

**Installation and setup guide**

User Guide

NetApp-Jenkins Plugin User Guide

The NetApp-Jenkins integration is an end-to-end framework from creating code repository until zipping the successful builds in the artifactory location. All of these processes are run as containers using Docker and persistent NetApp storage with NetApp Docker volume plugin (nDVP).

**Quick Start**

**It is assumed that your environment consists of a VM or a physical host which is running a flavor of Linux. (All validation has been done with a RHEL 7.2 host and Google Chrome browser, use this as a best practice) This solution can be deployed on a VM or on a physical server.**

1. **Host requirements**

* RAM: 8GB and above
* Networking: Access to internet
* Operating System: Flavor of Linux (RHEL 7.2 recommended)

1. **Setup Docker engine on RHEL Host**

Follow the steps illustrated in the link below to setup the Docker engine:

<https://docs.docker.com/engine/installation/linux/rhel/>

1. **Install Netappdvp on RHEL Host**

Follow the steps illustrated in the link below to setup Netappdvp:

<https://github.com/NetApp/netappdvp>

Configure the ontap-nas.json file with the storage details of your ONTAP system.  
\*\*\* With NetAppDVP 1.2 we have to include

**“storageprefix”=**  “”

in the ontap-nas.json file to avoid the netappdvp prefix for volumes.

1. **Setup OnCommand API services 1.2RC1**

**\*\*\* It is a best practice to install APIs on a new RHEL VM with network connectivity and 4 gb or more RAM**

Download APIs 1.2RC1 from the link below. Navigate to OnCommand API services -> select platform -> Linux -> GO:

<https://mysupport.netapp.com/NOW/cgi-bin/software>

|  |  |  |
| --- | --- | --- |
| |  |  | | --- | --- | | **https://mysupport.netapp.com/images/trans_spacer.gifhttps://mysupport.netapp.com/images/right-arrow_small.gif OnCommand API Services 1.2RC1** |  | |

Click on View & Download

Follow the steps illustrated in the link below to setup the OnCommand API services 1.2 RC1. Navigate to 1.2RC1 -> All Documents -> Installation and setup guide:

<https://mysupport.netapp.com/documentation/productlibrary/index.html?productID=62040>

**As a summary of the documentation in the above link, the following are the steps that have to be followed:**

* Download the RPM: “mysql57-community-release-el7-8.noarch.rpm” from yum repo “<http://dev.mysql.com/downloads/repo/yum/>” into a convenient directory on your VM**\*\*\***
* Install this RPM using “yum install mysql57-community-release-el7-8.noarch.rpm”
* Run command “cd /etc/yum.repos.d”
* View contents of the repo file using “vi mysql-community.repo”
* In this file, Under the section, “mysql56-community”: make enabled=1, others should be 0, especially “mysql57-community” **\*\*\***
* Run command “yum repolist enabled” and check that mysql56-community exists
* Get RPM “java-1.8.0-oracle-1.8.0.71-1jpp.1.el6\_7.x86\_64.rpm” and install **(If your VM is running any JRE 1.8.0.71 or later, then skip this step)**
* Copy binary “OnCommand-API-services-1.2.0-XXXXXXXXXXX.bin” to /opt **( Get the binary file from the location -** <https://mysupport.netapp.com/NOW/cgi-bin/software> **)\*\*\***
* Run command “chmod +x <binary\_file>”
* Run binary using “./<binary\_file>”

\*\*\* Absolute requirement

1. **Adding an ONTAP storage controller as a data source in API services**

A successful installation of APIs 1.2 RC1 should display a screen as follows:

“OnCommand API Services installed successfully.

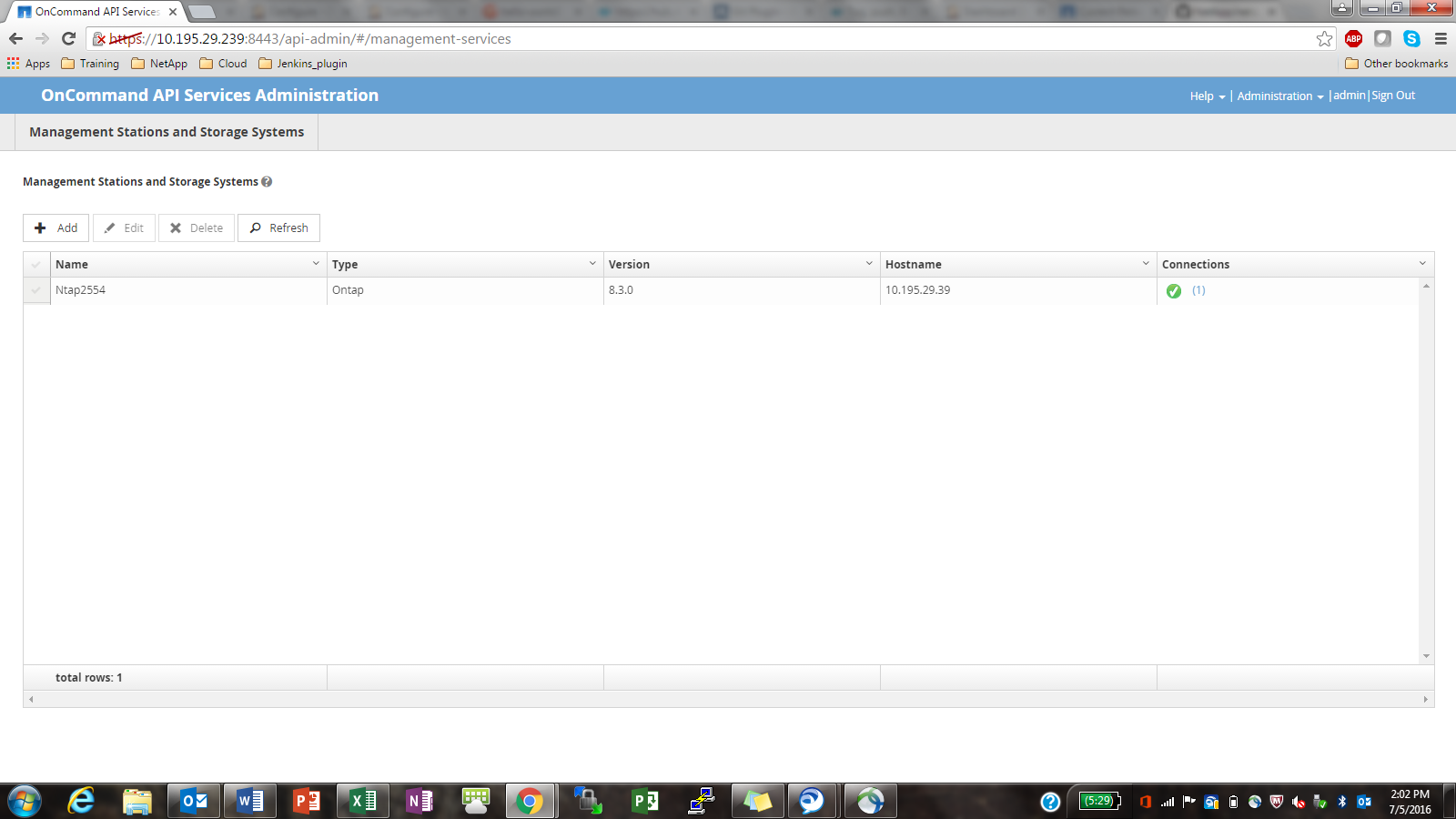
Use a web browser and one of the following URL(s) to configure and access the OnCommand API Services GUI.

<https://xx.xx.xx.xx:8443/api-admin/>

Use a web browser and the following URL to access OnCommand API Services documentation.

<https://xx.xx.xxx.xxx:8443>”

* Open a web browser and enter the URL “<https://xx.xx.xx.xx:8443/api-admin/>” where xx.xx.xx.xx is the IP address of the host machine where APIs is installed
* Enter the username and password chosen during the installation process to view the screenshot below



* Click on +Add and enter the details of your ONTAP Cluster
* Verify that the instance of ONTAP is added as shown in the above screenshot

1. **Verify if APIs 1.2 is working with Google Chrome Postman extension**

* Install the postman add-on on Chrome browser. Follow the link below to do so.

<https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=en>

* Change the request type to “get”
* In the URL space, enter https://{ip}:8443/api/1.0/ONTAP/volumes/ to get a list of volumes in the cluster
* In the headers option, add 3 of them

Key: Accept Value: application/json

Key: Content-type Value: application/json

Key: Authorization Value Basic [64-bit-encrypted user:password]\*\*\*

\*\*\*Remember to have logged into APIs dashboard through the chrome browser and keep the api-admin page open as shown in the above screenshot before running postman.

