

Software Developer at Harness

A PROJECT REPORT

Submitted in partial fulfilment of the requirements for the award of the degree

of

BACHELOR OF TECHNOLOGY (B.Tech.)

in

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by

**Aditya Kashyap
199202111**

Under the Supervision of

**Dr. Tejpal
Associate Professor , ECE**



(Department of Electronics and Communication Engineering)

MANIPAL UNIVERSITY JAIPUR

JAIPUR-303007

RAJASTHAN, INDIA

July/2023

DEPARTMENT OF ELECTRONICS AND COMMUNICATION
MANIPAL UNIVERSITY JAIPUR, JAIPUR – 303 007 (RAJASTHAN), INDIA

JAIPUR

Date: 10 July 2023

CERTIFICATE

This is to certify that the project titled **Software Engineer at Harness** is a record of the bonafide work done by **Aditya Kashyap** (199202111) submitted in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology (B.Tech) in **(Discipline)** of Manipal University Jaipur, during the academic year 2022-23.

Dr. Tejpal

Project Guide, Dept of ECE
Manipal University Jaipur

Dr. Manish Tiwari

HOD, Dept of ECE
Manipal University Jaipur



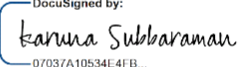
04th July 2023

TO WHOMSOEVER IT MAY CONCERN

This is to certify that Aditya Kashyap was an intern with us from 16th January 2023 to 14th July 2023.

We wish him success in sustaining and achieving higher levels of excellence.

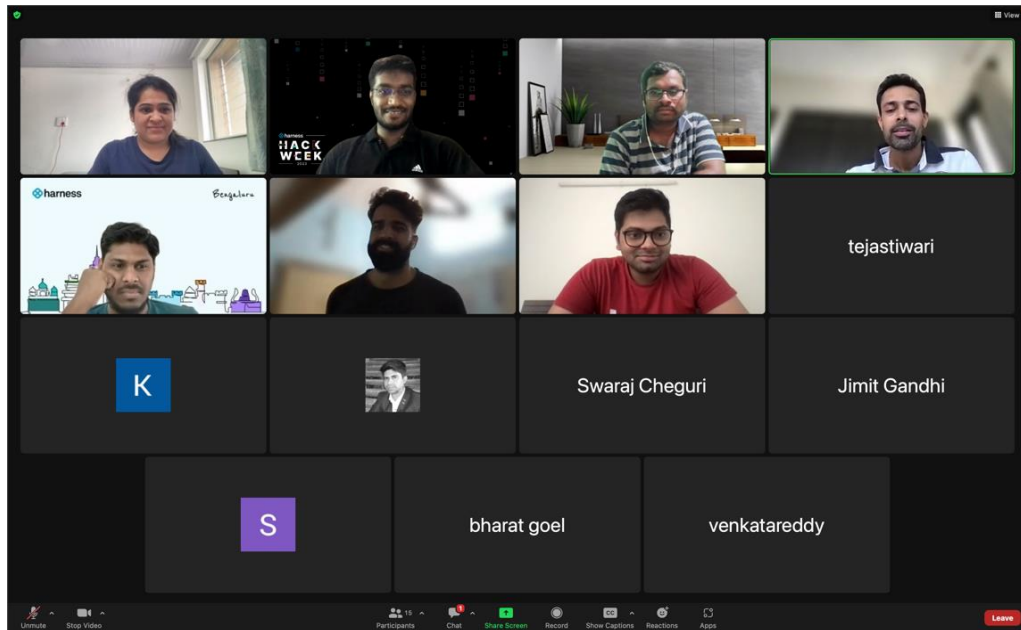
For **Harnessio R&D Labs India Private Limited**

DocuSigned by:

07037A10534E4FB...
Authorized Signatory

HARNESSIO R&D LABS INDIA PRIVATE LIMITED

Regd. Office: Indiqube Orion Khatha No. 1802/55/13/55/11B, Haralukunte Village, 24th Main Road, 2nd Sector HSR Layout
Bangalore 560102, India CIN No. U72900KA2018FTC113739; <https://harness.io> E-mail ID: luan.lam@harness.io

Geotagged photographs (3 to 4)



WFH – Zoom meeting of whole team during stand-up



Taking Joining Kit from Office



Sitting at my desk in Harness, Bangalore

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to the Dean of the Manipal University Jaipur, the Head of the Department of Electronics and Communication, and my project supervisor, Dr.Tejpal, for their guidance and support throughout this project.

I would also like to thank the company personnel from Harness for their cooperation and assistance in providing me with the necessary data and resources for this project. I am grateful to my manager at Harness Mr Nataraja M for constantly pushing me to learn new things and explore my new limits.

Finally, I would like to thank my family and friends for their support and encouragement throughout this internship at Harness.

Aditya Kashyap
(199202111)

ABSTRACT

In today's rapidly evolving technological landscape, ensuring the stability and automation of software systems is crucial. During my internship, I focused on addressing these critical aspects by implementing automation for new features and maintaining the stability of various entities. A key objective was achieving high code coverage and minimizing failures by developing robust corner cases. Additionally, I took on end-to-end ownership of Terraform for my team and was involved in the backend development of the product due to my exceptional performance.

To accomplish this project, I employed a methodology that encompassed several key elements. First, I utilized **TESTNG** for API automation, enabling comprehensive testing and validation of the system's functionality. This approach ensured thorough examination and seamless integration of newly introduced features into the existing framework while maintaining backward compatibility. Additionally, I utilized **Bazel** for backend development and **GO** for creating Terraform resources and data sources.

The outcomes of this project were significant and impactful. Through automation and addressing corner cases, the stability of the entities greatly improved. Code coverage increased substantially, resulting in a reduced number of failures and fewer customer-facing defects. Specifically, the number of CFDs (Customer Facing Defects) decreased from 25 per month to 4 per month for my team. Moreover, by implementing logging capabilities in our microservices, the debugging process became more efficient, reducing the resolution time for customer-facing defects from **96** hours to **72** hours.

In conclusion, this internship successfully achieved its objectives of automating new features, improving stability, and enhancing debugging capabilities. The utilization of **TESTNG** for API automation, **Java** for Harness-core development, and **Terraform** for resource management proved to be effective tools. The outcomes of this project, with statistical figures indicating the reduction in customer-facing defects and resolution time, emphasize the importance of prioritizing automation, stability, and efficiency in software development, ultimately leading to a more robust and reliable system.

LIST OF FIGURES

Figure No	Figure Title	Page No
1	Harness as a Product Snapshot	9
2	How Harness Helps ?	10
3	Harness Delegates Architecture	11
4	Architecture of CI module	12
5	CD Pipeline Example	12
6	Get Started with Feature Flag	13
7	Cluster in Kubernetes	16
8	Git Commands	17
9	Example of RestAssured API testing	19
10	Collection in MongoDB and collections have documents	20
11	Terraform Process	22
12	Harness Provider	22
13	Communication of Harness Delegates with other services	23
14	Setup Delegate (Step 1 - 3)	24
15	Setup Delegate (Step 4)	25
16	Setup Delegate (Step 5 - 8)	25
17	Suite File	27
18	Test class in Harness	28
19	CI Pipeline for Test	29
20	Test Results	29
21	Creation of New Resource	31
22	Test for Terraform Resource	32
23	Terraform Code Snippet	33
24	Sharing Learning with Others	34
25	Creating a useful pipeline for on-prem	34
26	Reducing time by creating custom Images	35
27	Appreciation for helping customer quickly	35
28	Total Pull Requests in Harness-core (BE)	35
29	Contribution in API automation	36
30	Contribution in Harness Terraform Provider	36

Contents		
		Page No
Acknowledgement		5
Abstract		6
List Of Figures		7
Chapter 1	INTRODUCTION	9
1.1	About The Company	9
1.2	About the Product	10
1.3	About the Work Done	14
Chapter 2	BACKGROUND THEORY	
2.1	Tools & Technology	15-22
Chapter 3	METHODOLOGY	23
3.1	Creating Delegate	23
3.2	Writing Test Case for API Automation using RestAssured	26
3.3	Terraform	30
Chapter 4	RESULT ANALYSIS	34-36
Chapter 5	FUTURE SCOPE	
5.1	Future of Harness	37
5.2	My Future at Harness	37
REFERENCES		38
PROJECT DETAILS		39

CHAPTER 1 - INTRODUCTION

1.1 – About the Company

Harness, the Modern Software Delivery Platform, provides a simple, safe and secure way for engineering and DevOps teams to rapidly release applications into production. Harness uses machine learning to detect the quality of deployments and automatically roll back failed ones, saving time and reducing the need for custom scripting and manual oversight, giving engineers their weekends back.

Harness performs various automated tasks in software pipelines such as automatically detecting performance and quality regression in canary deployments and automatic rollbacks and deployments, including self-service deployments with predefined role-based permissions. Harness helps the DevOps engineers to automate code testing, deployment, and rollbacks. It offers ready-made CD pipeline templates, such as blue/green, canary, and rolling deployments, and also allows you to use YAML code to build custom templates.

