**An Empirical Investigation Into a Large-Scale Devops Pipeline programs**

**ABSTRACT:**

Investigating a large number of devops pipeline programs and obtain statistical data that describe the content and patterns in devops pipelines. Getting insights into jenkinsfiles and other pipeline artifacts from open-source repositories . We propose 15 research questions , the reason behind the formulation of those questions and obtain useful information from 350 open source projects which make use of jenkins pipeline for Continuous Delivery and Continuous integration process.

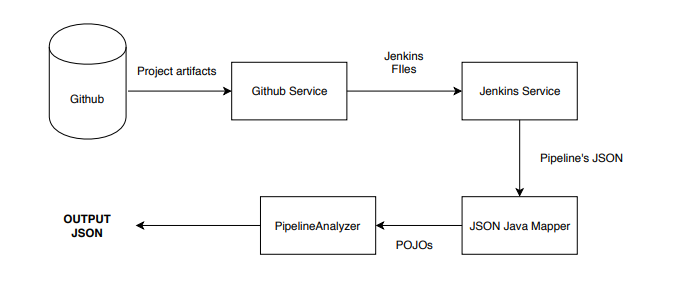
**OBJECTIVE:**

The goal of the project is to analyze pipelines in a number of steps. First, we obtain pipelines from various open-source repos from GitHub by searching and getting pipeline projects from Github. We obtain devops projects from the repository and create a program that reads in the devops configuration file and analyzes the structure of the pipeline. We then formulate the research questions to get useful insights from the available jenkins pipeline . We determine what utilities and procedures are used in stages of the pipeline, how they interact with the project artifacts . We then create abstractions based on which we design programming components of this project and determine how to present the results of the analyses.

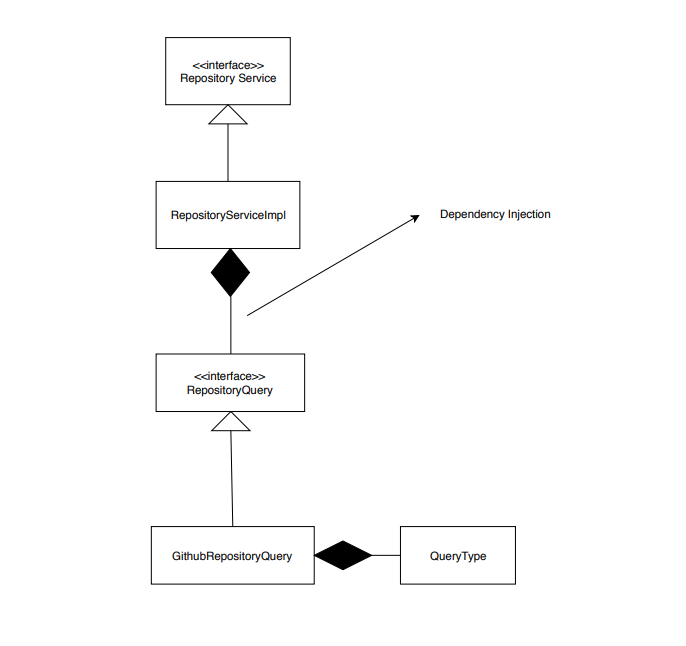
**ARCHITECTURE AND DESCRIPTION OF THE SYSTEM :**

The architecture of the system is show below in figure 1 and figure 2 .

We will now the discuss the architecture of the system describe below in detail .

****

**Figure 1:Architecture in pipe and filter representation**

****

**Figure 2 : Jenkins File Retrieval Component**

Our system consists of the following major components :

1. **Github Repository** : We fetch the open source pipeline projects from Github using Github Search API .
2. **Github Service** : The github service component is used to fetch pipeline programs and its various artifacts from the github repository . The following programming modules are use to perform different tasks related to github repository **RepositoryService , RepositoryQuery , Repository, RepositoryServiceImpl ,GithubRepositoryQuery** .
3. **Jenkins Service** : The Jenkins service component performs the task to convert the Jenkins file extracted from the open source pipeline programs to JSON for further analysis in the project . The following utility program **HttpUtils** is used to convert the Jenkins file to more parsable format i.e JSON representation . This was done because it makes is comparatively easier to use and interpret the pipeline using JSON representation .
4. **JSON Java Mapper** :

After doing comprehensive analysis of the the JSON representation of the different pipeline programs , we observed that we needed to create some abstractions for our pipeline programs . We created abstractions in form of class objects based on analysis and designed programming components of this project . ***The different classes were created for each distinct component of a pipeline like the Sections ( agent , post , stages , steps)***

***Directives (environment , options ,parameters , triggers ,stage ,tools***

***Input ,when ) .*** We made use of compositions also to formulate relationships between different components of the pipeline. This made it easier to map the JSON representation of our pipeline to different java POJO objects . To achieve the above task , we use the following mapper service **JsonJavaMapper** to create such abstractions in form of **pipeline** objects to analysis of jenkins programs .

1. **Pipeline Analyzer** : After creating the abstraction and representation of our pipeline programs in form of different objects , we then makes use of those objects to represent each of our Jenkins Program and perform analysis to get answers for our different research questions . The following implementation **PipelineAnalyzerImpl** provides different functions to provide answers for different research questions . The output of our pipeline analyzer is a set of json files which are then used to create bar charts and donuts charts to visualize the results of the analyzer .