\*\*\* Go to <https://www.base64encode.org/> and click on the encode option. Enter the “username:password” in this exact format and click on encode to get the [64-bit-encrypted user:password]

1. **Pull the Netapp-Jenkins, Netapp-Jenkins\_slave and GitLab Docker images on your Host**

* On your RHEL VM, run the command “***docker pull devopsnetapp/netapp-jenkins***” to pull the Jenkins master container image
* Run the command “***docker pull devopsnetapp/netapp-jenkins\_slave***” to pull the slave container image
* Run the command “***docker pull devopsnetapp/netapp-jenkins\_gitlab***” to pull the SCM container image.
* Run the command “***docker images***” and verify if these three images show up on the list

1. **Run the Jenkins master on the RHEL VM using the command:**

***docker run -i -d -t --name=Jenkins\_master -v /var/run/docker.sock:/var/run/docker.sock -p 1024:8080 -p 50000:50000 devopsnetapp/netapp-jenkins***

* The Docker socket is parsed from the host to the Jenkins master container to facilitate a sibling container setup using the “***-v /var/run/docker.sock:/var/run/docker.sock”*** option
* The port 1024 on the host is mapped to port 8080 on the container which is the default Jenkins port
* The port 50000 on the host is mapped to port 50000 on the container to facilitate JNLP based master-slave interaction
* **Edit the ontap-nas.json file as per your environment. It is located at /etc/netappdvp on the master container.**

1. **Verify that the Jenkins master is running using the command “*Docker ps*” The output should be as follows:**

**CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES**

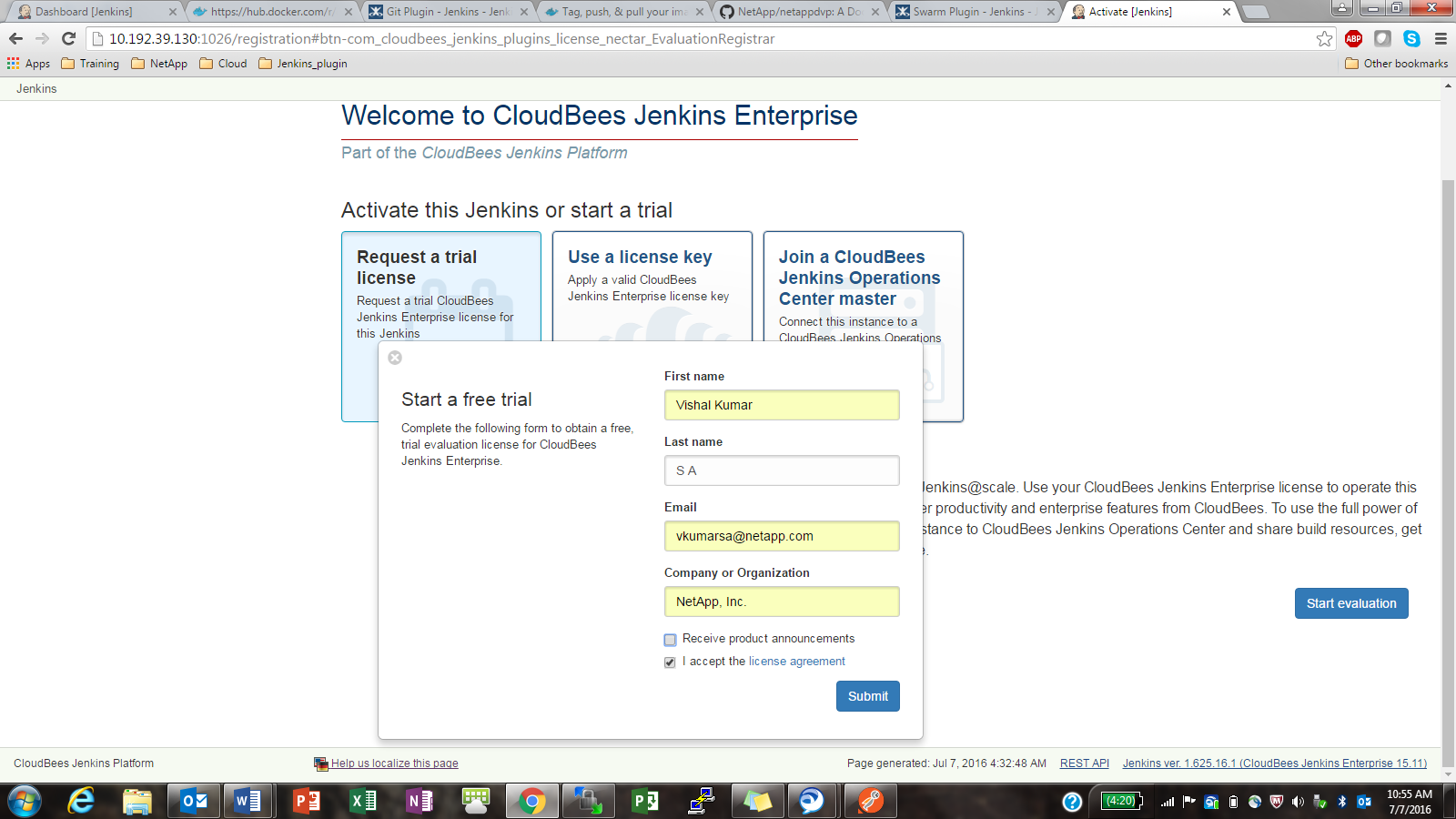
**60963927ce12 devopsnetapp/netapp-jenkins "/bin/tini -- /usr/lo" 40 hours ago Up 40 hours 0.0.0.0:50000->50000/tcp, 0.0.0.0:1024->8080/tcp elegant\_snyder**

1. **Accessing the Jenkins Dashboard from a web browser**

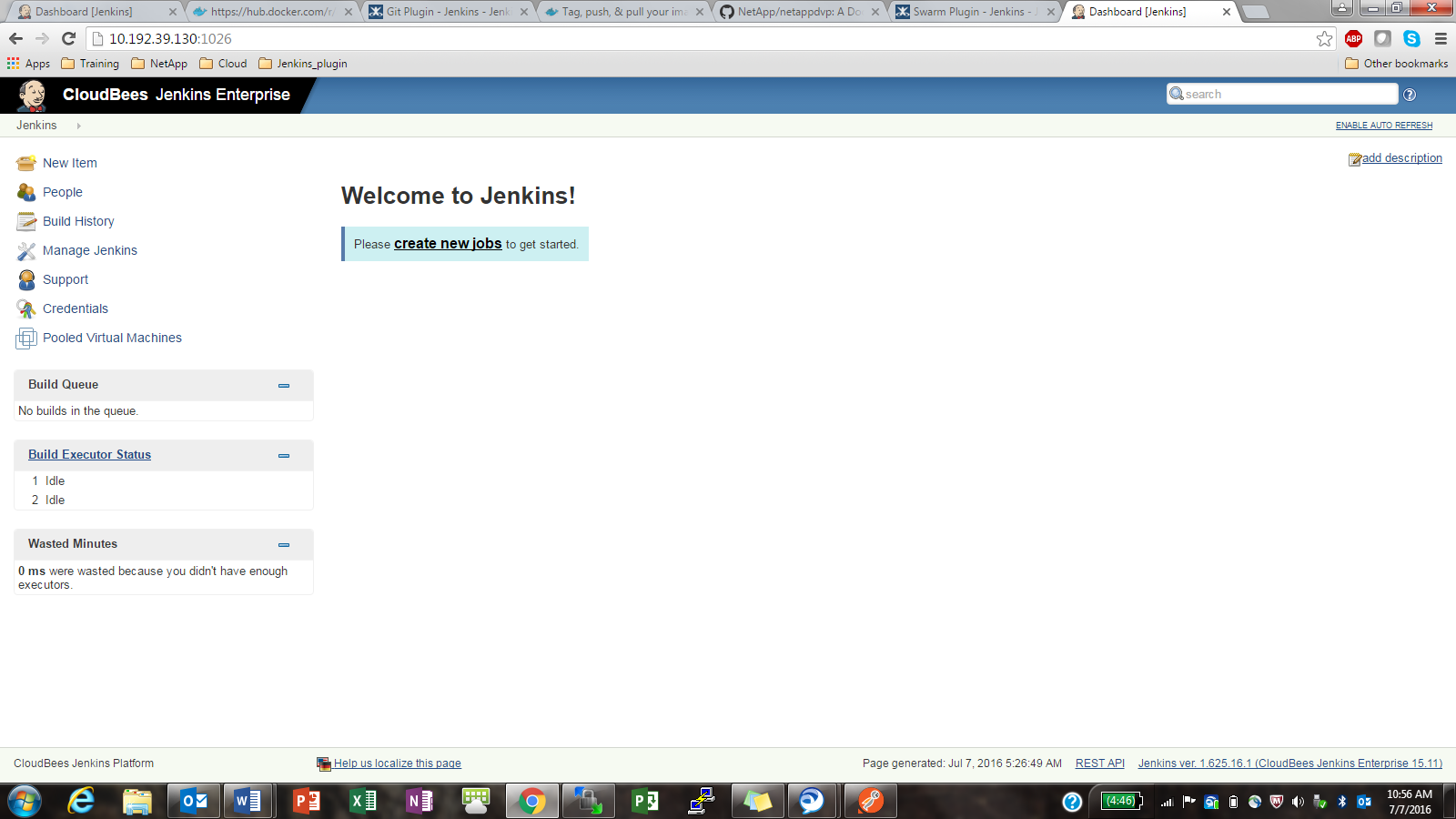
* Enter the “RHEL-VM-IPaddress:1024” on the web browser to view the license page for Enterprise Jenkins as shown below