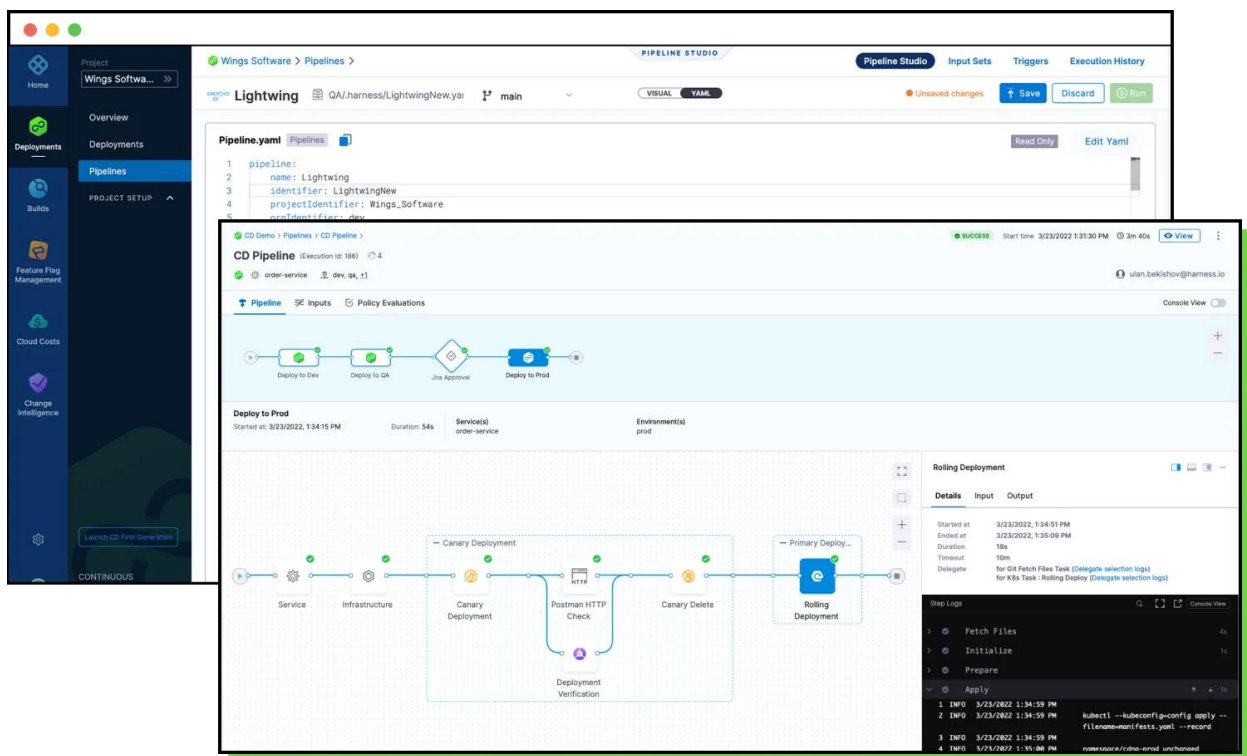


Fig 1

1.2 – About the Product

Harness is revolutionizing the software delivery process with its innovative Software Delivery Platform. Powered by artificial intelligence (AI), Harness simplifies and streamlines DevOps processes, offering a comprehensive suite of modules including Continuous Integration (CI), Continuous Deployment (CD), GitOps, Feature Flags, Cloud Costs, and much more. What sets Harness apart is its unique approach of providing fully integrated modules within a single pipeline, ensuring a seamless and cohesive experience for software delivery.

The Harness platform consists of multiple interconnected modules, designed to work harmoniously with each other. These modules can be utilized independently, integrating with your existing tooling, or combined to form a powerful and unified pipeline that spans CI, CD, and Feature Flags. This unified approach enables teams to effortlessly manage the entire software delivery lifecycle from end to end. Moreover, the platform incorporates metadata that enhances Cloud Cost Management, providing valuable insights and control over cloud expenditure.

With its AI-driven capabilities and comprehensive feature set, Harness empowers organizations to optimize their DevOps processes, enabling faster and more efficient software delivery. Whether you choose to leverage individual modules or leverage the full power of the unified pipeline, Harness ensures a streamlined and scalable approach to software delivery, driving productivity and success for development teams.

1.2.1 - Components and Harness Modules Integration

The way Harness modules integrate with the software development life cycle (SDLC) is depicted below .

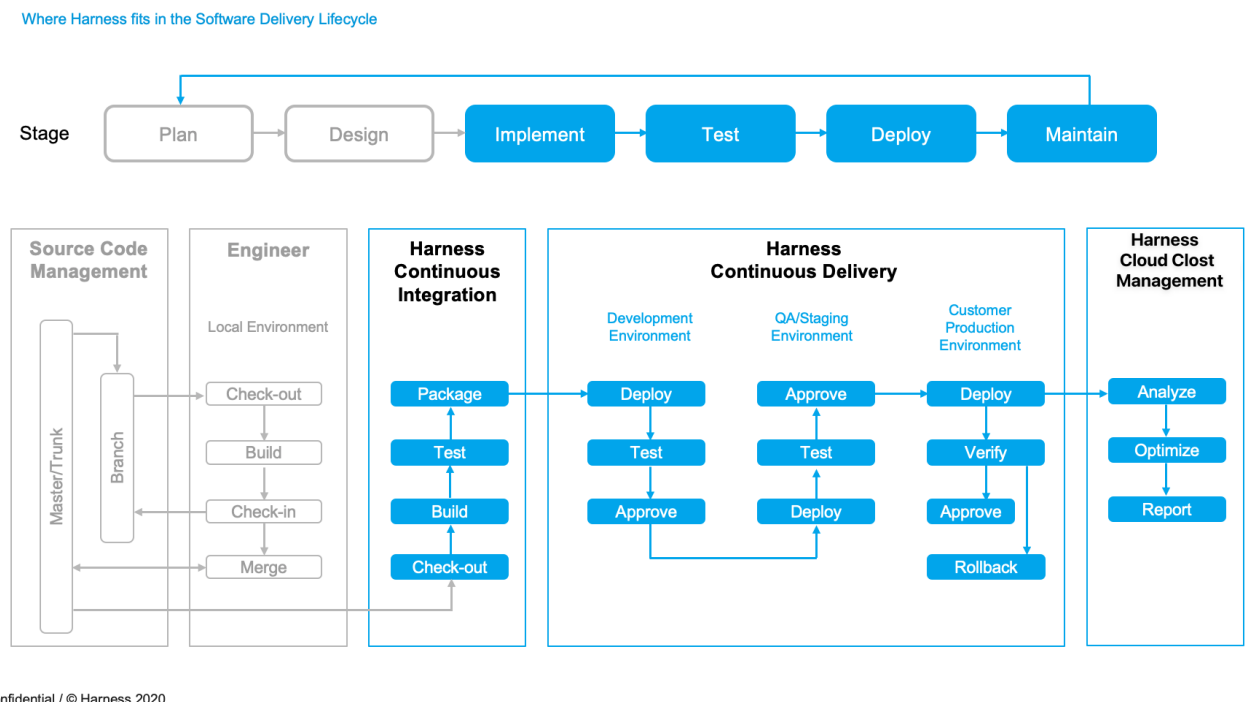


Fig 2

Out of all these , there are two main components of the Harness platform:

- **Harness Manager**: The manager stores your CI/CD and various configurations and manages your pipelines. Alternatively, you can manage pipelines through Git. You can manually trigger pipelines in the manager or set up automatic triggers in response to Git schedules, events, new artifacts, and more. You can use the SaaS version of the manager or self-manage it in your infrastructure.
- **Harness Delegate**: The delegate can be installed in your environment, connecting it to the Harness Manager. Delegate performs task using various assets, such as container orchestration platforms, monitoring systems, and artifact repositories.

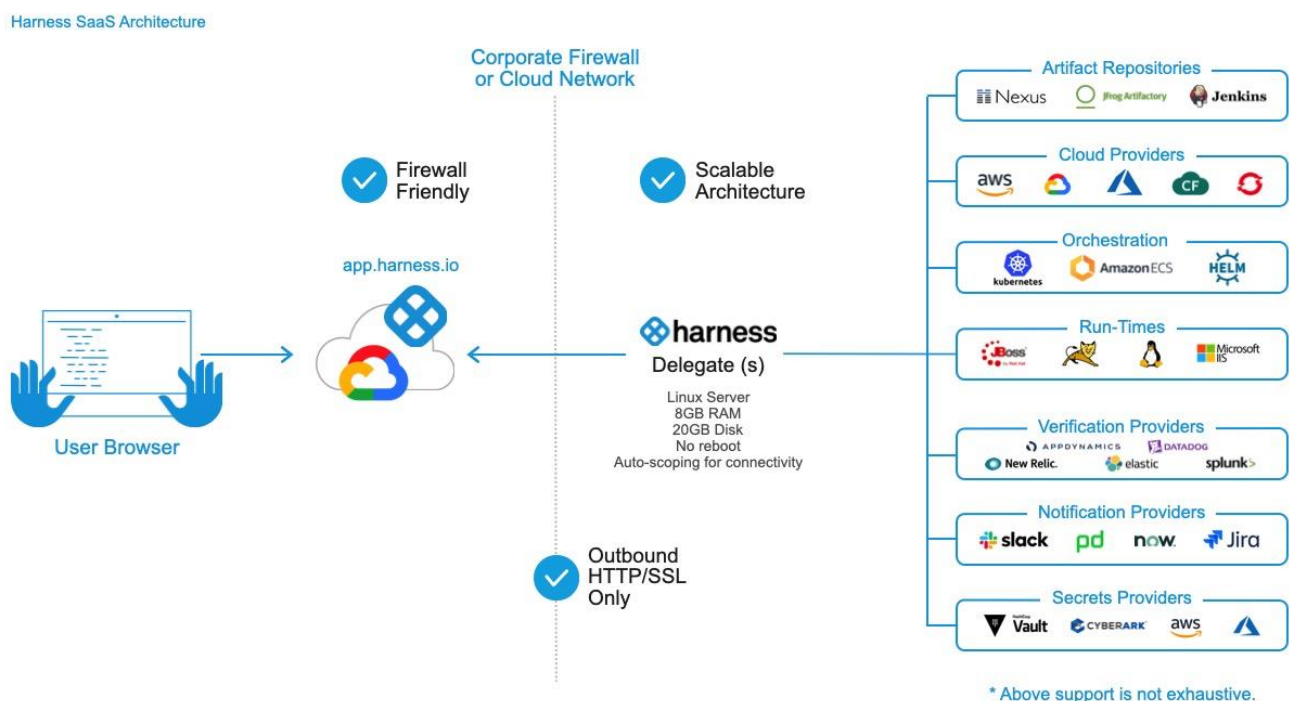


Fig 3

1.2.2 - Brief Overview About Each Module

Continuous Integration : Continuous Integration is automated builds that can be triggered by some sort of event, such as a code check-in, or merge, or on a regular schedule. The end goal of a build is to be deployed somewhere, and the main goal of Continuous Integration is to build and publish that deployable unit.

