### **APP DEPLOY AUTOMATION WORK FLOW SOP**

#### **Document Information**

* **Document Title: SOP**
* **Prepared By: ADA Team**

Table of Contents

[**APP DEPLOY AUTOMATION WORK FLOW SOP** 1](#_Toc170394404)

[**1. Introduction** 2](#_Toc170394405)

[**2.Pipeline creation in Harness:** 2](#_Toc170394406)

[**2.1 Project:** 3](#_Toc170394407)

[**2.2 Environments:** 3](#_Toc170394408)

[**2.3 Services:** 4](#_Toc170394409)

[**2.4 Pipelines:** 5](#_Toc170394410)

[**3.Stages:** 5](#_Toc170394411)

[**4.Connectors:** 6](#_Toc170394412)

[**5.Delegates**: 7](#_Toc170394413)

[**6.Credentials and permissions:** 7](#_Toc170394414)

[**7.Variables:** 7](#_Toc170394415)

[**8.Triggers:** 8](#_Toc170394416)

[**9.Advanced settings:** 8](#_Toc170394417)

[**10.Input sets and overlays:** 9](#_Toc170394418)

[**11.Conditional Execution:** 9](#_Toc170394419)

[**12.Failure strategies:** 9](#_Toc170394420)

[**13.CD Overview:** 10](#_Toc170394421)

[**14.CloudBees/DCP Overview:** 10](#_Toc170394422)

[**15.Component & Component creation Process Creation - HIP Project:** 10](#_Toc170394423)

[**16.RPA Pipeline Deployment for QE and Prod and Add new Bot to the Pipeline:** 11](#_Toc170394424)

[**17.DCP Pipeline Overview:** 11](#_Toc170394425)

[**18.OCI Snapshot and  Inv Sync:** 11](#_Toc170394426)

[**19.BoDT and PW Change:** 11](#_Toc170394427)

[**20. Artifact Management** 12](#_Toc170394428)

[**21.Projects, Applications** 12](#_Toc170394429)

[**22.Environments, Pipelines** 13](#_Toc170394430)

[**23.Application Processes, Procedures** 14](#_Toc170394431)

[**Key URL’s mentioned :** 16](#_Toc170394432)

# **1. Introduction**

The purpose of this document is to provide a comprehensive guide for the App deploy Automation Team's daily activities. It outlines detailed procedures for managing pipeline creation in harness, Creating the components, Environment’s, Application Process and Procedures in Cloudbees. The document also includes creation Harness Pipelines.

# **2.Pipeline creation in Harness:**

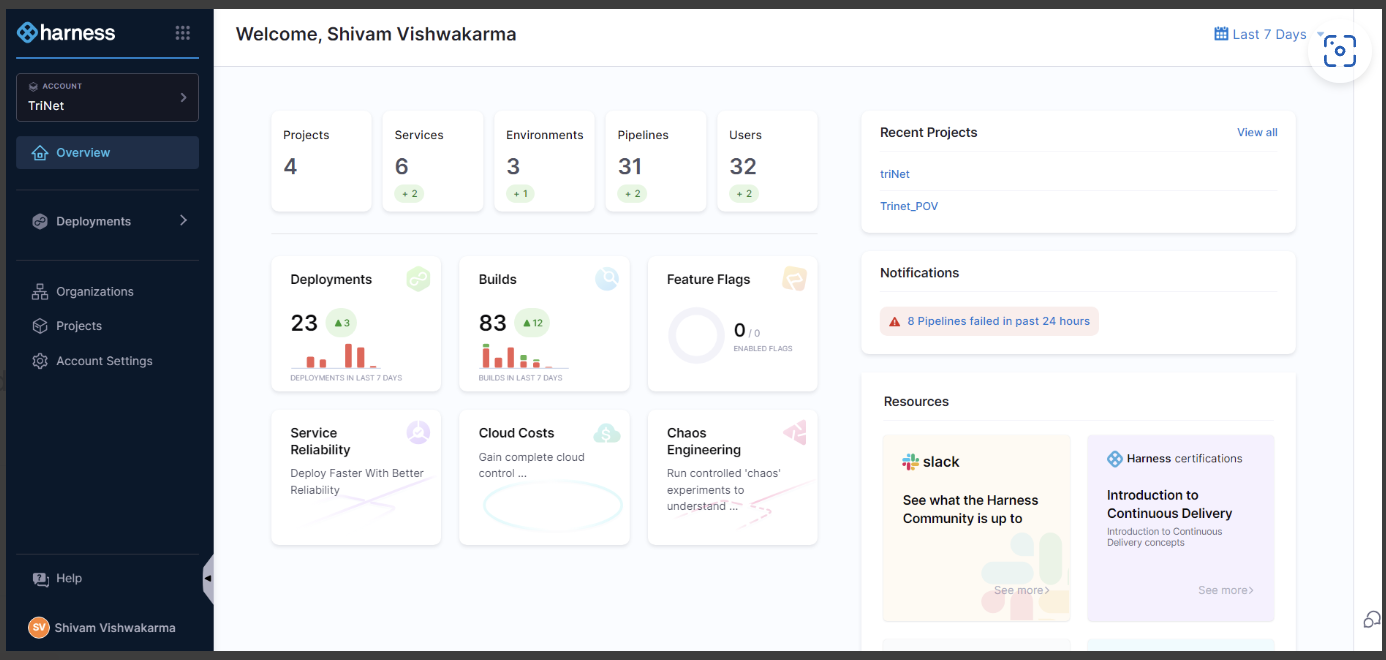
From Harness homepage you can directly navigate to -

* Projects
* Account settings
* Deployments
* Organizations

Also, you can see counts of below fields.

* Projects
* Services
* Environments
* Pipelines
* Deployments
* Builds

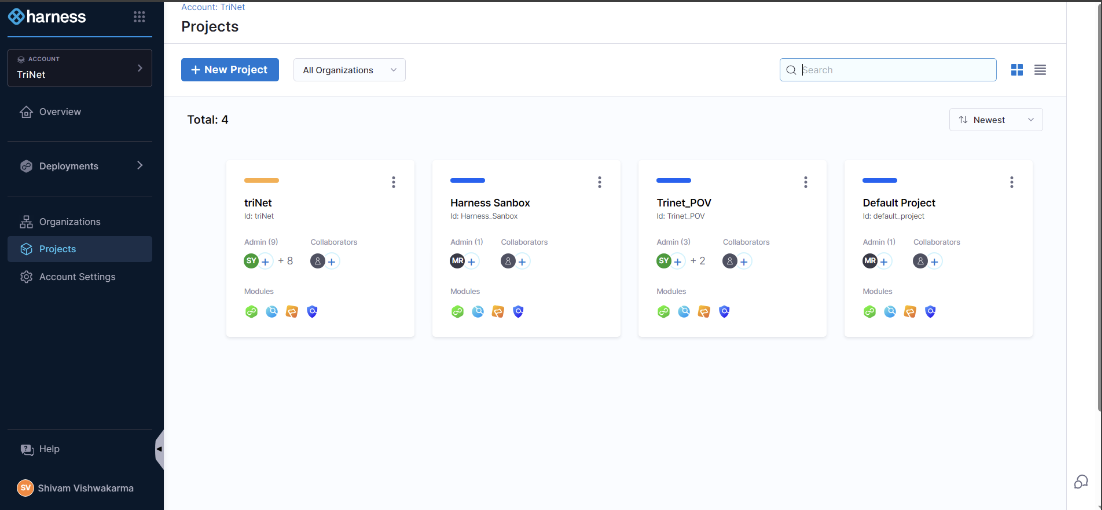
From top right you can use filter to see the count of builds and deployments (Organization level)



# **2.1 Project:**

A Harness Project contains Harness Pipelines, users, and resources that share the same goal. For example, a Project could represent a business unit, division, or simply a development project for an app.