**Formulations of your research questions.**

1. What are the most frequent post-condition blocks in the post section within jenkins pipelines?

The post section defines one or more additional steps that are run upon the completion of a Pipeline’s or stage’s run (depending on the location of the post section within the Pipeline). Post can support any of of the following post-condition blocks: **always, changed, fixed, regression, aborted, failure, success, unstable, and cleanup**. These condition blocks allow the execution of steps inside each condition depending on the completion status of the Pipeline or stage. *Gathering information about the different post condition blocks may help other developers to create such post-conditions in the future and facilitate infrastructure code reuse.*

1. What are the most frequent agent type in the stage section within jenkins pipelines?

The agent section specifies where the entire Pipeline, or a specific stage, will execute in the Jenkins environment depending on where the agent section is placed. This section must be defined at the top-level inside the pipeline block, but stage-level usage is optional. The different agent types are : **any , none , label , node , docker**. *Having information about the most commonly used agent types will tell us how agents are used by different developers and which agent types are preferred for development of jenkins programs . This will guide developers in creation of more efficient pipeline programs in the future .*

1. What are the most frequent steps type in the stage section within jenkins pipelines ?

Steps are the major execution part of the pipeline. Pipelines have multiple commands that execute in multiple stages and steps. *Having knowledge about what steps are used across different stages and their functionality will help us in having information about the most useful steps which can be used for future development of pipeline programs .*

1. What is the correlation between number of triggers and number of stages in a jenkins pipeline ?

Triggers in a pipeline are directives that retrigger the pipeline. Stages are the different phases (**build , test , deploy , etc** ) a pipeline executes from start to end. *Finding the relationship between the number of triggers and number of stages using a correlation coefficient tells us about how number of stages are dependent on the triggers used. The coefficient of correlation was between 0-1, implying that these two are not strongly correlated but the correlation still exists.*

1. What is the number of projects that have user-defined parameters?

Pipeline programs make use of different parameter which can be user defined or existing parameters . *Having information about how many projects make use of user defined parameters tells us about how percentage of user define new parameters in different projects and for what purposes are those parameters used .*

1. What are the most frequent type of the environment variables within jenkins pipelines ?

Pipeline programs make use of different environment variables which can be user defined or existing variables . *Having information about how projects make use of different environment variables tells us about how frequently are environment variables used and for what purposes are those variables used .*

1. Which are the most frequent User-Defined Parameters in a jenkins file ?

Pipeline programs make use of different parameter which can be user defined or existing parameters . *Having information about how projects make use of different User-Defined Parameters tells us about how frequently are such parameters used and for what purposes are those variables used .*

1. What is the correlation between the number of stages and number of steps in a pipeline ?

Steps form important part of different stages of a pipeline . Stages are the different phases (**build , test , deploy , etc** ) a pipeline executes from start to end. *Finding the relationship between the number of steps and number of stages using a correlation coefficient tells us about how number of stages are dependent on the steps used. The coefficient of correlation was between 0-1, implying that these two are not strongly correlated but the correlation still exists.*

1. Which tools in jenkins pipeline are most used and least used across different pipelines ?

A devops pipeline has usage of various tools. *This research enabled us to point out the most used and least used tools in the pipeline. This would help in understanding which tools are more or less popular amongst developers.*

1. What are most frequent When conditions types used in a jenkins pipeline?

The when directive allows the Pipeline to determine whether the stage should be executed depending on the given condition. The when directive must contain at least one condition. If the when directive contains more than one condition, all the child conditions must return true for the stage to execute. *The when block has multiple built-in conditions. This method is used to analyze the most frequently used conditions inside when block. Each condition serves a particular purpose. This analysis enables developers to select the right condition for their need.*

1. What are the most common used stages category in a pipeline ?

Stages are most frequently used components of a pipeline . *Having information about which stages is a pipeline categorized into which help us in organizing developers to design pipeline in a efficient way .*

1. What the commonly used types of agent arguments in a pipeline ?

Agents define where the entire Pipeline or a stage will execute.Agents can be defined at pipeline level or at stage level.This research question is self generated withe the motive that developer know at which agent is more suitable for which stage of the pipeline. This find the most frequently used agent inside the stage block.

.

1. What are the most commonly used commands in steps of different stages of a pipeline ?

Steps are the major execution part of the pipeline.Pipelines have multiple commands that execute in multiple stages and steps. *For a developer ,getting the most commonly used commands provide an insight of the most used commands by fellow developers. Some commands are specific to the stage in the pipeline. A more specific classification is done further that provides frequently used commands in each stage of a pipeline.*

1. Which commands are frequently used in build , test and deploy stages of a pipeline ?

A Devops pipeline has its stages mainly classified into build, test and deploy.There could be multiple miscellaneous stages added but the more significant ones are build, test and deploy. In each steps of these stages several commands executed. So,we have categorized the commands as per the stages they are part and the we find out the frequencies of these commands in build, test and deploy stages across several projects. *This helps developers understand which is the right commands to execute at which stage of the pipeline.*

1. Which Linux and Windows commands are frequently used in various steps of the pipeline stages ?

We have categorized the most commonly used Linux commands and windows commands executed in various steps of pipeline. They are depicted by ‘sh’ for linux and ‘bat’ for windows. Though they are system specific, it will help developers get most used commands for their respective systems. This categorization also avoid confusion of linux users using windows commands and vice versa, saving time and effort.

**IMPLEMENTATION**

**1.frequentPostConditions:** This method check the most used conditions in the post condition block from always, changed, fixed, regression, failure, success, unstable and cleanup.

**2.frequentAgentTypes:** The frequentAgentTypes method find the most frequently used agent parameters from any, none, label, node and docker in the pipeline block.

**3.frequentStepTypes:** The frequentStepTypes method finds the frequency of the *most used type of statements inside step block* like sh, bat,git,withMaven etc.

**4. analyzeUserDefinedParameters:** In this method the *most commonly used user defined parameters* are found.

**5.frequentEnvVarTypes:** The *most used environment variable* are found.

**6.** **frequentUserDefinedParameters:** This method finds the *most frequently used user defined parameters* across pipelines.

**7.mostUsedTools:** This method finds the most used tools from *jdk, maven and gradle.*

**8.leastUsedTools:** This method finds the least used tools from *jdk, maven and gradle.*

**9. triggerStagesCorrelation**: This method generates a list number of trigegrs and number of stages in each pipeline.Using a *correlation function we generate a correlation of triggers and number of stages.*

**10.frequentWhenCondition:** This method finds the *most used condition inside when block.*

**11.frequentStageTypes:**This method analyzes the *most used stage types* inside stages block.

**12.frequentAgentArgumentTypes:** Agent are declared inside pipeline block as well as inside stages block. This method analyzes the most used agent inside the stage block.

**13.stagesStepsCorrelation:** This method finds the correlation between stages and steps in a pipeline by using a correlation function.

**14.commonCommandsInSteps:** This method answers 3 different questions.Firstly,it finds the most commonly used commands inside steps block. Also we have further categorized into stage level as well as user environment level. Secondly,Stage level categorization is done by classifying commands used for Build, Test and Deploy stages. The third type is it finds the most frequently used Windows and Linux commands.