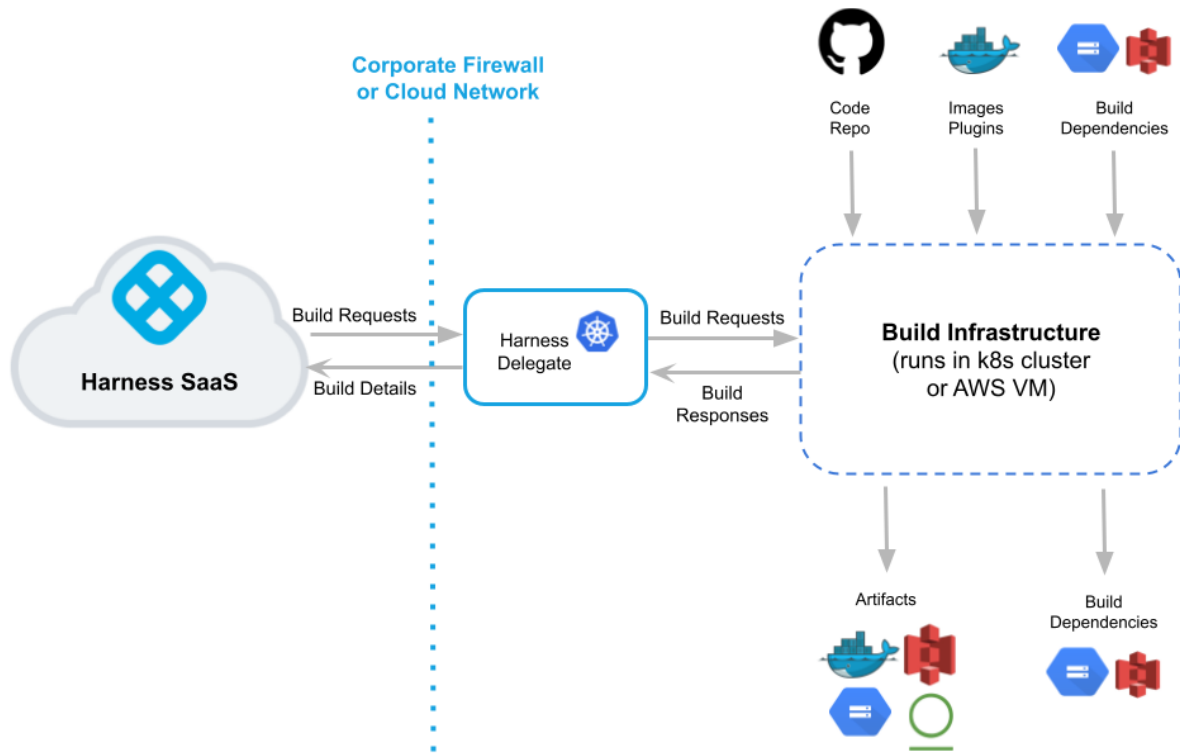


Fig 4

Continuous Delivery & GitOps (CD) : The primary objective of continuous delivery (CD) is to efficiently deliver a packaged artifact into a production environment. CD encompasses the automation of the entire delivery process, including the deployment phase. Within CD, various responsibilities are handled, such as infrastructure provisioning, change management (ticketing), artifact deployment, change verification, and ongoing monitoring. An essential aspect of CD is ensuring that any deployment or changes occur seamlessly and only when there are no issues detected, thereby maintaining the stability and reliability of the production environment.

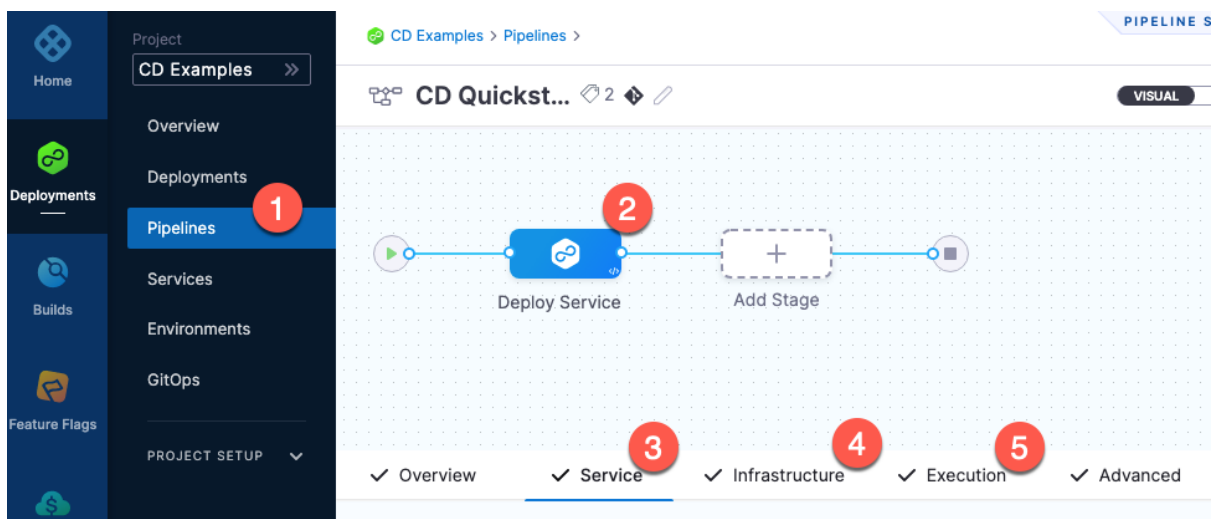


Fig 5

Feature Flags : Harness Feature Flags is a feature management solution that allows you to change your software's functionality without deploying new code. It does this by letting you hide code or behaviour without having to ship new versions of the software.

This module is really powerful as

- Reduce risk by testing new features before you deploy them to everyone.
- Improve the customer experience by ensuring that everyone has the same access to features, regardless of their location or device.
- Increase agility by giving you the flexibility to change your software's functionality quickly and easily.

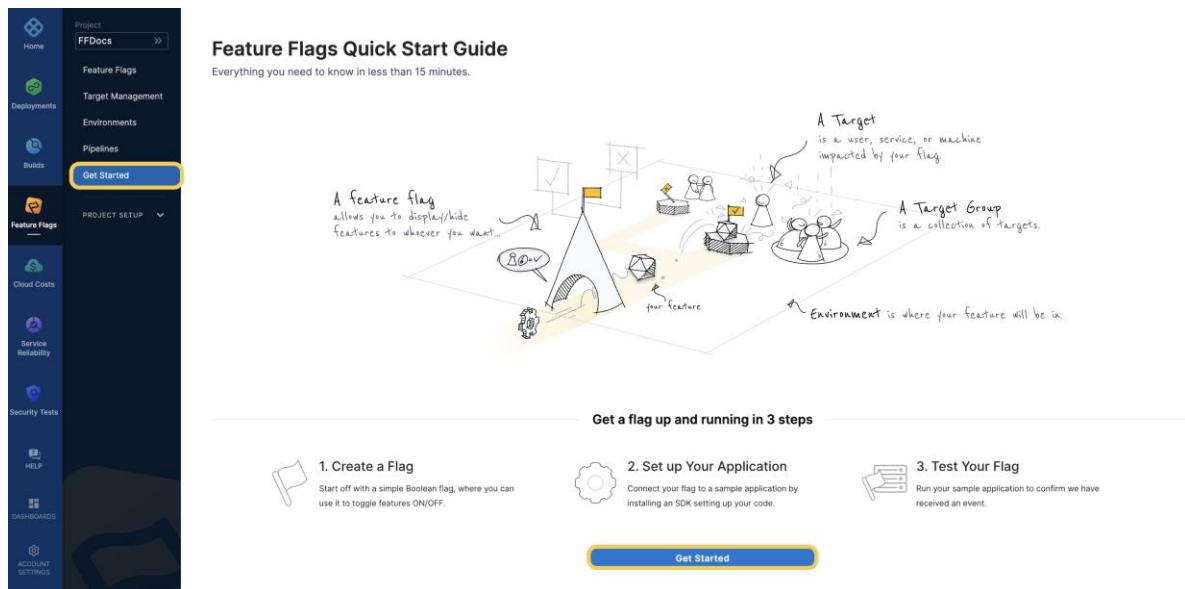


Fig 6

Cloud Cost Management : It is a powerful tool that can help users save money on their cloud costs. It is easy to use and integrates with a variety of cloud providers, including AWS, Azure, and Google Cloud Platform. If you are looking for a way to optimize your cloud costs, Harness CCM is a great option to consider. CCM offers a variety of features to help users optimize their cloud costs, including:

- **Cost anomaly detection**: CCM can identify unexpected spikes or drops in cloud costs, which can help users identify and fix cost-related issues.
- **Cost forecasting**: CCM can forecast future cloud costs based on historical data, which can help users plan their budgets and avoid overspending.
- **Cost optimization recommendations**: CCM can provide users with recommendations on how to optimize their cloud costs, such as right-sizing resources, using reserved instances, or leveraging spot instances.
- **Cost governance**: CCM can help users enforce cost policies, such as budget limits or resource usage quotas.

1.3 About the Work Done

I am currently an intern as a Software Engineer in the Platform Team at Harness. I have been given the opportunity to work on multiple crucial features and put my engineering knowledge into practice. My job role includes handling the quality of the product, automation, and development of BE APIs, fixing bugs, and maintaining Terraform for my team.

I like how interns are never treated as interns at Harness. This is exactly what is required for a pre-final year student to get out of their comfort zone and solve real-world problems. I spent my initial months understanding the product completely rather than focusing on what I was supposed to do according to my job description. I now realize the importance of this.

I have worked extensively to stabilize the resources and entities that I own to a large extent. I have also worked on saving costs for the company, which was appreciated by my team members and senior vice president.

The major work of stabilization was done on the Terraform side. I increased the stability of the Platform Team-owned resources from 74% to 99.2%. As I mentioned earlier, Harness Delegates is a major component, and it brings a lot of costs. Initially, we had 4 different delegates to get our job done, as they all had different capabilities. I merged them into a single delegate and increased the number of pods of that delegate to meet our requirements. This led to saving \ \$100 per month and brought the cost down from \$250 per month to \$150 per month. I have done this across all of our accounts, which has resulted in massive savings.

I am proud of the work that I have done during my internship at Harness. I have learned a lot and grown as an engineer. I am grateful for the opportunity to have worked on such crucial features and to have made a real impact on the company. I am confident that I have the skills and knowledge to be a valuable asset to any team.

I am excited to continue my career in software engineering and to use my skills to solve real-world problems. I am confident that I can make a significant contribution to any company that I join.



CHAPTER 2 - BACKGROUND THEORY

2.1 – Tools Used in Harness

In this section, we will discuss the different tools and technologies that Harness uses to make sure that the SDLC (software development life cycle) for any company remains simpler. Here we would be knowing the spine of Harness that has made it so successful in the market, that it has UPI has its customer, Apple and many more.

2.1.1 – Kubernetes

Kubernetes is an open-source container orchestration system that helps to manage containerized applications across a variety of environments, including physical, virtual, and cloud. It is a highly flexible tool that can be used to deploy and manage complex applications running on clusters of hundreds to thousands of servers.

Kubernetes provides a number of features that make it a powerful container management system, including:

- Autoscaling: Kubernetes can automatically scale your application up or down based on demand. This can help you to save money on cloud resources and ensure that your application is always running at the optimal level.
- Rollouts and rollbacks: Kubernetes can be used to deploy new versions of your application gradually, over time. This can help you to minimize risk and ensure that your application is always available.
- Health monitoring: Kubernetes can monitor the health of your application and automatically restart or replace unhealthy containers. This can help you to ensure that your application is always running smoothly.
- Load balancing: Kubernetes can distribute traffic across your application's containers, ensuring that no single container is overloaded.
- Secret management: Kubernetes can store and manage sensitive data, such as passwords and API keys, in a secure way. This is our default secret storage space for all the secrets that are created in Harness which is called as Google Key Management System

Kubernetes Basics

- Cluster: It is a collection of hosts(servers) that helps you to aggregate their available resources. That includes ram, CPU, ram, disk, and their devices into a usable pool.
- Master: The master is a collection of components which make up the control panel of Kubernetes. These components are used for all cluster decisions. It includes both scheduling and responding to cluster events.

- Node: It is a single host which is capable of running on a physical or virtual machine. A node should run both kube-proxy, minikube, and kubelet which are considered as a part of the cluster.
- Namespace: It is a logical cluster or environment. It is a widely used method which is used for scoping access or dividing a cluster.

