User Manual

**Getting started with Powership**

List of tools integrated within the toolchain:

|  |  |  |
| --- | --- | --- |
| **Tool** | **Functionality** | **Version** |
| HCL UrbanCode Velocity | Reporting | 1.2.3 |
| HCL UrbanCode Deploy | Deployment | 7.0.2.1 |
| HCL AppScan on Cloud | Static Analysis for security | 1.0.0 |
| HCL OneTest | Functional Testing | 9.5 |
| Jira | Planning | Not applicable/External service |
| GitHub | Source code hosting | External service – Tested on 2.17 version |
| Jenkins | Build and Orchestration of Delivery Pipelines | 2.176 |
| SonarQube | Static Source Code Analysis for quality | To be updated |
| Jacoco | Code Coverage Tool | 3.0.4 |
| Grafana | Visualizing time series data for Infrastructure and Application analytics | 6.1.3 |

**System Requirements:**

**Software requirements**

To install Powership, the following software is required:

* + Gcloud (Google cloud) for Google cloud cluster authentication for the command line option
  + Kubectl
  + Helm v2.13
  + Docker

**Hardware requirements**

To install Powership, the Google cloud Kubernetes cluster must have the following minimum configuration:

* + Node size - 4
  + Total cores - 16 vCPUs
  + Total memory - 60.00 GB

## **Installing Powership**

Powership can be installed in Google Cloud console .Powership can also be installed in “default” namespace of the Kubernetes cluster from ubuntu/linux instance.

1. To install Powership Tech Preview in the default namespace of the Kubernetes cluster In Google Cloud, follow the below steps
2. Login to Google Cloud account.
3. Goto Navigation Menu, select Kubernetes engine and then click on 'Clusters'.
4. Make sure you are connected to the respective project for eg. blackjack .
5. Select the cluster name (for eg solutions-team-test-mars ) and click on connect button.
6. In Connect to the cluster pop-up , click on 'Run in Cloud Shell'.
7. Click on 'Start Cloud Shell' button.
8. Connect to the cluster where Powership must be deployed:

gcloud container clusters get-credentials <cluster-name> --zone <master-zone> --project <GCP project name>

1. Clone the git repository to get the below scripts.(this is available scripts folder)

* install-cert-manager.sh
* powership-cluster-setup.sh

**git clone https://github.com/pnp-powership/setup-files.**

1. Run the following shell script.

./install-cert-manager.sh

1. To initialize the Helm, run the following commands:

* helm init
* helm update

1. Run the powership-cluster-setup.sh script with the valid username and password .

./powership-cluster-setup.sh [username] [password]

Use command chmod 777 powership-cluster-setup.sh in case of permission issue.

1. Deploy Powership by running the following command:

* helm install powership-repo/powership -n <release\_name>

replacing <release\_name> with a value of your choice.

1. Verify that Powership is successfully installed and services are in Running status by running the following command:

* kubectl get pods

Wait and repeat the above command until the pods are in Running status and the command output shows Ready 1/1

1. Get Powership application IP\_address by running the following command:

* kubectl get svc

Copy the IP\_address that is reported in the EXTERNAL\_IP column of the command output.

1. Launch Powership in a web browser, in the below format,

http://watcher.IP\_ADDRESS.nip.io:80

1. Login to Powership by using default Keycloak credentials:

* username: user
* password: pass

Note:

If any issues while installing the Powership or solution, please delete the installed Powership instance by using the below command and reinstall the same.

helm del --purge <release\_name>

1. To Install Powership in “default” namespace of the Kubernetes cluster from “ubuntu/linux” instance.
2. Install gcloud by using the below command

**curl https://sdk.cloud.google.com | bash**

**exec -l $SHELL**

verify gclound installed or not by executing

**gcloud version**

1. Install kubectl, by below command

**gcloud components install kubectl**

1. verify kubectl, installed or not by executing

**kubectl version**

1. Download helm 2.14 version .tar.gz from https://github.com/helm/helm/releases and run the below commands
   * 1. **tar -zxvf helm-v2.14.1-linux-amd64.tar.gz**
     2. **cp linux-amd64/helm /usr/local/bin/helm**

this will copy usr/local/bin directory

verify the version of helm by command executing the below command.

**helm version**

1. Login to Google Cloud Project with the command:

**gcloud auth login**

1. Connect to the cluster where Powership must be deployed:

**gcloud container clusters get-credentials <cluster-name> --zone <master-zone> --project <GCP project name>**

1. Create service account and install Tiller using the following steps:
   * **kubectl --namespace kube-system create sa tiller**
   * **kubectl create clusterrolebinding tiller --clusterrole cluster-admin --serviceaccount=kube-system:tiller**
   * **kubectl patch deploy --namespace kube-system tiller-deploy -p '{"spec":{"template":{"spec":{"serviceAccount":"tiller"}}}}'**
2. Execute the below command to initialize the helm
   * **helm init**
   * **helm update**
3. Run the powership-cluster-setup.sh script with the username and password provided by HCL Powership team:

**./powership-cluster-setup.sh [username] [password]**

1. Deploy Powership by running the below command. Replace release\_name by your preferred name,

**helm install powership-repo/powership -n <release\_name>**

1. Verify Powership installed properly and services are in Running state by executing the below command,

**kubectl get pods**

1. Get the Powership application URL by running the following steps:

**kubectl get svc**

1. Launch Powership in a web browser, in the below format,

**http://watcher.IP\_ADDRESS.nip.io:80**

1. Login to Powership by using default Keycloak credentials:

username: user

password: pass

**Note:**

1. If any issues while installing the Powership or solution, please delete the installed Powership instance by using the below command and reinstall the same.

**helm del --purge <release\_name>**

**Configuring the sample project**

To configure JPetStore sample project and play with it, you must run the following steps:

1. Integrating Jira and GitHub with HCL UrbanCode Velocity (UCV)
2. Integrating Jira with GitHub
3. Deploying JPetStore sample project
   1. Configuring HCL UrbanCode Deploy (UCD) with Jenkins
   2. Configuring HCL UrbanCode Velocity (UCV) with HCL UrbanCode Deploy (UCD)
4. Configuring Urbancode Velocity and Urbancode Deploy in Jenkins

**Integrating Jira and GitHub with HCL UrbanCode Velocity (UCV)**

1. Login to UCV with valid username and password.

UN/PWD:admin/admin.

2.Modify the base url by adding the /**featureFlags** and click **Enter**

For Eg: <https://velocity.ipaddress.nip.io/featureFlags>

1. In the field “Enter a feature name to enable it “, enter” **portfolio** “ -> click **Enable**
2. From the header, click **Value Streams**
3. Then click **Create** button to create a volume stream-> Enter the name, description, and choose the team from the dropdown menu
4. In the created Value stream, click **“Download Sample”** to download the **JSON** file and, replace the “**integration”** array with your values.
5. To create JIRA API token access, <https://id.atlassian.com/manage/api-tokens>
6. From the dialog that appears, enter a memorable and concise **Label** for your token and click **Create**.
7. Click **Copy to clipboard**, then use the same token below instead of “jira-token”.

"integrations": [

{

"type": "ucv-ext-jira",

"tenant\_id": "5ade13625558f2c6688d15ce",

"name": "[WHATEVER NAME YOU WANT]",

"disabled": true,

"properties": {

"baseUrl": "[CHANGE TO YOUR URL EX:https://jkeimport.atlassian.net]",

"username": "[USERNAME EX: bryantschuck@gmail.com]",

"password": "[Jira-token]",

"jiraProjects": [

"[YOUR JIRA PROJECT KEY EX: JKP]"

]

}

},

{

"type": "ucv-ext-github",

"tenant\_id": "5ade13625558f2c6688d15ce",

"name": "github-repository",

"disabled": true,

"properties": {

"url": "[REPO URL EX: https://github01.hclpnp.com/BryantS/sample-repository]",

"name": "[NAME OF THE REPO EX: sample-repository]",

"owner": "[OWNER OF THE REPO EX: BryantS]",

"apiUrl": "https://api.[GITHUBURL]",

"token": "[YOUR GITHUB TOKEN - Can be obtained from github admin, app tokens]"

}

}

]

1. Replace the “**phases”** array to make it more realistic:

"phases": [

{

"name": "Planning",

"description": null,

"stages": [

{

"name": "Backlog",

"query": "issue.status=\"Backlog\" or issue.status=\"To Do\"",

"wipLimit": null,

"gates": null

},

{

"name": "For Development",

"query": "issue.status=\"Selected for Development\"",

"wipLimit": null,

"gates": null

}

]

},

{

"name": "Development",

"description": null,

"stages": [

{

"name": "In Progress",

"query": "issue.status=\"In Progress\"",

"wipLimit": null,

"gates": null

},

{

"name": "Review",

"query": "issue.status!=Done and pr.status=open",

"wipLimit": null,

"gates": null

},

{

"name": "Merged",

"query": "(pr.status=MERGED or pr.status=closed) AND deployment.env!=DEV and deployment.env!=QA and deployment.env!=PROD and build.status!=success",

"wipLimit": null,

"gates": null

},

{

"name": "Build",

"query": "build.status=success AND deployment.env!=DEV AND deployment.env!=QA AND deployment.env!=PROD AND build.startTime < 5h",

"wipLimit": null,

"gates": null

}

]

},

{

"name": "Deployment",

"description": null,

"stages": [

{

"name": "DEV",

"query": "deployment.env=DEV AND deployment.env!=QA AND deployment.env!=PROD",

"wipLimit": null,

"gates": null

},

{

"name": "QA",

"query": "deployment.env=QA AND deployment.env!=PROD",

"wipLimit": null,

"gates": null

},

{

"name": "PROD",

"query": "deployment.env=PROD AND deployment.startTime < 12h",

"wipLimit": null,

"gates": null

}

]

}

],

1. Click **Upload** button under **Value streams** to upload the updated json file.

**Integrating Jira with GitHub**

1. Login to Jira with valid credentials
2. In the bottom-left corner, click the **Settings** button
3. Under Jira **Settings**, click **Apps**
4. In the search bar, search for **Github for Jira**
5. Click on “Get app” button and install.
6. Click **Github** for Jira card and install it.
7. After installation, click “Get Started” link.
8. Click **Add** an Organization
9. Add all the necessary Github details
10. You can see your Github is added under Add an Organization

**Deploying JPetStore sample project**

**a. Configuring HCL UrbanCode Deploy (UCD) with Jenkins**

1. Login to UCD tool with valid username and password
2. Click the **Settings** tab, then navigate to **System settings**
3. Change the Url for External Agent URL, External User URL to your UCD application Url. For eg https://deploy.35.226.129.237.nip.io
4. In the **General Settings** tab, under **Agent for Version import(Component Settings)**, select Agent Name for eg(  **“ucd-Agent” )** andclick **Save.**
5. In the **Licensing section,** remove the FlexNet Server Path and paste the text #DEVMODE#

and click on “save” button.