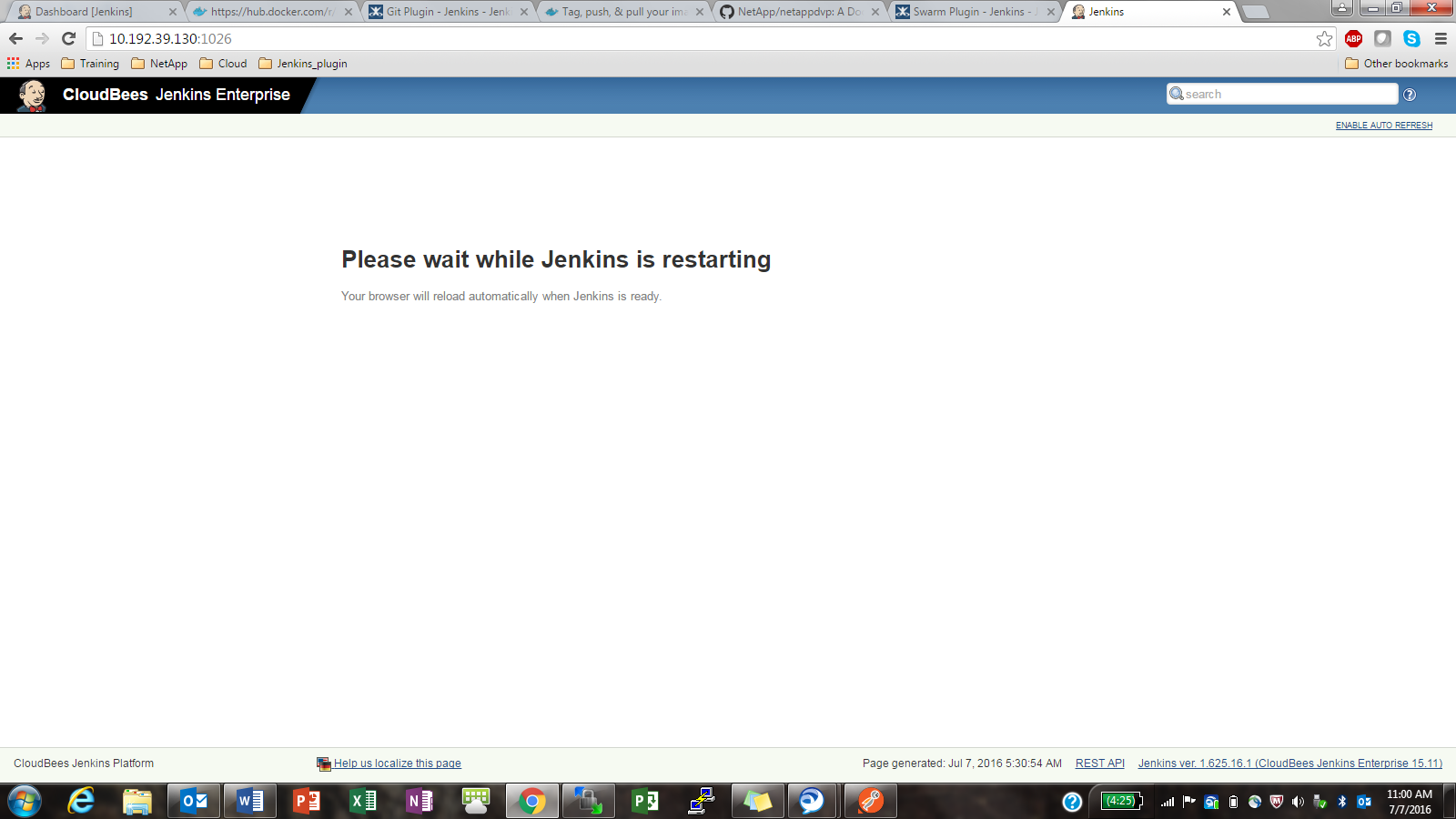
* Either request a trail license key or Use an Enterprise Jenkins license key
* When you request a trail license key, the tab as shown below pops up and fill out the required details



* Click on submit to view the Enterprise Jenkins Dashboard as shown below

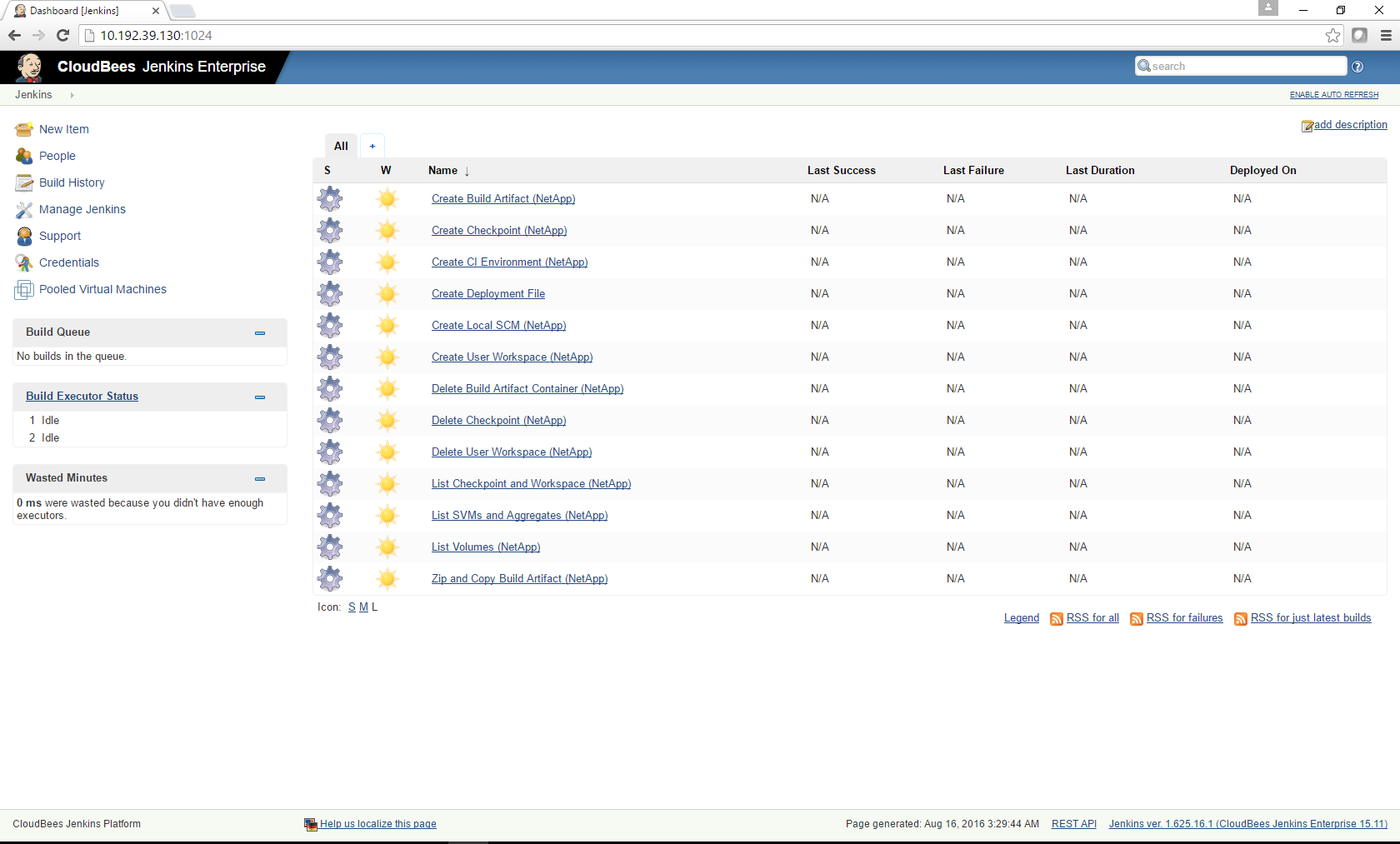


* In the URL space after the IP:1024, type in /safeRestart for the configurations to take effect. Eg. [***http://10.192.39.130:1024/safeRestart***](http://10.192.39.130:1024/safeRestart)
* The screen below appears and wait for Jenkins to restart