Why we need Kubernetes ?

Kubernetes is needed because it can help to solve a number of problems that are associated with containerized applications, such as:

- Downtime: Kubernetes can help to minimize downtime by automatically rolling out new versions of your application gradually, over time. This can help you to ensure that your application is always available, even if there are problems with a particular version.
- Scalability: Kubernetes can help you to scale your application up or down based on demand. This can help you to save money on cloud resources and ensure that your application is always running at the optimal level.
- Security: Kubernetes can help you to secure your application by providing features such as secret management and pod-level security. This can help you to protect your application from unauthorized access and malicious attacks.
- Manageability: Kubernetes can help you to manage your application more effectively by providing features such as health monitoring, logging, and auditing. This can help you to identify and troubleshoot problems more quickly and easily.
- Containers are lightweight and portable: Containers are self-contained units of software that include everything that an application needs to run, such as the application code, its dependencies, and its runtime environment. This makes them easy to deploy and move between different environments, such as development, staging, and production.
- Containers are reliable: Containers are designed to be reliable and resilient. If one container fails, Kubernetes can automatically restart it. This helps to ensure that your application is always available.

Filter: Select property name or value

<input type="checkbox"/> Status	Name ↑	Location	Number of nodes	Total vCPUs	Total memory	Notifications	Labels
<input type="checkbox"/>	automation-cluster	us-central1-c	5	68	272 GB	4 Scaling issues	— ⋮
<input type="checkbox"/>	performance-cluster	us-central1-c	2	4	16 GB	Pods unschedulable	— ⋮
<input type="checkbox"/>	pl-play-cluster-pr	us-central1-c	12	60	240 GB	3 Scaling issues Node upgrade available	— ⋮

Fig 7

2.1.2 – Git and GitHub

Git is a distributed, open-source version control system that enables developers and data scientists to track code, merge changes, and revert to older versions.

When you make changes to a file in Git, the changes are saved in your local directory. They are not yet part of the Git development history. To create a commit, you need to first stage the changed files. Staging is the process of adding files to the staging area, which is a temporary holding area for changes that you want to commit.

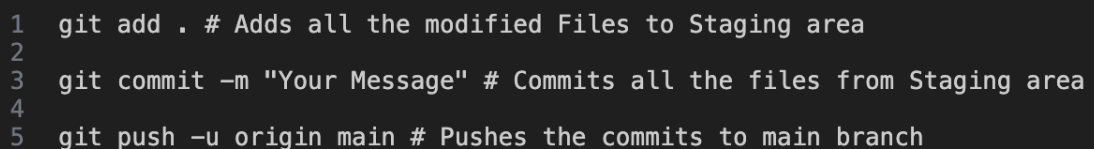
Once you have staged the changes that you want to commit, you can package them into a commit with a message describing the changes. The commit message is important because it helps you to keep track of the changes that you have made to your code.

The three states of files in Git are:

- Modified: A modified file is a file that has been changed but has not yet been staged.
- Staged: A staged file is a file that has been added to the staging area and is ready to be committed.
- Committed: A committed file is a file that has been added to the Git development history.

Committing your changes is important because it creates a snapshot of your code at a particular point in time. This snapshot can be used to revert to older versions of your code if necessary.

The basic commands that is used in life of every Software Engineer are :

A terminal window with a dark background and three colored window control buttons (red, yellow, green) in the top-left corner. It contains five lines of text, each preceded by a line number from 1 to 5.

```
1 git add . # Adds all the modified Files to Staging area
2
3 git commit -m "Your Message" # Commits all the files from Staging area
4
5 git push -u origin main # Pushes the commits to main branch
```

Fig 8

2.1.3 - Java

Java is a high-level, general-purpose, object-oriented programming language that is designed to be simple, robust, secure, and portable. Java code is compiled into bytecode that can be run on any platform that has a Java Virtual Machine (JVM).

Here are some of the key features of Java:

- Simple: Java syntax is simple and easy to learn, even for beginners.
- Object-oriented: Java is an object-oriented language, which means that everything in Java is an object. Objects have data and behavior, and they can interact with each other.
- Robust: Java is a robust language that is designed to be reliable and secure. It has a number of features that help to prevent errors and security vulnerabilities, such as garbage collection and exception handling.
- Secure: Java is a secure language that is designed to protect users from security threats. It has a number of features that help to prevent malicious code from running, such as bytecode verification and sandboxing.
- Portable: Java code is portable, which means that it can be run on any platform that has a JVM. This makes it a good choice for developing software that needs to be deployed to multiple platforms.

Java is a popular programming language that is used to develop a wide variety of software, including web applications, mobile apps, and desktop applications. It is a powerful and versatile language that can be used to create high-quality software.

2.1.4 – Rest Assured Testing Framework

Rest Assured is an open-source Java library that can be used to test RESTful APIs. It provides a number of features that make it easy to automate API testing, including:

- A simple and intuitive API: Rest Assured has a simple and easy-to-use API that makes it easy to get started with API testing.
- The ability to test a wide variety of RESTful APIs: Rest Assured can be used to test a wide variety of RESTful APIs, regardless of the underlying technology.
- Powerful features for asserting and verifying API responses: Rest Assured provides a number of powerful features for asserting and verifying API responses, such as JSON path matching and XML path matching.
- The ability to extend with custom assertions and plugins: Rest Assured can be extended with custom assertions and plugins, which makes it even more powerful and flexible.

Rest Assured integrates well with Maven and our Automation repository is a maven project so it benefits a lot, which makes it easy to build and run API tests. It also has a number of built-in features for reporting and debugging API tests.

Why we chose Rest Assured ?

One reason why we need Rest Assured is to test the APIs that are used by popular applications like Google Maps. When you open Google Maps and look for a place to go, you may see nearby restaurants and travel options from leading providers. These APIs are not Google products, but Google uses them to provide its services. Rest Assured can be used to test these APIs to make sure that they are working properly and that they are returning the correct data.

Another reason why we need Rest Assured is to automate API testing. In today's fast-paced world, it is essential to be able to automate API testing. This allows you to test your APIs quickly and easily, without having to manually test them every time you make a change. Rest Assured is a relatively easy-to-use tool, even for beginners. This makes it a good choice for teams that do not have a lot of experience with API testing.

Finally, Rest Assured is open source. This means that it is free to use and there is a large community of users and contributors. This can be helpful if you need help with using Rest Assured or if you need to find information about how to use it.



```
1 import io.restassured.RestAssured;
2 import io.restassured.response.Response;
3 import org.testng.Assert;
4
5 public class RestAssuredExample {
6
7     public static void main(String[] args) {
8         // Set the base URI for the API
9         RestAssured.baseURI = "https://api.github.com/";
10
11         // Create a request to the /users/octocat endpoint
12         Response response = RestAssured.get("/users/octocat");
13
14         // Assert that the response status code is 200
15         Assert.assertEquals(response.getStatusCode(), 200);
16
17         // Assert that the response body contains the key "login" with the value "octocat"
18         Assert.assertEquals(response.body().jsonPath().get("login"), "octocat");
19     }
20 }
```

Fig 9

2.1.5 – MongoDB

MongoDB is an open-source, document-oriented database and leading NoSQL database. It is written in C++ and is a cross-platform database that provides high performance, high availability, and easy scalability. MongoDB works on the concept of collections and documents.

A collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema, meaning that documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

A document is a set of key-value pairs. Documents have a dynamic schema, which means that documents in the same collection do not need to have the same set of fields or structure. Common fields in a collection's documents may hold different types of data.

Here are some of the reasons why MongoDB is being used in Harness:

- Document-oriented storage: Data is stored in the form of JSON-style documents. This makes it easy to store and query data that is naturally organized in a document-like format.
- Indexing on any attribute: MongoDB allows you to index on any attribute of a document. This makes it possible to quickly and efficiently query data based on specific criteria.
- Replication and high availability: MongoDB supports replication, which means that your data is stored on multiple servers. This ensures that your data is always available even if one server fails.
- Auto-sharding: MongoDB automatically shards your data across multiple servers as your data grows. This ensures that your database can scale to handle large amounts of data.

Overall, MongoDB is a powerful and versatile database that is well-suited for a variety of use cases. If you are looking for a database that is document-oriented, scalable, and fault-tolerant, then MongoDB is a great option to consider.



Fig 10

2.1.6 – Terraform

Terraform is an open-source infrastructure as code software tool that enables you to define infrastructure resources in a declarative configuration file. Terraform can manage existing and new resources across public and private clouds. It can also manage on-premises resources.

Terraform works by using a provider plugin for each supported cloud or service. These plugins allow Terraform to interact with the APIs of the cloud or service to create, update, and delete resources.

The Terraform workflow consists of three stages:

- Plan: In this stage, Terraform analyses the current state of your infrastructure and compares it to your desired state, as defined in your configuration file. Terraform then generates a plan that describes the changes it needs to make to your infrastructure to achieve the desired state.
- Apply: In this stage, Terraform executes the plan that it generated in the previous stage. This may involve creating new resources, updating existing resources, or deleting resources.
- Destroy: In this stage, Terraform deletes all of the resources that it created in the Apply stage.

Here are some of the benefits of using Terraform:

- Quick : It is very quick in creating new things and managing them in comparison to creation of things via API .
- Easier Adoption : The user friendliness way to write code makes it easier for big giants to adopt to this as they can manage different services from one place.
- Consistency: Terraform uses a declarative configuration file, which means that you can define your infrastructure in a way that is easy to understand and maintain.
- Reusability: Terraform configuration files can be reused across multiple projects, which can save you time and effort.
- Portability: Terraform can be used to manage infrastructure on a variety of cloud platforms and on-premises systems.
- Automation: Terraform can be used to automate the provisioning and management of your infrastructure, which can save you time and effort.
- Version control: Terraform configuration files can be version controlled, which makes it easy to track changes and roll back to previous versions.