You can create new project by simply clicking New Project.

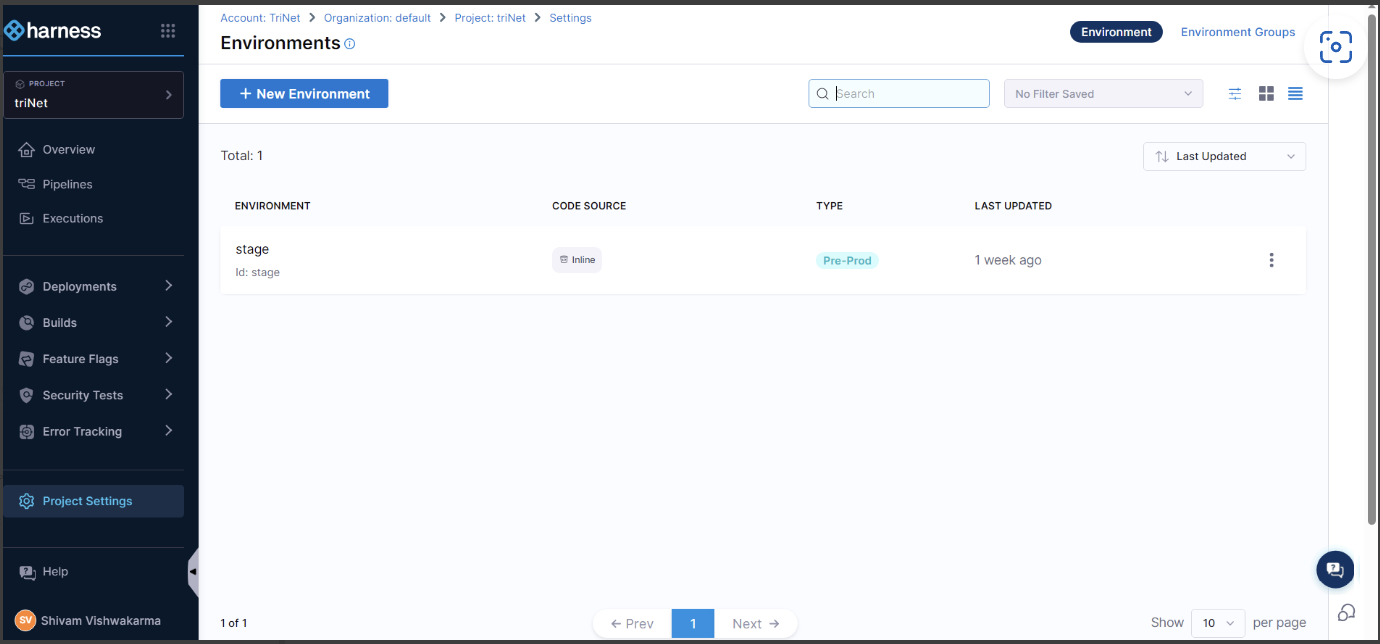


# **2.2 Environments:**

Environments represent your deployment targets (QA, Prod, etc). Each environment contains one or more **Infrastructure Definitions** that list your target clusters, hosts, namespaces, etc.

You can create environments from:

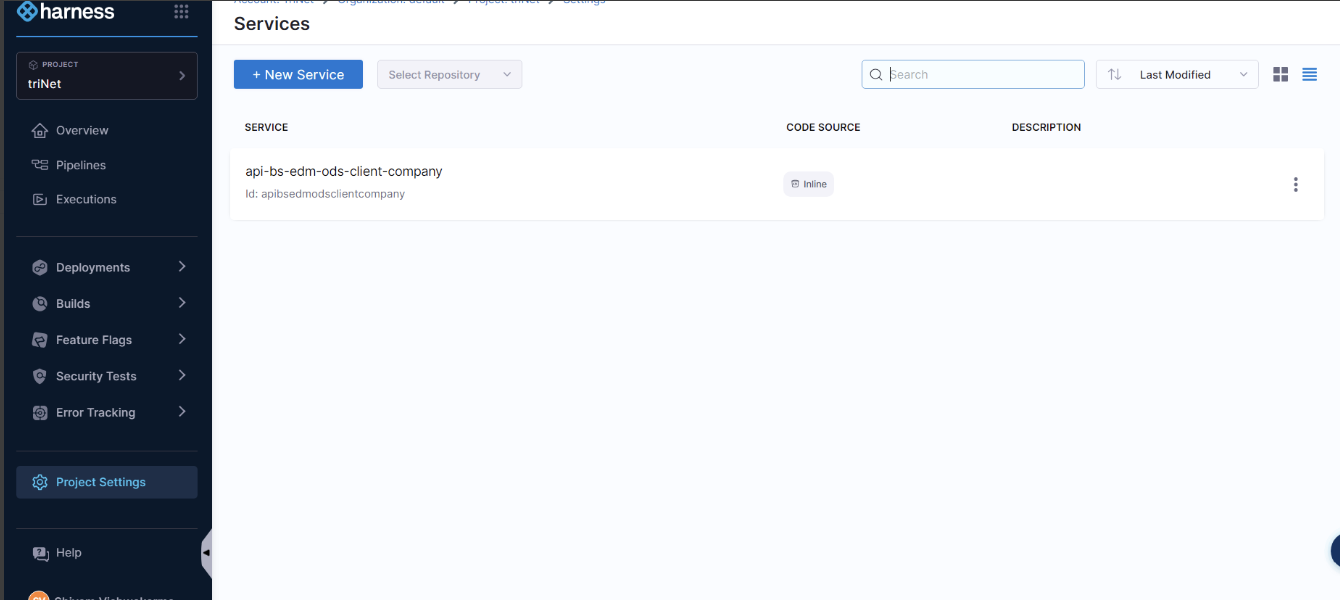
* An Account
* An organization
* Within a pipeline
* Outside a pipeline



# **2.3 Services:**

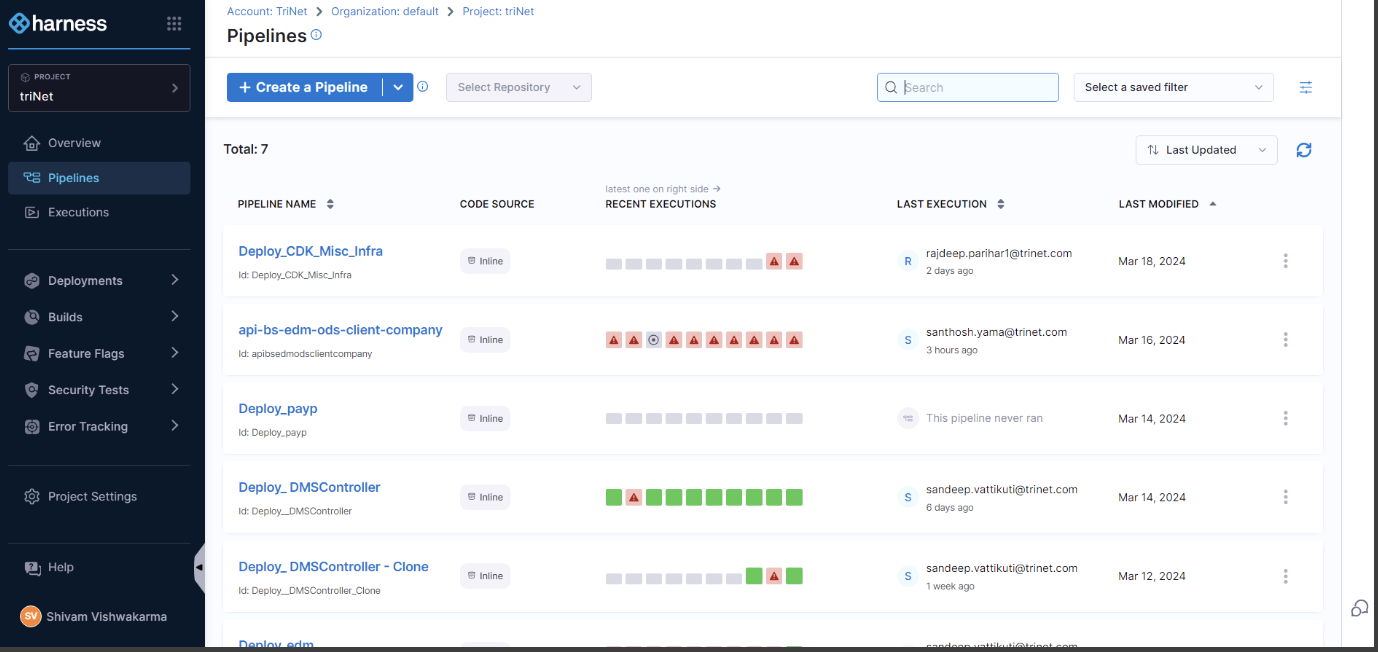
Services represent your microservices and other workloads. Each service contains a **Service Definition** that defines your deployment artifacts, manifests or specifications, configuration files, and service-specific variables.