1. Download the **Tomcat Apache UCD Plugin** and install it in UCD .

To install **Tomcat Apache UCD Plugin**in UCD**,**run the following steps:

(

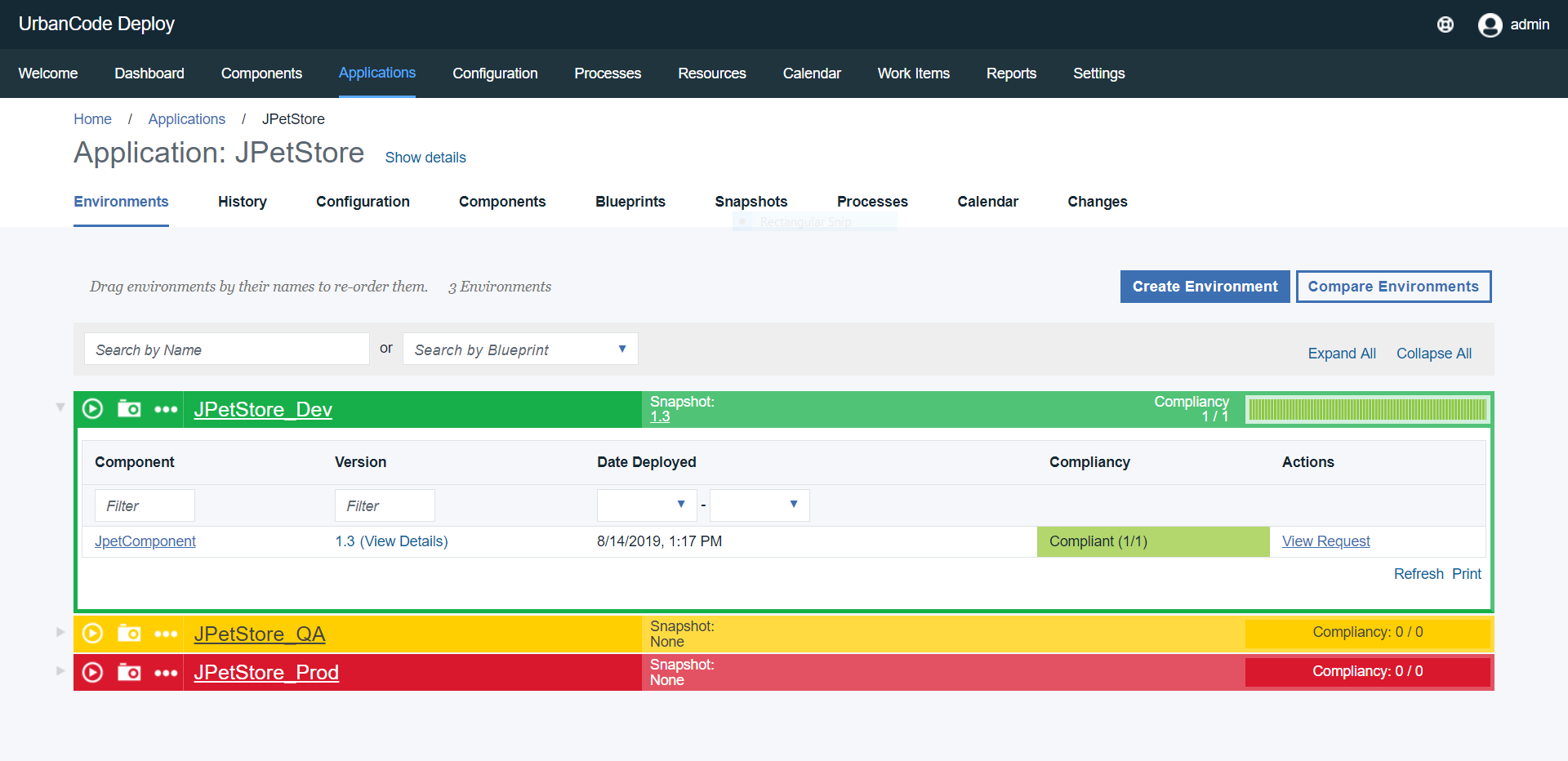
1. Download  **Tomcat Apache UCD Plugin**from[here](https://developer.ibm.com/urbancode/plugin/tomcat-ibmucd/)
2. Login to UCD
3. Navigate to the **Settings** tab
4. Under **Automation section**, link on **Automation Plugins**
5. Click **Load plugin**
6. Upload Tomcat 7.zip

)

1. Upload the ‘**jpetstore-app.json’** file which will be available under **UCD\_Integration** folderin the **JPetStore Git repository.**

* Click on “Applications” tab.
* Click on “Import Application
* Click on “Choose File” link and browse for ‘**jpetstore-app.json’**  file and click on submit.

1. Click on the **Resources** tab .By default three resources are present in UCD i.e **JPetStore-Dev, JPetStore-Prod, JPetStore-QA.**
2. By hovering over the **Resource name**, you can see the **Actions** link( click on 3 dots)
3. Hover over the “Actions” link next to Resource name and Click on “Add Agent’.
4. Hover over the “Actions” link next to Agent Name and Click on “Add Component’.
5. Select the component eg ( **Jpet-component)** and click on **Save.**
6. Click on “Applications” tab, Click on the Application name( eg. Jpetstore).
7. By Clicking on Arrow button next to Resource name, verify that the component ( **Jpet-component)**  is present.



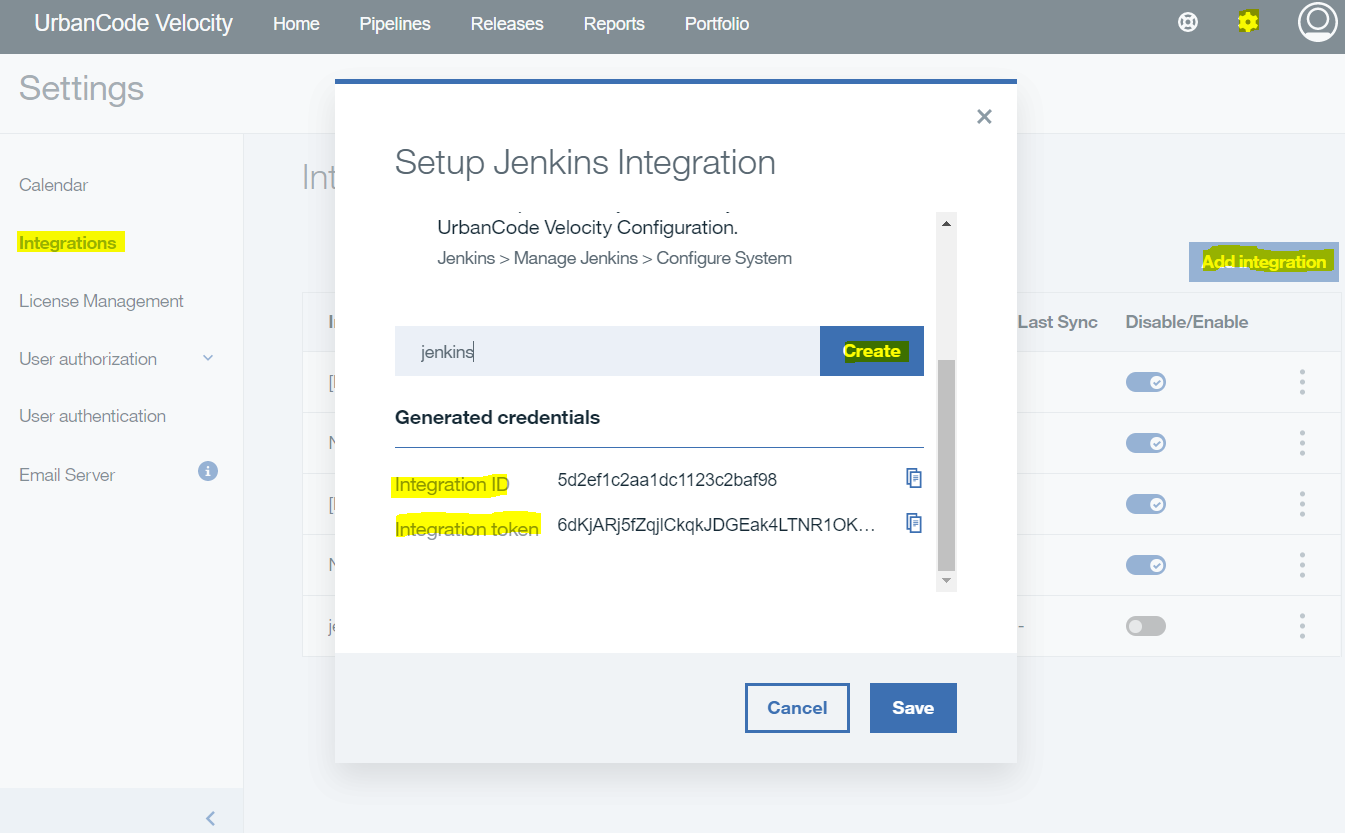
**b. Configuring HCL UrbanCode Velocity (UCV) with HCL UrbanCode Deploy (UCD)**

We must create pipeline in UCV to track the status of the JPetStore Project in UCD. Run the following steps:

