**Rules Start Only on Merge:**

**A screenshot of a computer

Description automatically generated**

* **if: '$CI\_PIPELINE\_SOURCE != "merge\_request\_event"'**: This condition checks if the pipeline source is not a merge request event. If it's not a merge request, the rule evaluates to true.
* **when: never**: This specifies that the job should never be scheduled automatically. The **never** keyword means that the job won't be run as part of the regular pipeline execution flow; instead, it needs to be manually triggered or triggered through specific conditions.

**Test Vtpexe Helper Job:**

**A screenshot of a computer program

Description automatically generated**

* **stage: test-helper-apps**: This job runs in the **test-helper-apps** stage.
* **script**: Executes the script, in this case, echoing "testing vtpexe."
* **before\_script and after\_script**: These are executed before and after the main script, respectively.
* **rules**: Specifies the rules for job execution. It references the **rules-start-only-on-merge** rule set, ensuring that the job is only scheduled if the pipeline source is not a merge request event. Additionally, it includes a condition based on changes in the **scripts/vtpexe-helper/\*\*/\*** directory.

**Build Vtpexe Helper Job:**

**A screen shot of a computer program

Description automatically generated**

* **stage: build-helper-apps**: This job runs in the **build-helper-apps** stage.
* **environment**: Specifies the deployment environment.
* **script**: Executes the script, in this case, echoing "Building vtpexe."
* **before\_script and after\_script**: These are executed before and after the main script, respectively.
* **rules**: Specifies the rules for job execution. It references the **rules-start-only-on-merge** rule set, ensuring that the job is only scheduled if the pipeline source is not a merge request event. Additionally, it includes a condition based on changes in the **scripts/vtpexe-helper/\*\*/\*** directory.

**Deploy Vtpexe Helper Job:**

**A screenshot of a computer program

Description automatically generated**

* **stage: deploy-helper-apps**: This job runs in the **deploy-helper-apps** stage.
* **script**: Executes the script, in this case, echoing "deploy-vtpexe-helper."
* **before\_script and after\_script**: These are executed before and after the main script, respectively.
* **rules**: Specifies the rules for job execution. It references the **rules-start-only-on-merge** rule set, ensuring that the job is only scheduled if the pipeline source is not a merge request event. Additionally, it includes a condition based on changes in the **scripts/vtpexe-helper/\*\*/\*** directory.

**Explanation:**

* The **rules-start-only-on-merge** rule set is used to control when these jobs are scheduled, ensuring that they run only when the pipeline is triggered explicitly through an API call (not a merge request event).
* Each job specifies its stage, script to be executed, and any additional configurations such as environment variables, before and after scripts, and rules based on changes in specific directories

**rules-starting-if-not-push-or-merge**A black screen with green text

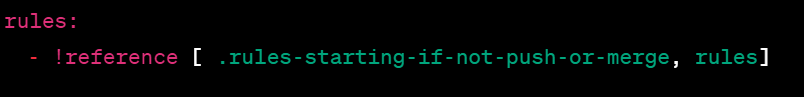
Description automatically generated

* **if: '$CI\_PIPELINE\_SOURCE == "api"'**: This condition checks if the pipeline source is an API trigger. The **$CI\_PIPELINE\_SOURCE** variable in GitLab CI contains information about how the pipeline was triggered. If it equals "api," it means the pipeline was triggered programmatically through the API.
* **when: never**: This specifies that the job should never be scheduled automatically. The **never** keyword means that the job won't be run as part of the regular pipeline execution flow; instead, it needs to be manually triggered or triggered through specific conditions.

**Explanation:**

* This rule set is designed to ensure that the jobs using this rule set will only run when the pipeline is triggered explicitly through an API call. It prevents these jobs from being automatically scheduled for regular events like pushes or merge requests.

**Usage:**

* This rule set is then referenced in other jobs in the main pipeline, such as in the **prepare-integration-environment-job** and **report-integration-result-job** jobs:
* By referencing this rule set in other jobs, it enforces the condition that those jobs should only run when the pipeline is triggered explicitly through an API call, providing more control over when certain jobs are executed.
* **Before Script:**A screen shot of a computer code

  Description automatically generatedActivates the virtual environment for the CI job.
* Initializes Poetry, a dependency manager for Python projects.
* Activates the Poetry virtual environment.
* Installs RClone, a command-line program to manage files on cloud storage.
* Sends a state message to the backend indicating that the CI job has started.

**Prepare Integration Environment Job:**

* A screen shot of a computer

  Description automatically generated
* Executes in the **initialize-integration** stage and inherits rules from the **rules-starting-if-not-push-or-merge** rule set.
* Specifies the deployment environment for the job.
* Executes a script to prepare the integration trigger.
* Lists files in the working directory.
* Displays the content of the **integration\_vars.env** file.
* Defines artifacts to be saved, including environment variable files.