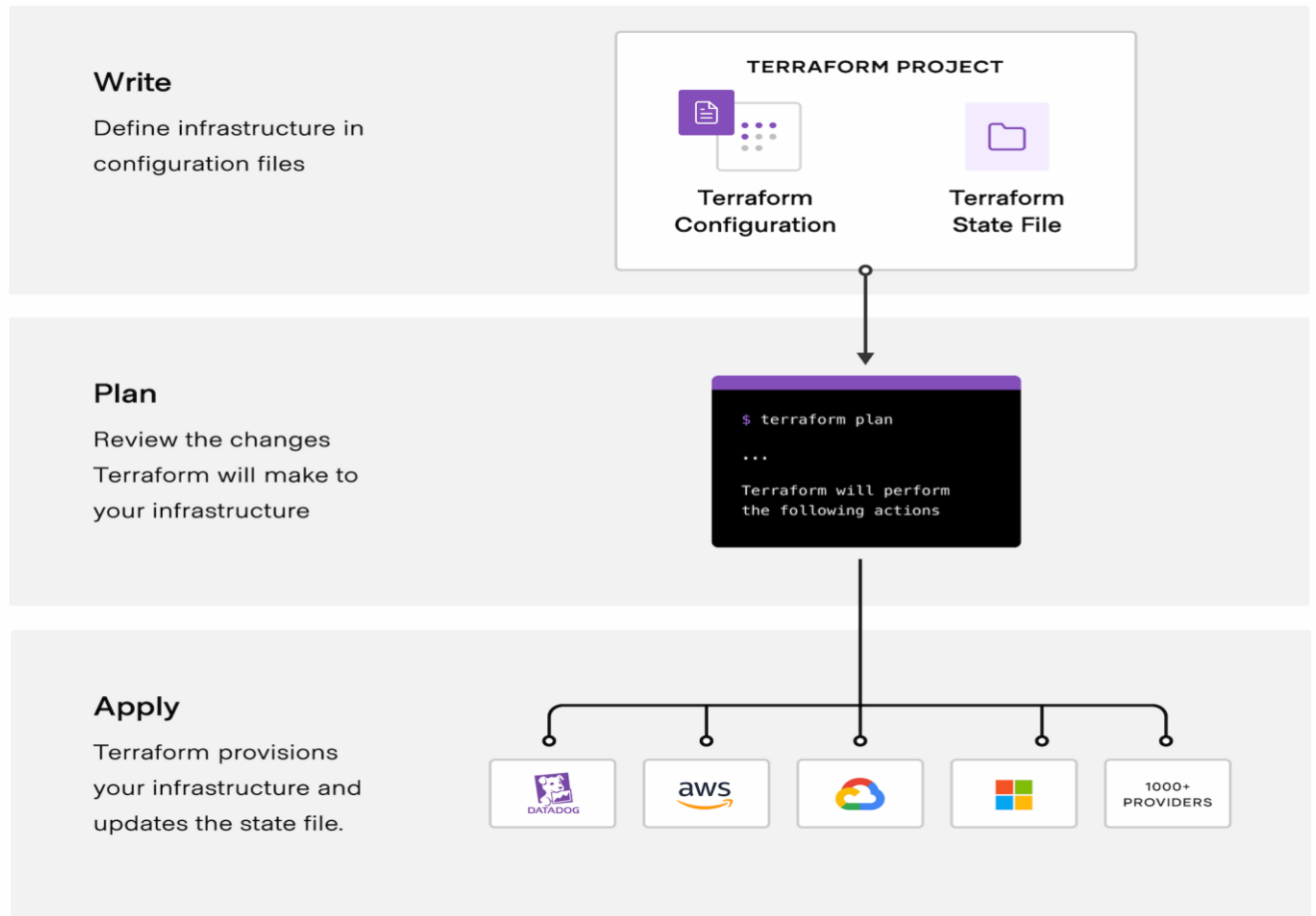


Fig 11

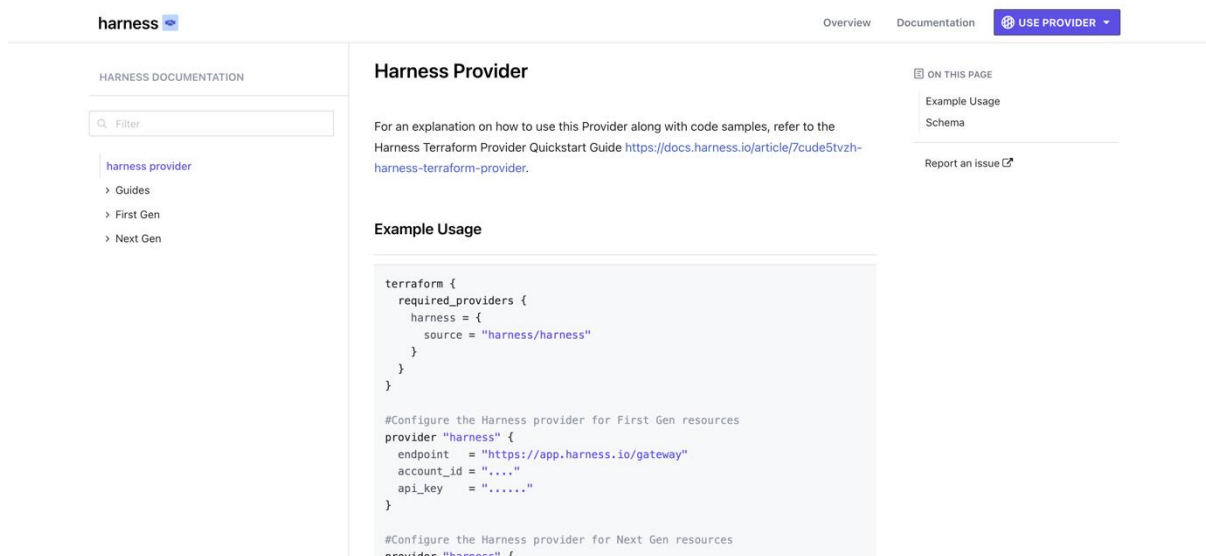


Fig 12

CHAPTER 3 - METHODOLOGY

3.1 – Creating Delegates

Harness Delegate is a service that you run in your local network or virtual private cloud (VPC) to connect your artifacts, infrastructure, collaboration, verification, and other providers with Harness Manager. The first time you connect Harness to a third-party resource, Harness Delegate is installed in your target infrastructure, such as a Kubernetes cluster. After the delegate is installed, you can connect to third-party resources. The delegate performs all operations, including deployment and integration.

Here are some of the key benefits of using Harness Delegate:

- Simplicity: Harness Delegate is easy to install and configure.
- Scalability: Harness Delegate can be scaled to support many third-party resources.
- Security: Harness Delegate is secure and can be configured to meet your specific security requirements.
- Reliability: Harness Delegate is reliable and can be used to ensure that your deployments are successful.
- Extensibility: Harness Delegate can be extended to support new third-party resources.

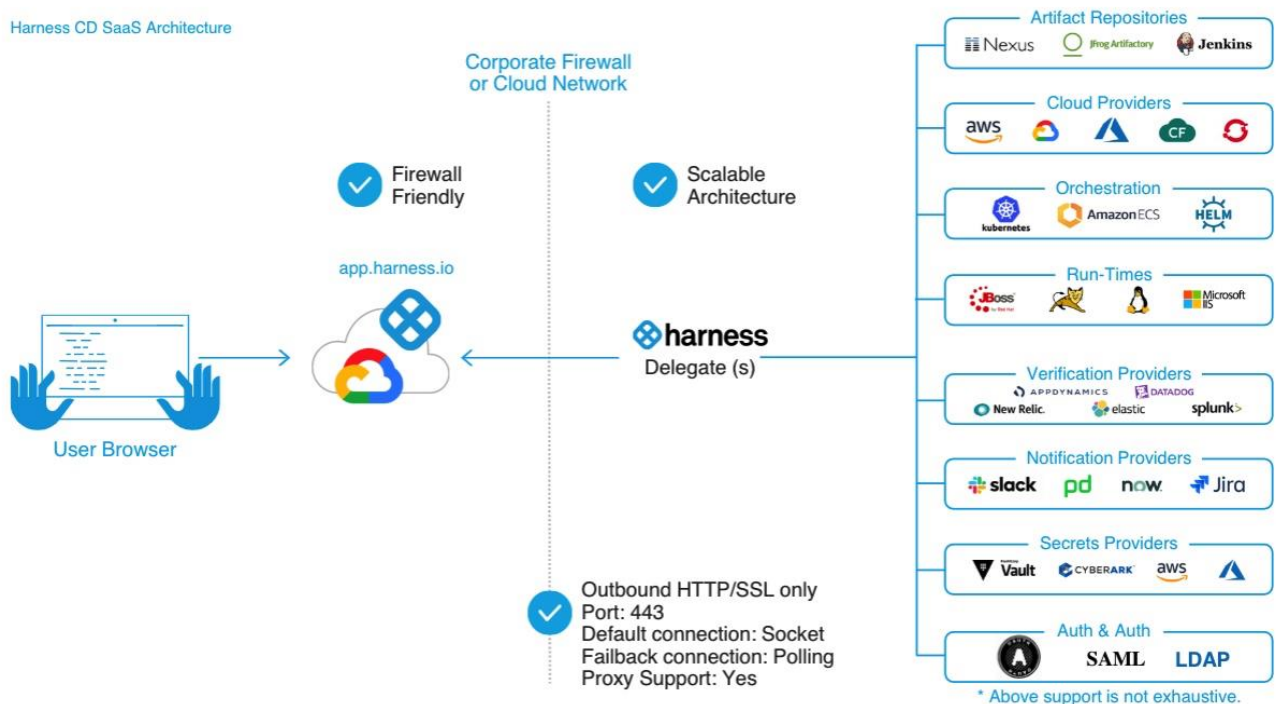


Fig 13

Steps to create a Delegate:

1. Log in to your Harness account.
2. Go to the Account Resources page.
3. Click on Delegates.
4. Click on Install Delegate.
5. Click on the Kubernetes manifest tab.
6. Enter a name for your delegate.
7. Click on the Download YAML button.
8. Execute the command that is written in the install delegate window by connecting to your Kubernetes cluster.

Once you have completed these steps, your Harness delegate will be installed and configured. You can then use the delegate to connect to your Harness environment and perform operations such as deployments and integrations.