**HOW TO RUN:**

For this project we are using a jenkins plugin:

[jenkinsci](https://github.com/jenkinsci)/[pipeline-model-definition-plugin](https://github.com/jenkinsci/pipeline-model-definition-plugin)

This is used to setup a jenkins server that is later used to convert the jenkins files to corresponding json files. The steps to setup the server are given on the above github page in the README section.

Once the jenkins server is setup, we are ready to setup our pipeline-analyzer project.

1. Clone the code from bitbucket
2. In the analyzer.properties file do the following configurations:

***githubAuthorization=token cba9d36f5c0583f62a3965755f9eadd651d8c27d***

This is the github user token for using the github developer api. Only change the actual token part. The string “token” is essential, do not remove.

***githubBaseUrl=***[***https://api.github.com/***](https://api.github.com/)

This is the base url for the Github developer api

Following are some configurations that are used to construct the URL used to query Github:

***githubPath=search/code***

***githubQueryString=jenkinsfile.txt***

***githubRef=searchresults***

***githubSearchPath=in:path***

***githubUserAgent=https://api.github.com/meta***

***githubSearchType=Code***

Following URL is the endpoint of the local jenkins server that was generated as a result of the setup task of the pipeline-model-definition-plugin project. Please make sure you only replace the domain name and the port according to your requirement.

***jenkinsURL=http://localhost:8080/jenkins/***

Following is the output directory of our project used to generate output files. Most of our output files are in the form of json files.

***outputDirectory=D:\\git\\output\\***

1. In the log4j.properties file, do the following configuration:

This the location of the log output.

***log4j.appender.FILE.File=D:\\git\\adarsh\_hegde\_ashwani\_khemani\_srinath\_kv\_cp\\log.out***

1. In the config.json file give the configuration for the output directory that will be used by the Chart JS code. **Please note :** This should be same as the output Directory configured in the analyzer.properties file.