You can create services from:

* An account
* An organization
* Within a pipeline
* Outside a pipeline
* 

# **2.4 Pipelines:**

A CD Pipeline is a series of Stages where each Stage deploys a Service to an Environment. It can perform many additional CD operations, including, but not limited to:

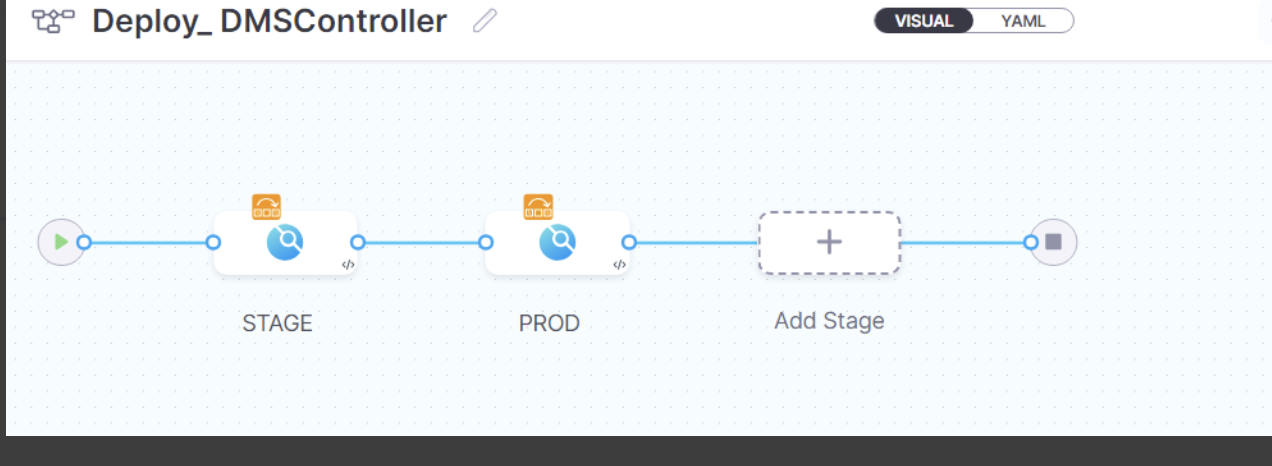
* Propagating Services and their definitions across Stages.
* Approvals using integrations like Jira and ServiceNow.
* Synchronizing stage deployments using barriers.
* Notifications. And many other operations.
* 

# **3.Stages:**

A CD Stage is a subset of a Pipeline that contains the logic to perform one major segment of the deployment process. Stages are based on the different milestones of your release process, such as Dev, QA, and prod releases, and approvals.

See the following:

* [Add a Stage](https://developer.harness.io/docs/platform/pipelines/add-a-stage) : [Add a stage | Harness Developer Hub](https://developer.harness.io/docs/platform/pipelines/add-a-stage/)
* [Add a Stage Template Quickstart](https://developer.harness.io/docs/platform/templates/add-a-stage-template) : [Create a stage template | Harness Developer Hub](https://developer.harness.io/docs/platform/templates/add-a-stage-template/)
* [Stage and Step Conditional Execution Settings](https://developer.harness.io/docs/platform/pipelines/step-skip-condition-settings) : [Define conditional executions for stages and steps | Harness Developer Hub](https://developer.harness.io/docs/platform/pipelines/step-skip-condition-settings/)



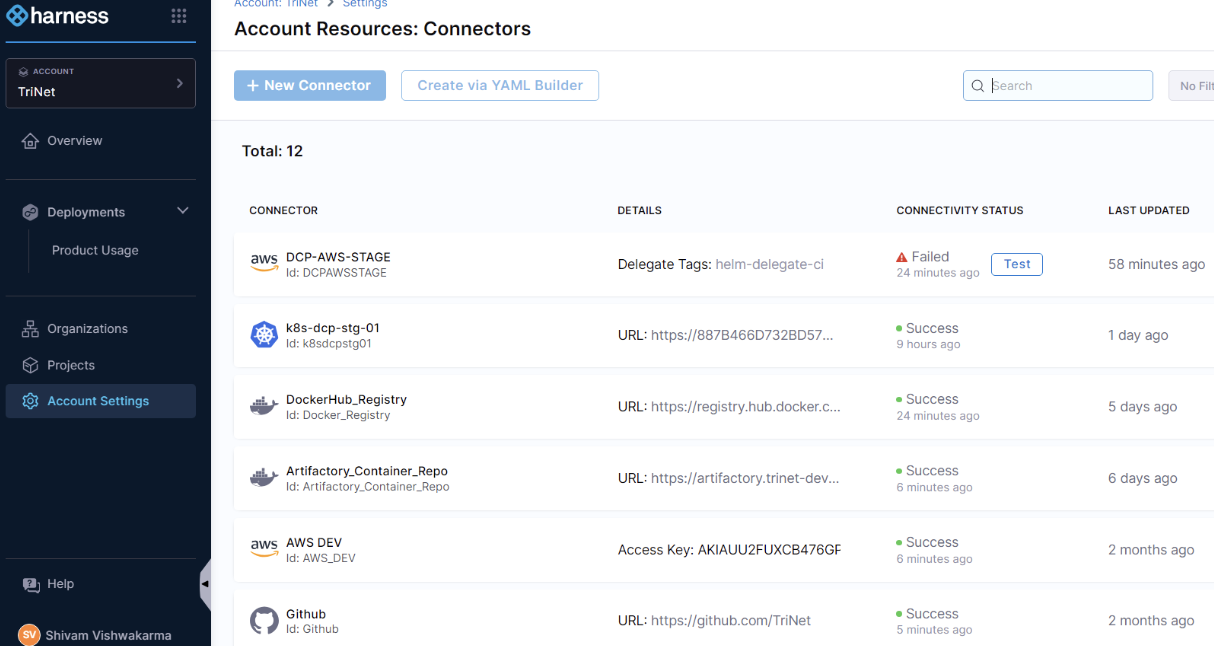
# **4.Connectors:**

Connectors contain the information necessary to integrate and work with 3rd party tools such as Git providers, artifact repos, and target infrastructure.

Harness uses Connectors at Pipeline runtime to authenticate and perform operations with a 3rd party tool.