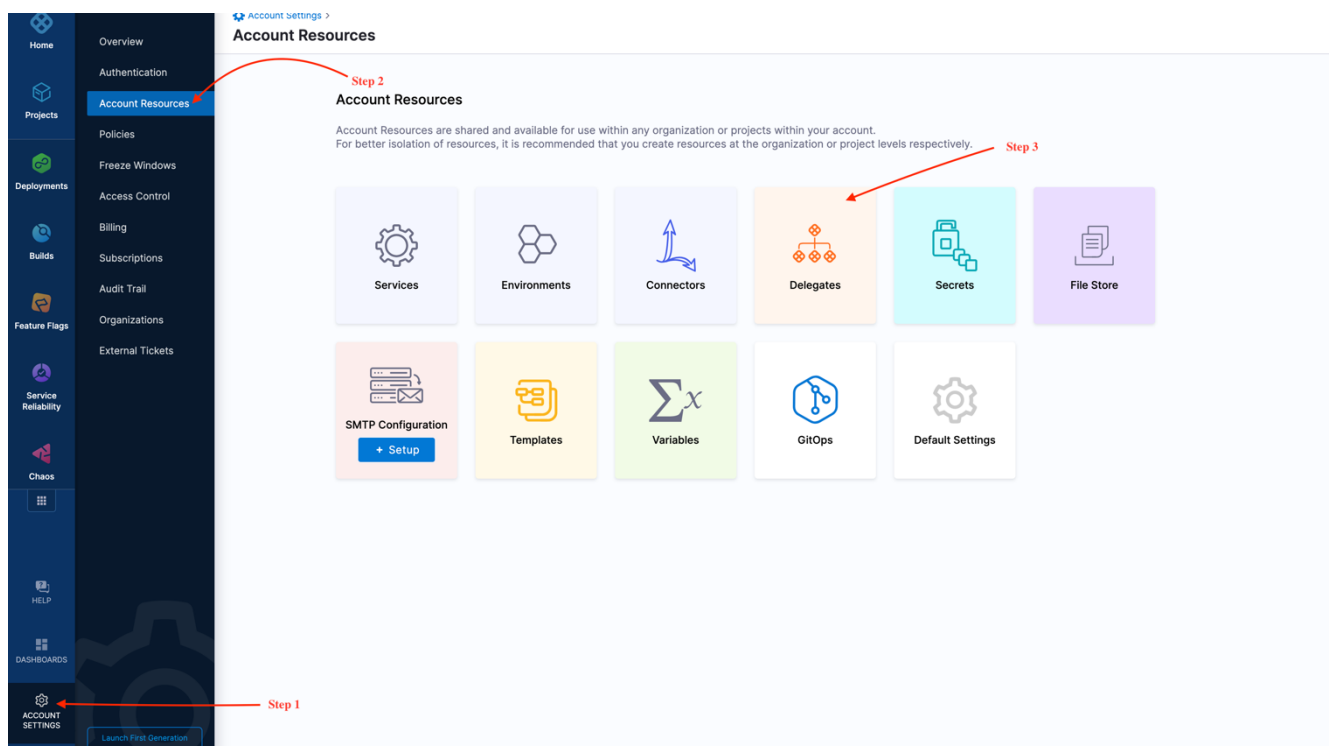


Fig 14

[platform-team] Project Delegates

+ New Delegate Step 4

Search No Filter Saved

Total: 5 Status (A→Z, 0→9)

DELEGATE	TAGS	VERSION	INSTANCE STATUS	LAST HEARTBEAT	CONNECTIVITY STATUS
pi-play-automation-cluster _pi_play_automation_cluster	AUTO UPGRADE: ON pi-play-automation-cluster	23.06.79707	Expiring in 2 months	51 seconds ago	Connected

Fig 15

New Delegate

[Switch back to old delegate install experience](#)

Delegates are worker processes that run on your infrastructure to execute tasks on behalf of the Harness platform. [Learn more about Delegates](#), [watch Video](#)

Select where you want to install your Delegate

☒ Kubernetes ☐ Docker

Install your Delegate

Step 5

Delegate Name
Step 6

YAML file options

Step 7

This YAML has all configuration variables pre-populated with values specific to your Harness account. Additionally, the auto upgrade configuration is set to ON so that the delegate version installed will always remain in sync with the version available on the Harness Manager.

Copy the YAML to a machine with kubectl installed and with access to your Kubernetes cluster. Run the following command to install the Harness Delegate in your Kubernetes Cluster.

```
kubectl apply -f harness-delegate.yaml Step 8
```

[Delegate sizing guide - modify the replica count to run specific number of parallel deployments/builds](#)

For Advanced configuration options, refer to [documentation](#)

Verify Delegate connection to Harness Manager

Verify once you have completed the steps above to make sure its installed properly

Fig 16

Once the delegate is connected, you can view some details about it in the Harness UI. These details include:

- The version of the delegate that is being used.
- The heartbeat of the delegate, which ensures that the delegate is healthy.
- The status of the delegate, which indicates whether the delegate is online or offline.

3.2 – Writing Test Case for API Automation using RestAssured

TestNG and RestAssured are two popular tools used for API automation testing. TestNG is a Java-based testing framework that allows you to write and run tests in a structured way. RestAssured is a Java library that allows you to make HTTP requests to APIs and verify the responses.

Together, TestNG and RestAssured can be used to automate a wide variety of API tests, including:


- Functional testing: This type of testing ensures that the API is performing as expected. For example, you can use TestNG and RestAssured to test that a specific endpoint returns the expected response.
- Performance testing: This type of testing ensures that the API can handle a given load. For example, you can use TestNG and RestAssured to test that the API can handle a certain number of requests per second.
- Security testing: This type of testing ensures that the API is secure from attack. For example, you can use TestNG and RestAssured to test that the API is not vulnerable to common attacks such as cross-site scripting (XSS) and SQL injection.

Testing is an important part of the software development lifecycle. It helps to ensure that software is reliable, secure, and meets the needs of users. By using TestNG and RestAssured, you can automate API testing and save time and effort.

Here are some of the benefits of using TestNG and RestAssured for API automation testing:

- Reusability: TestNG and RestAssured are both reusable frameworks, which means that you can use the same tests for different APIs. This can save you time and effort in the long run.
- Scalability: TestNG and RestAssured can be scaled to support a large number of APIs. This is important if you are developing a large-scale application.
- Flexibility: TestNG and RestAssured are flexible frameworks that can be used to test a wide variety of APIs. This makes them a good choice for a variety of projects.
- Easy to use: TestNG and RestAssured are both easy to use frameworks. This makes them a good choice for beginners and experienced developers alike.

This suite file consists of one class that has different Test cases to execute. This suite file has a test tag that takes various inputs for our case we have given name which specifies the Test name for reference purpose and preserve-order that gives you the assurance that the order in which classes would be mentioned, they would be executed in same manner.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE suite SYSTEM "http://testng.org/testng-1.0.dtd">
3 <suite name="plng-account-level-2fa">
4
5     <test verbose="2" name="plng-account-level-2fa" preserve-order="true">
6         <classes>
7             <class name="io.harness.rest.plng.Authentications.AccountLevel2FATest"></class>
8         </classes>
9     </test>
10 </suite>
11
```

Fig 17

3.2.1 Showing how Testcases are written in Harness.

The `@BeforeTest` annotation specifies that the method annotated with it will be run before any of the test cases in the class are run. This is useful for setting up any shared resources that will be used by the test cases, such as database connections or file handles.

The `@Test` annotation specifies that the method annotated with it is a test case. When the test suite is run, each test case will be executed in a separate thread. This ensures that the results of one test case do not affect the results of another test case.

The `@AfterTest` annotation specifies that the method annotated with it will be run after all of the test cases in the class have been run. This is useful for cleaning up any shared resources that were used by the test cases, such as closing database connections or deleting temporary files.

The test case that you described first fetches the original account details so that they can be restored after the test is complete. Then, it changes the name of the account to a random name that is prefixed with "pl_auto_name_" and followed by three random letters. Finally, it

verifies that the name change was successful and reverts the account name back to the original value.

This test case ensures that the account name can be changed successfully and that the original name can be restored. It is a good practice to include `@BeforeTest`, `@Test`, and `@AfterTest` annotations in your test cases to ensure that they are executed in a consistent and predictable order.