**Trigger Integration Pipelines Job:**

* A screenshot of a computer program

  Description automatically generated
* Executes in the **trigger-integration-pipelines** stage for the specified deployment environment.
* Sets a variable **TRIGGER\_BRANCH** with the branch to be triggered.
* Displays messages and content of the **integration\_vars.env** file.
* Uses the **trigger** command to initiate the child pipeline.
* Specifies dependencies on the **prepare-integration-environment-job**.
* Inherits rules from the **rules-starting-if-not-push-or-merge** rule set.

**Report Integration Result Job:**

* A screenshot of a computer program

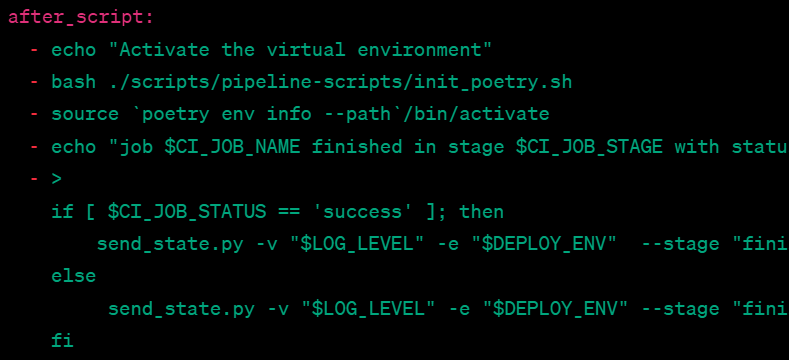
  Description automatically generated
* Executes in the **report-integration-result** stage for the specified deployment environment.
* Performs checks and reports on the status of simulations.
* Gets integration states from the backend.
* Always runs, regardless of the job status.
* Specifies dependencies on the **prepare-integration-environment-job** and **trigger-integration-pipelines-job**.
* Specifies an after script that activates the virtual environment, reports job status, and sends a state message to the backend.

**Print Integration Pipeline Parameter Job:**

* A screenshot of a computer

  Description automatically generated
* Executes in the **testing** stage for non-push or non-merge request events.
* Displays various information, including the test run ID, pipeline creation timestamp, environment variables, contents of **/etc/resolv.conf**, and the branch name.

**After Script:**



**Explanation:**

* **echo "Activate the virtual environment"**: Outputs a message indicating that the virtual environment is being activated.
* **bash ./scripts/pipeline-scripts/init\_poetry.sh**: Executes a Bash script (**init\_poetry.sh**) related to initializing Poetry, a Python dependency manager.
* **source poetry env info --path/bin/activate**: Sources (activates) the virtual environment using the information obtained from Poetry.
* **echo "job $CI\_JOB\_NAME finished in stage $CI\_JOB\_STAGE with status $CI\_JOB\_STATUS"**: Outputs a message indicating the completion of the job, including the job name, stage, and status.
* **if [ $CI\_JOB\_STATUS == 'success' ]; then ... else ... fi**: A conditional statement checking the value of **$CI\_JOB\_STATUS**. If the job was successful, it executes the commands within the **if** block; otherwise, it executes the commands within the **else** block.
  + In the **if** block:
    - **send\_state.py -v "$LOG\_LEVEL" -e "$DEPLOY\_ENV" --stage "finished-integration" --status "successful" --message "$CI\_JOB\_NAME Job successful executed"**: Sends a state message, indicating that the job has been successfully executed, to a destination (backend) with various parameters such as log level, deployment environment, stage, status, and a custom message.
  + In the **else** block:
    - **send\_state.py -v "$LOG\_LEVEL" -e "$DEPLOY\_ENV" --stage "finished-integration" --status "failed" --is\_failed True --message "$CI\_JOB\_NAME Job failed TODO Find Reason"**: Sends a state message, indicating that the job has failed, to the same destination. It includes information about the log level, deployment environment, stage, status (failed), a flag indicating failure (**--is\_failed True**), and a custom message.
* **rules: - !reference [ .rules-starting-if-not-push-or-merge, rules]**: Specifies the rules for job execution, referring to the **rules-starting-if-not-push-or-merge** rule set. This ensures that the job is scheduled only when the pipeline is explicitly triggered through an API call and not a push or merge request event.

**Purpose:**

* The **after\_script** section here is responsible for performing tasks after the main script has executed. In this case, it activates a virtual environment, outputs job completion information, and sends a state message to a backend based on the success or failure of the job. This allows for post-job actions, reporting, and communication with external systems.