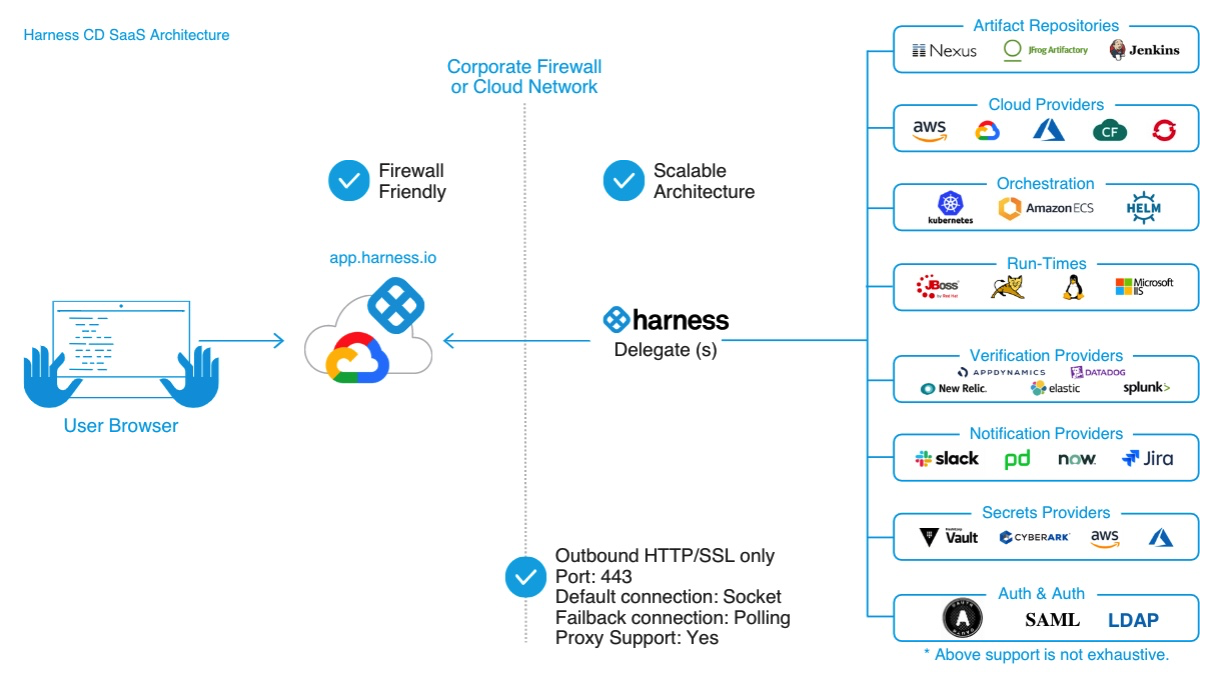
For example, go to:

* [Add a Kubernetes cluster connector | Harness Developer Hub](https://developer.harness.io/docs/platform/connectors/cloud-providers/add-a-kubernetes-cluster-connector/)
* [Docker Connector Settings Reference | Harness Developer Hub](https://developer.harness.io/docs/platform/connectors/cloud-providers/ref-cloud-providers/docker-registry-connector-settings-reference/)
* [Git connector settings reference | Harness Developer Hub](https://developer.harness.io/docs/platform/connectors/code-repositories/ref-source-repo-provider/git-connector-settings-reference/)



# **5.Delegates**:

The Harness Delegate is a software service you install in your environment that connects to the Harness Manager and performs tasks using your container orchestration platforms, artifact repositories, monitoring systems, etc.



# **6.Credentials and permissions:**

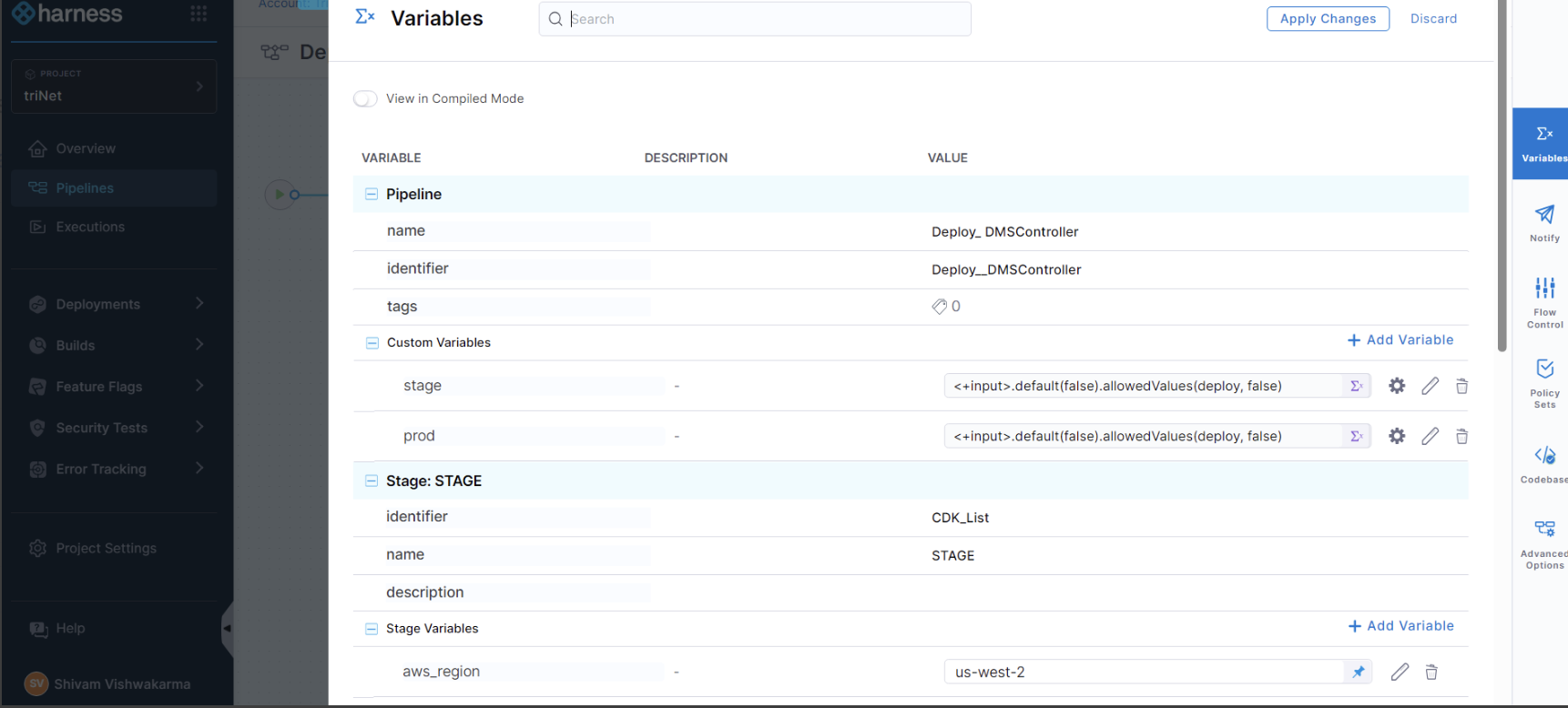
The Delegate uses the credentials set up in the Connectors used by the Pipeline to perform deployment tasks.

The Delegate also needs permissions in the target environment to execute deployment tasks. These permissions are granted in the Delegate config file or the environment account you use when installing the Delegate.

# **7.Variables:**

Pipeline and Stage variables are custom variables you can add and reference in your Pipeline and Stage. They're available across the Pipeline. You can propagate and override their values in later stages.

