1. List down and explain in a paragraph the various tools which are helpful to set up a CI/CD environment and provide your preference of toolset for a CI/CD environment that you will besetting up , along with the Pros and Cons. Provide a high-level diagram showing the various components involved.

Continuous integration and delivery is the best practice for DevOps and agile development. Here are the top CI/CD tools with their key features so we can pick the right one & integrate the power of automation in your CI/CD pipeline.(My)

Jenkins:

Being platform-agnostic, Jenkins is compatible with Windows, Linux operating system, and Mac OS and runs on ports 8080 and 8443.

The easy installation process as a self-contained Java program; and configuration of Jenkins can be carried out using a web interface that contains on-the-fly error checks and built-in help.

Plugin architecture provides nearly endless possibilities for what Jenkins can do, making it quite extensible.

Numerous plugins for integration with different test automation tools for the test pipeline

Jenkins can also be integrated with major cloud platforms like Google Cloud, Amazon EC2, Atlassian Cloud, VMWare vSphere, etc., using plugins.

Easy work distribution across multiple machines ensures faster builds, tests, and deployments across multiple platforms.

TeamCity

Easy installation and configuration of the TeamCity server with extensive documentation

Setting up the first pipeline takes less than 2 minutes and can be created through a simple UI without writing any code.

Integrates with build & test tools, keeps a history of tests, flags unstable tests as flaky, and can also help identify the exact commits that cause failures in the pipelines.

Real-time reporting helps dig into an issue, resolve and assign it to the project members.

By integrating TeamCity with IDE, you can build, check, and run automated tests without committing any code or creating any branches – and keep your code base clean at all times.

Superior Docker support. The tool provides integration with cloud providers like AWS, GCP, Azure, and VMware.

GitLab:

Individual members of the organization can have more granular access controls to the repositories.

Auto-scaling GitLab CI runners can efficiently manage and save 90% on EC2 costs, which is essential for a parallel testing environment.

Easy management of git repositories with great access permissions

The active and progressive community provides out-of-the-box support, which does not require modification in additional plugin installation.

Improved collaboration with merge management systems and easy merge requests. There is support for almost every version control system and build environment.

Buddy:

Super fast deployments based on changesets

Build pipelines in visual UI with more than 120 ready-to-use actions

Pipeline cache for artifacts, static files, Git repositories, and Docker layers

Export pipeline configuration to YAML or script the entire process top to bottom

Run pipelines on push, manually, or with Slack commands & webhooks

Environment templates to create, customize, and reuse testing environments

GoCD:

Manual triggers to deploy any verified application version to the intended environment.

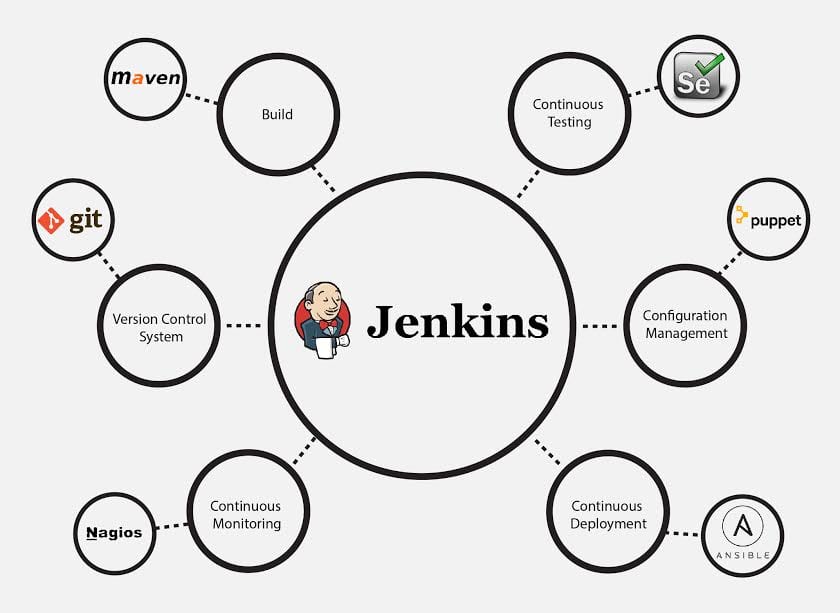
Support the execution of tests written in most languages or frameworks. The tool’s agent grid provides parallel and cross-platform execution.

Comparing the content(files and commit messages) across any two arbitrary builds is invaluable while troubleshooting a broken pipeline.