***{***

***"outputDirectory":"D:\\git\\output\\"***

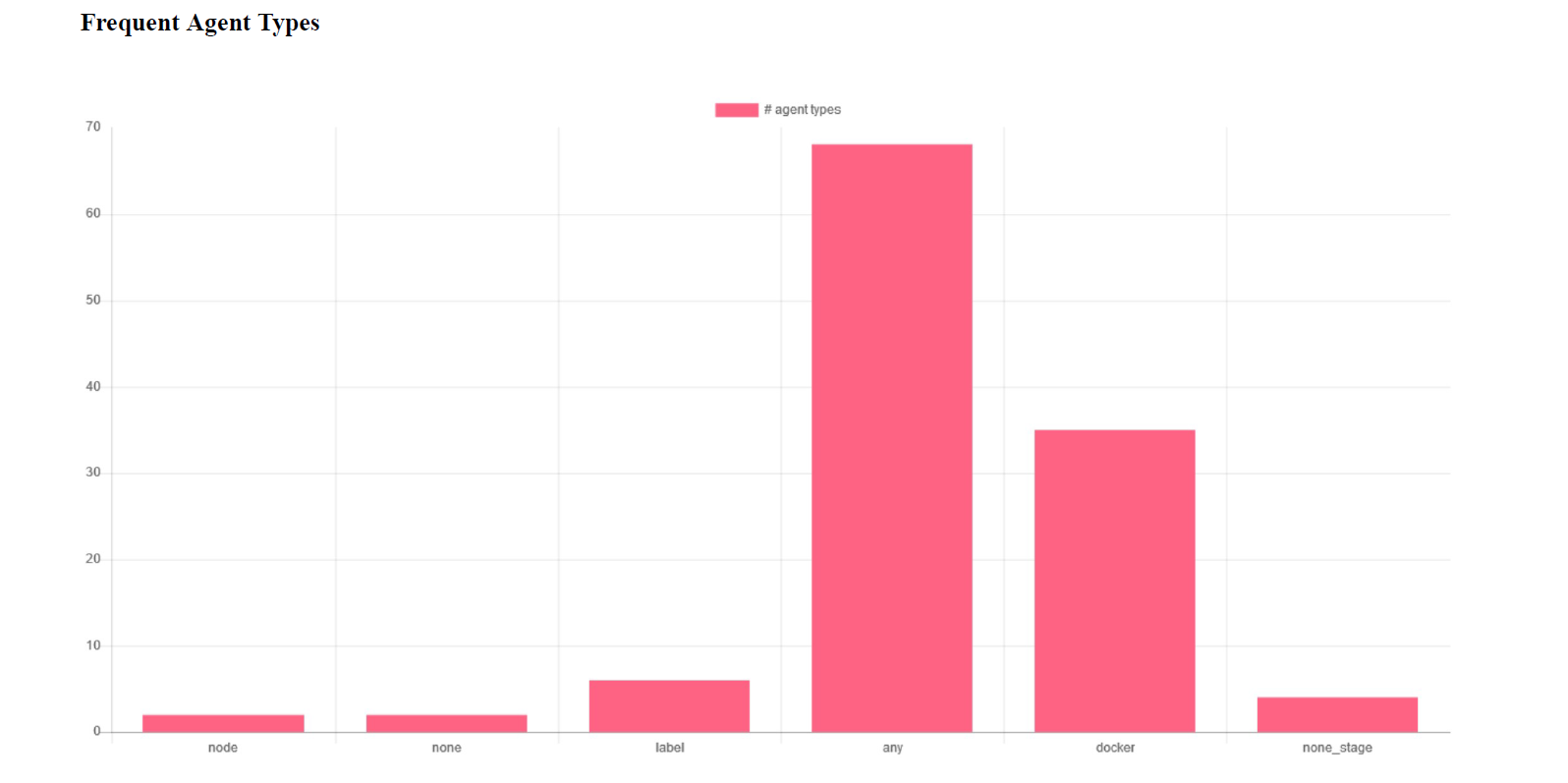
***}***

1. Now the project is setup and we can run the application and the test cases.
2. To run the code:

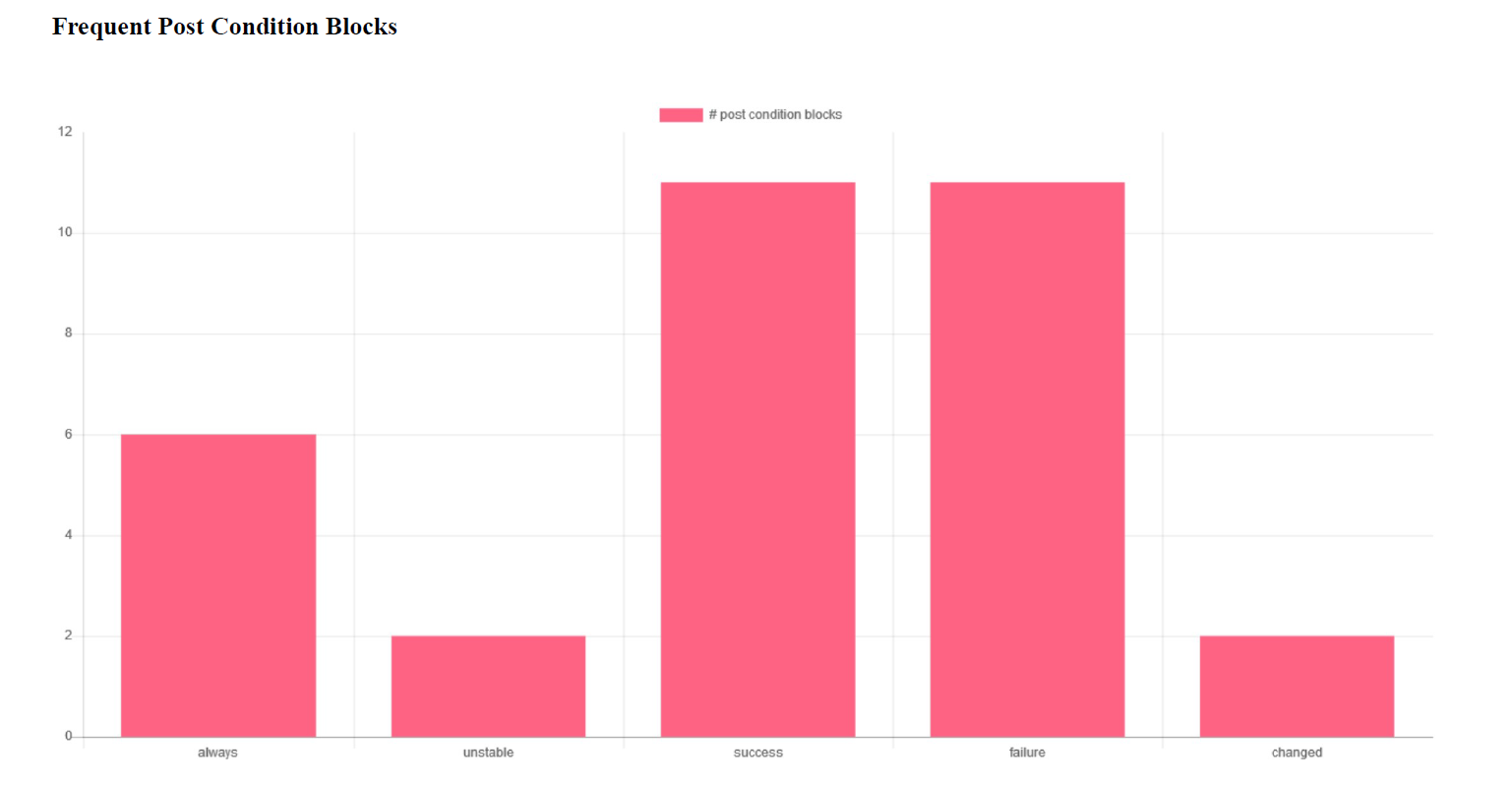
***Execute Main.java***

1. To execute test cases, execute all the tests classes in the test directory.
2. To view the output of our research questions. Copy the **charts.html** in the output folder in the root of the project to the output directory you had configured. Anyways, all the outputs are already there in output folder for viewing before execution. It is very important that the charts.html is in the same directory as the output files.
3. Open the charts.html in the browser(this only works on Firefox and IE due to the CORS restrictions in browsers).
4. The charts for various research questions are there for viewing. Following are some screenshots.

**RESULTS AND EXPLANATION**



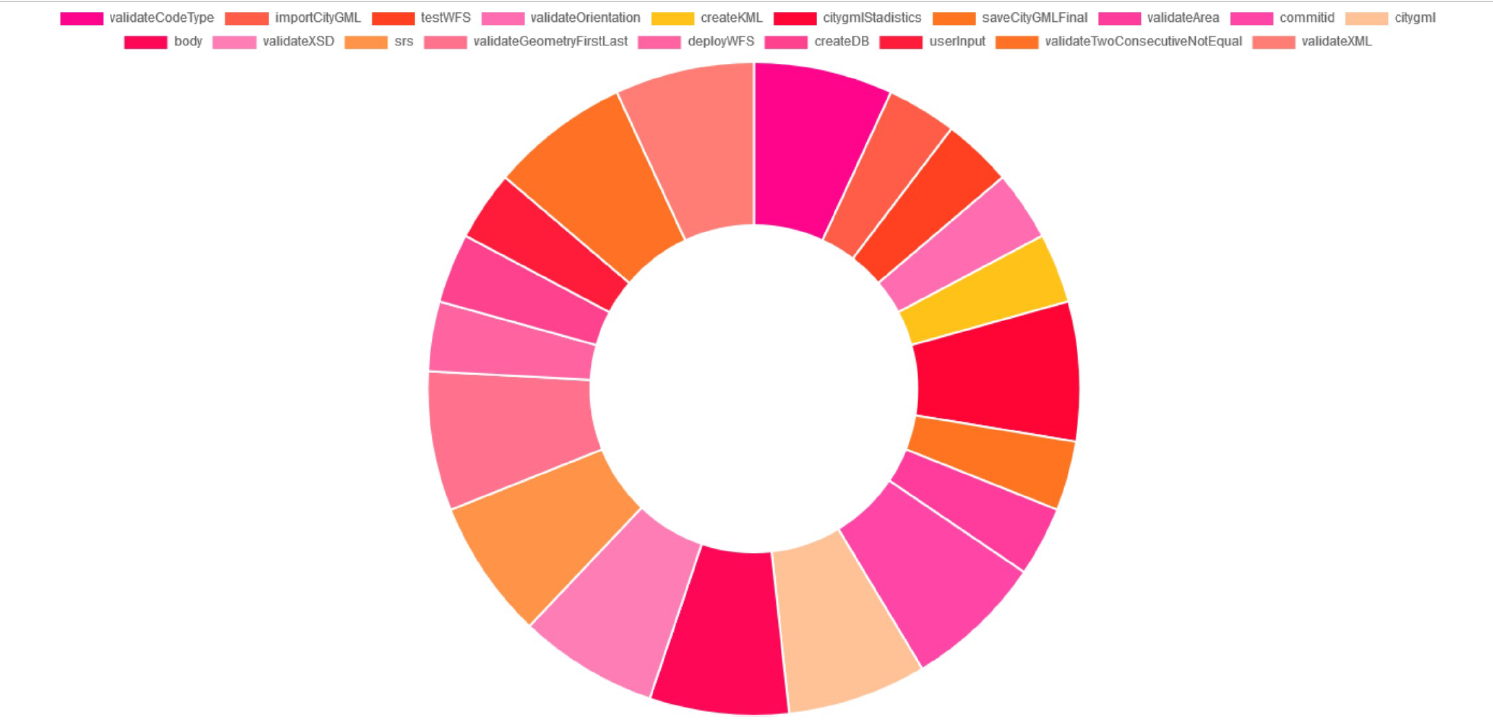
The above diagram show the different agent types and the frequency of each type of agent across different pipeline programs .