For more information, go to [Use Harness expressions | Harness Developer Hub](https://developer.harness.io/docs/platform/variables-and-expressions/harness-variables/)

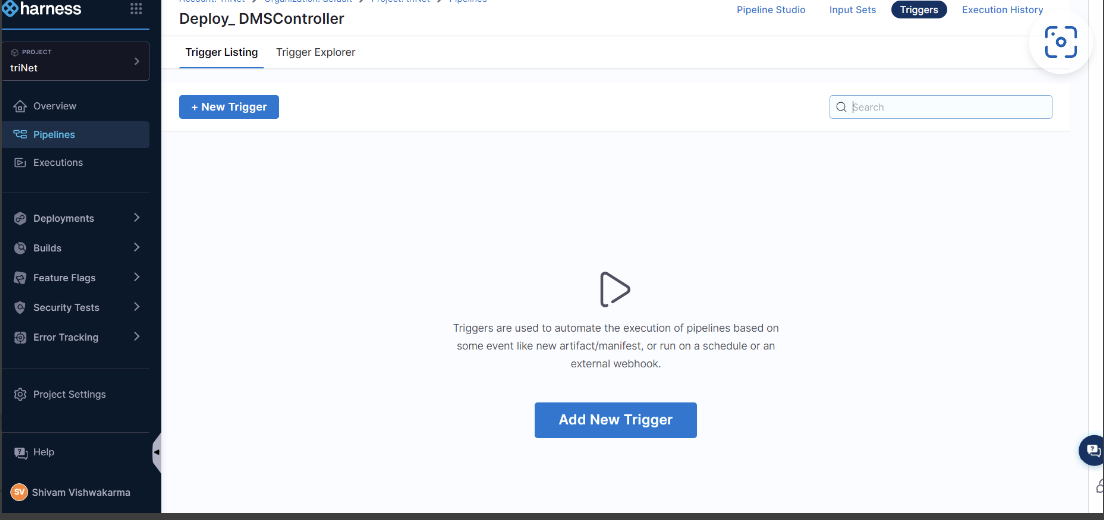


# **8.Triggers:**

You can run your Pipelines manually or use triggers to initiate their execution.

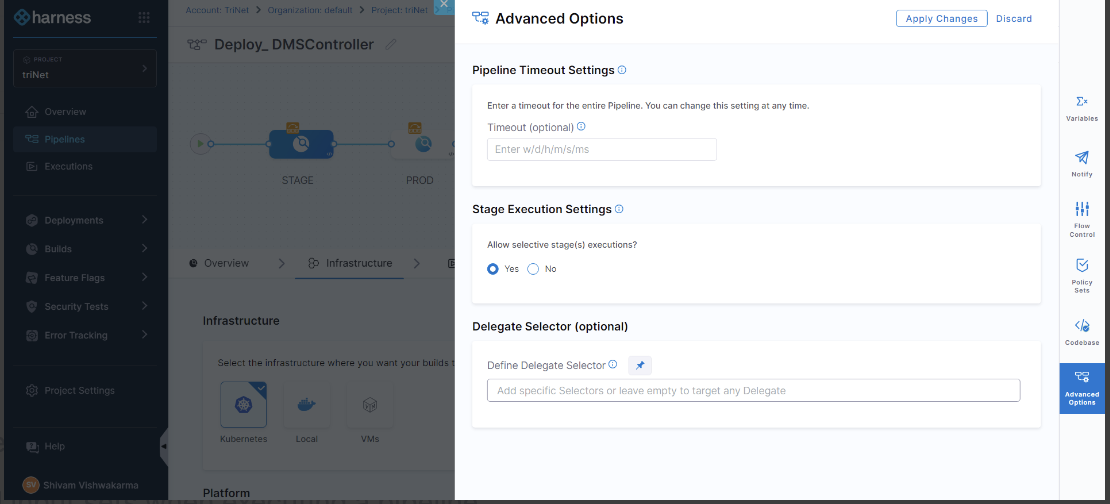
You can trigger a Pipeline based on Git events, manifest changes, schedules, new artifacts, etc.

For examples, go to:

* [Trigger pipelines on a new artifact | Harness Developer Hub](https://developer.harness.io/docs/platform/triggers/trigger-on-a-new-artifact/)
* [Trigger pipelines on new Helm chart | Harness Developer Hub](https://developer.harness.io/docs/platform/triggers/trigger-pipelines-on-new-helm-chart/)
* [Trigger pipelines using Git events | Harness Developer Hub](https://developer.harness.io/docs/platform/triggers/triggering-pipelines/)
* 

# **9.Advanced settings:**

Pipelines, Stages, and steps have advanced settings to control the flow of operations.



# **10.Input sets and overlays:**

Harness input sets are collections of runtime inputs for a pipeline provided before execution.

Overlays are groups of input sets. Overlays enable you to provide several input sets when executing a pipeline.

With input sets and overlays, you can make a single pipeline template that can be used for multiple scenarios. Each scenario can be defined in an input set or overlay and simply selected at runtime.

For more information, go to [Input sets and overlays | Harness Developer Hub](https://developer.harness.io/docs/platform/pipelines/input-sets/)

# **11.Conditional Execution:**

You can set conditions on when you run Stages and steps. For example, Execute This Stage Only if Prior Pipeline or Stage Failed.

The stage Conditional Execution applies to all steps that do not have their own Conditional Execution. A step's Conditional Execution overrides its stage's Conditional Execution.

For more information, go to [Define conditional executions for stages and steps | Harness Developer Hub](https://developer.harness.io/docs/platform/pipelines/step-skip-condition-settings/)

# **12.Failure strategies:**

A failure strategy defines how your Stages and steps handle different failure conditions.

The failure strategy contains error conditions that must occur for the strategy to apply, and actions to take when the conditions occur.

Failure strategies are a critical pipeline design component that determine what fails a step or stage and what to do when the failure occurs.

For more information, go to [Define failure strategies for stages and steps | Harness Developer Hub](https://developer.harness.io/docs/platform/pipelines/failure-handling/define-a-failure-strategy-on-stages-and-steps/).

# **13.CD Overview:**

1. Walk through on Cloud Bees tool Overview, ADA process and work we do.
2. Discussed on Components, Component Process.
3. Discussed on Application and Application process and how these are mapped with different Components.
4. Discussed on Snapshot and it contains artifacts name and its version details.
5. Discussed on different Environments and the pipeline that we have for all the Environments.
6. Received mandatory Confluence links for ADA Process.
7. Discussed on MyAccess page and how to request for access.

# **14.CloudBees/DCP Overview:**

1. Walk through on ADA process and work we do.
2. How the artifacts been fetched to DCP and PEO application.
3. Discussed on Components, Component Process.
4. Discussed on Application and Application process and how these are mapped with different Components.
5. Common steps which are available in Application process (Set\_Env, Fetch KVP, Set\_rolling\_deploy\_property, Jason\_ingest).
6. Discussed on Snapshot and it contains artifacts name and its version details.
7. Cloudbees\_deployments are the repo where release team keeps all the artifacts.
8. Discussed on different Environments and the pipeline that we have for all the Environments.
9. We are storing all the automation scripts in the location: apps/data/cloudbees/worker\_agents/deployment\_scripts/cloudbees/**example.sh.**
10. Discussed on Cloud bees Worker nodes, in total we have 5 worker nodes in that we are using wrkr01, wrkr03 for Non prod, Wrkr02 for DCP and wrkr04, wrkr05 is for Prod.
11. Discussed on code catch for Demo01 Env, we perform code catch on every Friday to Demo01 from Prod. This we do on the code available till Thursday.

# **15.Component & Component creation Process Creation - HIP Project:**

1. Discussed on Component creation process for HIP Project and it’s checks to perform before creating.
2. Discussed on Dashboards, in which we can check the number of Deployments and its status for the particular time period.
3. Discussed on where to check the code which is written for each widget of Dashboard.
4. Received path for python Automation scripts: /apps/data/cloudbees/worker\_agents/deployment\_scripts/cloudbees/python.