Elimination of bottlenecks with trivial parallel execution across platforms, versions, pipelines, and branches

Easily reusable pipeline configurations via a template system

Note: Jenkins is my preference of toolset for a CI/CD environment



1. How do you manage security credentials in a CI/CD environment, such that only the relevant

people who need access to the credentials will be able to access it? For Example, you could

take Jenkins as an example environment (or other environments based on your experience).

Provide the approach you will take.

To maximize security, credentials configured in Jenkins are stored in an encrypted form on the controller Jenkins instance (encrypted by the Jenkins instance ID) and are only handled in Pipeline projects via their credential IDs.

Jenkins can store the following types of credentials:

1. Secret text - a token such as an API token (e.g. a GitHub personal access token),
2. Username and password - which could be handled as separate components or as a colon separated string in the format username:password (read more about this in Handling credentials),
3. Secret file - which is essentially secret content in a file,
4. SSH Username with private key - an SSH public/private key pair,
5. Certificate - a PKCS#12 certificate file and optional password, or
6. Docker Host Certificate Authentication credentials.

Configuring credentials

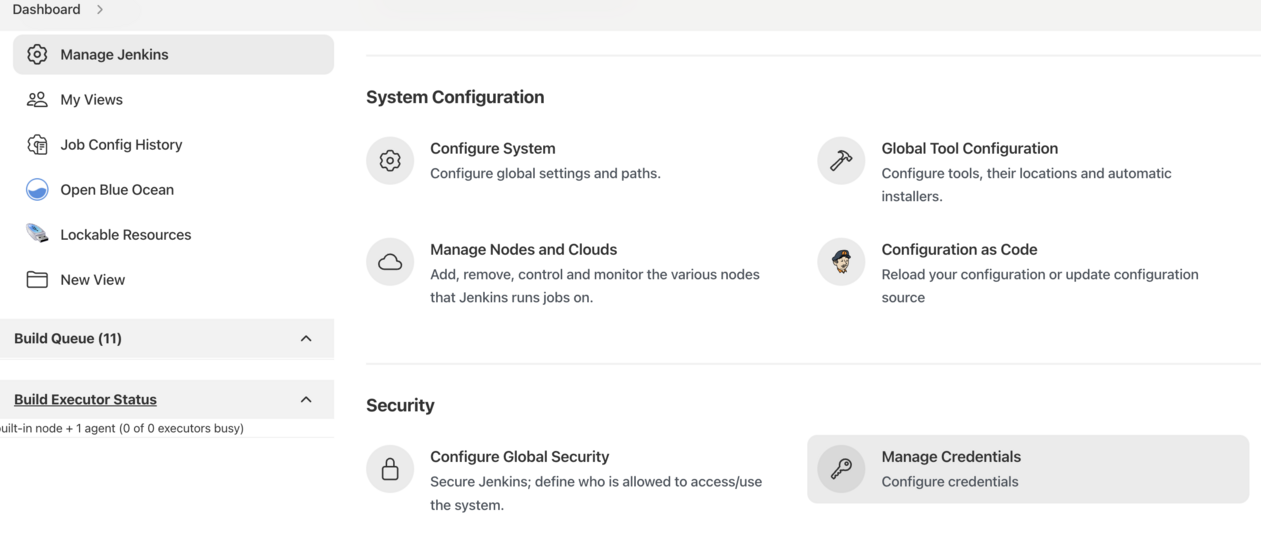
This section describes procedures for configuring credentials in Jenkins.

Credentials can be added to Jenkins by any Jenkins user who has the **Credentials > Create** permission (set through **Matrix-based security**). These permissions can be configured by a Jenkins user with the **Administer** permission.

### Adding new global credentials

To add new global credentials to your Jenkins instance:

1. If required, ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).
2. From the Jenkins home page (i.e. the Dashboard of the Jenkins classic UI), click **Manage Jenkins > Manage Credentials**.