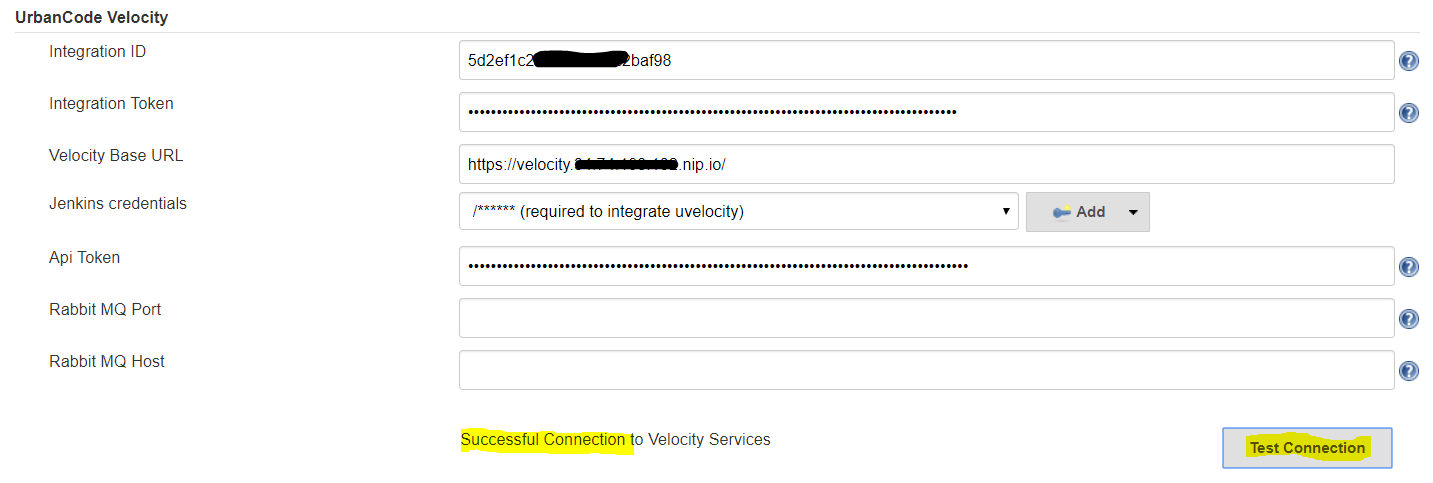
1. Login to UCV with valid username and password
2. Click on **Value Stream** tab and click the **Create+** link which is in the top left.
3. Provide the name, the description, and select the team, then click **Save.**
4. Click on “**Pipeline**” tab, then click on “**Add App**”.
5. Click **Add app** and select the option **UrbanCodeDeployment.**Providethe **Application Name** and click **Save**
6. Click the **plus sign** under the **Input** section
7. Select the application name and process name from UCD and click **Save** for the respective environments (Dev, QA, PROD)
8. Once the application is deployed in UCD, it should display the recent successful version of the application

**Configuring Urbancode Velocity and Urbancode Deploy in Jenkins:**

1. **Urbancode Velocity integration:**
   1. Login into Urbancode Velocity as an Administrator and click on the “Settings” icon.
   2. Navigate to “Integrations” part in the Settings page and click “Add integration” button.
   3. In the pop-up window, give a name for the Jenkins integration and click “Create” button as below.

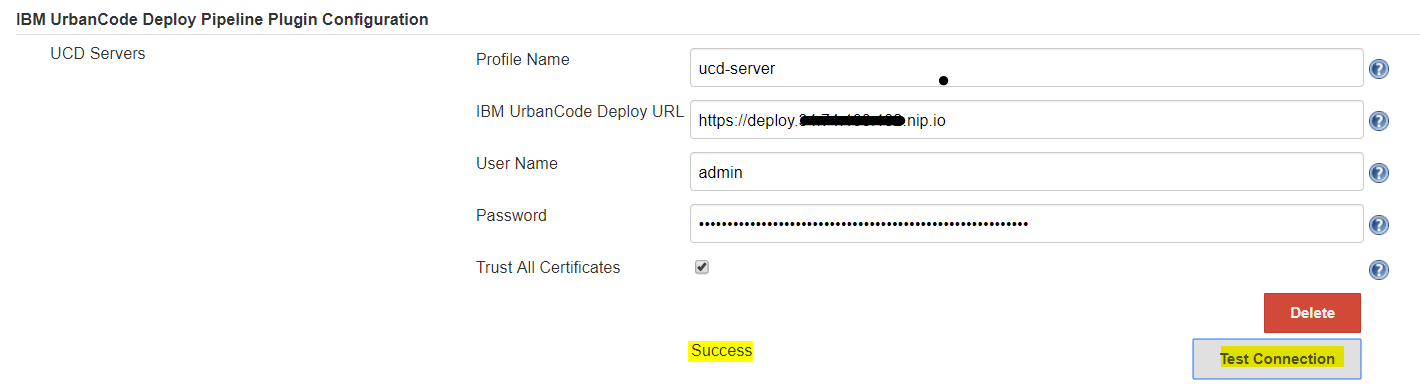


* 1. Then use this “Integration ID” and “Integration token” in Jenkins 🡪 Manage Jenkins 🡪 Configure System 🡪 Urbancode Velocity part as below,



* 1. For the “Api Token” in the above screen shot, login into Velocity as Admin and navigate to “My profile” option by clicking on the user icon. Then generate the API token and use the same in Jenkins for Configuration.
  2. After the configuration, click “Test Connection” to check the integration between Jenkins and Urbancode Velocity is successful or not.

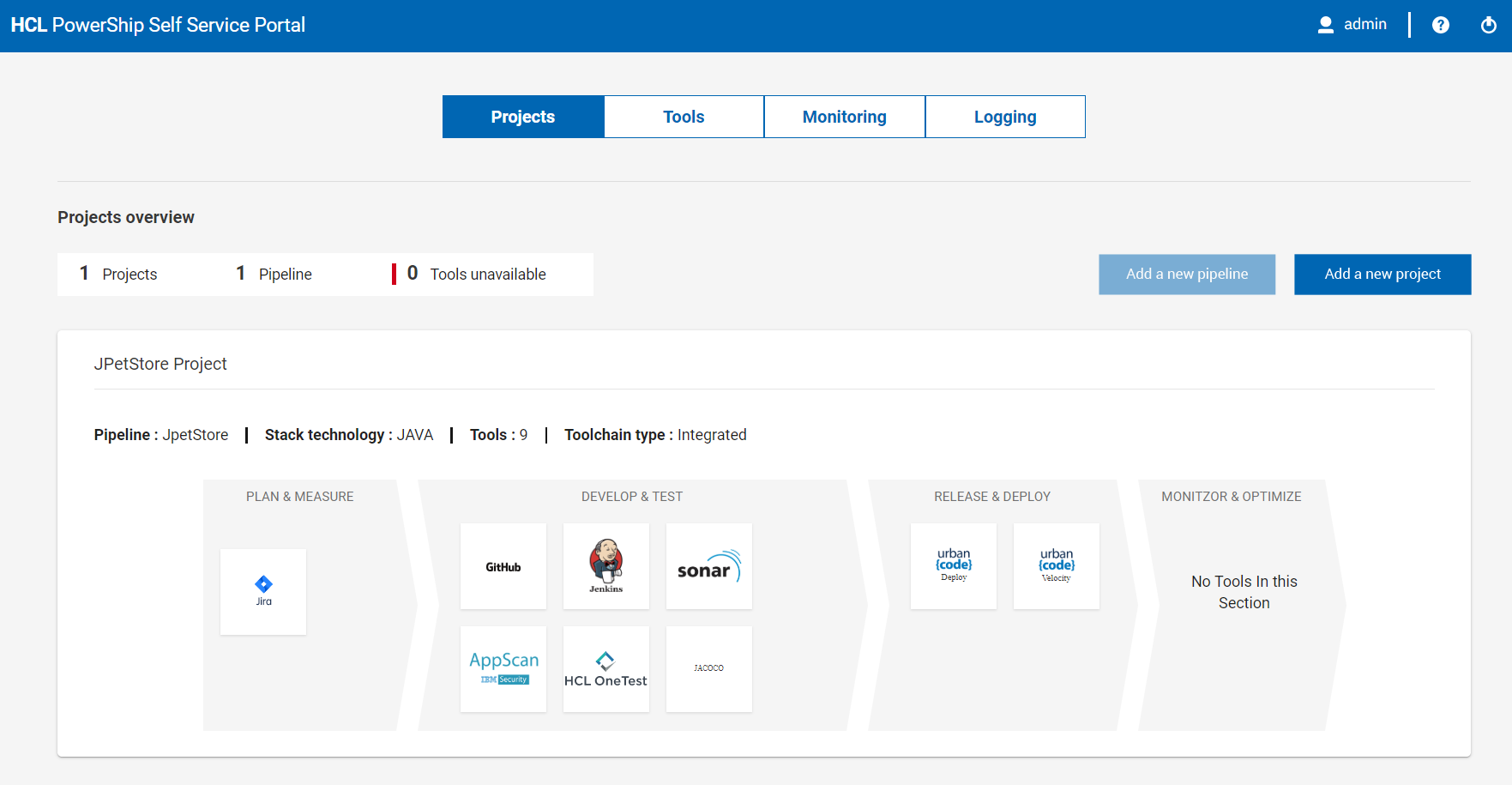
1. **Urbancode Deploy integration:**
   1. Login into Jenkins as Admin, then navigate to Manage Jenkins 🡪 “Configure system” option.
   2. Then provide the required values as below in Urbancode Deploy configuration part and click “Test Connection” to check the connection between Jenkins and UCD are correct.
   3. The connection should be SUCCESS if integration happened correctly.