# **16.RPA Pipeline Deployment for QE and Prod and Add new Bot to the Pipeline:**

1. ADA Team will receive the request to add new bot.
2. To run pipeline at first, we need to select Environment type, Update the CR number, select the bot that need to be deployed and then we need select the version received from CR.
3. For every bot we create one application process and for each bot a task is created in pipeline.
4. Discussed on how to add the new components from the existing Components and to add component definition.
5. Whenever the request receives to us to add the bot in pipeline, release team will add the artifacts in cloudbees artifacts management.
6. Discussed on how to create the application process (Unlike other applications, we create one application process for one bot).
7. once we are done with the component and application process there we come back to pipeline and add/copy from existing task for requested bot.
8. We update the details, Definition and conditions tabs with new bot name.
9. We update the Parameter details for each bot.
10. Once we add bot application process, task and then the parameter in order to check if that is all done correctly, we'll just go to one new run and update anyone default CR number and check newly added bot name is coming and its version is showing.

# **17.DCP Pipeline Overview:**

1. Discussed on Difference between Microservices Pipeline and DCP Pipeline.
2. Most of the components in DCP under development.
3. Discussed on Application Processes and the steps.
4. Discussed on Component process.
5. Discussed on Procedures and checked the logs.
6. For DCP we are following the Blue-Green Deployment Strategy.

# **18.OCI Snapshot and  Inv Sync:**

1. During the Migration time from urban code to Cloudbees, we generated the Snapshots from urban code and Cloudbees by using the python script we merged the code and removed the duplicates.
2. In Cloud bees under Environment Inventory, we can check for each application which version is deployed.
3. we have inventory procedures under this path: " Common utilities- Procedures- getEnvInventory.
4. We have Separate Procedures for config and Api exist in cloudbees.

# **19.BoDT and PW Change:**

1. Bodt -Back office digital transformation- oracle fusion All Objects are under custom folder.
2. Two types of objects are migrated xdoz,xdmz
3. Cr is used to track migrations.
4. Data object-no extension, data model end with dm.
5. CSV files consist of Target path: RiceID: Jira ID.
6. If the object has already pushed in the past, that will be deleted and push the new one.
7. AD team will deploy the changes and ADA team will fix if any pipeline issues.
8. Changesets contains migration related information which environment to migrate, Jira information.
9. We have to change the passwords for all the accounts in stage and prod environments, be it RabbitMQ or Redis for every three months.
10. We will receive CSV file via email App admin or there is a location in SharePoint where the where there is a list of all accounts.
11. There will be a change list for back end and front end and which we have to follow for doing the password change.
12. So, all the passwords in our project are currently being stored in beyond trust, so there is a Python file vault BT.PY which usually converts the CSV to YAML file.
13. once that is converted, that particular YAML file, we'll give it as input to another Python file, which is BT change password.PY.
14. Also, refer to this page as well for additional details: [Backend Prod Password change - Urban Code - Confluence (trinet-devops.com)](https://confluence.trinet-devops.com/display/UC/Backend+Prod+Password+change)

# **20. Artifact Management**

CloudBees also maintains an artifact management repository. As part of CD process for PEO apps, CD pipelines picks the artifact from CloudBees Artifact Management repository and deploys to environments. However for DCP apps, CloudBees pulls the artifact directly from Jfrog Antifactory as the end to end CI/CD pipeline is automated till deployment to Production.

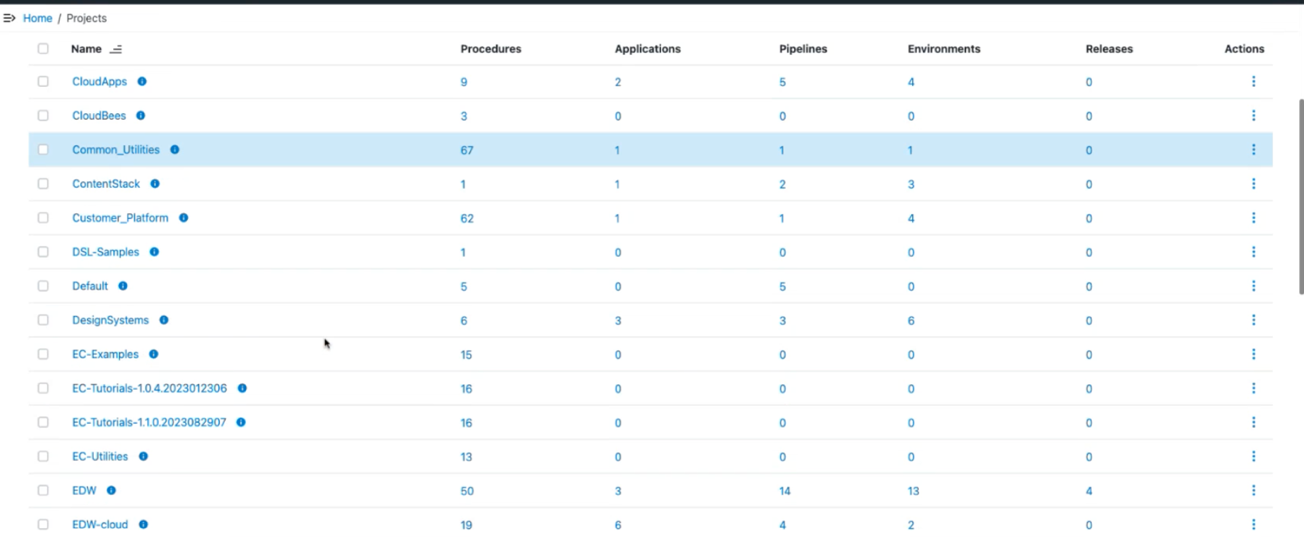
A screenshot of a computer

Description automatically generated

# **21.Projects, Applications**

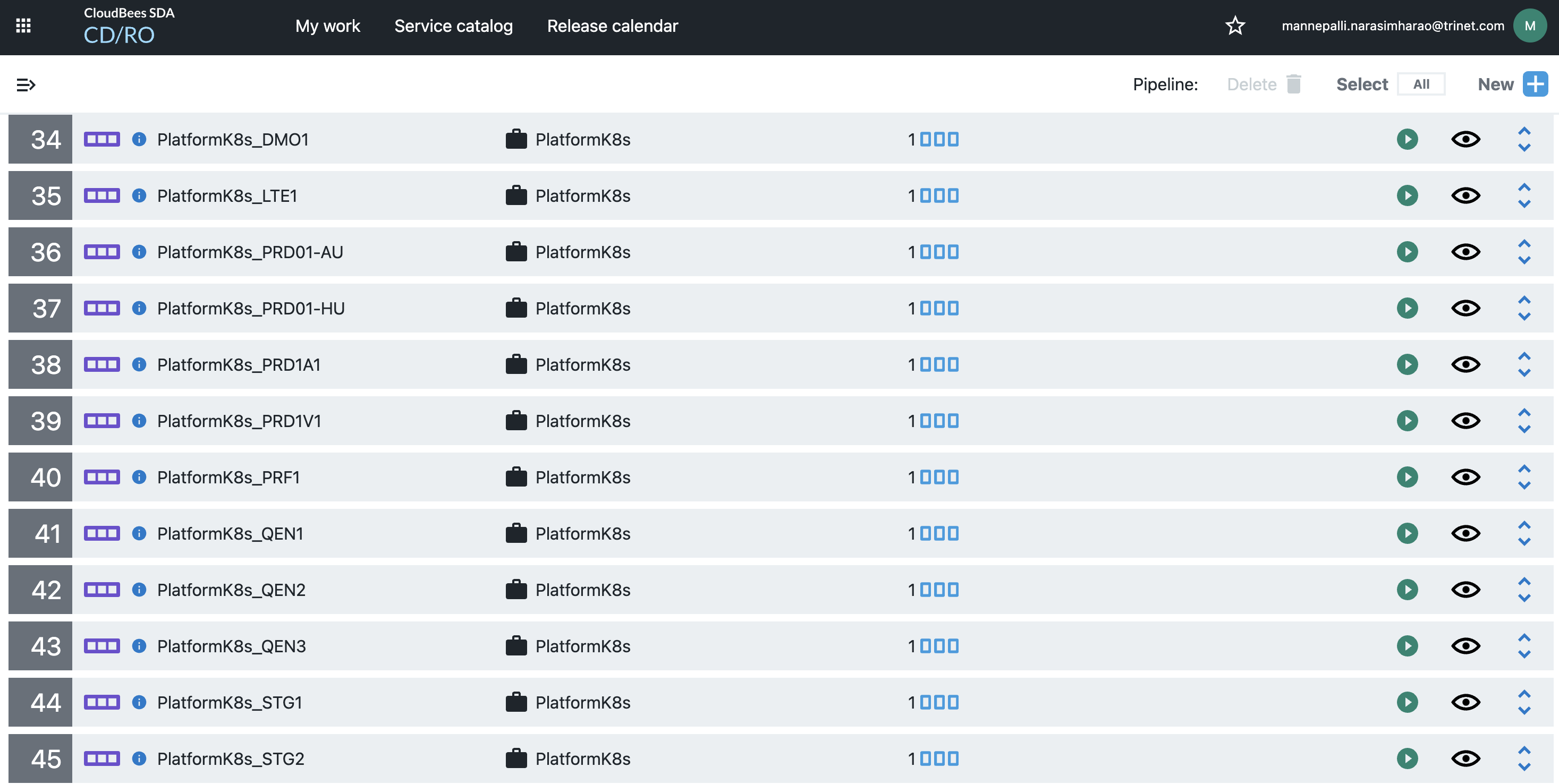
Within CloudBees CDRO, there are multiple projects. And projects contains Applications. Some applications contains components.