```
1 public class AccountNameChangeTest extends AbstractTest {
2     String PL_ACCOUNT_PREFIX = "pl_auto_name_";
3     AccountNameHelper accountNameHelper = new AccountNameHelper();
4     String actualName = null;
5
6     @BeforeTest(alwaysRun = true)
7     public void actualName() {
8         RequestSpecification requestSpecification = GenericRequestBuilder.getRequestSpecificationObjectNG();
9         Response response = accountNameHelper.getAccountDetails(requestSpecification, defaultAccountId);
10        Assert.assertEquals(response.statusCode(), HttpStatus.SC_OK, "Account Details Should have been fetched");
11        actualName = response.jsonPath().getString("data.name");
12    }
13    @Test
14    public void accountNameChangeTest() {
15        RequestSpecification requestSpecification = GenericRequestBuilder.getRequestSpecificationObjectNG();
16        String newName = commonHelper.createRandomName(PL_ACCOUNT_PREFIX, 5);
17
18        Response response = accountNameHelper.updateName(requestSpecification, defaultAccountId, newName);
19        Assert.assertEquals(response.statusCode(), HttpStatus.SC_OK, "Name Change should happen");
20    }
21
22    @AfterTest(alwaysRun = true)
23    public void revertToOriginalName() {
24        RequestSpecification requestSpecification = GenericRequestBuilder.getRequestSpecificationObjectNG();
25        Response response = accountNameHelper.updateName(requestSpecification, defaultAccountId, actualName);
26        Assert.assertEquals(
27            response.statusCode(), HttpStatus.SC_OK, "Name change should have been reverted to original name");
28    }
29 }
30
```

Fig 18

3.2.2 How tests are executed in CI pipeline.

This is the pipeline that we use for running tests of authentication related test cases.

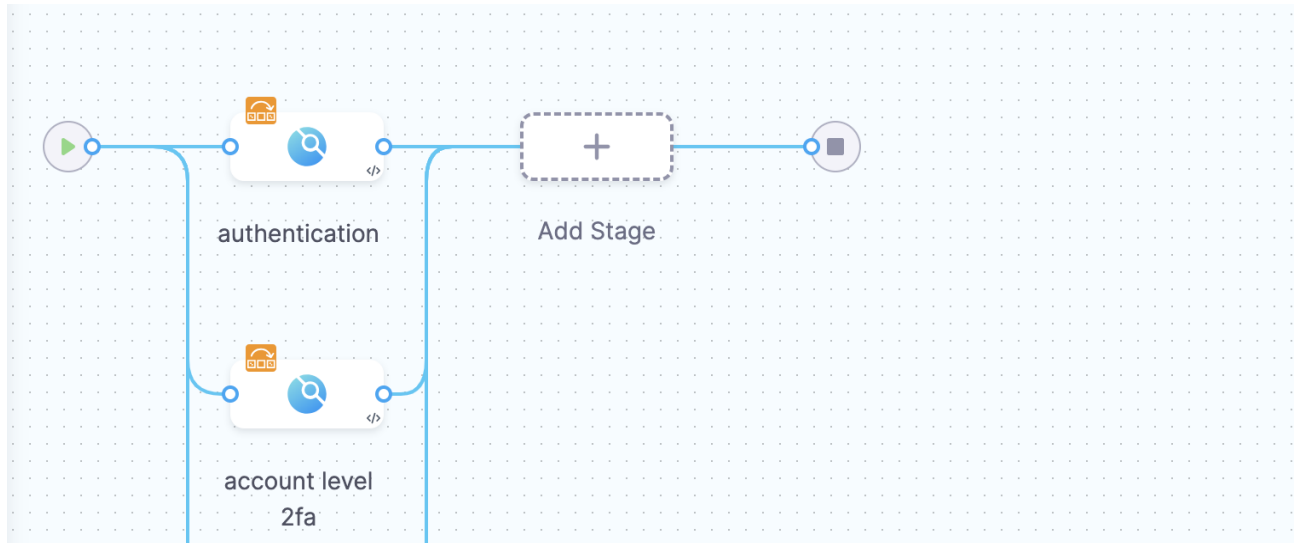


Fig 19

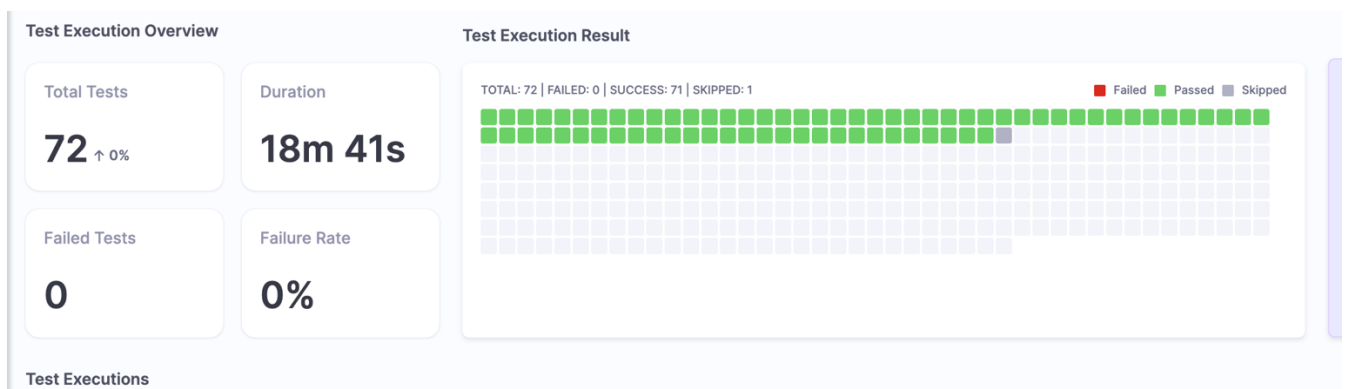


Fig 20

The test results tab gives us complete overview about the total number of tests that were executed and along with that the time it takes to execute all tests. It also gives me the classes along with the methods that were executed but that can't be shared over here as those are confidential information.

3.3 – Terraform.

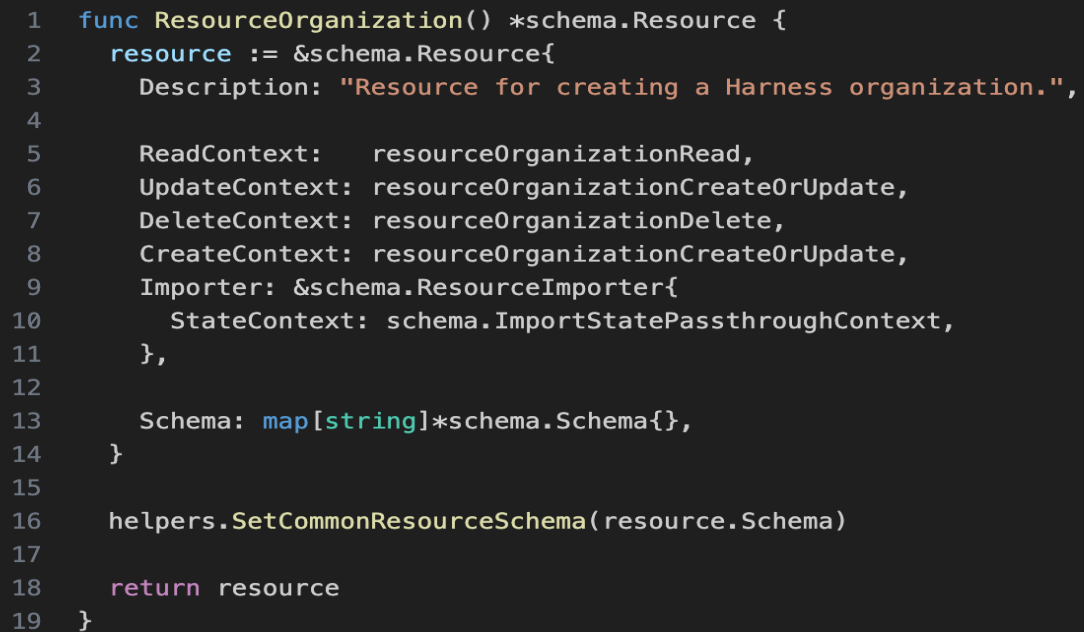
The Harness provider for Terraform is a set of resources that allow you to manage Harness resources from within Terraform. The provider includes resources for managing organizations, pipelines, applications, and more.

The Harness provider for Terraform uses the following schema for resource creation:

- ReadContext: This is a GET call that is used to get the data for a particular resource. For example, the ReadContext for a harness_organization resource would be used to get the data for an organization.
- UpdateContext: This is a POST call that is used to update the data for a particular resource. For example, the UpdateContext for a harness_organization resource would be used to update the name or description of an organization.
- DeleteContext: This is a DELETE call that is used to delete a particular resource. For example, the DeleteContext for a harness_organization resource would be used to delete an organization.
- CreateContext: This is a POST call that is used to create a new resource. For example, the CreateContext for a harness_organization resource would be used to create a new organization.
- Importer: This is a way of creating a resource by using the command line terminal. For example, the Importer for a harness_organization resource would be used to create a new organization using the harness create organization command.

These contexts are used to create the resources that are defined in your Terraform configuration file. For example, if you have a resource block that defines a harness_organization resource, the ReadContext for that resource would be used to get the data for the organization, the UpdateContext would be used to update the data for the organization, the DeleteContext would be used to delete the organization, and the CreateContext would be used to create a new organization.

The Importer context is used to create resources that are not defined in your Terraform configuration file. For example, if you want to create a new organization using the command line terminal, you will use the Importer context for the harness_organization resource.



```

1  func ResourceOrganization() *schema.Resource {
2      resource := &schema.Resource{
3          Description: "Resource for creating a Harness organization.",
4
5          ReadContext:    resourceOrganizationRead,
6          UpdateContext:  resourceOrganizationCreateOrUpdate,
7          DeleteContext:  resourceOrganizationDelete,
8          CreateContext:  resourceOrganizationCreateOrUpdate,
9          Importer: &schema.ResourceImporter{
10             StateContext: schema.ImportStatePassthroughContext,
11         },
12
13         Schema: map[string]*schema.Schema{},
14     }
15
16     helpers.SetCommonResourceSchema(resource.Schema)
17
18     return resource
19 }

```

Fig 21

3.3.1 – Testcases for a Terraform resource.

The TestAccResourceOrganization test function creates a new organization resource in Harness, and then updates the name of the organization. The function then verifies that the organization was created and updated successfully.

The function takes the following parameters:

name: The name of the test.

id: The ID of the organization.

updatedName: The updated name of the organization.

resourceName: The name of the resource in Terraform.

The function first calls the acctest.TestAccPreCheck function to make sure that the Harness provider is installed and configured correctly. Then, it calls the ProviderFactories function to get the provider factories for the Harness provider.

Next, the function calls the CheckDestroy function to verify that the organization resource can be destroyed successfully. Then, it calls the Steps function to execute the test steps.

The Steps function calls the testAccResourceOrganization function twice. The first time, it passes the original name of the organization. The second time, it passes the updated name of the organization.

The testAccResourceOrganization function creates a new organization resource in Harness with the specified name. It then returns the ID of the organization.

The Check function verifies that the organization resource was created successfully. It does this by checking the id and name attributes of the resource.

The ImportState and ImportStateVerify parameters are used to test the import functionality of the organization resource.

Overall, the TestAccResourceOrganization test function verifies that the organization resource can be created, updated, and imported successfully.

```
1 func TestAccResourceOrganization(t *testing.T) {
2
3     name := t.Name()
4     id := fmt.Sprintf("%s_%s", name, utils.RandStringBytes(5))
5     updatedName := fmt.Sprintf("%s_updated", name)
6     resourceName := "harness_platform_organization.test"
7
8     resource.UnitTest(t, resource.TestCase{
9         PreCheck: func() { acctest.TestAccPreCheck(t) },
10        ProviderFactories: acctest.ProviderFactories,
11        CheckDestroy: testAccOrganizationDestroy(resourceName),
12        Steps: []resource.TestStep{
13            {
14                Config: testAccResourceOrganization(id, name),
15                Check: resource.ComposeTestCheckFunc(
16                    resource.TestCheckResourceAttr(resourceName, "id", id),
17                    resource.TestCheckResourceAttr(resourceName, "name", name),
18                ),
19            },
20            {
21                Config: testAccResourceOrganization(id, updatedName),
22                Check: resource.ComposeTestCheckFunc(
23                    resource.TestCheckResourceAttr(resourceName, "id", id),
24                    resource.TestCheckResourceAttr(resourceName, "name", updatedName),
25                ),
26            },
27            {
28                ResourceName: resourceName,
29                ImportState: true,
30                ImportStateVerify: true,
31            },
32        },
33    })
34 }
```

Fig 22

3.3.2 – Writing Terraform Code

As a common person we would be not much worried about what is going in the backend but rather than, how can we do try this out?

So initially you need to install Terraform in your machine, post that you can start writing terraform code and save the file extension as .hcl which represents that it is a terraform file

Sharing one small example of creating a new organization in Harness without going to the Harness UI and without clicking something over there. As you can see below that only these 2 lines are doing my whole job of creating new organization.