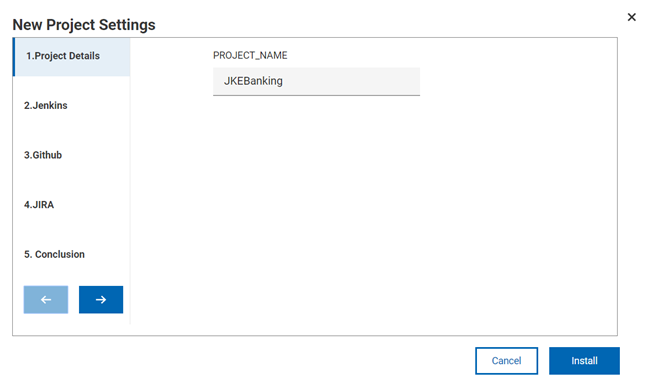
# **Creating a new project with an existing Pipeline**

From Powership Self Service Portal, you can add a new project to the same pipeline. Run the following steps:

1. From the **Project** tab, click **Add a new project**



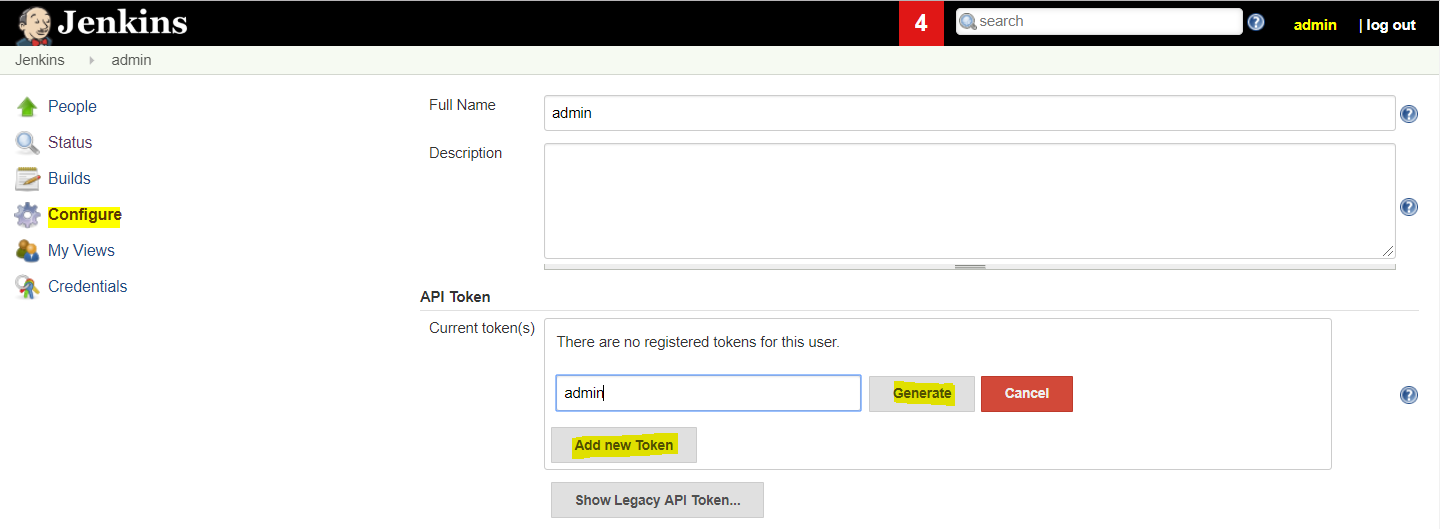
1. Enter the project name, for example JKEBanking, and click the right arrow.



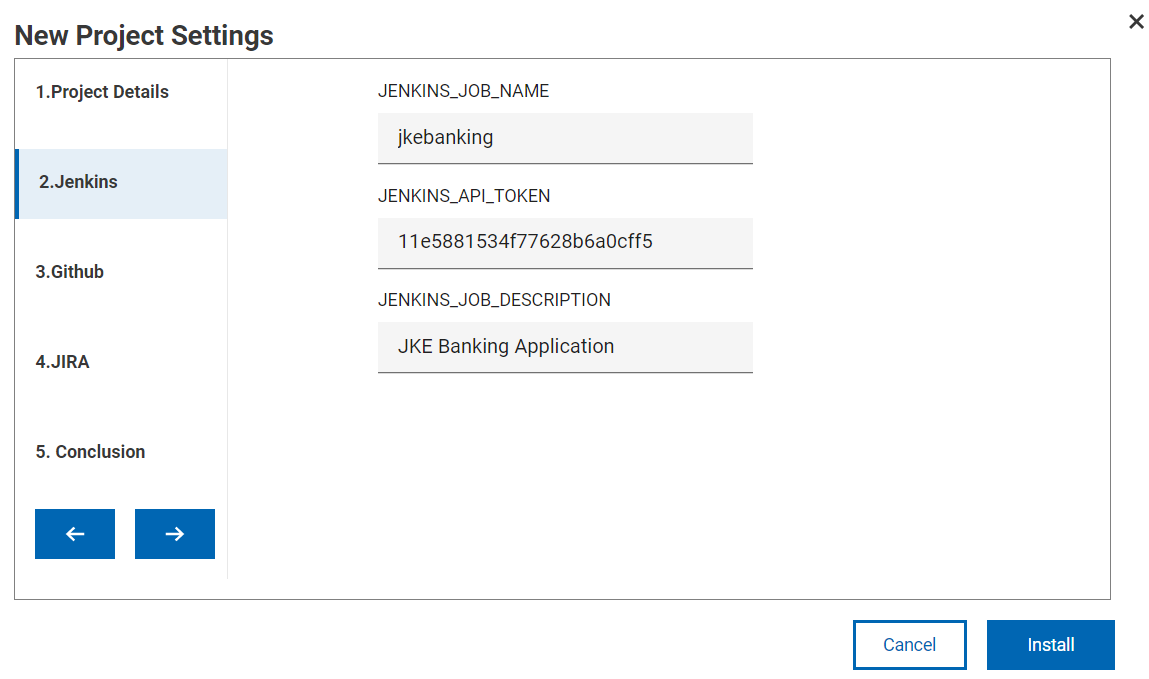
1. For **Jenkins**, provide JENKINS\_JOB\_NAME, JENKINS\_API\_TOKEN, and JENKINS\_JOB\_DESCRIPTION.

For Jenkins API token creation, run the following procedure:

* Login to Jenkins
* Click on the username on the top left side of the Jenkins panel
* In the User Configuration panel, click **Configure**and then click**Add new Token**
* Provide a unique token name and click **Generate** to generate the API token for admin user
* Copy the token and click the **Save** button to save it



4.Click the right arrow to move to the next tool.

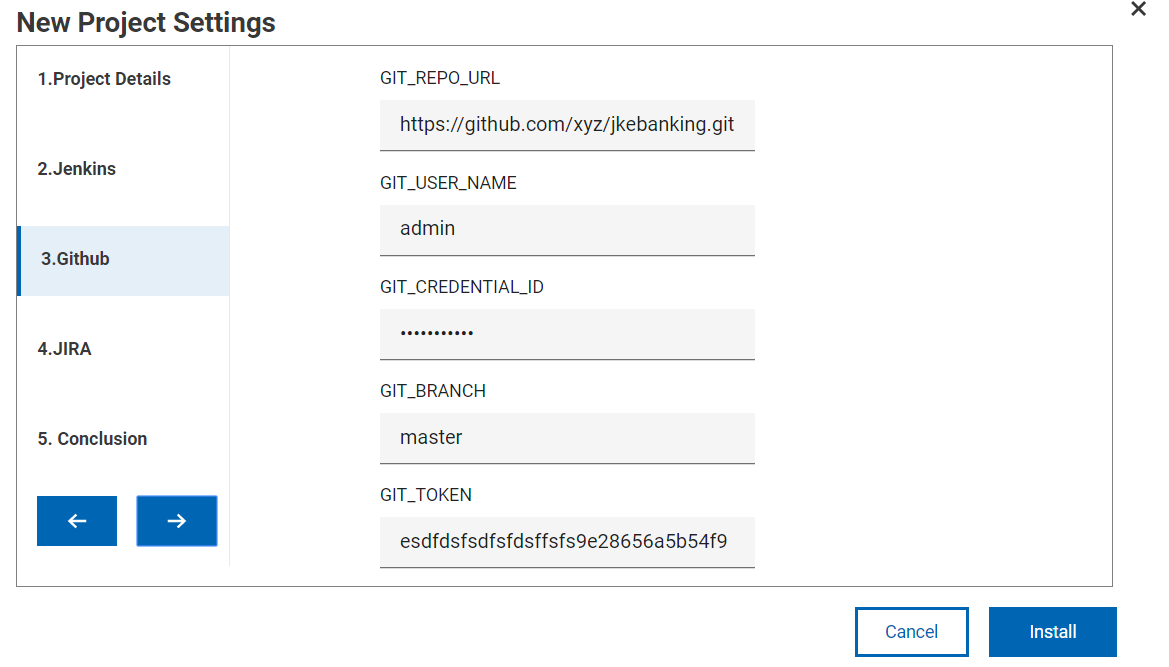


5.For GitHub, provide the information required to manage your project in the build process.

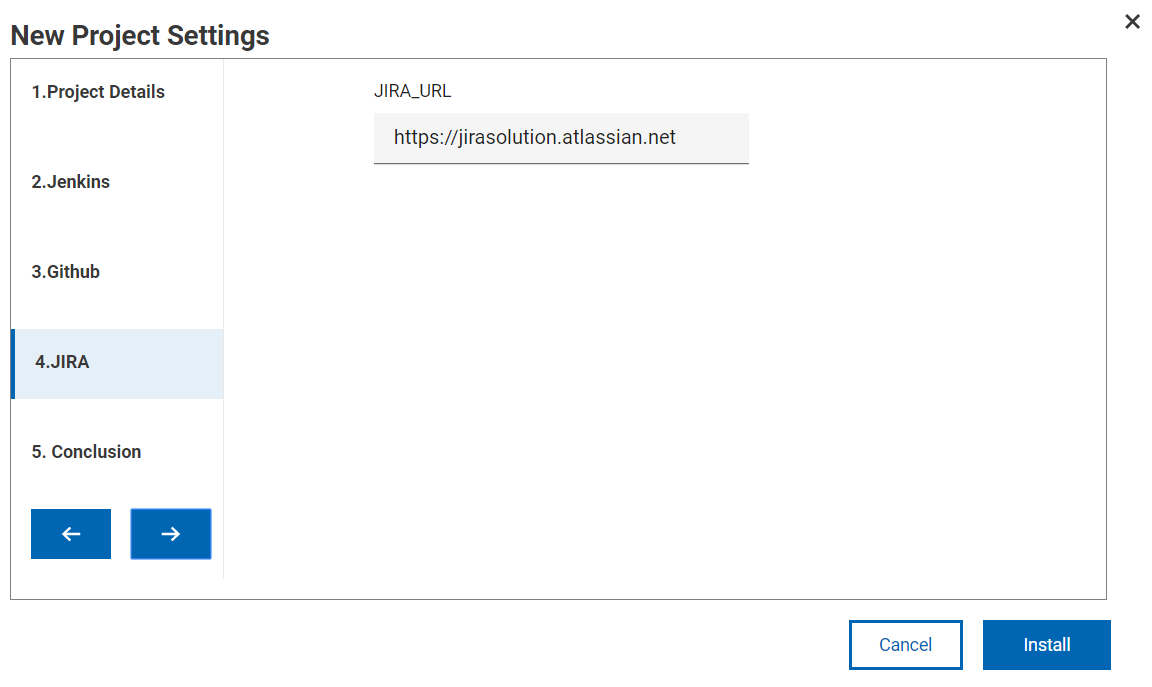
Note :GIT\_CREDENTIAL\_ID must be unique.