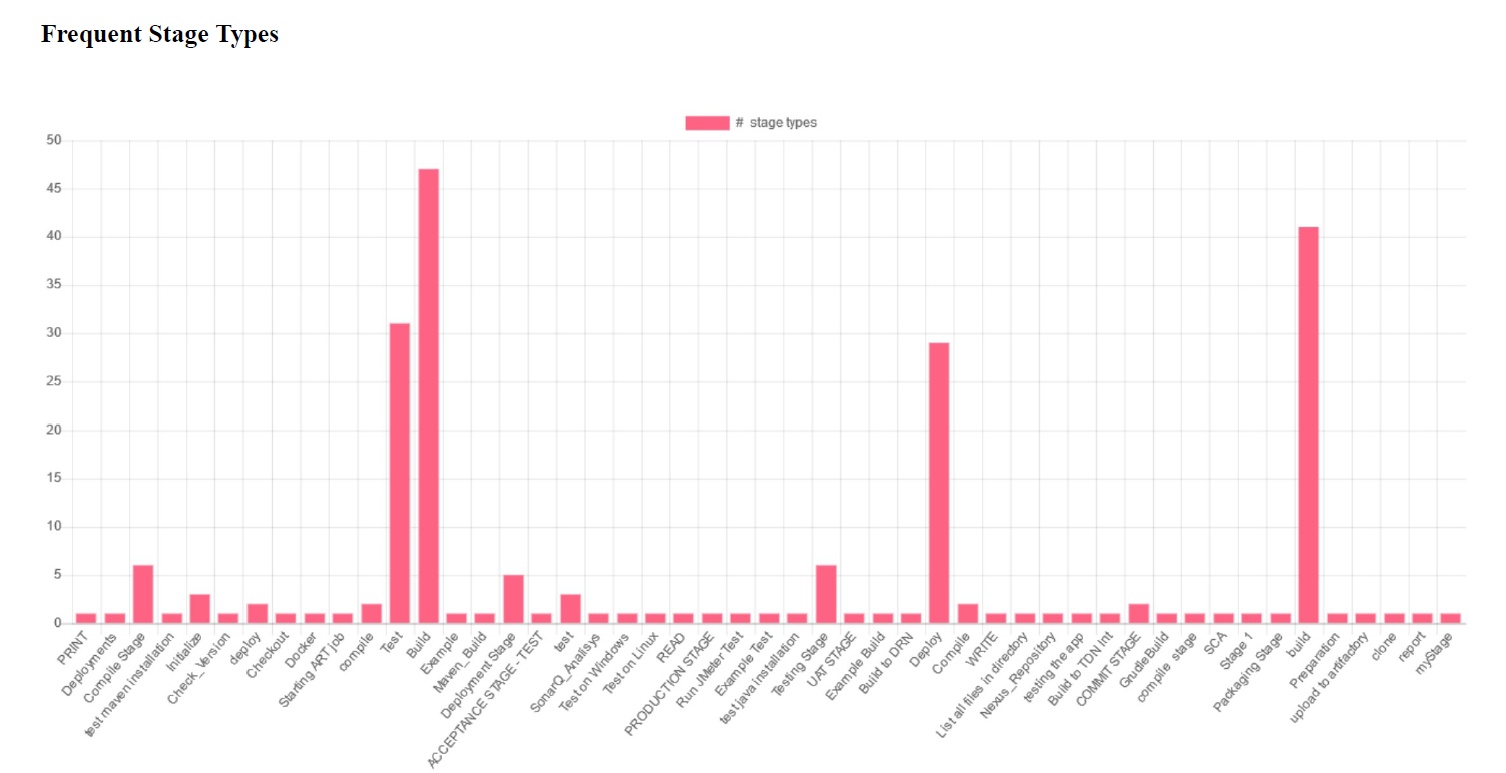


The above diagram show the different post condition blocks and the frequency of each type of condition block across different pipeline programs .