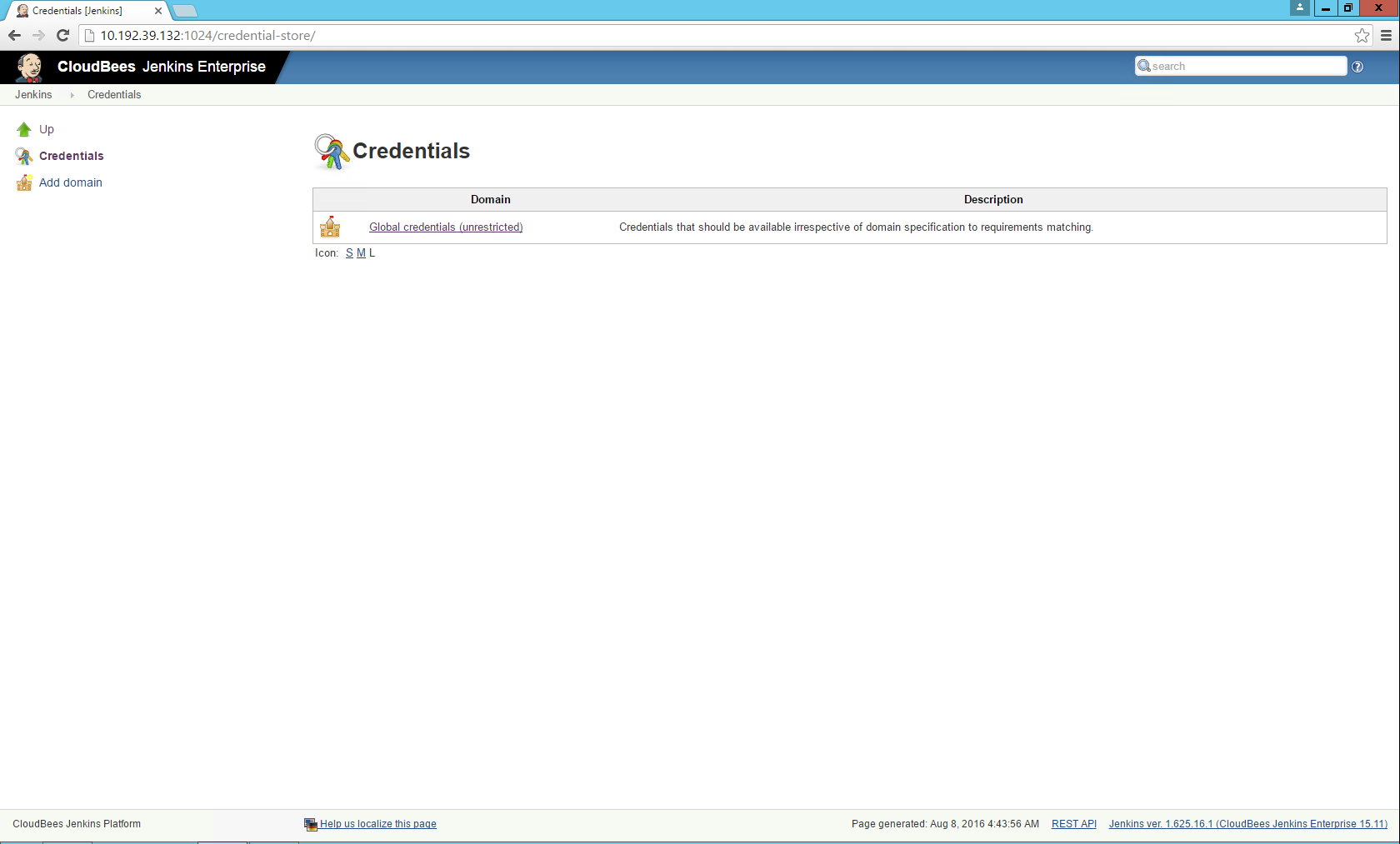
\*\* If this is taking too much time, open a new tab and enter the IP\_address:1024 after a minute or two.

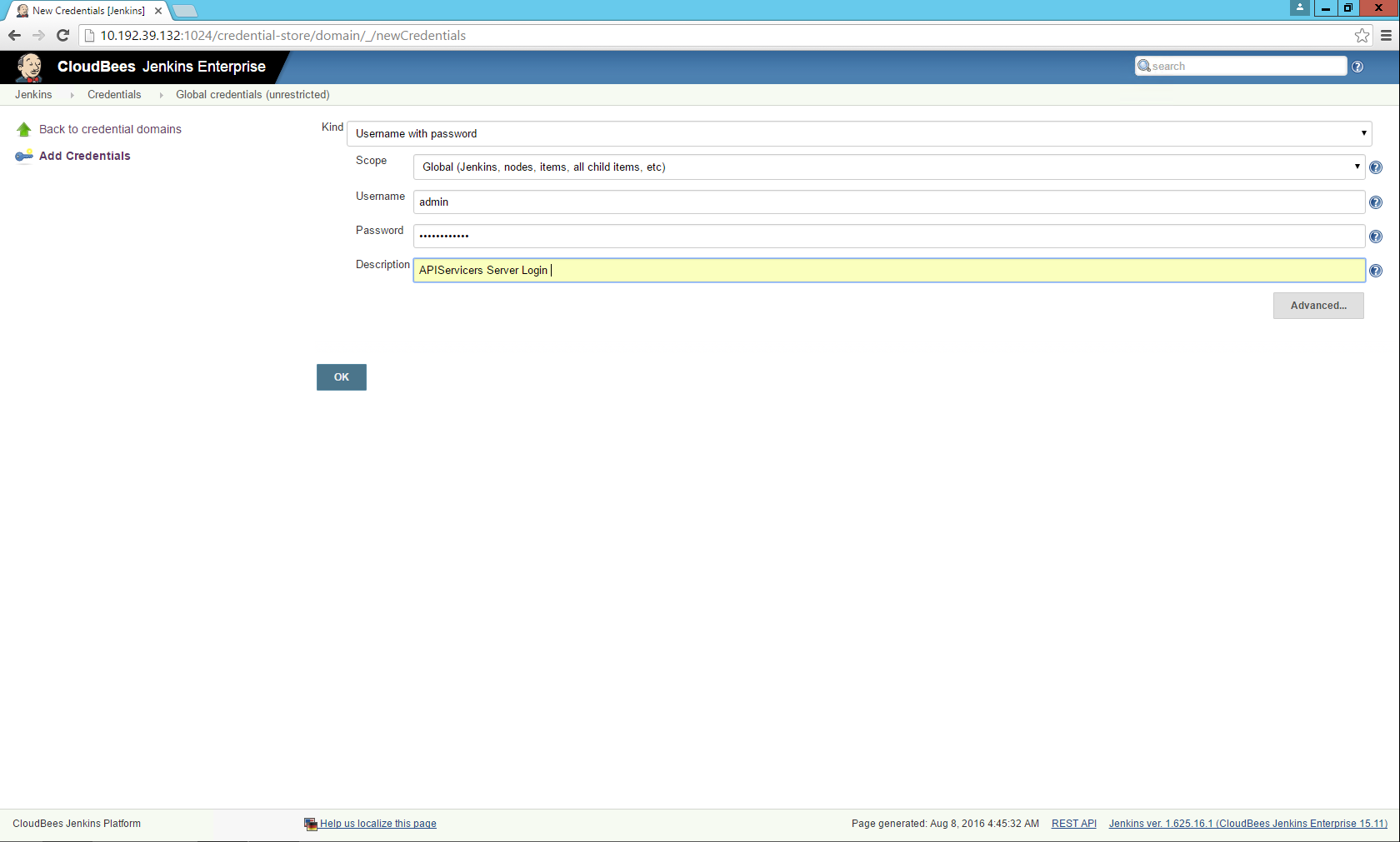
* All the builder templates will appear on the dashboard and are ready to use as below

