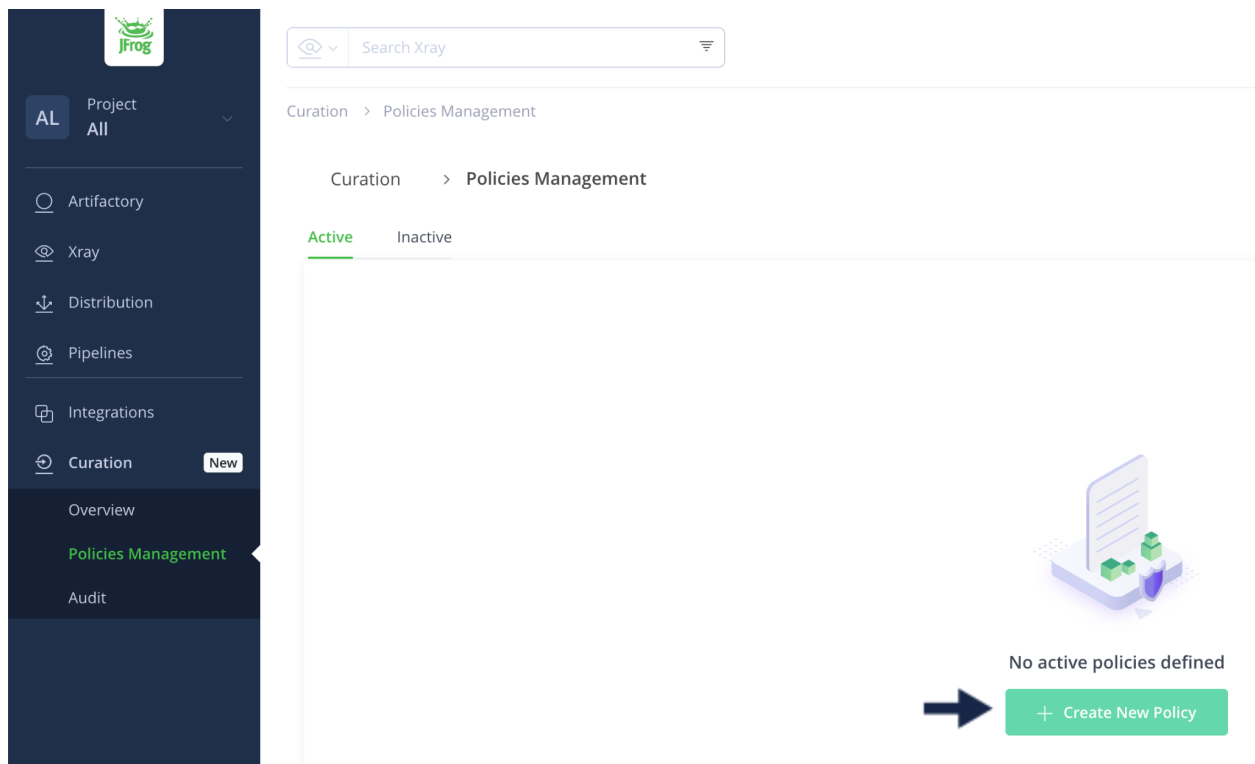

2. From the left bar, select 'Curated Repositories'

3. Turn On curation for the 'npm-remote' repository. This will enable you to enforce curation policies on this repository.

Phase #2 - Creating a Curation policy

4. Click on the back arrow at the Platform Configurations banner and from the left bar select 'Curation' -> 'Policies Management'

5. Click on 'Create New Policy'

6. Choose a name for the policy and click 'Next'

7. Choose the specific 'npm-remote' repository. This means the policy will be enforced only on the 'npm-remote' repository. Click 'Next'.

8. Now it's time to choose a condition for the policy. Take a look at the different options, some are security related, some are license related and some are operational.

Curation enables you to pick the right OSS dependency based on different types of criterias. For this lab, let's choose 'Malicious Package' as the condition. Click 'Next'.

9. Here we can add a waiver that will exclude specific packages from the policy being created. A waiver can be added also after policy creation. Let's skip and click on 'Next'.
10. Currently, Curation has two different actions: 'Block', which will block the download request and return a proper error message, or 'Dry Run', which will only simulate the curation flow. Let's choose 'Block' and click on 'Save Policy' on the bottom right.

Congratulations, you've created your first curation policy!