To create GIT\_TOKEN, see [GitHub documentation](https://help.github.com/en/articles/creating-a-personal-access-token-for-the-command-line).

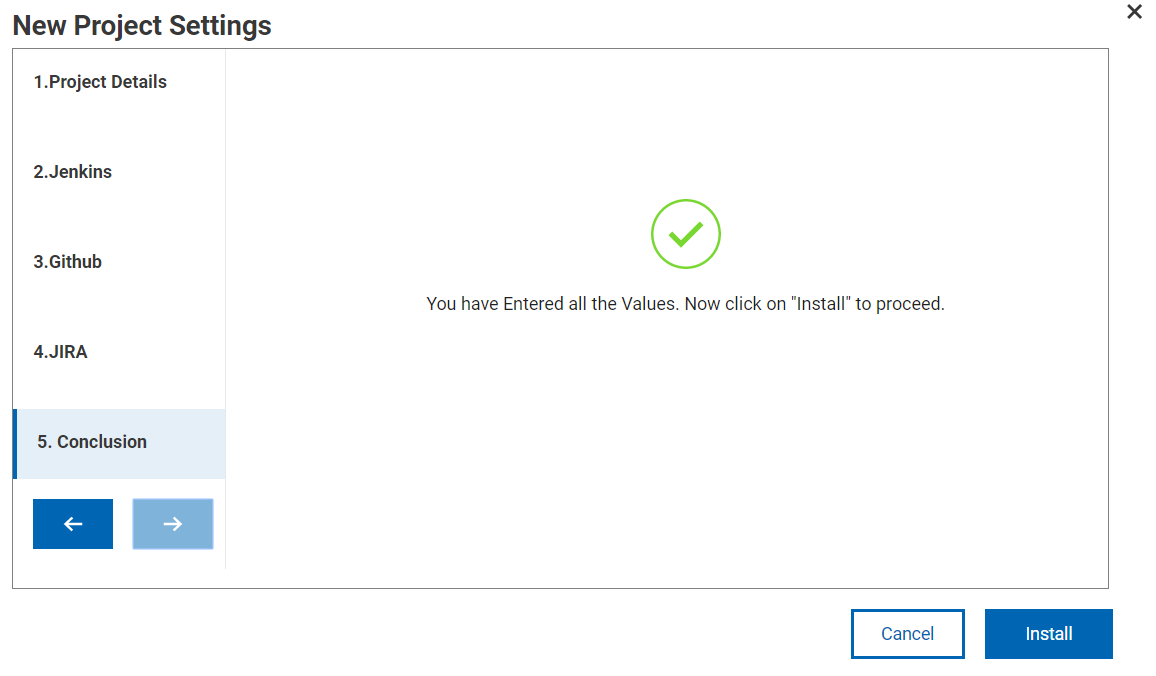
Click the right arrow to move to the next tool.



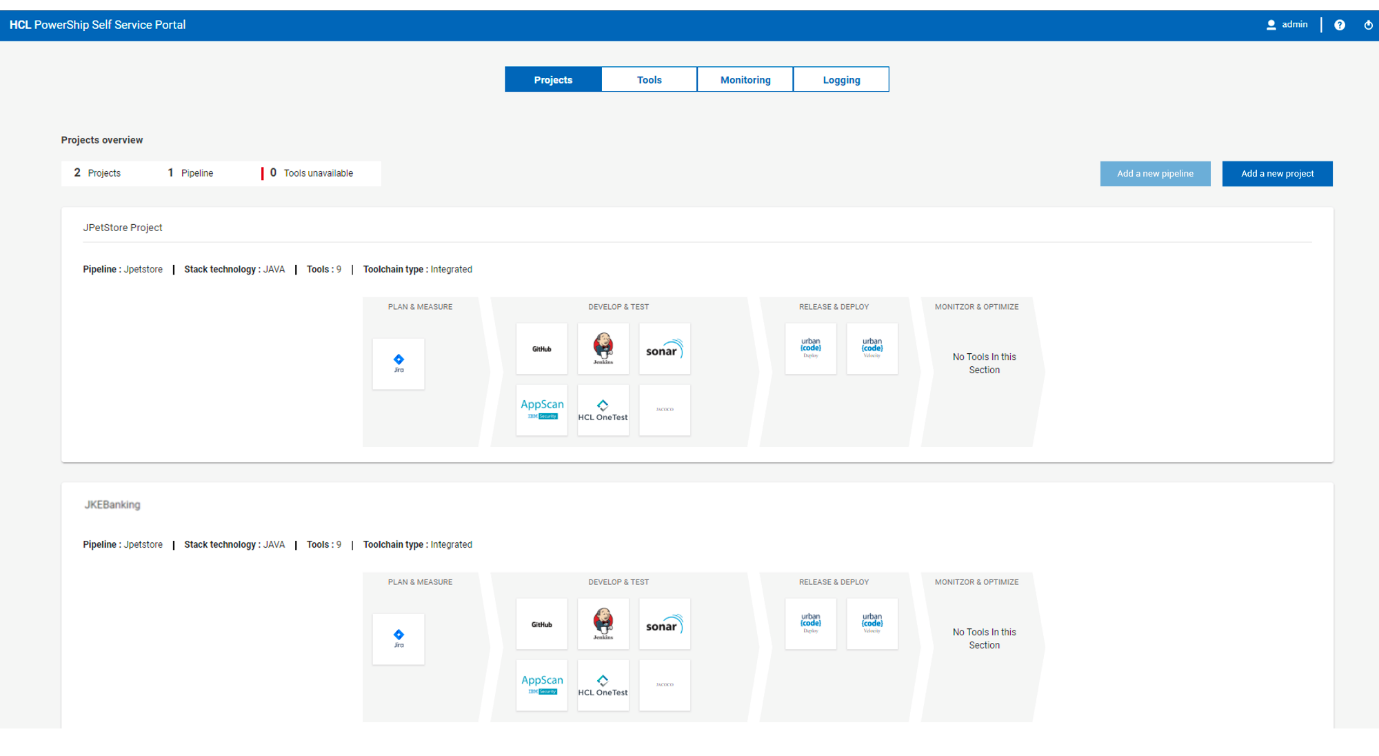
6.For **Jira**, provide the Jira URL. Click the right arrow to proceed.



7.Click **Install.**



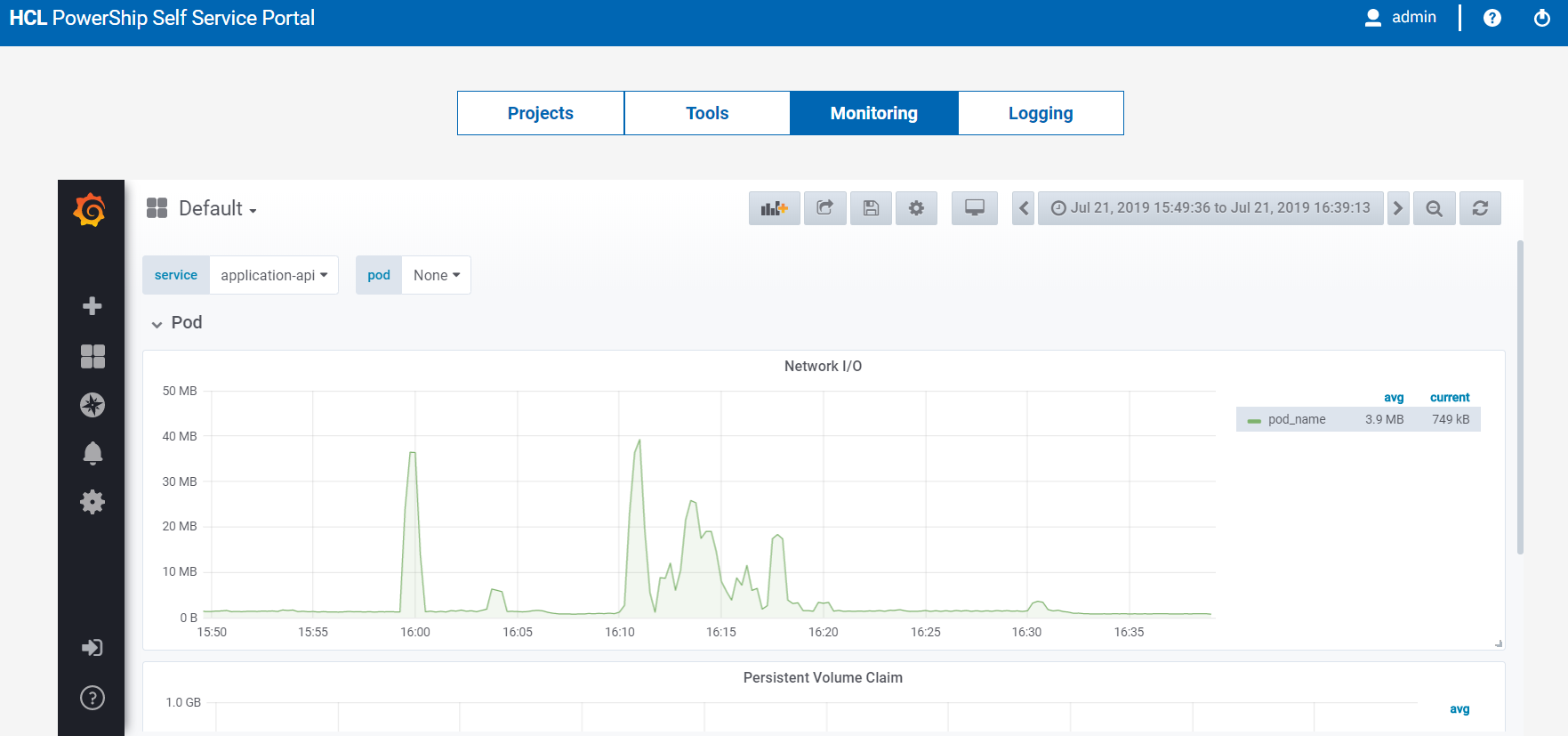
8.Your project is created in Jenkins and is visible in the Powership Dashboard.