# Let’s assume we have this repository/module by the name of the **integration-service** in a

version control system of choice. Assignment is to design a CI/CD Pipeline for the above

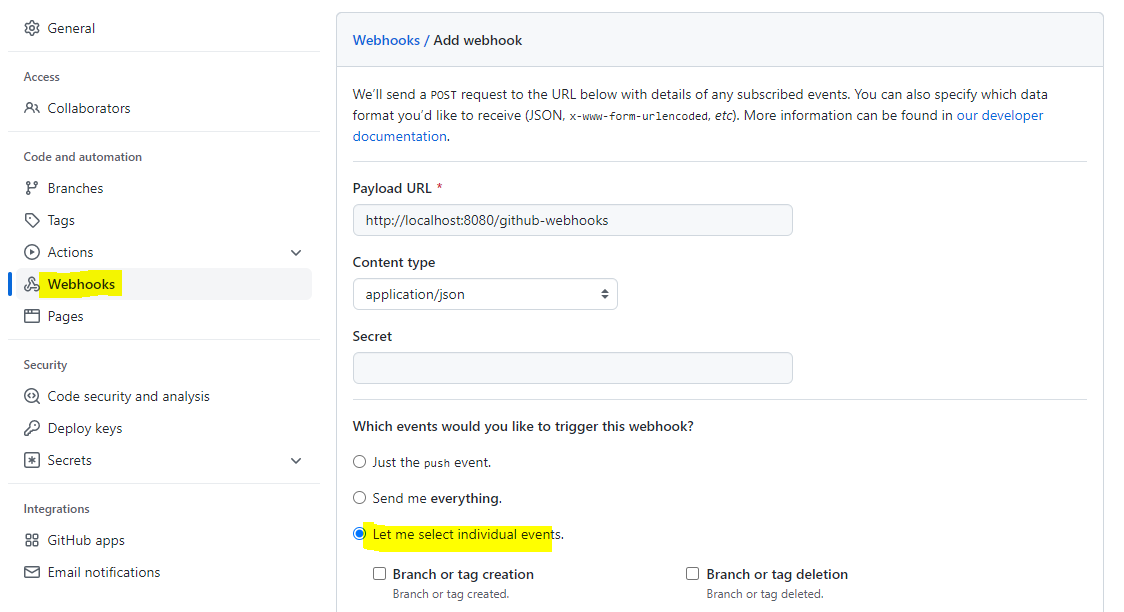
module. Below are the requirements mentioned for the CI/CD pipeline (Jenkins is the tools

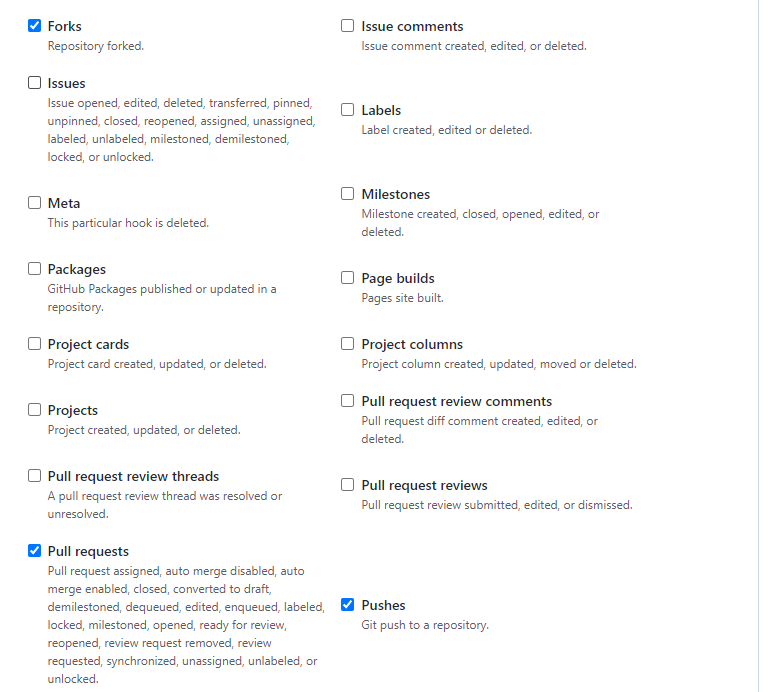
of preference, feel free to chose otherwise)