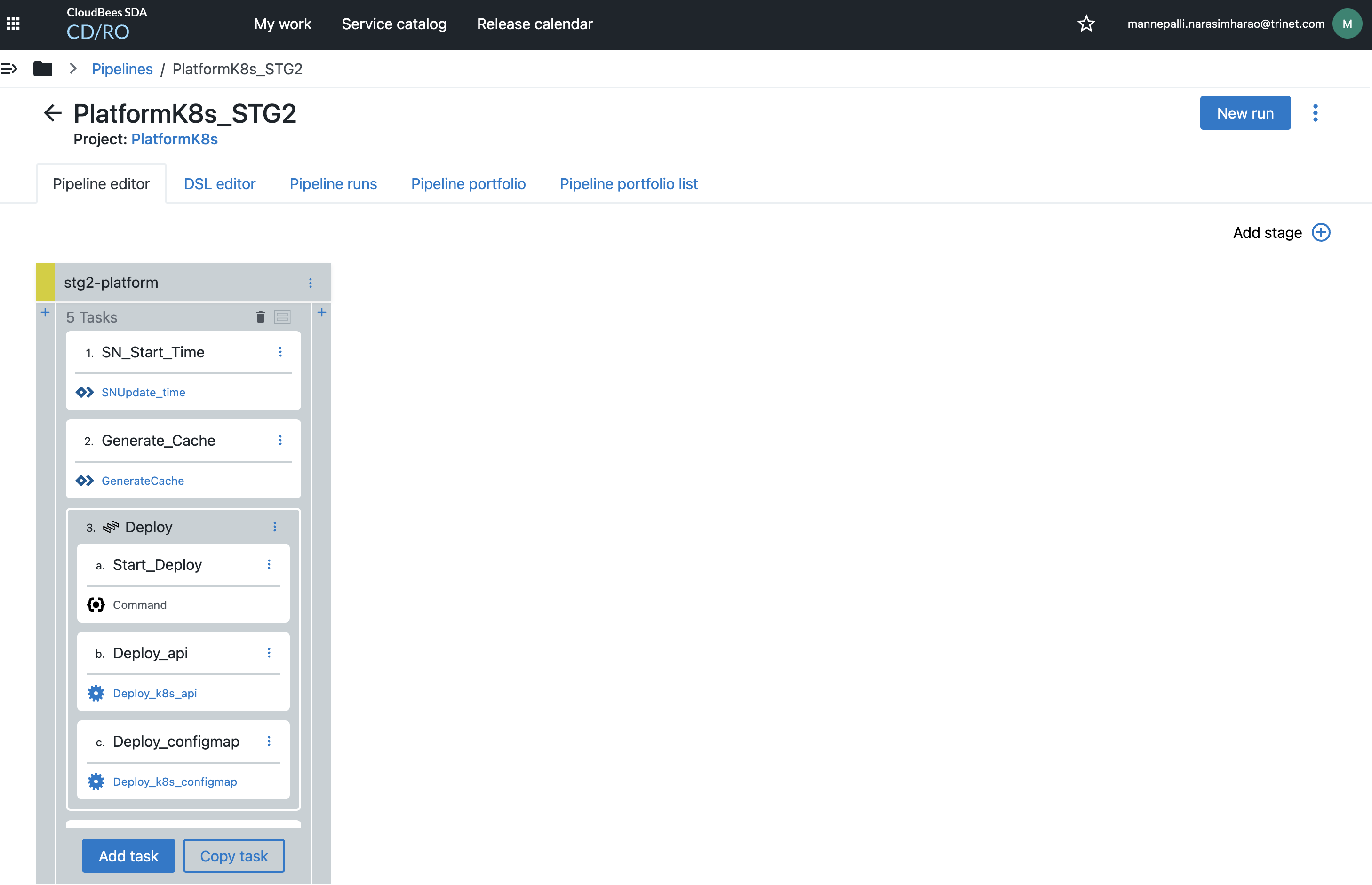
Example Projects: Customer\_platform, Platform



# **22.Environments, Pipelines**

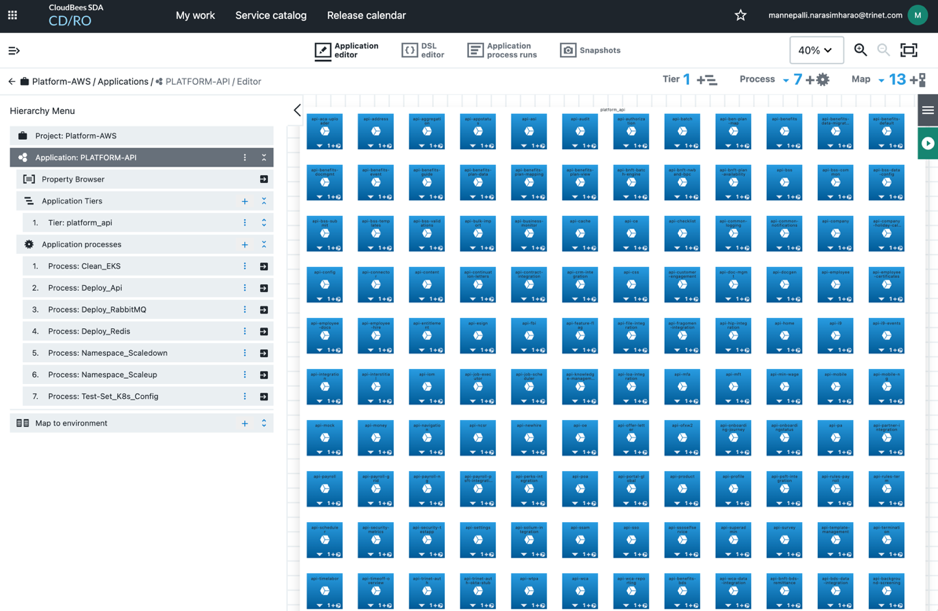
For any deployment, environment specific pipelines are already configured. Each pipeline will have tasks. And each task calls an application process.