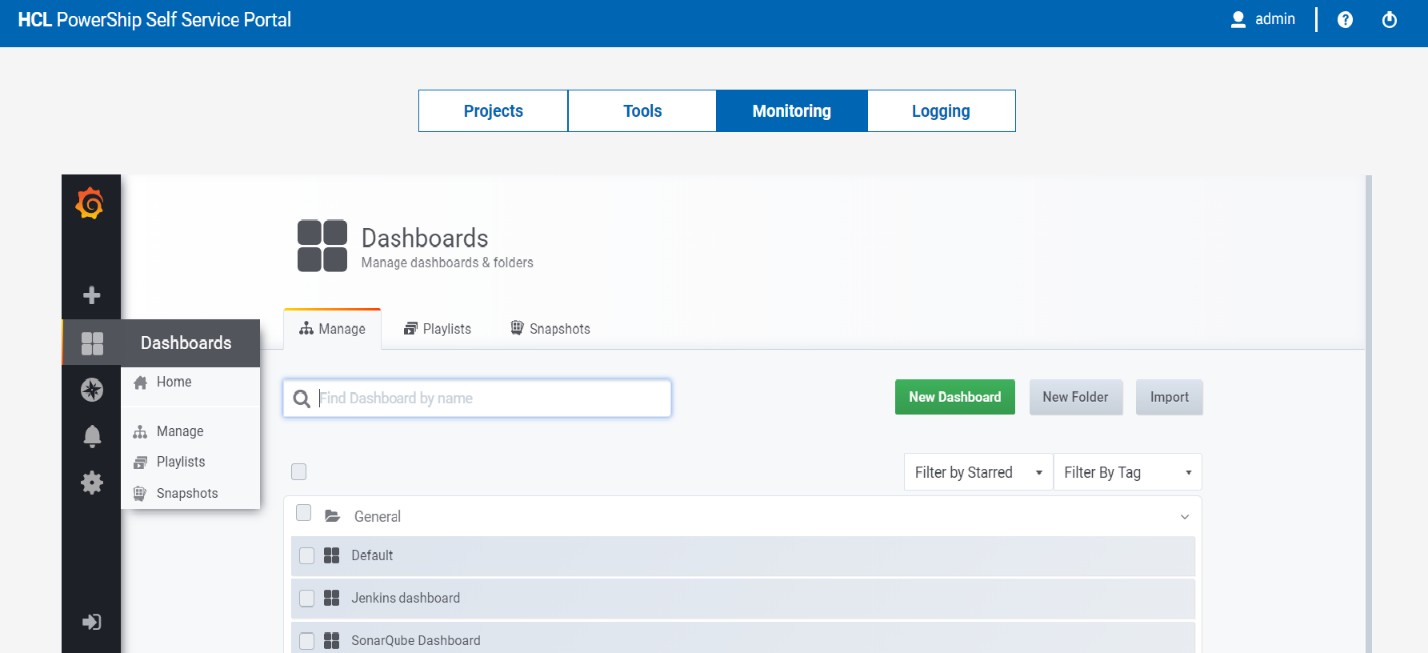
9. Follow the steps described in [Configuring the sample project](https://solutions.hcldoc.com/HCL_Powership/!SSL!/Responsive_HTML5/Getting_Started_with_Powership/Configuring_the_sample_project.htm) to configure JKEBanking project.

# **Monitoring**

Powership uses Grafana tool for monitoring the Kubernetes cluster pods and containers, Jenkins server details and SonarQube metrics for each project. The Grafana monitoring dashboard will look like the below,



To switch to Jenkins and Sonarqube dashboards, click the “Manage” option under the Dashboard as below and choose the respective dashboard.



# **Logging**

# **The Logging feature will give the detailed logging information of each pods, services and volumes. This we can get it in the Powership “Logging” tab. In the sample logging window will look like,**

# 

# **To get the pods/services log details, choose “Pods” and click the “Notepad” icon as below to see the individual of the pod/services,**

# 

# **Jenkinsfile** modification

# **Powership creates Pipeline based jobs in Jenkins. It automatically creates the “Jenkinsfile” and pushes the same to the user specified Git repository. By default, it has the template/place holders for various pipeline stages. The user must modify the values according to their project. Please refer Jenkinsfile which is in your GIT repository after creating the New project in Powership and change the same as per the below instructions.**

# **Stage(‘SCM’)**

# 

# **From the above code snippet change the below place holders,**

# **<BRANCH\_NAME> - The actual branch name of the Git repo which you want to use it for the build.**

# **<GIT\_REPO\_URL> - The actual GIT repo url in the form of “ssh”**

# **<GIT\_CREDENTIAL\_ID> - The user credential ID which you have provided while create New project in Powership.**

# **Stage(‘Build’)**

# 

# **By default, “package” goal mentioned for the Maven build. If you want to change it to any other Maven goal, you can change otherwise leave as it is.**

# **Stage(‘Cucumber’)**

# 

# **This “Cucumber” stage will help you in executing cucumber test scripts if you have it in your project. If you do not have the cucumber test scripts, comment/remove this stage from the Jenkinsfile.**

# **From the above screen shot replace the cucumber test class name instead of “<CUCMBER\_TEST\_CLASS\_NAME> and update the jsonReportDirectory value as well.**

# **Stage (‘SonarQube Analysis’)**

# 

# **This stage will execute the Sonarqube analysis of your maven project. Please change <PROJECT\_KEY> and <PROJECT\_NAME> as per your project details.**

# **Stage(‘Appscan’)**

# 

# **This stage will execute the Appscan analysis of your project. Appscan used to perform static (code level) and dynamic (Web application level) analysis. The above code snippet we can get it from “Pipeline syntax” of the Jenkins instance. In the Pipeline syntax, choose “appscan” and fill the required fields to get the actual code snippet for the Appscan analysis.**

# **Stage(‘Publish Artifacts to UCD’)**

# 

# **From the above screen shot, update the yellow highlighted values based on the “Urbancode Deploy” server details which is specifically created for your Project. Please refer “New project section” of this manual for more details.**

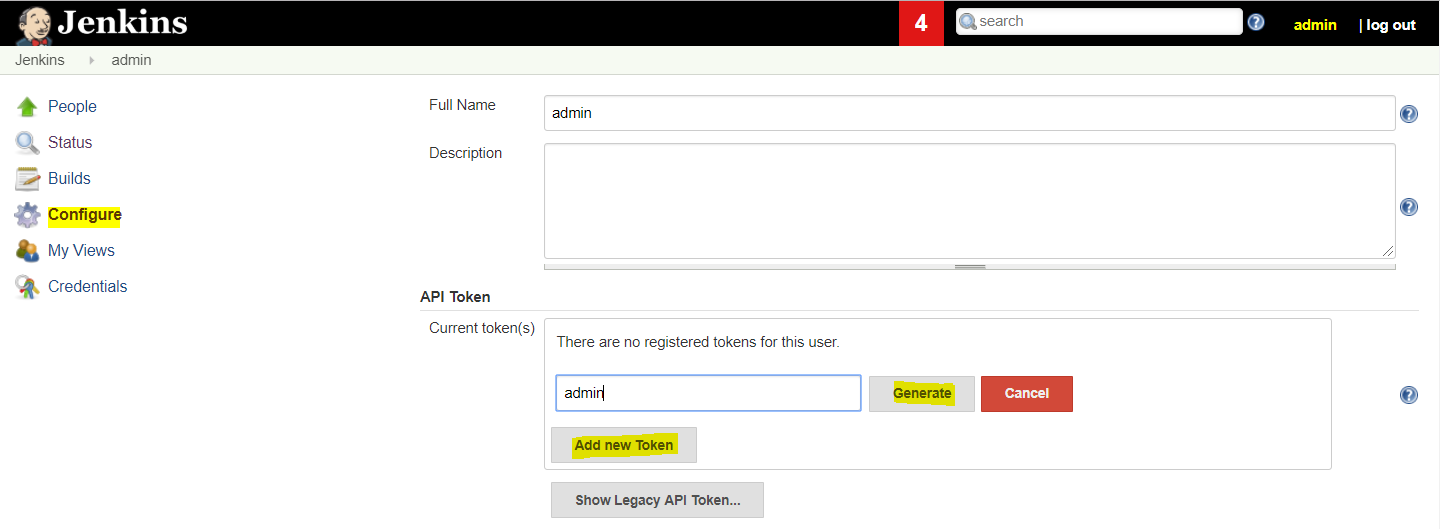
# **Stage(‘HCL One Test’)**

# 

# This stage will be used to execute HCL Onetest script of your project**.** You must upload HCL OneTest scripts from Powership UI to run them successfully.

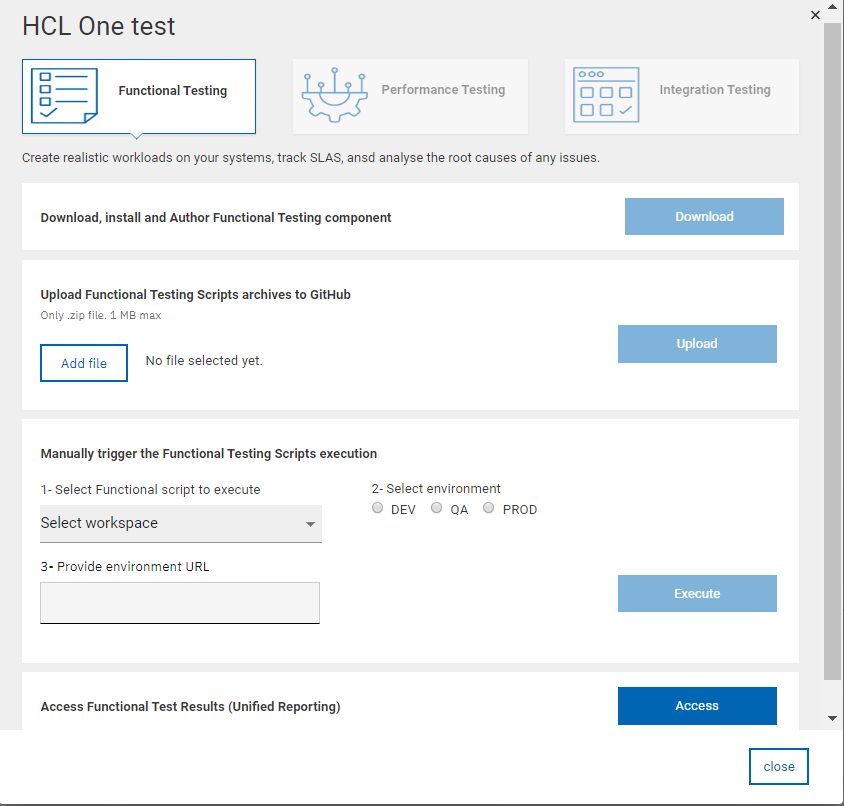
**Jenkins API token creation:**

1. To create Jenkins API token, login to Jenkins as Admin.
2. Click on the “ username “ link in the Jenkins UI and click on “configure”.
3. Then click on the “Add new Token” button in the configuration page, provide a unique token name and click “Generate” to generate the API token for admin user.
4. Once the token is generated, click on save and apply.
5. Use the generated token in “Powership” while creating “New Project”.