```
1 resource "harness_platform_organization" "test" {  
2     identifier = "identifier"  
3     name       = "soskcdx"  
4 }
```

Fig 23

CHAPTER 4 – RESULT ANALYSIS

After working on a variety of languages, tools, and software, I have received compliments from my team and customers for my on-call support. I have also significantly boosted my GitHub profile by contributing to open-source repositories. The biggest takeaway from my internship is the industrial knowledge I have gained. My mentors have been incredibly helpful and supportive, and I am grateful for their guidance.

Initially, it was challenging to keep up with the technically sound workforce. However, I was able to multi-task and work past the stipulated time to understand the backend of the technology. I am glad that I was given this opportunity as it has allowed me to develop myself through the various challenges presented to me.

Here are some key points that I have learned from my internship:

- Never hold yourself back. Give your best in everything you do, as anything less is a disservice to yourself.
- Build good relationships with your colleagues. They will make working in a competitive environment more enjoyable.
- Don't be afraid to ask for help. Your managers are there to support you, so don't hesitate to reach out if you need assistance.
- Share your learnings with others .

Agenda for tomorrow's **Bi-weekly team sync up to showcase the work and share learning**

1. Project copy feature by **@Aditya Kashyap**.
2. Infra evolution integration by **@Nataraja M**
3. Just in Time provisioning by **@Vikas Maddukuri** and **@Mayank Verma**
4. Favourites framework by **@Avinash Madhwani**, **@Boopesh** **@Richa**



Fig 24



Fig 25

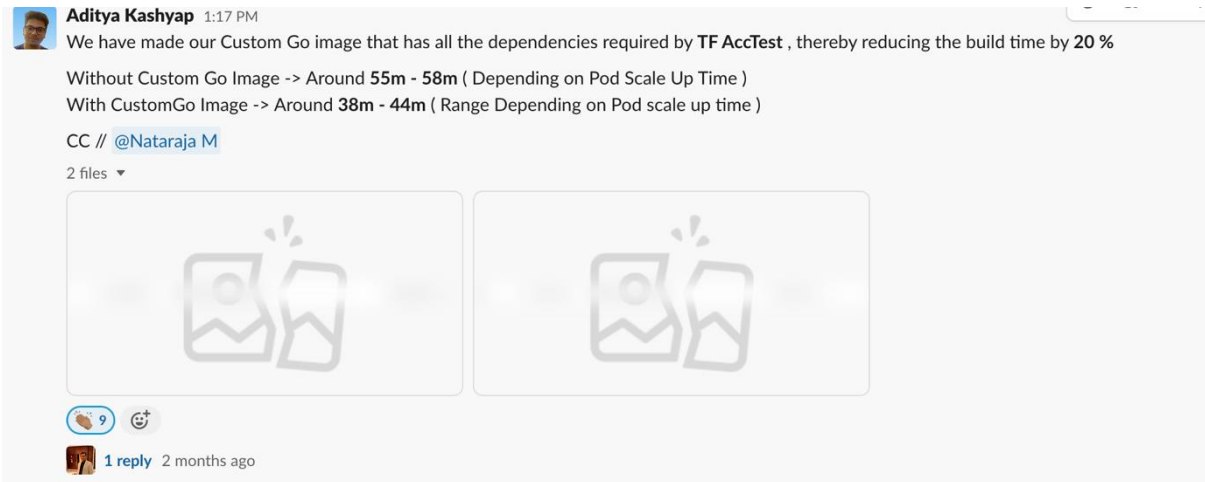


Fig 26

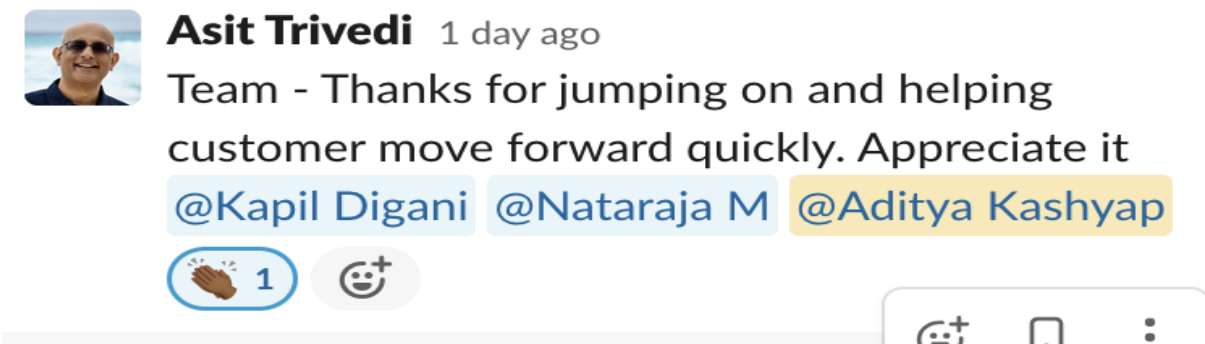


Fig 27

Clear current search query, filters, and sorts

<input type="checkbox"/>	1 Open	25 Closed	Author	Label	Projects	Milestones	Reviews	Assignee	Sort
<input type="checkbox"/>	fix: [PL-38926]: Added Missing Comma	#50254 by adityaar24 was merged 15 hours ago · Review required							1 of 3 tasks
<input type="checkbox"/>	fix: [PL-39304]: Fix for Variable API schema	#50206 by adityaar24 was merged 2 days ago · Approved							2 of 3 tasks
<input type="checkbox"/>	fix: [PL-39304]: HotFix for Variable API schema	#50202 by adityaar24 was closed 2 days ago · Review required							3 tasks
<input type="checkbox"/>	fix: [PL-39304]: Fix for Variable API schema	#50200 by adityaar24 was merged 2 days ago · Approved							2 of 3 tasks
<input type="checkbox"/>	feat: [PL-38926]: Added More entities to copy script	#50010 by adityaar24 was merged 4 days ago · Approved							3 tasks
<input type="checkbox"/>	fix: [PL-39304]: Fixed the API documentation of variables to update the type	#49864 by adityaar24 was merged 3 days ago · Approved							2 of 3 tasks
<input type="checkbox"/>	fix: [PL-39304]: Fixed the API documentation of variables to update the type	#49787 by adityaar24 was merged 2 weeks ago · Approved							
<input type="checkbox"/>	fix: [PL-29458]: Removed Query Param Account Identifier From the API	#49743 by adityaar24 was merged last week · Approved							3 tasks
<input type="checkbox"/>	feat: [PL-38926]: Added More entities to copy script								

Fig 28

Filters		is:pr author:adiyaar24 is:closed	Labels 19	Milestones 0	New pull request
Clear current search query, filters, and sorts					
<input type="checkbox"/>	0 Open	37 Closed	Author	Label	Projects
<input type="checkbox"/>	Auth test Fixed for SAML ✓				
	#5465 by adiyaar24 was merged 2 weeks ago · Review required				
<input type="checkbox"/>	fix: [PL 39262] - Test Updation in respect to new changes ✓				1
	#5318 by adiyaar24 was merged on Jun 5 · Review required				
<input type="checkbox"/>	PL-38960 Editing Name for the account and Validating ✓				1
	#5216 by adiyaar24 was merged on May 23 · Review required				
<input type="checkbox"/>	PL-38959 Added Test Case for Trigger Validation ✓				1
	#5215 by adiyaar24 was merged on May 23 · Review required				
<input type="checkbox"/>	Coverage Improvements for Google SM With Delegate ✓				1
	#5108 by adiyaar24 was merged on May 11 · Review required				
<input type="checkbox"/>	Automation Added For Google SM with Delegates ✓				1
	#5104 by adiyaar24 was merged on May 10 · Review required				
<input type="checkbox"/>	Added DockerFile for Custom Go Image ✓				
	#5092 by adiyaar24 was merged on May 9 · Review required				
<input type="checkbox"/>	Secret TOTP Updated for 2FA Account ✓				
	#4838 by adiyaar24 was merged on Apr 6 · Review required				
<input type="checkbox"/>	Added GCP In Pre Requisite Class ✓				
	#4767 by adiyaar24 was merged on Mar 28 · Review required				
<input type="checkbox"/>	Suite File Changes ✓				
	#4748 by adiyaar24 was closed on Apr 10 · Review required				
<input type="checkbox"/>	2Fa at User Level Fixed ✓				

Fig 29

Filters		is:pr author:adiyaar24 is:closed	Labels 19	Milestones 0	New pull request
Clear current search query, filters, and sorts					
<input type="checkbox"/>	1 Open	31 Closed	Author	Label	Projects
<input type="checkbox"/>	ER: [PL-39021]: Updated the documentation for Resource Group ✓				3
	#580 by adiyaar24 was merged 3 weeks ago · Approved				
<input type="checkbox"/>	Revert "feat: [PL-38818] Added Read Support for all connectors by name" ✓				1
	#552 by adiyaar24 was merged on May 25 · Approved				
<input type="checkbox"/>	feat: [PL-38818] Added Read Support for all connectors by name ✓				4
	#541 by adiyaar24 was merged on May 17 · Review required				
<input type="checkbox"/>	PL-38654 Secret File Not Getting Created with Tags separated by comma ✓				2
	#534 by adiyaar24 was merged on May 11 · Review required				
<input type="checkbox"/>	Filter Tests Added ✓				2
	#530 by adiyaar24 was merged on May 5 · Review required				
<input type="checkbox"/>	Updation of SSH Tests ✓				1
	#519 by adiyaar24 was closed on May 5 · Review required				
<input type="checkbox"/>	Updated Test For Connectors ✓				6
	#509 by adiyaar24 was merged on May 11 · Review required				
<input type="checkbox"/>	[SPG - 2584] UserGroup & Organisation Fix ✓				
	#507 by adiyaar24 was merged on Apr 7 · Review required				
<input type="checkbox"/>	Doc Updation for Org got skipped ✓				

Fig 30

CHAPTER 5 – FUTURE SCOPE OF WORK

5.1 The Future of Harness

Harness has the potential to disrupt the market with its user-friendly interface and ease of use. It also provides a wide range of external services out of the box, which saves companies the time and effort of having to maintain their own infrastructure.

In the coming year, we will see significant advances in developer efficiency and experience. As companies of all sizes look for ways to weather the twin challenges of the talent shortage and economic recession, they will increasingly invest in automation and process improvements. This will lead to a proliferation of internal developer platforms, a new focus on key development metrics, and much greater scrutiny of cloud computing costs. All of these changes will benefit developers and businesses alike.

The future of DevSecOps will move towards platform engineering and will rely heavily on artificial intelligence (AI). AI will be used to optimize build execution times, increase efficiency, and provide security. We will also see more automation in the form of bots that can run runbooks and fix production issues.

GitOps will become the standard for infrastructure as code (IaC) and will have a strong foothold in application CI/CD. IaC will also adapt to the serverless space, gaining even more traction. As more solutions are developed to address containerization and virtualization issues, developers will have an easier time building and deploying applications.

5.2 My Future at Harness

I will continue to contribute to the backend development of Harness, including fixing bugs, creating new APIs, and making sure that the product is backward compatible. I will also be working on the Terraform side of the business. Terraform is becoming increasingly important as more companies adopt complex infrastructure. I will be working on adding more support for Terraform and on stabilizing the product for my team.

I am excited about the future of Harness, and I am confident that I can make a significant contribution to the company's success. I am also looking forward to working on Terraform, which is a technology that I am passionate about. I believe that Harness and Terraform are well-positioned to disrupt the market and I am excited to be a part of that journey.

REFERENCES

1. Harness Website (<https://harness.io>)
2. Developer Hub Harness (<https://developer.harness.io>)

PROJECT DETAILS

<i>Student Details</i>			
Student Name	Aditya Kashyap		
Register Number	199202111	Section / Roll No	ECE – A
Email Address	Adityakashyap3190@gmail.com	Phone No (M)	8851590289
<i>Project Details</i>			
Project Title	Software Engineer Harness		
Project Duration	6 Months	Date of reporting	16 Jan
<i>Organization Details</i>			
Organization Name	Harness		
Full postal address with pin code	24 th Cross Road, HSR Sector 2, Near Garden layout - 560102		
Website address	https://harness.io/		
<i>Supervisor Details</i>			
Supervisor Name	Nataraja M		
Designation	QA Manager		
Full contact address with pin code			
Email address	nataraja@harness.io	Phone No (M)	
<i>Internal Guide Details</i>			
Faculty Name	Dr Tejpal		
Full contact address with pin code			
Email address	Tej.pal@jaipur.manipal.edu	Phone No (M)	8079076316