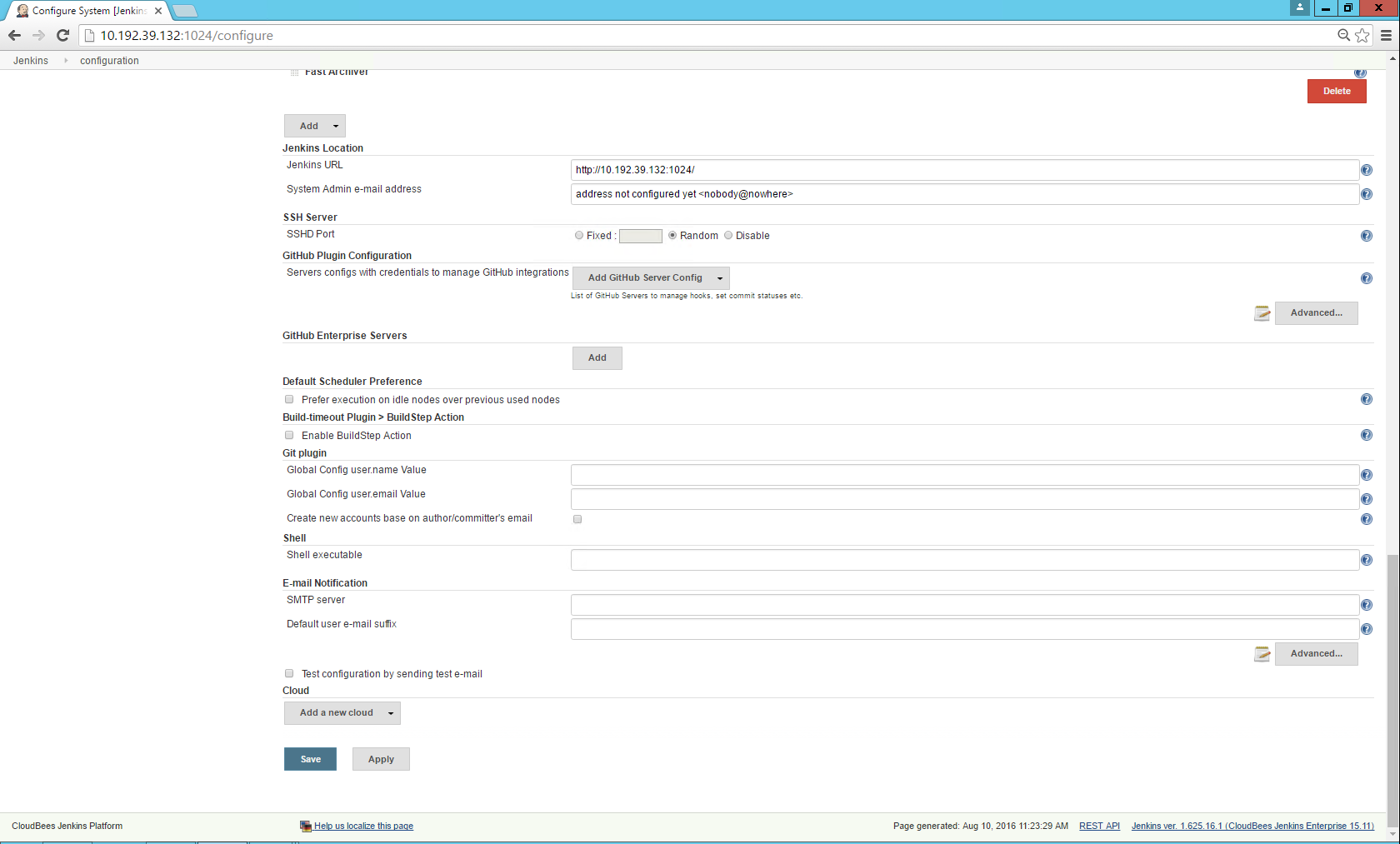


1. **Setting up API services username & password & Saving the JenkinsMaster URL\*\*\*\***

* Click on the credentials link on the left side column on the Jenkins Dashboard
* Click on Global credentials
* Click on add credentials on the left side
* Enter the APIs username and password along with a relevant description

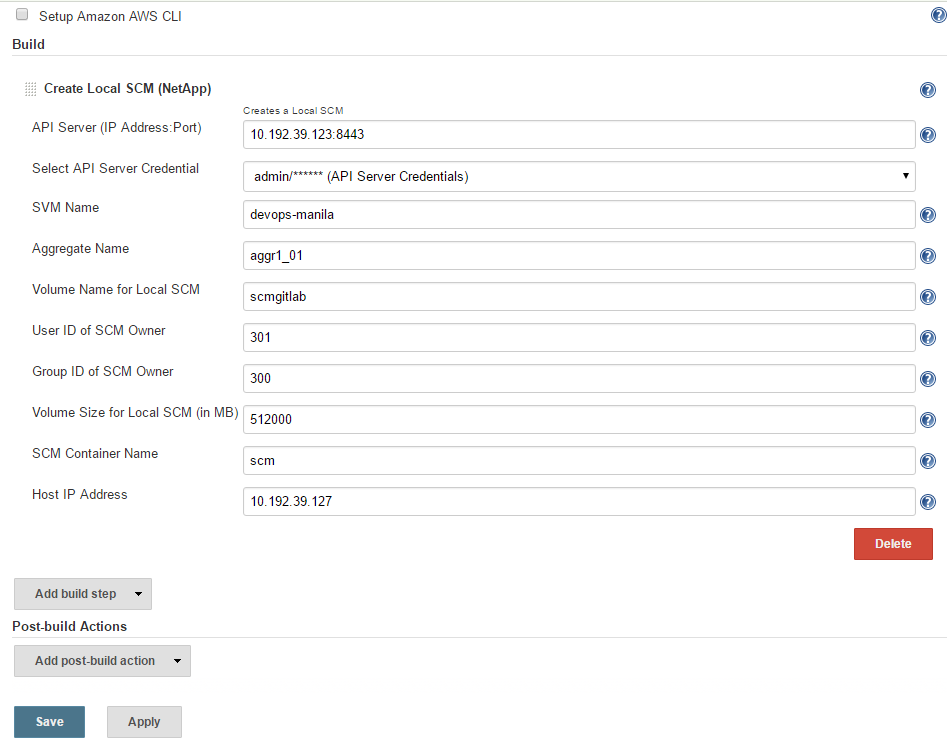




**11.1 SAVE THE JENKINS MASTER URL :** By default the Jenkins master URL will appear in Manage Jenkins > Configure System  
  
Click Save \*\*\*\*\*\* (This is mandatory)  
 ****

1. **Setting up your CI environment**
2. Launching the GitLab local repository on NetApp.

* Create a new job by clicking on new item on the Jenkins dashboard
* Enter item name eg. “**Demo1**” and Select the first option > Freestyle project and click on >OK
* Scroll down the configuration page and click on “add build step” drop down list
* Search for “**Create Local SCM(NetApp)**” in the list and select it
* A configuration tab as shown in the figure below appears.



* Enter the APIs IP-address:port, select the APIS user/pass from the dropdown list, and enter the SVM and aggregate names
* Enter the name of the volume to run the local SCM on. **eg. scmgitlab**
* Enter the UID and GID of the volume to be created

\*\*\* Verify that this UID and GID is created on the ONTAP controller

* Enter the size of the volume in MBs and give it a simple container name **eg. scmgitlab**

\*\*\* Verify that Netappdvp is running on the host before building this project (2 instances of NetAppdvp should be running, to check use :

>ps ax | grep netappdvp)

If not then run NetAppDVP:

>**netappdvp --config=/etc/netappdvp/ontap-nas.json &**

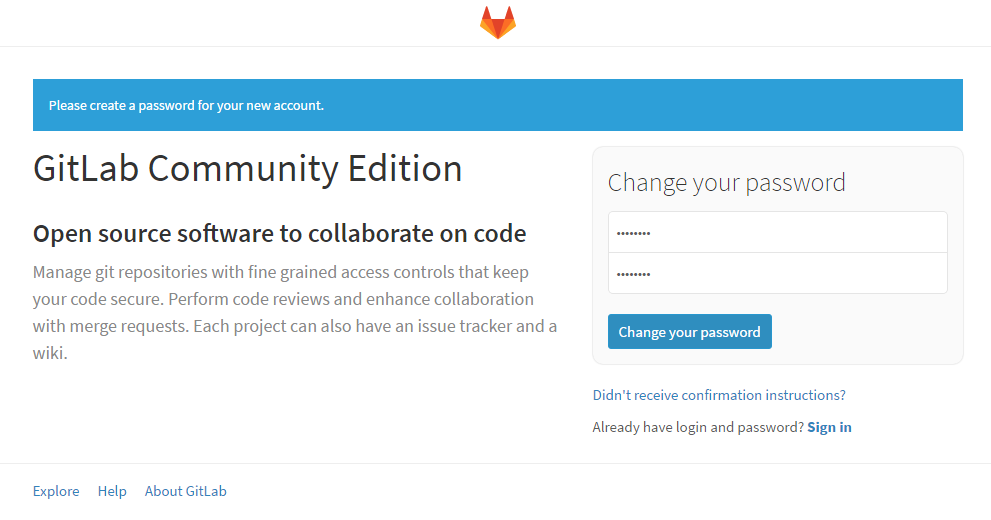
\*\*\* Run “**df**” command on host and verify that there are not stale file handles. If there are then unmount that path and continue

* Here the **Host IP Address** is the IP address of the server on which the GitLab container will run.
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process
* To verify the running of the GitLab container, access the host and give a “**docker ps**” command to view the running containers (make note of the container-id)
* For further verification, enter the context of this container using “**docker exec –it [container-id] /bin/bash**”and enter the command “**df**” to see if the Netapp 3 separate volumes mounted on to the container. One volume for data, second volume for config (tagged as \_config) and third volume for logs (tagged as \_logs)
* Once the job is successfully built, open up the browser and go to http://<Host\_IP\_Address>.