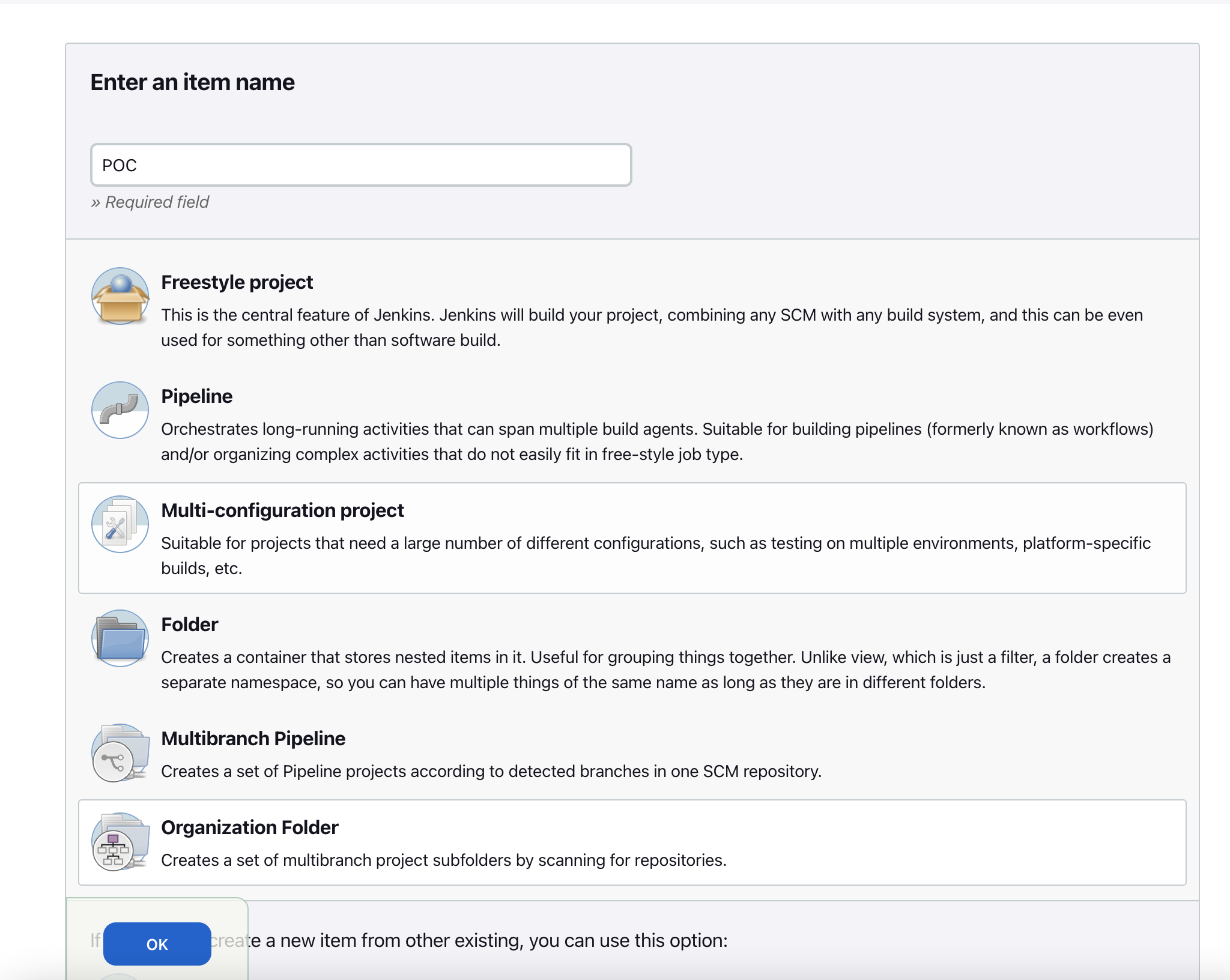
To trigger the pipelines automatically based on the push, pull and merge requests configured the webhook **in** the repository settings **as** below snapshot.