Phase #3 - Testing your Curation policy

11. Open your terminal and make sure you are in a session in your ec2 instance.
12. 'npm config' has been preset to point to npm-remote repository.
13. Run an 'npm install' command. We will try to download the malicious package 'cors.js'.
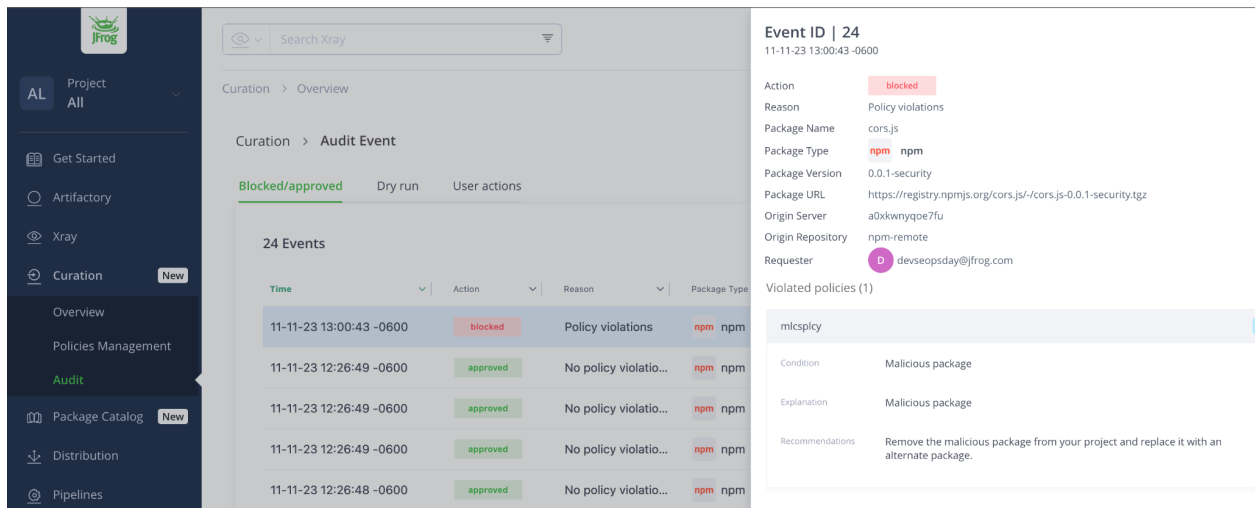
```
jf npm install cors.js
```

14. You should be able to receive the following message, if you did - congratulations, Curation
    was configured successfully. Otherwise, go back to Phase #2 or ask for help.

```
asimp@asimp-mac devsecopslondon %
asimp@asimp-mac devsecopslondon % jf npm install cors.js
11:00:43 [ Error] error while running '/usr/local/bin/npm install cors.js': exit status 1
npm WARN config `_auth`, `_authToken`, `username` and `_password` must be scoped to a registry. see `npm help npmrc` for more information.
npm ERR! code E403
npm ERR! 403 403 Forbidden - GET https://dsod23lom07.jfrog.io/artifactory/api/npm/npm-remote/cors.js/-/cors.js-0.0.1-security.tgz
npm ERR! 403 In most cases, you or one of your dependencies are requesting
npm ERR! 403 a package version that is forbidden by your security policy, or
npm ERR! 403 on a server you do not have access to.
{
  "error": {
    "code": "E403",
    "summary": "403 Forbidden - GET https://dsod23lom07.jfrog.io/artifactory/api/npm/npm-remote/cors.js/-/cors.js-0.0.1-security.tgz",
    "detail": "In most cases, you or one of your dependencies are requesting\na package version that is forbidden by your security policy, or\non a server you do not have access to."
  }
}

npm ERR! A complete log of this run can be found in:
npm ERR!     /Users/asimp/.npm/_logs/2023-11-11T19_00_41_462Z-debug-0.log
```

Phase #4 - Inspecting the Curation event

15. Go back to the JPD UI, and click on 'Curation' -> 'Audit'. Here you should be able to see the
    last event, which is a rejection of your npm install request.

16. In addition, if you click on 'User Actions', you should be able to see the audit trail of the policy
    you created in Phase #2.
    Inspect the information provided on this action.

Congratulations! You have completed Lab 2

Phase #5 - BONUS - Inspecting the package in JFrog Catalog

17. Go back to the UI, and click on 'Package Catalog'. Search for the 'cors.js' package.

18. What kind of information does the catalog provide on cors.js? What is the core issue and what is the remediation?