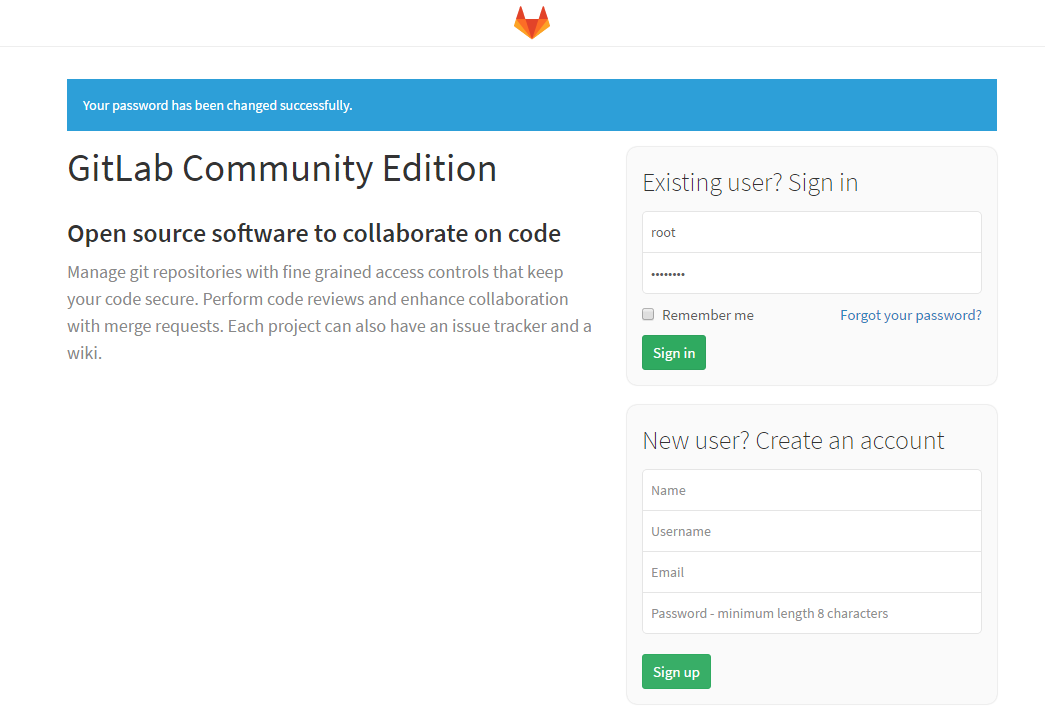
(Port 80 is being used to serve the web interface)

Note: The page can take a few minutes to load.

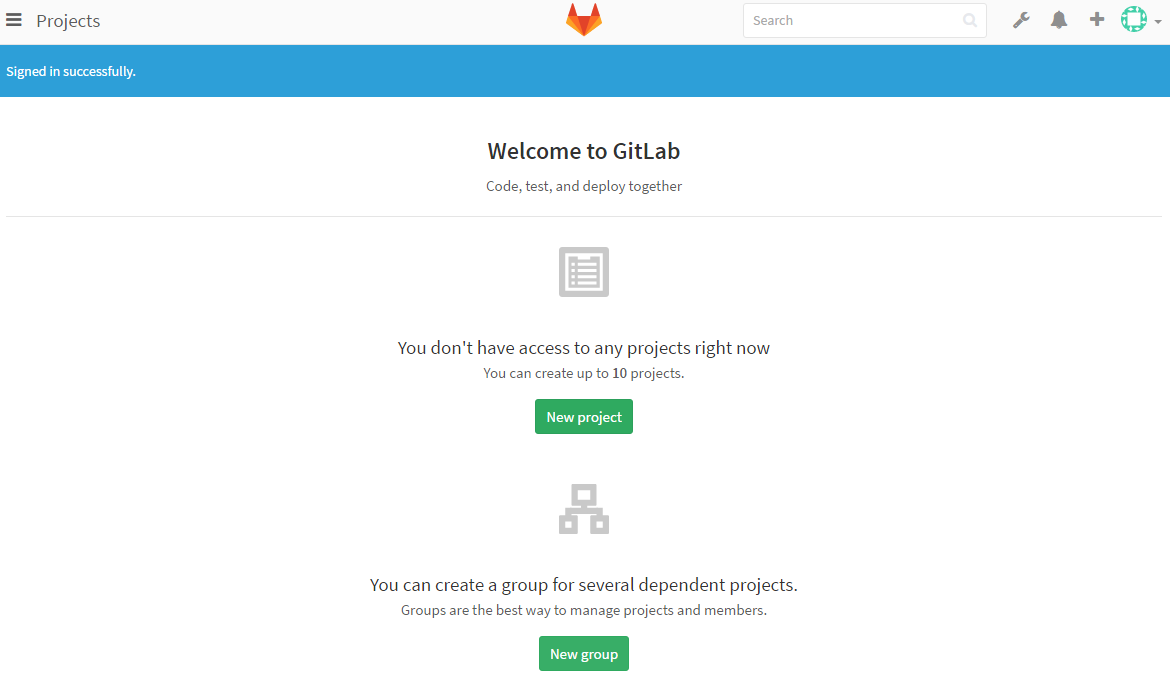
* The following screen should appear. Enter a new password.



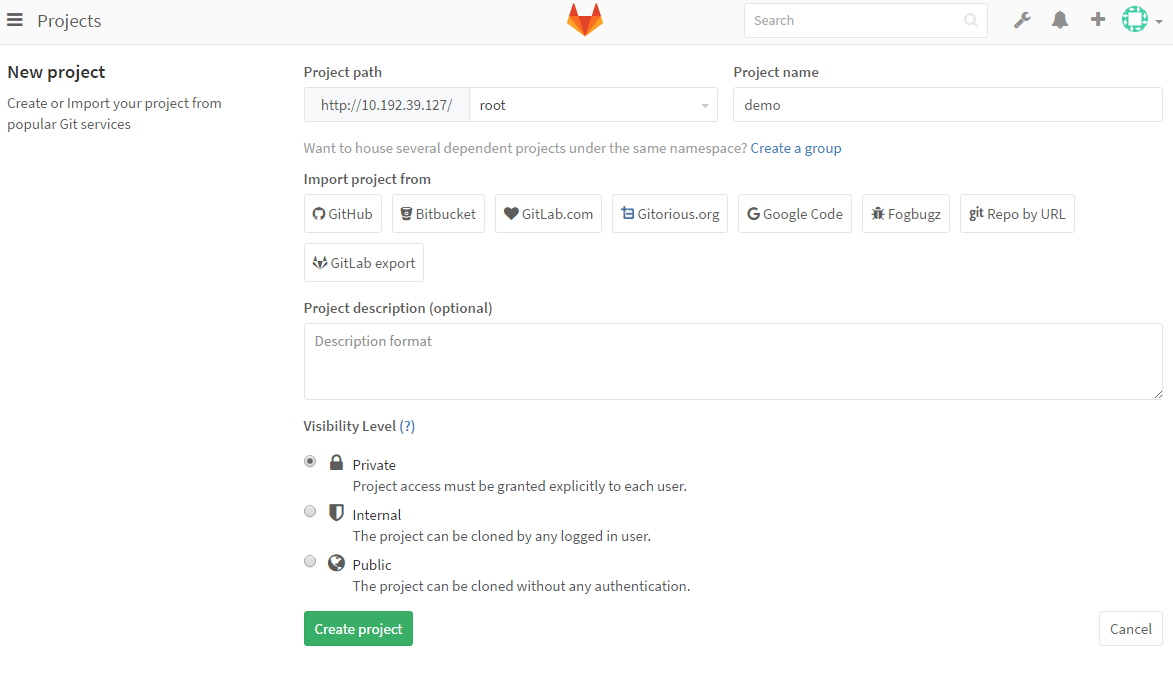
* On the next page, sign in into the dashboard. The default username is **root.** This can be modified later.