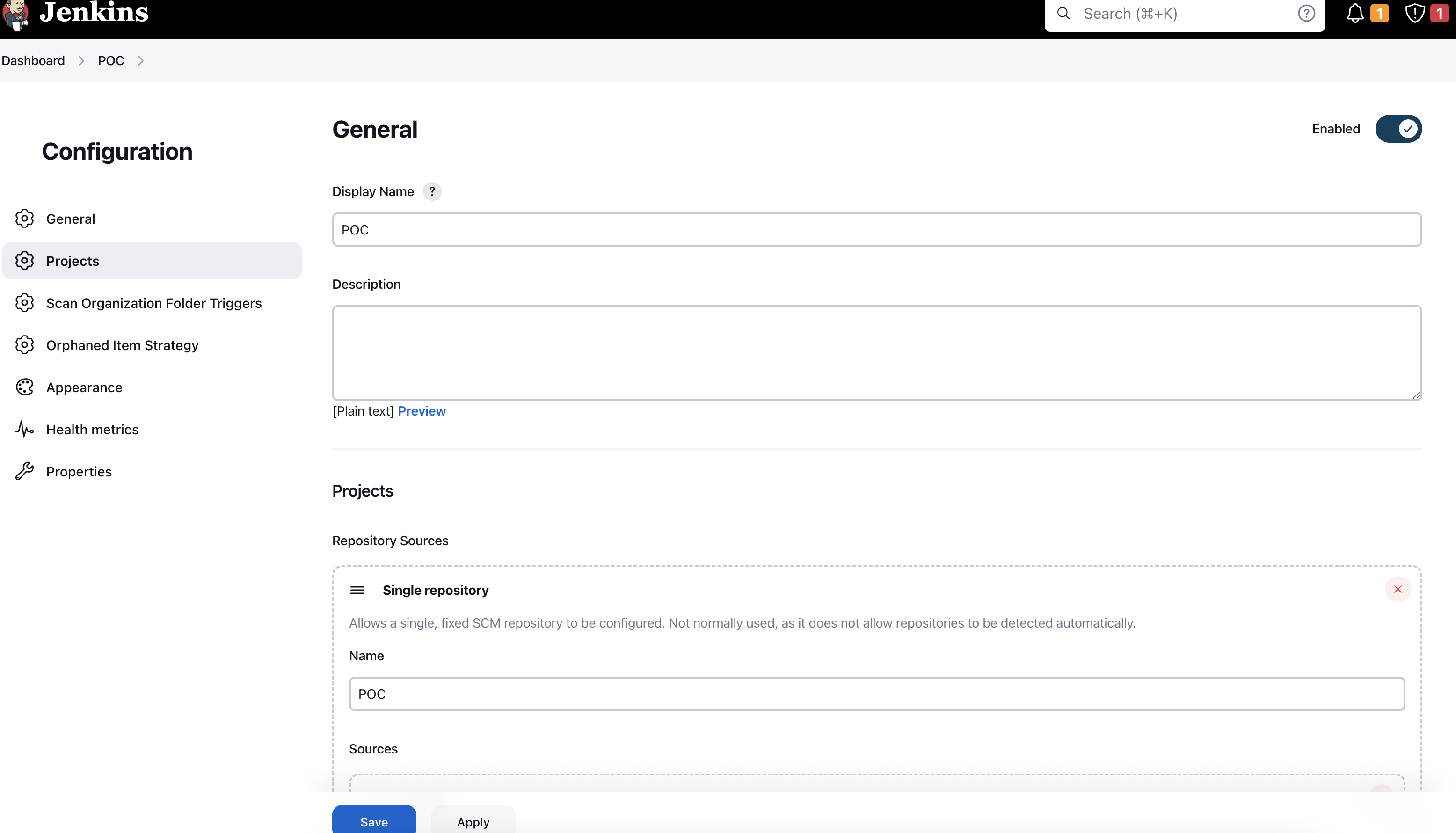
* We can create webhook configuration in GitHub settings and create jobs to git scm poll to trigger pull request/ push changes.
* We have other way also, using organization folder, where we can map all repo’s under project. While creating we can provide below configuration to mention which branches it should invoke and another configuration.