**Frequent Environment Variable Types**

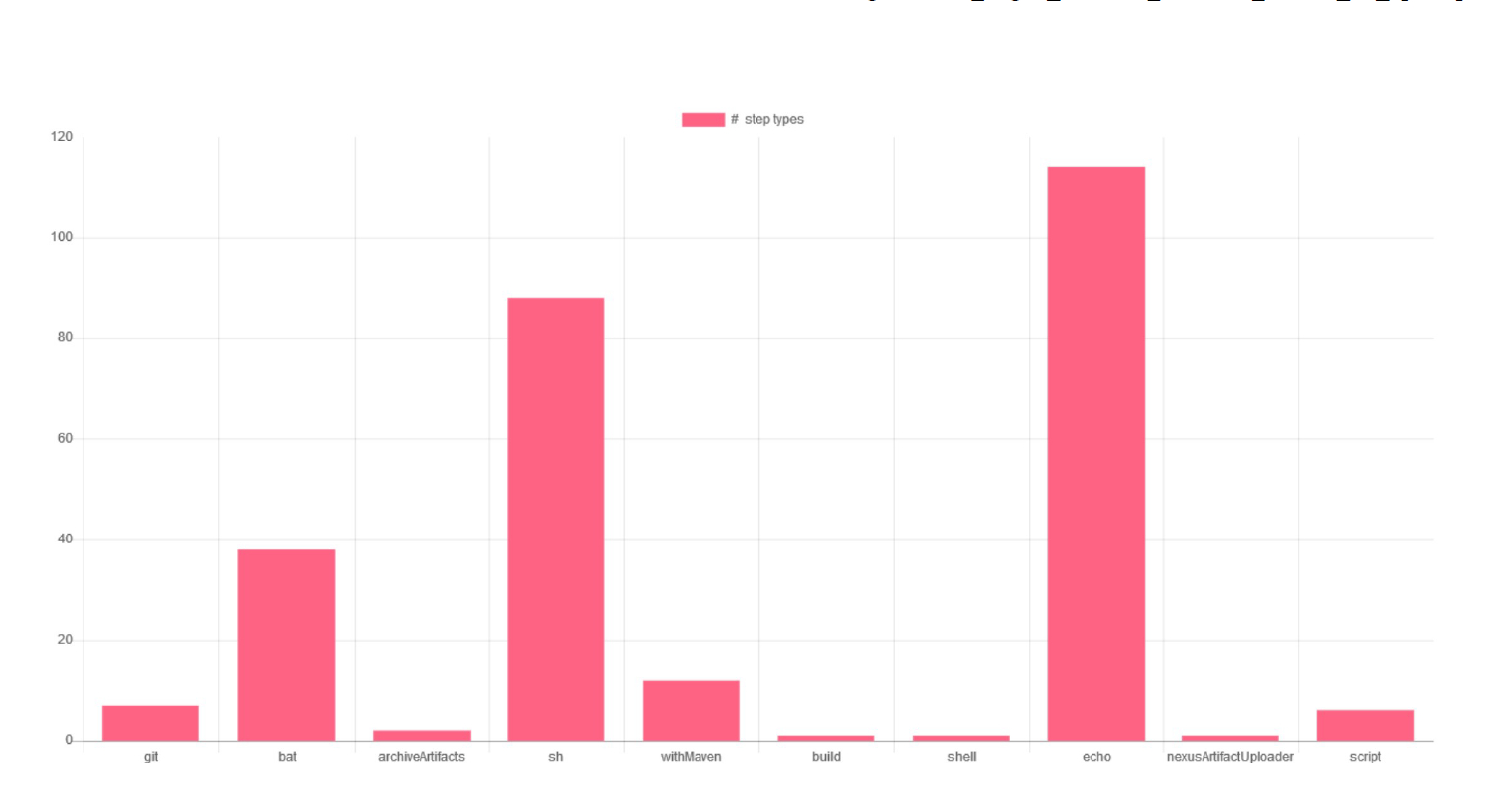
The below diagram show the different environment variable and the frequency of each type of environment variables across different pipeline programs .



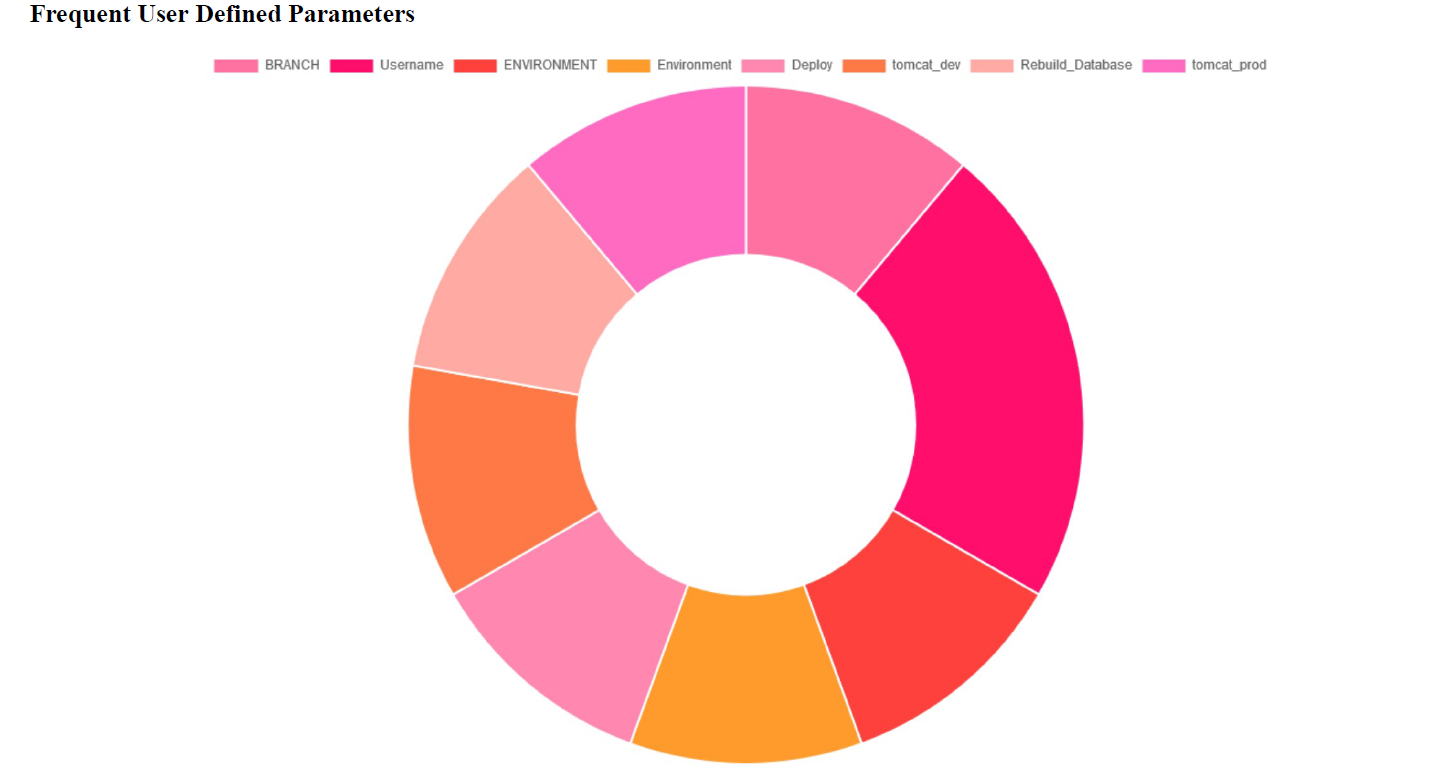


The above diagram show the different stage type and the frequency of each type of stage across different pipeline programs .