* On the next screen, click on **New Project.**

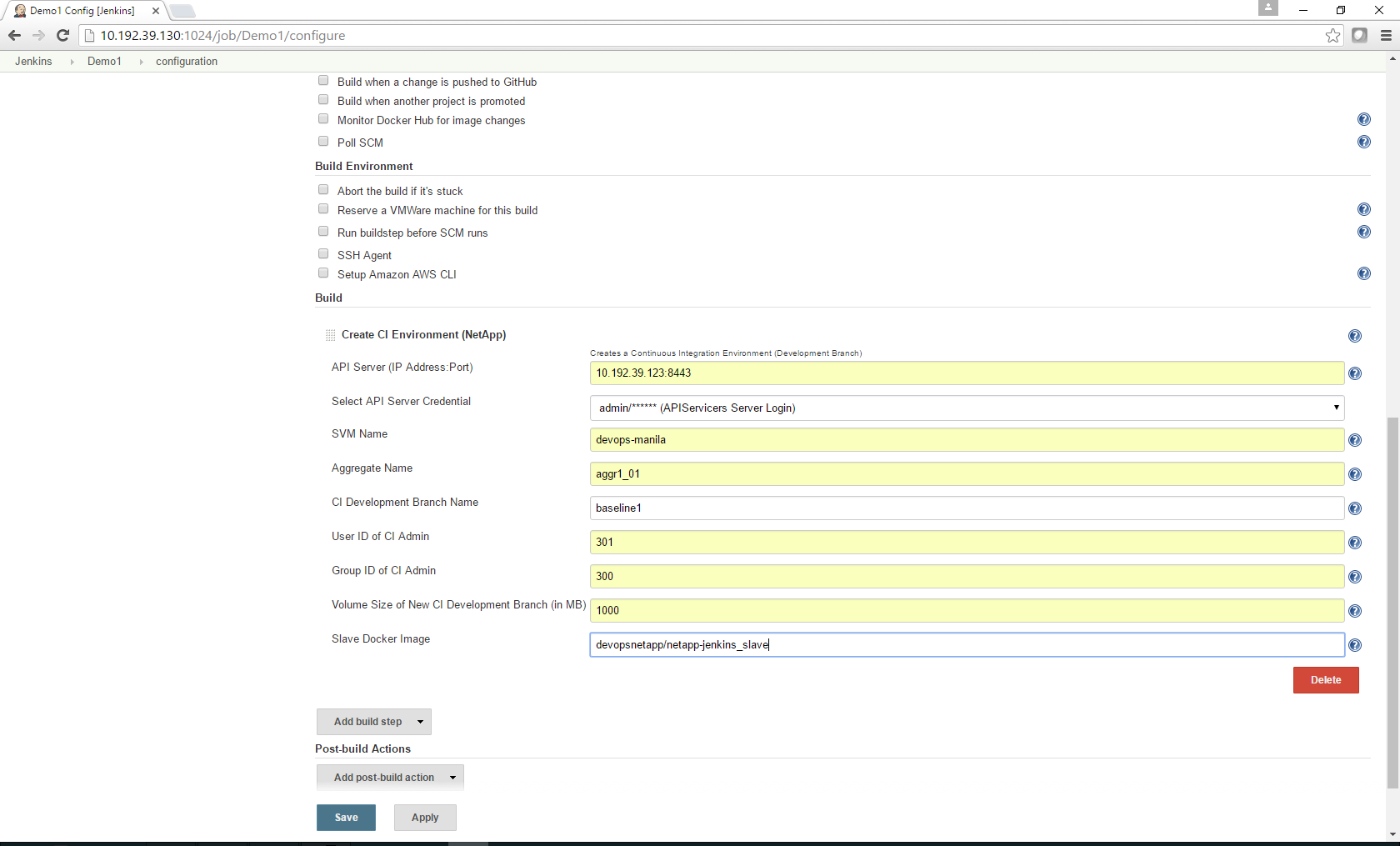


* Provide the requested details and a new local repo will be ready for use.



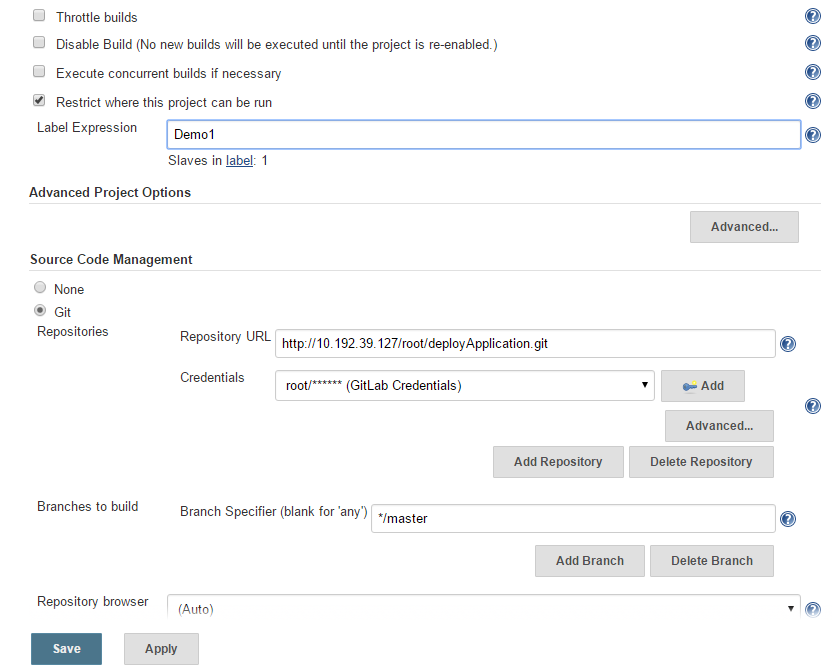
1. Creating the Baseline container with NetApp volume mounted

* On your Jenkins Dashboard click on the “Demo1” project which was created as part of step1) and scroll down to build step
* Delete the existing build steps by clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**Create** **CI Environment (NetApp)**”
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the SVM and Aggregate name to create the volume from ( To view How to list all the available SVMs and Aggregates in the cluster from the Jenkins Dashboard, skip to step17))
* Enter the name of the new Development branch to create **eg. baseline1**
* Enter the GID and UID for the volume (\*\*\* This Unix user and group has to be created in advance on the ONTAP controller)
* Enter the size of the new volume to be created in MBs
* Enter the name of the slave docker image as “devopsnetapp/netapp-jenkins\_slave”
* Refer to the following screenshot to see how the configuration should look like
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process
* To verify the running of the Baseline container, access the host and give a “**docker ps**” command to view the running containers.
* For further verification, enter the context of this container using “**docker exec –it [container-id] /bin/bash**”and enter the command “**df**” to see if the Netapp volume is mounted at /workspace/Demo1



1. Using the Baseline container with NetApp volume mounted

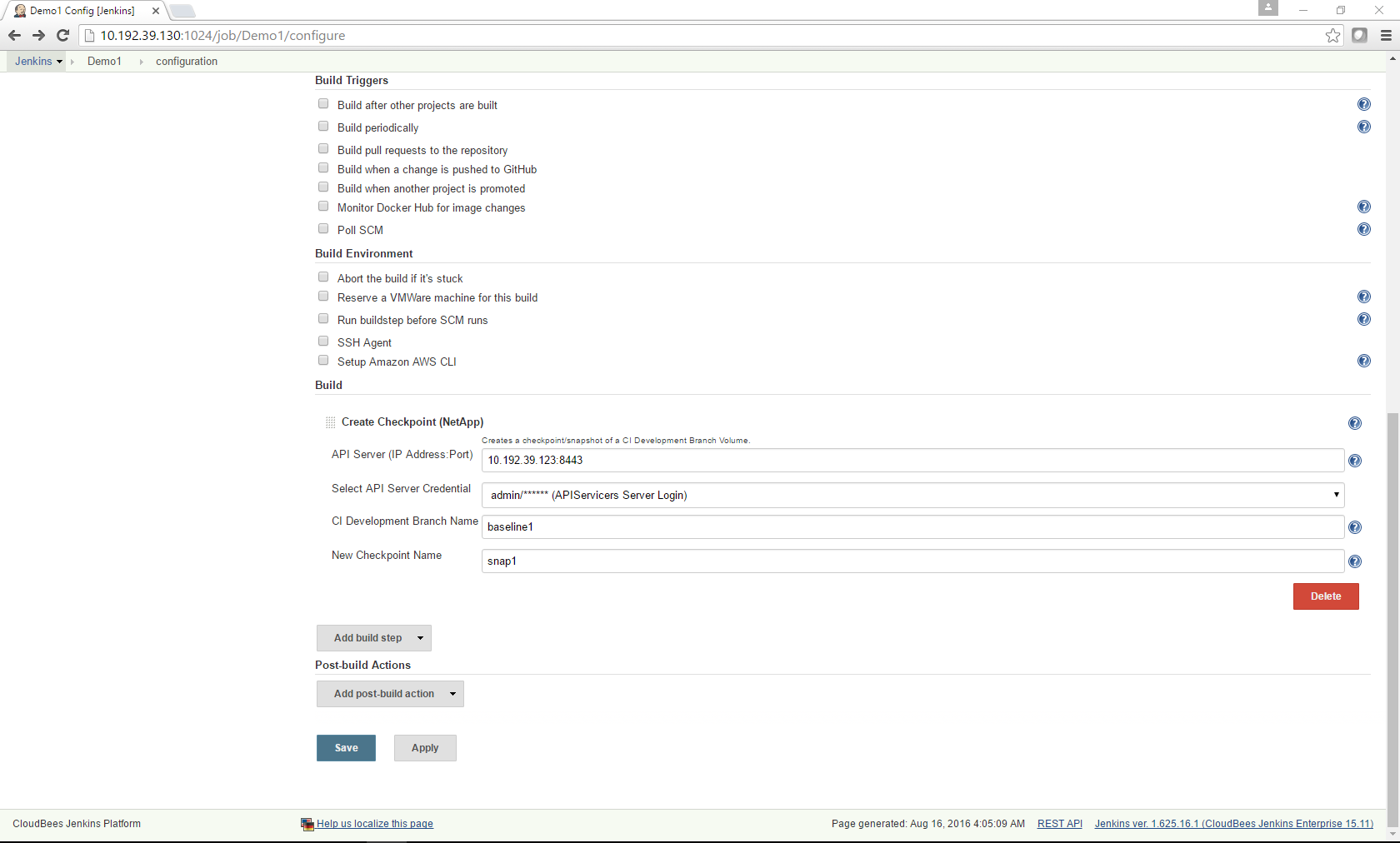
* First the code has to be pulled onto the baseline container & volume
* Create a new job by clicking on new item on the Jenkins dashboard
* Choose the SCM option as Git as shown below and enter the GitLab URL in the repository URL field
* Add credentials for the GitLab user and password created earlier and select it from the drop down list under credentials
* Select Restrict where the project can Run. You need to give it the previous project name.   
  eg: Demo1



* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “**Console output**” to view the output of the process
* To verify if the code got pulled on to the Baseline container, access the host and give a “**docker ps**” command to view the running containers and check for the container’s name eg. “bl1” (make note of the container-id)
* Then enter the context of this container using “**docker exec –it [container-id] /bin/bash**”and Change Directory to **/workspace/Demo1** and list the contents of the directory. It should be that of the local repository.