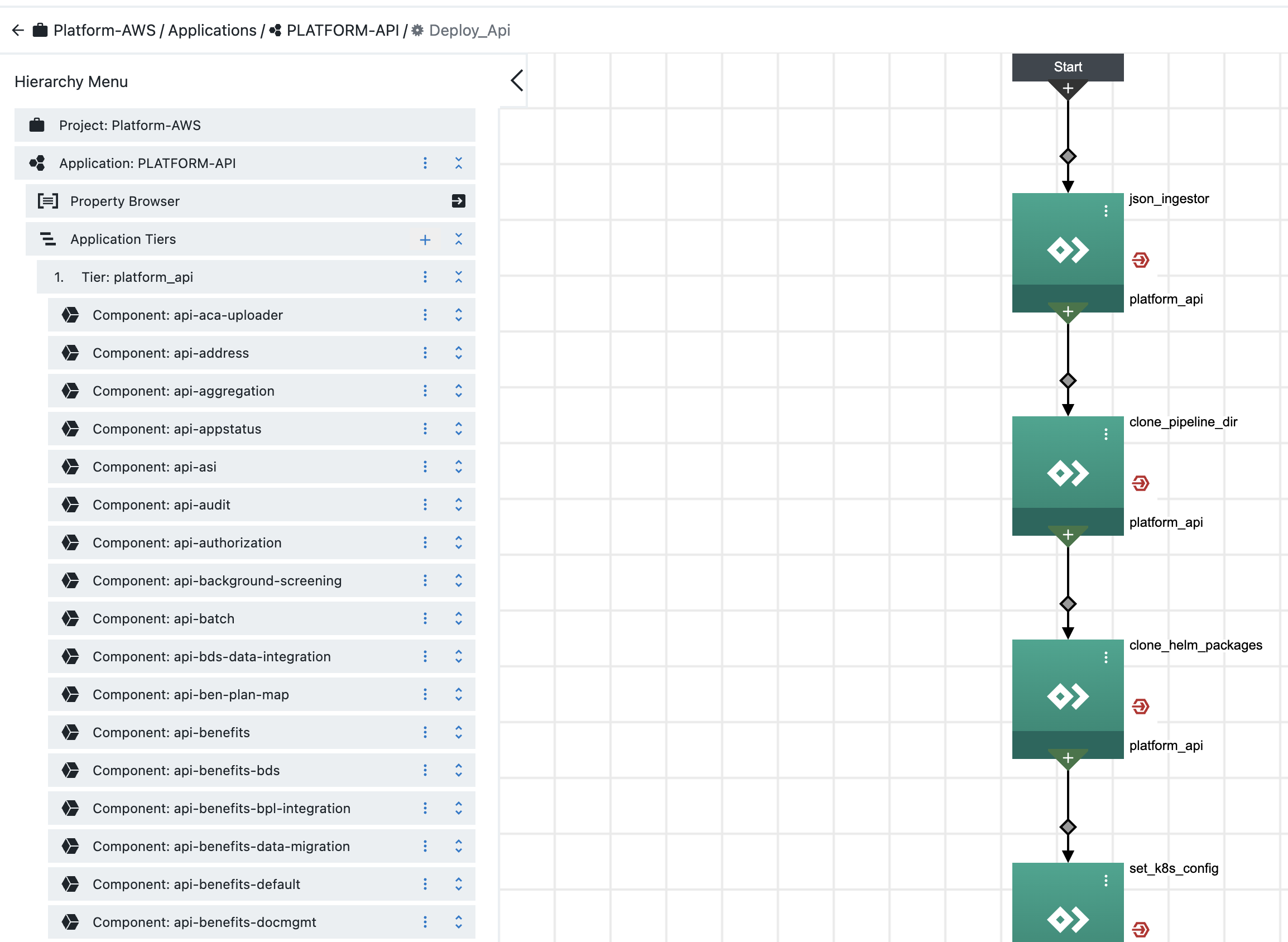
# **23.Application Processes, Procedures**

When developer or App Deploy team triggers pipelines (in case of PEO App only. DCP apps does not have pipelines. Application process is directly called from Jenkins pipeline for DCP apps.), the tasks added in the pipeline calls the application process, which will run the entire deployment process by executing multiple steps like commands or procedures etc.

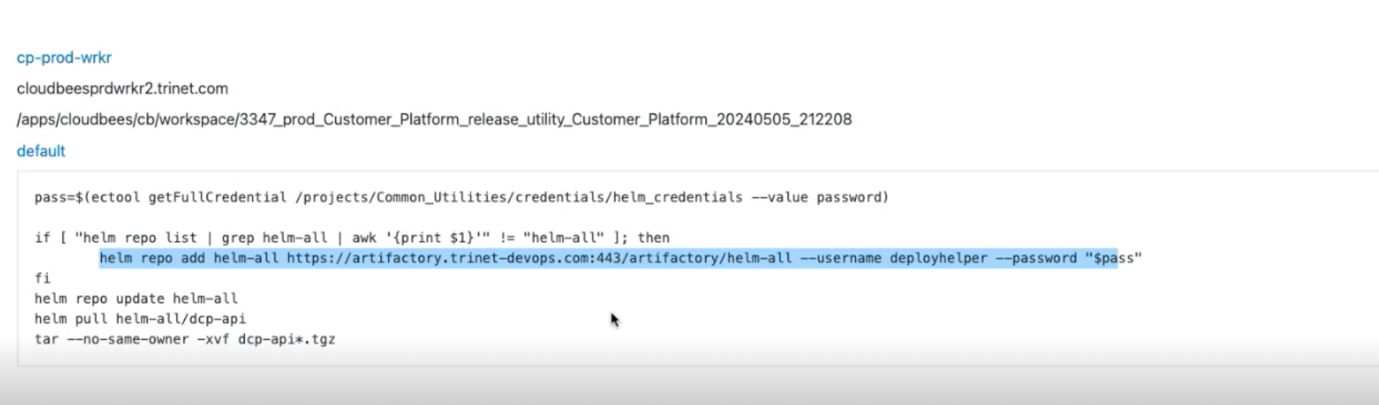


In the above screen, on the left pane, we can see the application processes that can be used for deployments.

Here is a screenshot of how application process looks like:



Each step in the process might invoke a procedure, which will have multiple steps and they can be command executions. Here is screenshot of command from a procedure.



* Specific parameters are assigned to each environment within the application process.
* Multiple environment pipelines can be executed parallelly.
* However, each environment pipeline can only be initiated once at a time.
* If an environment pipeline is triggered multiple times, subsequent runs are queued until the ongoing run concludes.
* Under the Access control we can define which team requires the access to different pipelines
* Developers have access to pipelines in order to deploy and test the application in Dev environment.
* QE also have the access to pipelines in order to deploy application in QE environments.
* For Prod pipelines only App Deploy and App Deploy Automation teams will have the access.
* Application Deployment (APP Deploy) Team is responsible for managing deployment requests.
* Upon receiving a deployment request, the team selects the appropriate Continuous Deployment (CD) pipeline.
* AppDeploy team follows the details provided in the Change Request (CR) and initiate the specific pipeline for deployment.

# **Key URL’s mentioned :**

CloudBees: <https://cloudbees.trinet.com/>

CloudBees Trainings: [Trainings and Recordings - CloudBees](https://confluence.trinet-devops.com/display/CLOUD/Trainings+and+Recordings)

KT Confluence: [16 - CloudBees CDRO Automation - Sandeep](https://confluence.trinet-devops.com/display/RM1/16+-+CloudBees+CDRO+Automation+-+Sandeep)

Harness: [Harness](https://app.harness.io/ng/account/TSJynUr6SmezTq0oNif7kg/all/overview)