Frequent Step Types

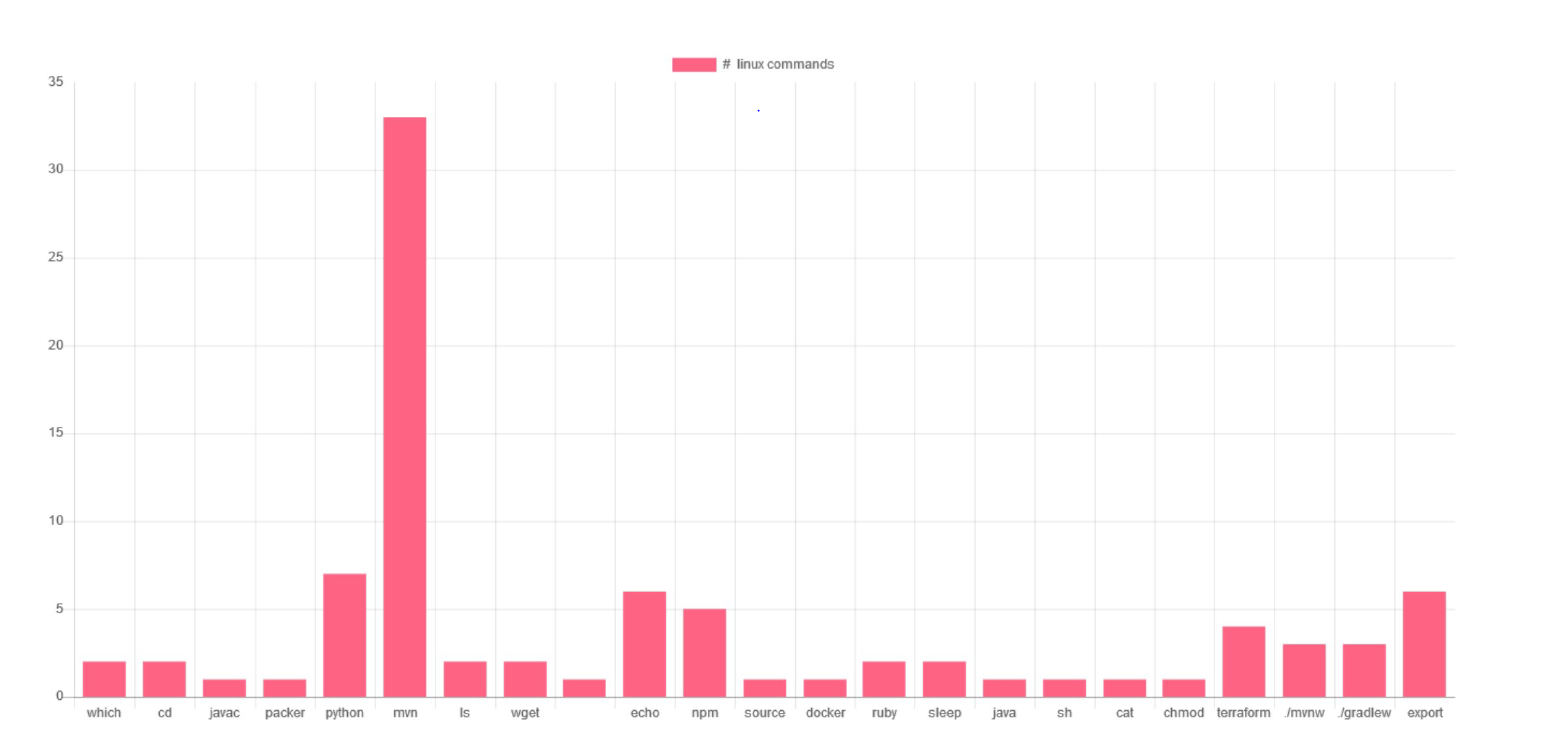


The above diagram show the different step type and the frequency of each type of step across different pipeline programs .