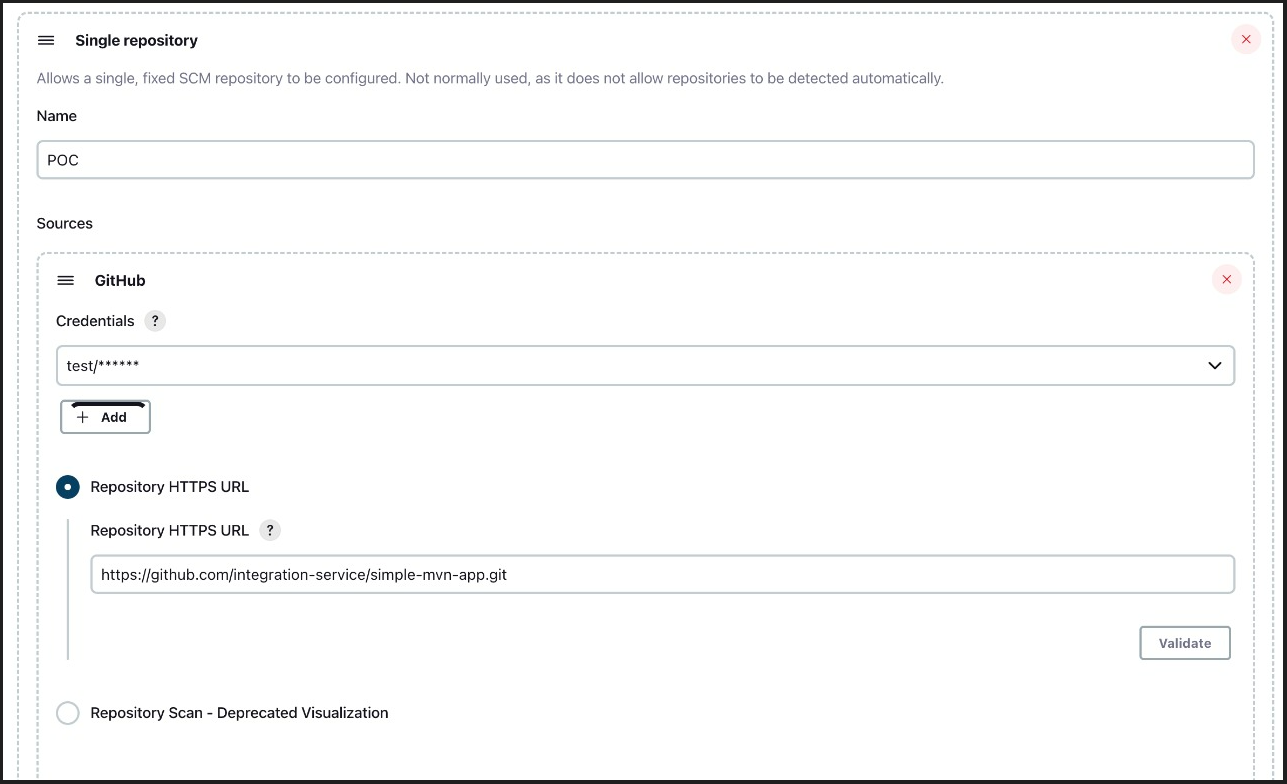
Create Organization folder

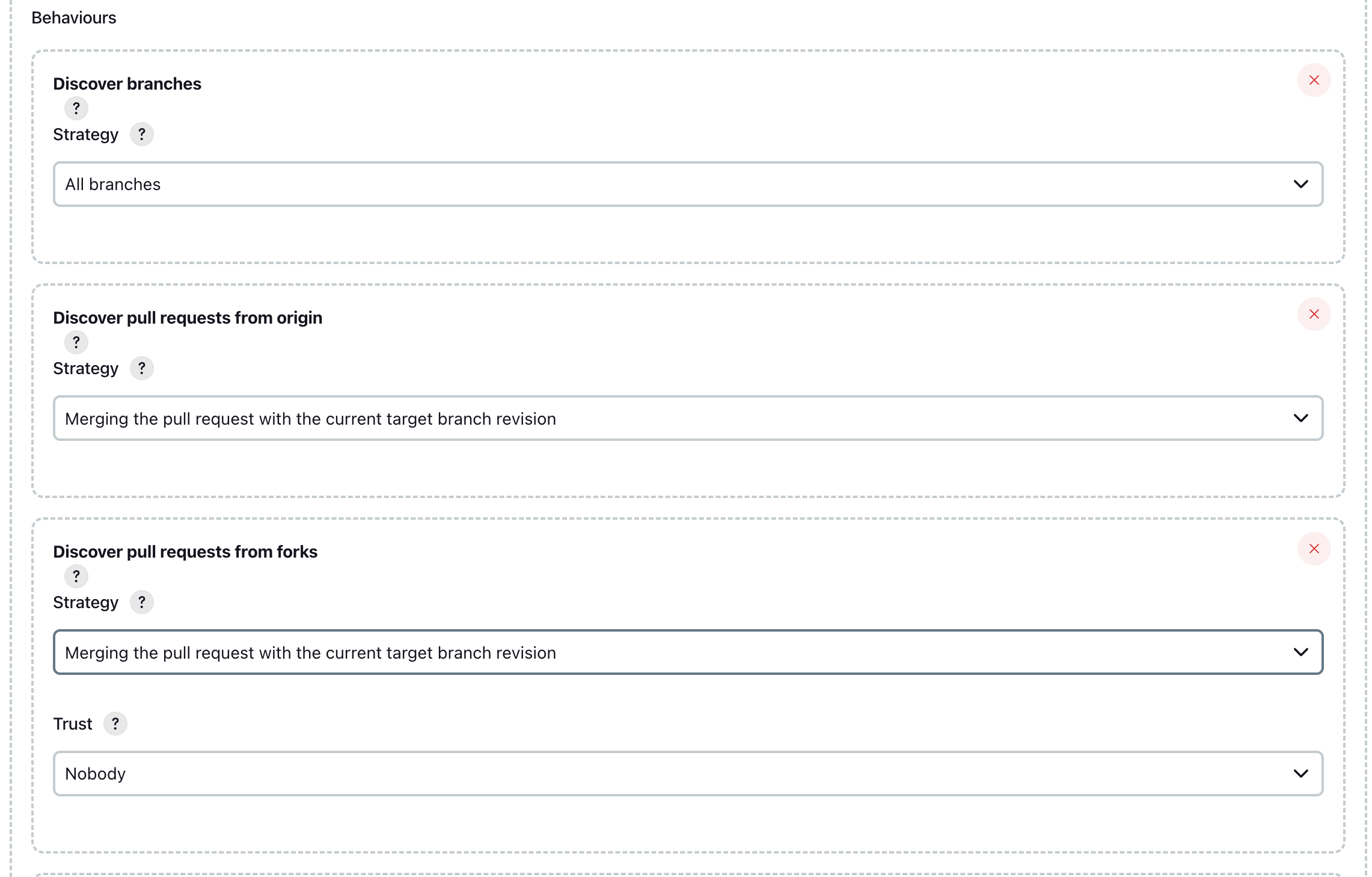