**HCL Onetest script execution:**

HCL Onetest scripts can be executed from Powership. This can be achieved by clicking the “HCL Onetest” in the Powership Projects page and this will open the below screen



As of now Powership supports Functional Testing scripts execution.

**Download:**

1. Login to the HCL Products & Platforms customer portal hosted by Flexera at:  [https://hclsoftware.flexnetoperations.com/flexnet/operationsportal/startPage.do](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fhclsoftware.flexnetoperations.com%2Fflexnet%2Foperationsportal%2FstartPage.do&data=02%7C01%7Csaravanakumar.velus%40hcl.com%7C24a059525eb745fa698f08d71a5a676e%7C189de737c93a4f5a8b686f4ca9941912%7C0%7C0%7C637006847890005238&sdata=1eZayPZ1FAQLJVOv1kGj23T%2FeZB%2BZfEykE1g9C6fDDw%3D&reserved=0)

using the following credentials:

Login: [HCLTestingProductsInternalAccount@hcl.com](mailto:HCLTestingProductsInternalAccount@hcl.com)

              Password: **Test2018**

1. From the top menu bar, click Downloads -> List Downloads
2. Click Test
3. Select product
4. Select files and click Download Selected Files

(Ignore the Readme file, as internal employees should refer to the internal Readme file referenced above.)

**Upload:**

In this section, we can upload the Functional test scripts which we have created for the application by using the “Add file” button.

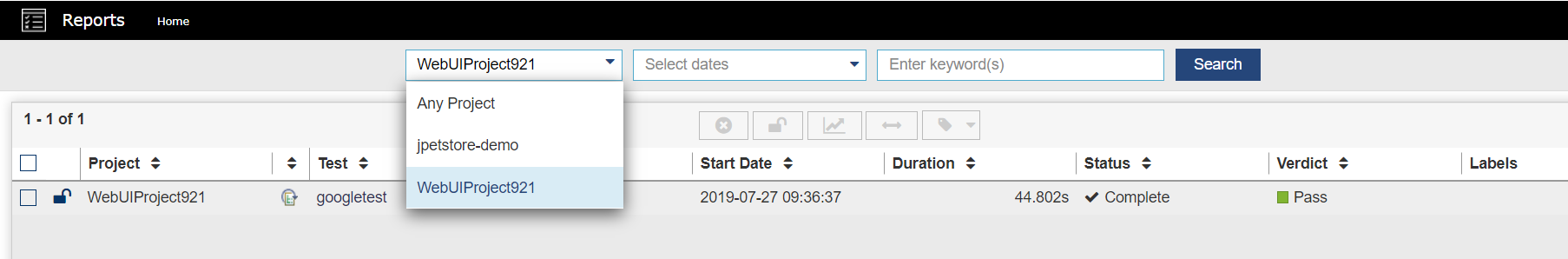
Once the file is added we can click “Upload” button, to upload the scripts to the test engine.

**Execute:**

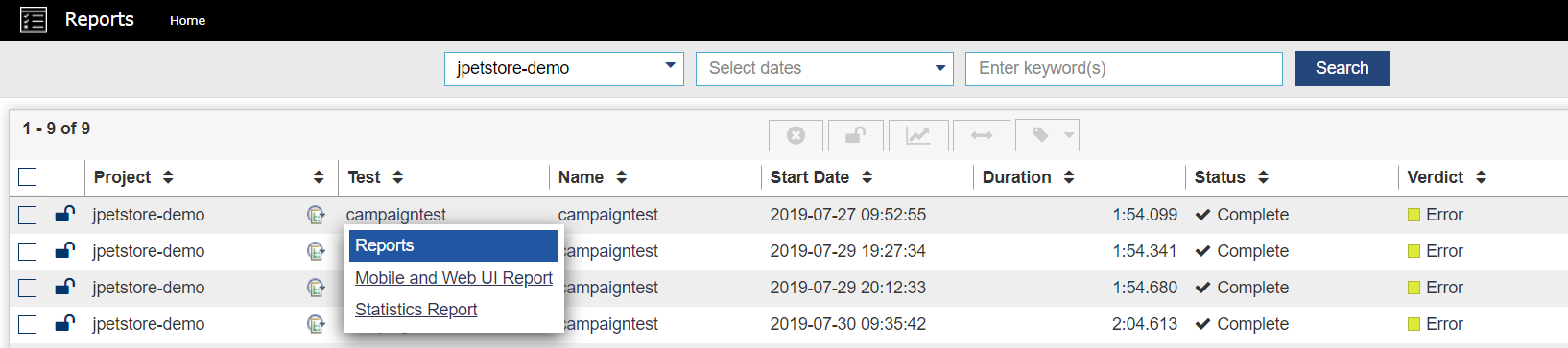
In this section, select the workspace where the test scripts are uploaded and select the environment and provide the environment url to execute the tests.

**Access:**

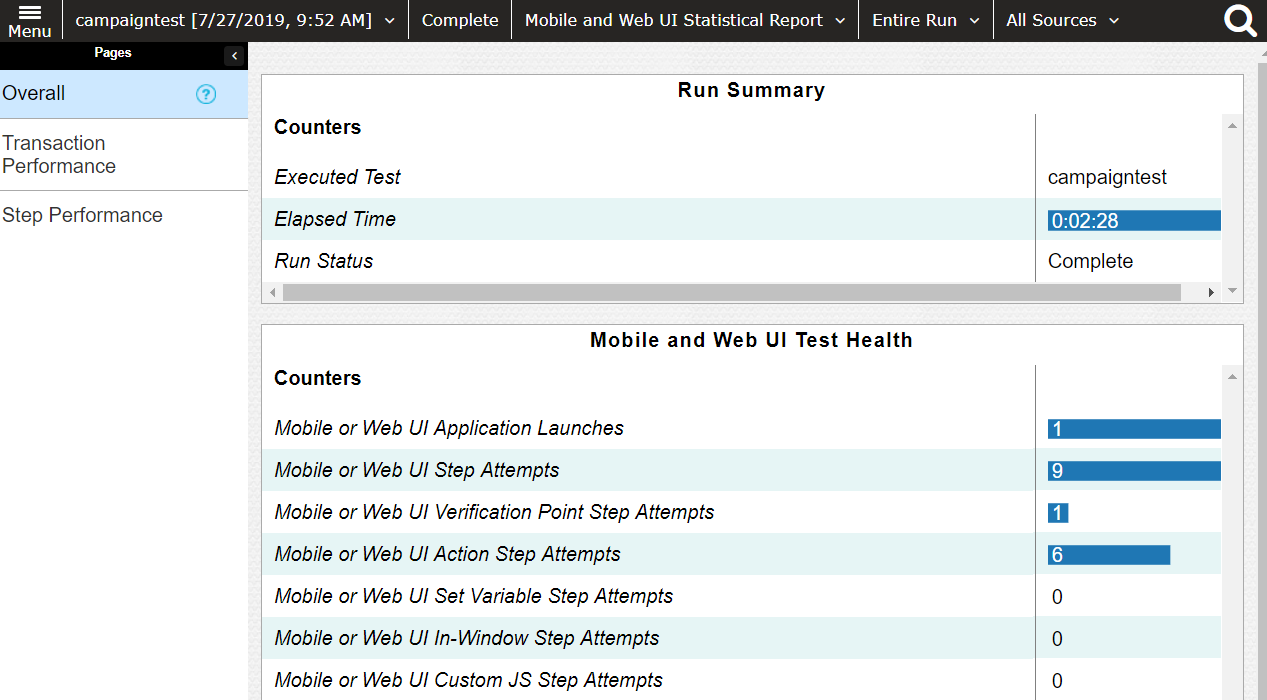
While clicking the “Access” button, it opens the HCL Onetest Unified reporting, where we can see all the test results for our projects.



From above screen, select the specific project and click “Search” to view the test results.



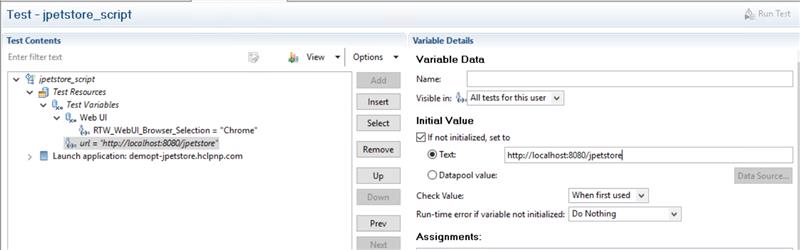
This will get two types of reports as shown above and click the respective link to view the detailed test results.



**Note:**

* + 1. While recording the user must create a Test Variable with name as "url". This will be replaced by Powership UI while executing the scripts.