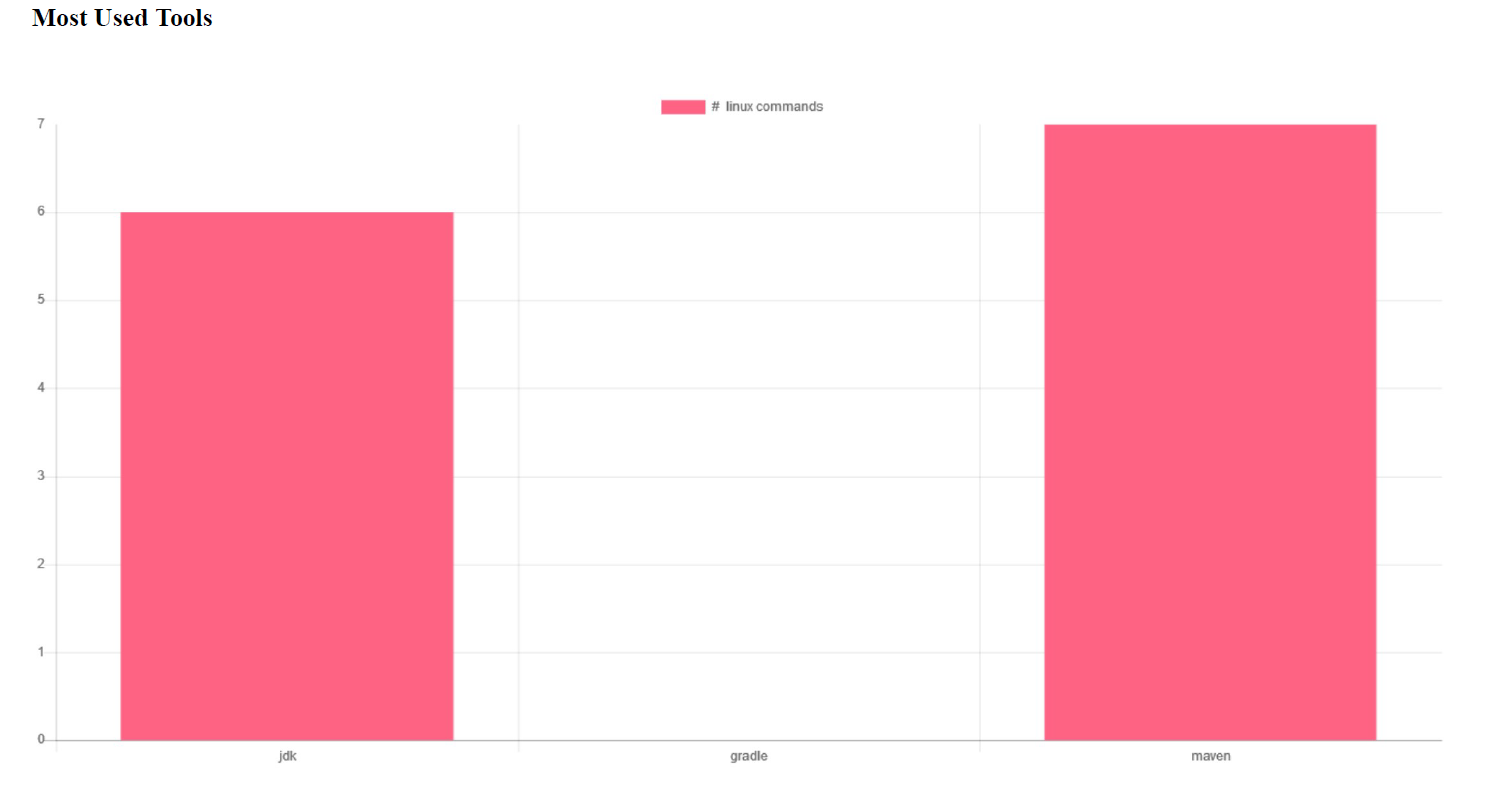


The above diagram show the different user defined parameters and the frequency of each type of parameter across different pipeline programs .

Frequent Linux Commands

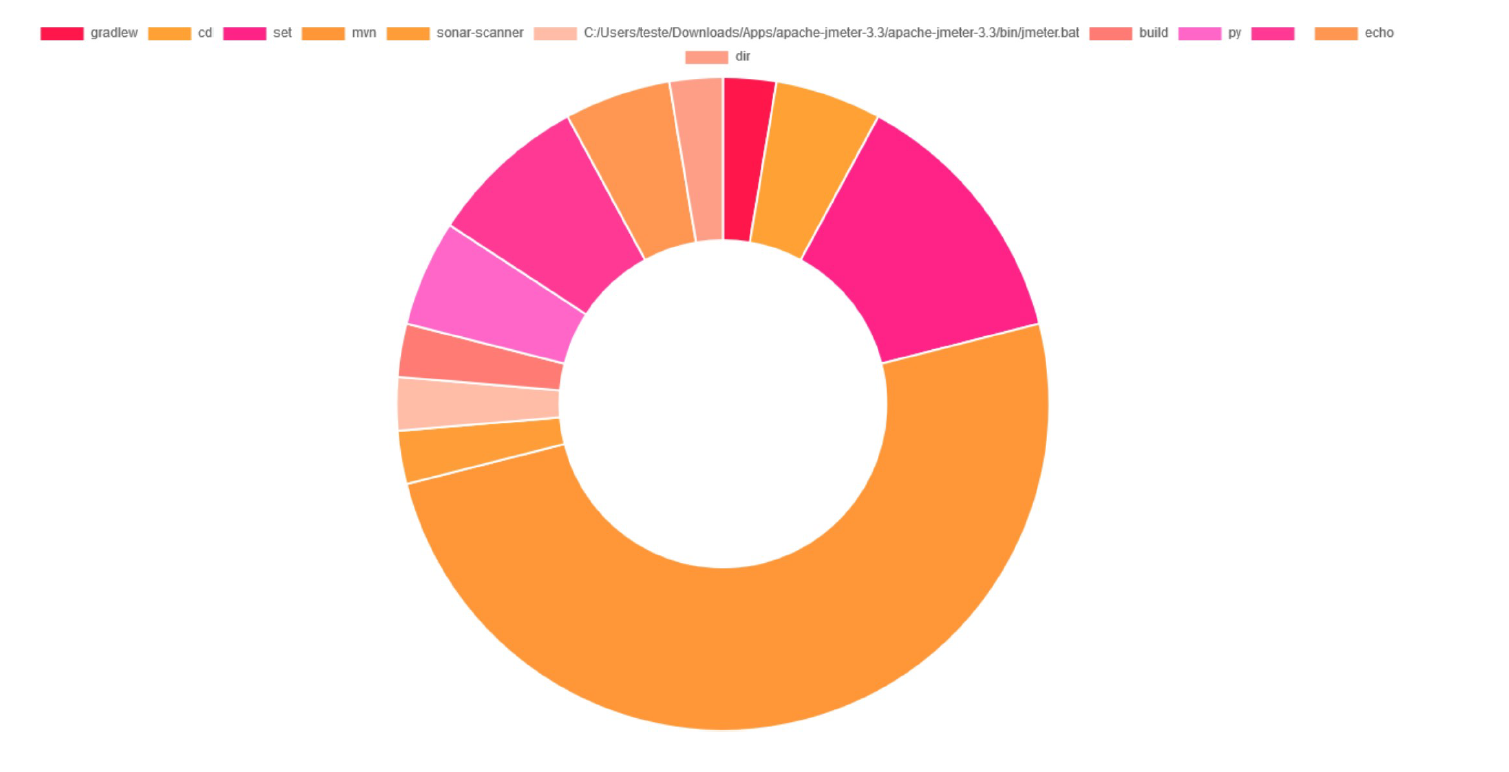


The above diagram show the different linux commands and the frequency of each type of command across different pipeline programs .



The above diagram show the different tools and the frequency of each type of tool across different pipeline programs .

Most Used Windows Commands

The above donut chart shows the most used Windows commands used inside steps clock across pipelines