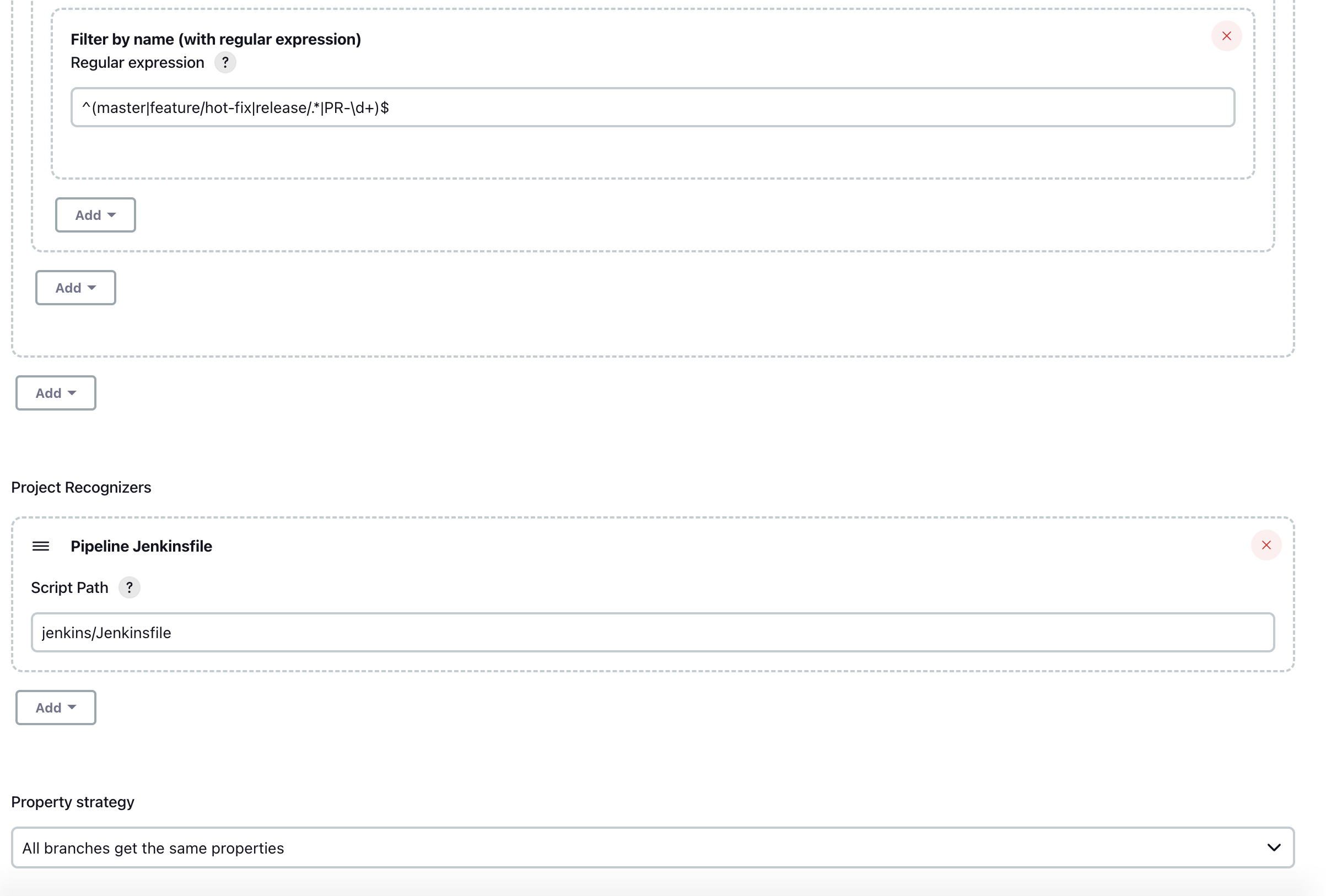


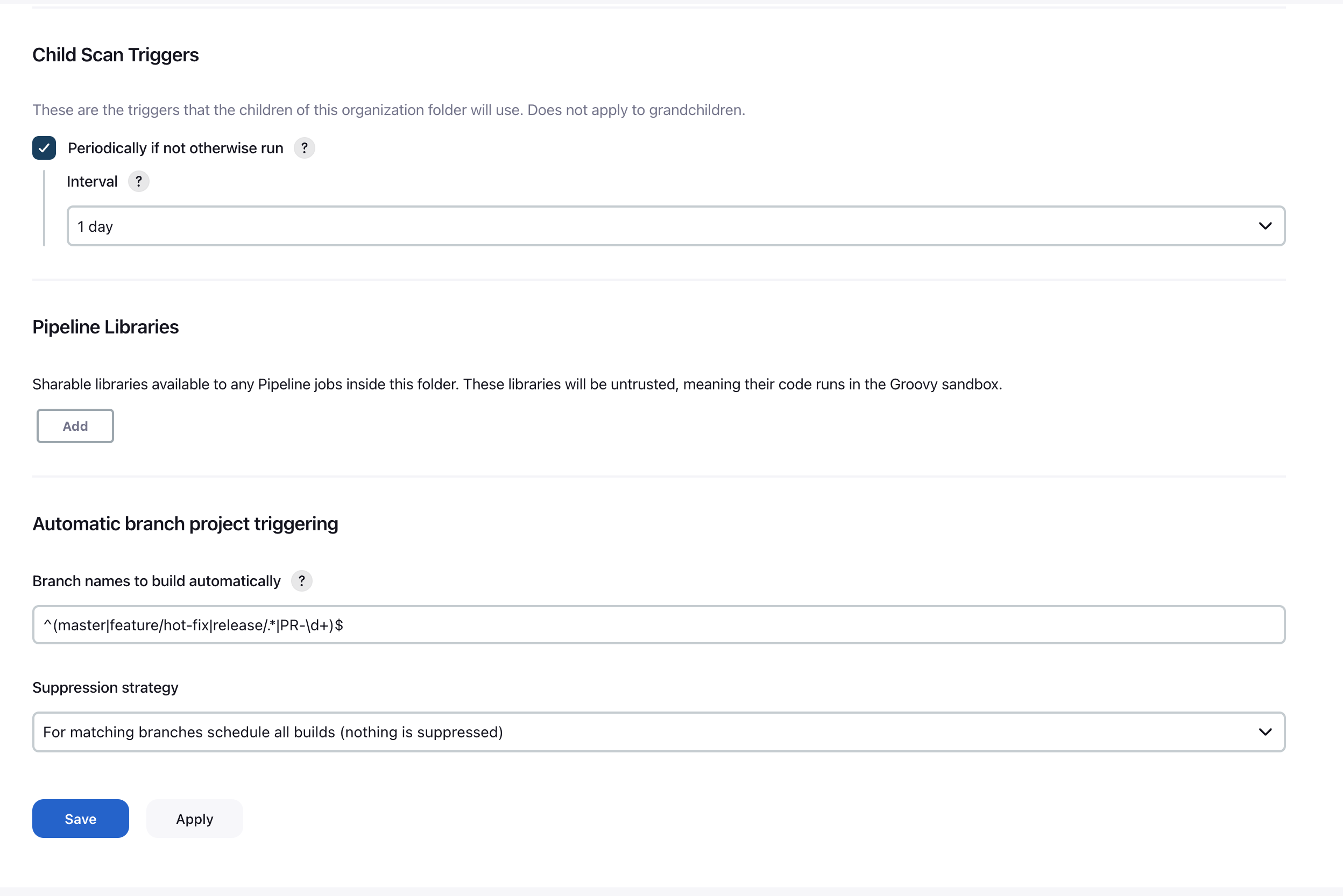












Note: For Jenkins pipeline code please have a look on pipeline code folder