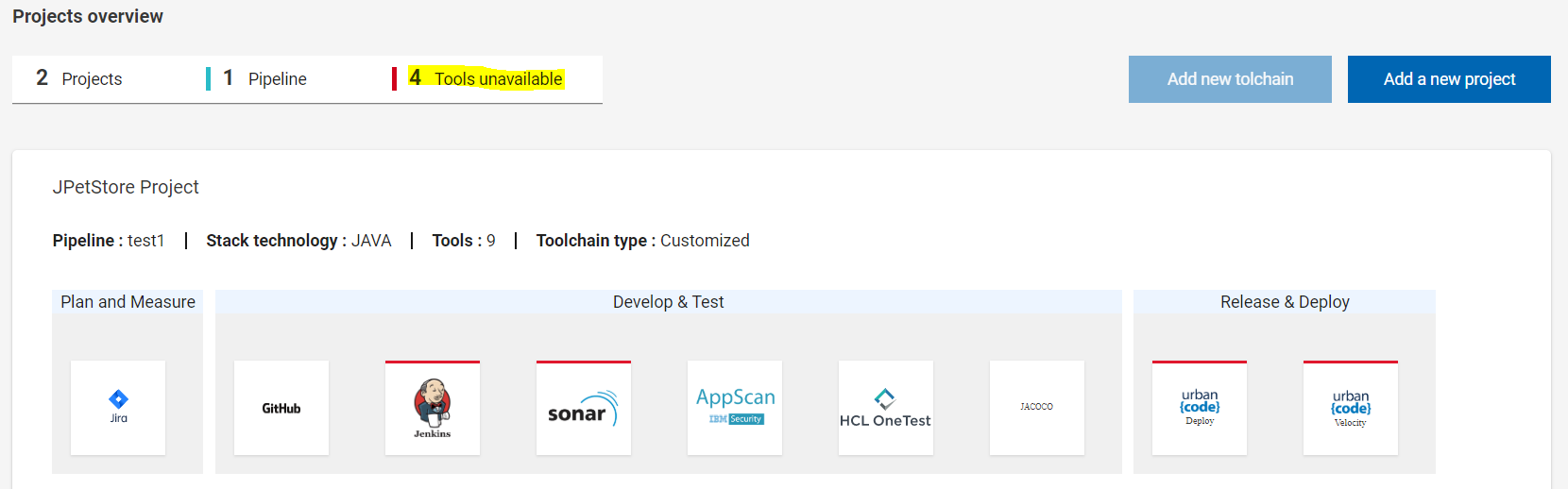
For eg: http://ip:8080/jpetstore



* + 1. Depends on the script size “Upload” and “Execute” functionalities will take time to get the results.

## **Troubleshooting**

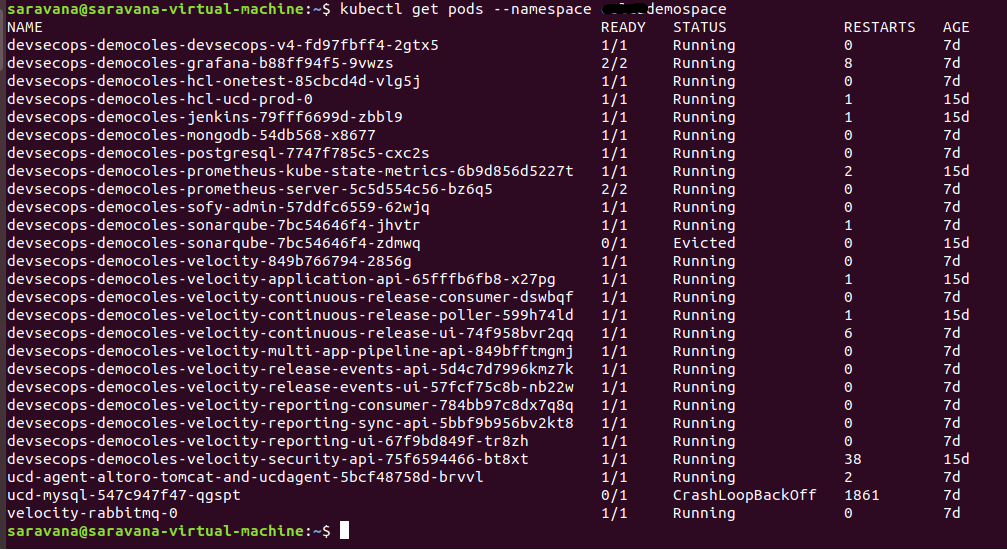
1. When creating a new pipeline, you might notice a red mark on top of some tool icons. This means that the tools are not available or not running in Kubernetes.
2. In the example below, the red mark is present for Jenkins, SonarQube, UCD and UCV tools.



* Run the command below and check if the pod is running.

**kubectl get pods –namespace namespace\_name**

* Wait until all tools are in running status



### **Where to find detailed logs of pods/services?**

Detailed logs of the pods/services are available in sofy-admin portal.

* Navigate to [https://sofy-admin.<LOAD\_BALANCER\_IP\_ADDRESS>.nip.io/#/](#/)
* To get the load balancer IP Address run the command

**kubectl get svc –namespace namespace\_name**

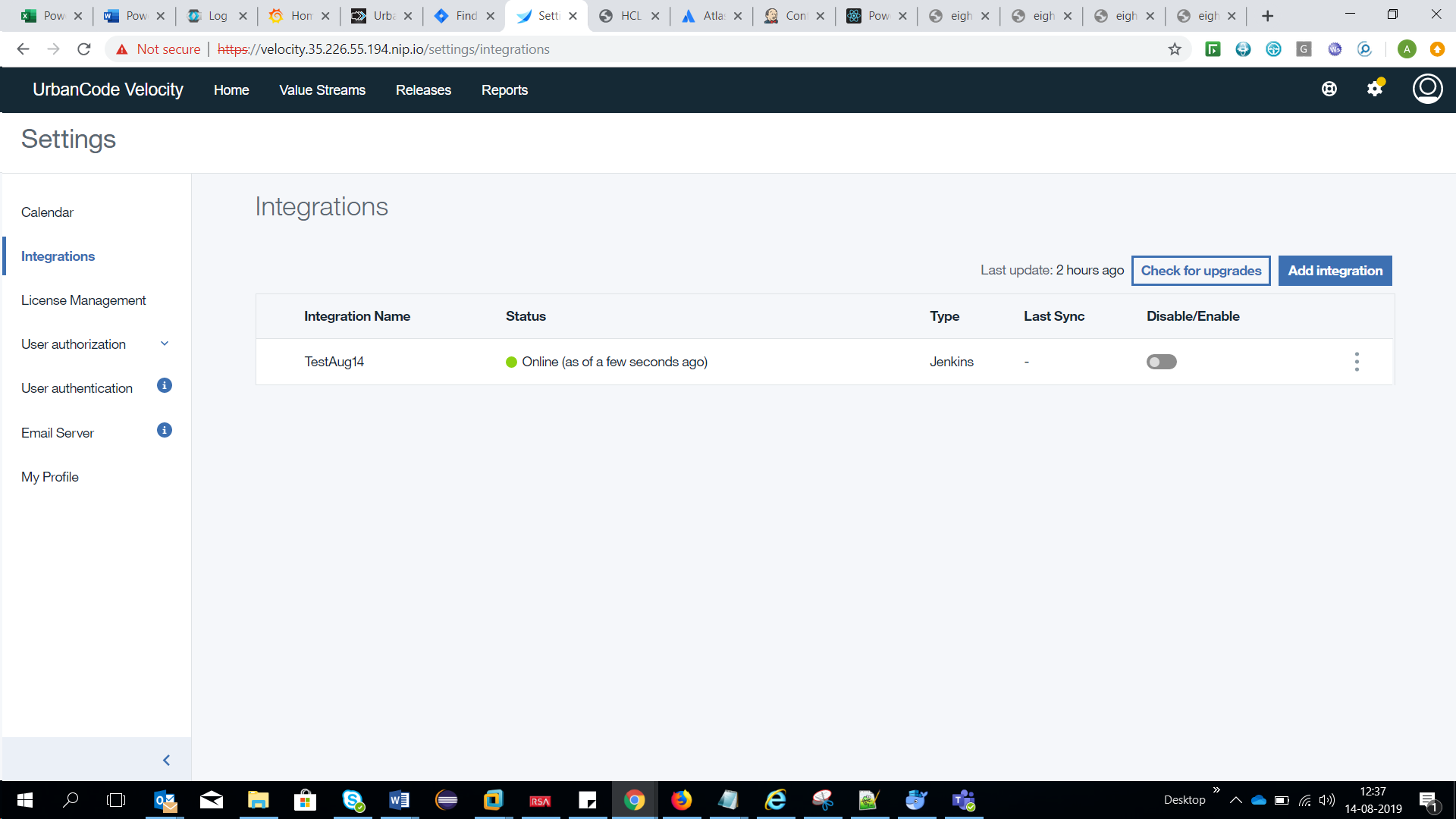
* and use the **EXTERNAL-IP** which is of type **LoadBalancer**.

1. In watcher application, if there is some problem in opening tools like

Jenkins, UCV or UCD.

Try to open the respective tools in incognito window.

1. Sometimes in UCV tool, Jenkin UCV integration is showing as offline status.(Note: this is bug in UCV tool).

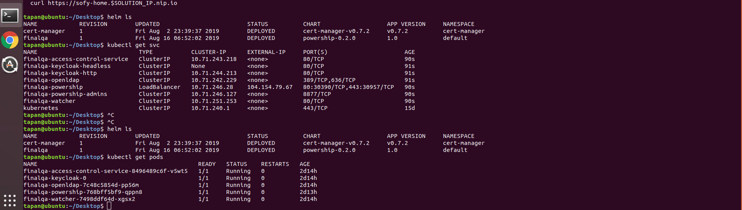


1. Sometimes while launching powership url(http://watcher.IP.nip.io) user may get this error.

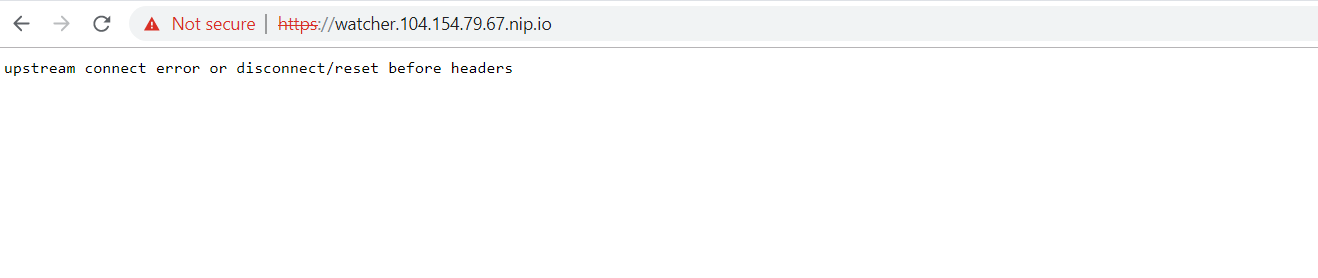
“upstream connect error or disconnect/reset before headers”.

Use the below command to check that all the pods running.

kubectl get pods



command : kubectl get pods



\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*