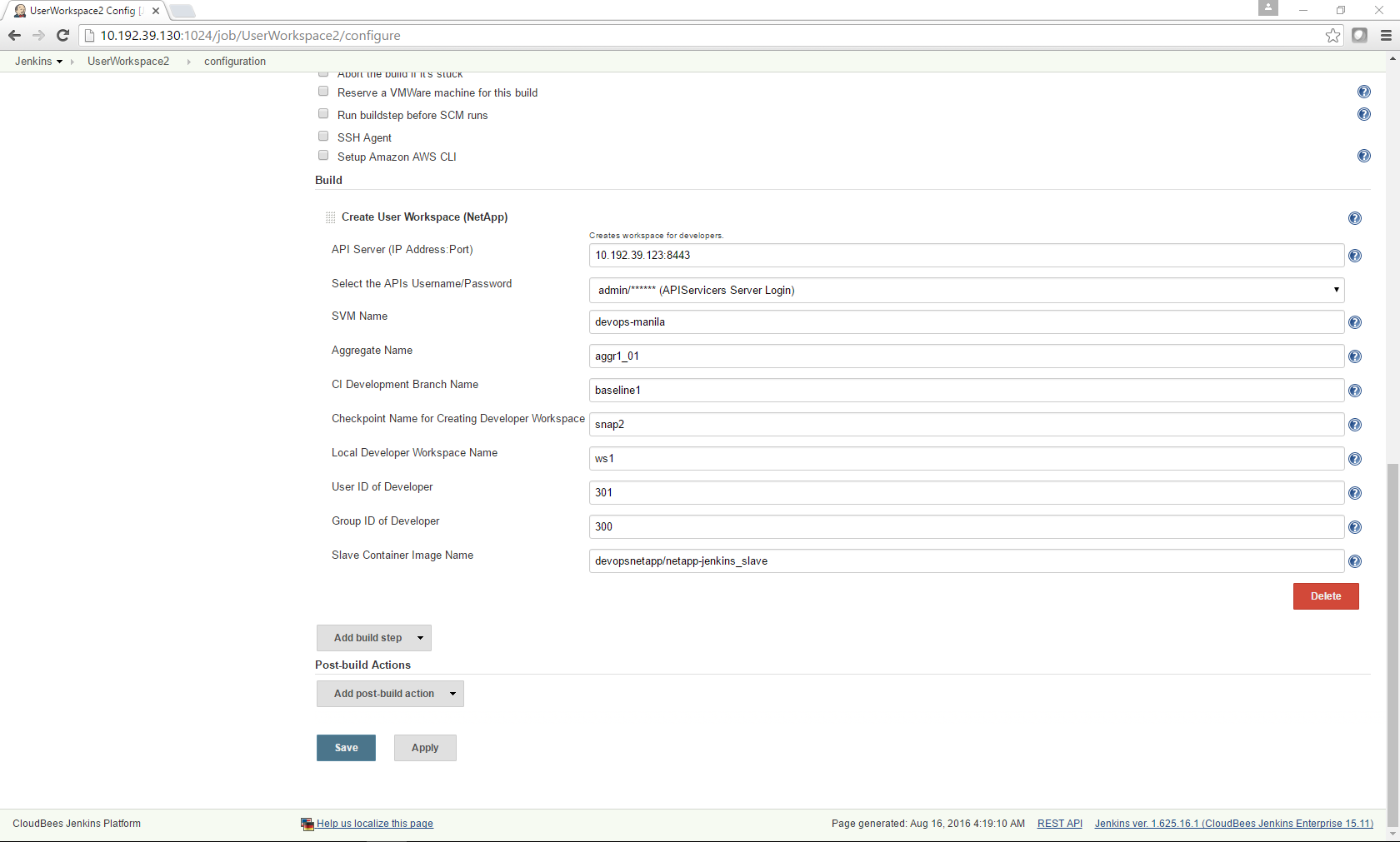
1. Taking a snapshot of the baseline volume

* On your Jenkins Dashboard click on the “Demo1” project which was created as part of step1) and scroll down to build step
* Delete the existing build steps by clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**Create Checkpoint (NetApp)**”
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the Development Branch Baseline for which the snapshot has to be taken eg. **baseline1**
* Finally, enter the name of the new snapshot to be created **eg. snap1**
* Verify configuration and select the final verification option. Refer to the screenshot below to see what the configuration should look like
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



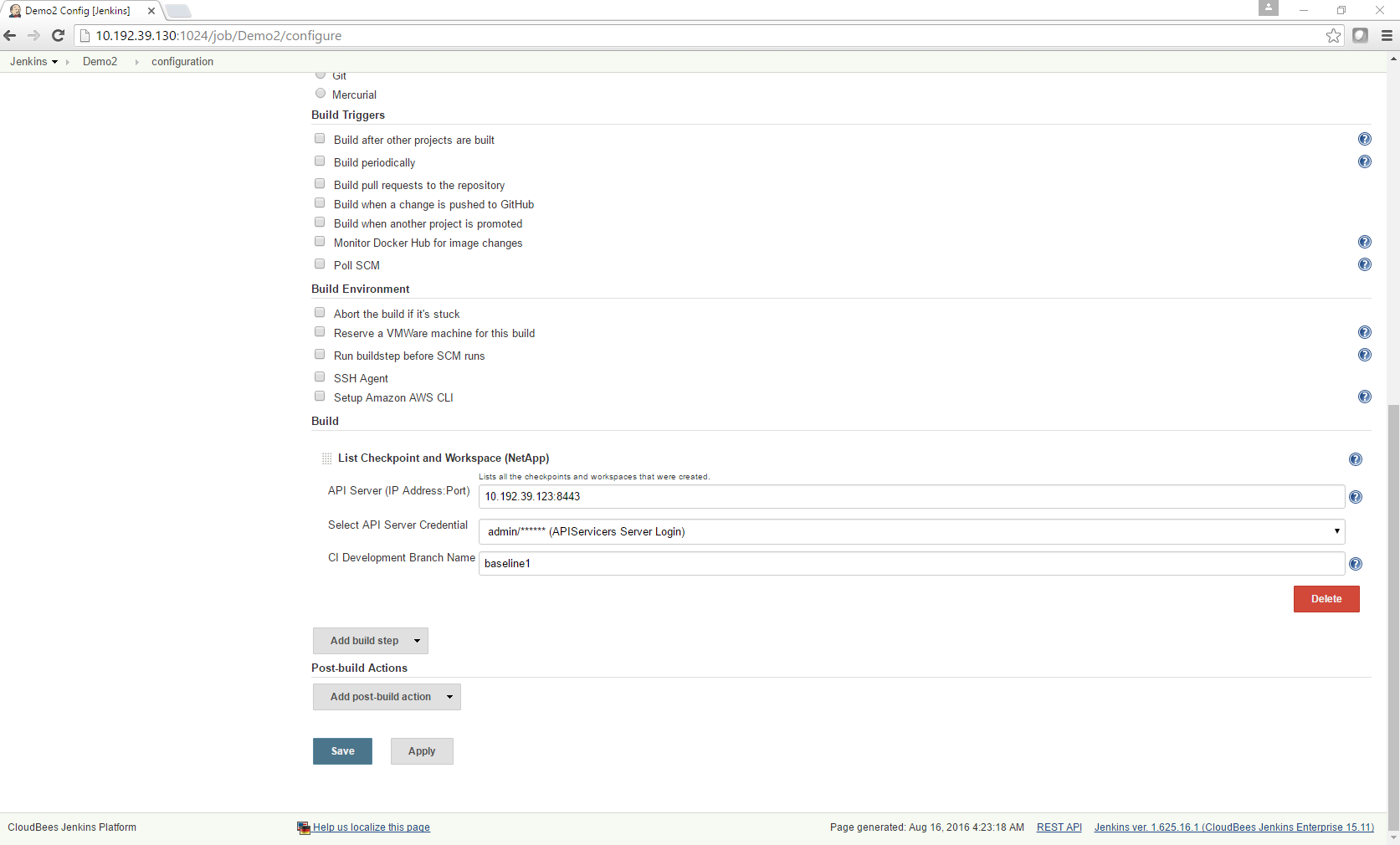
1. Workspace creation from the Baseline

* On your Jenkins Dashboard and Create a New Item>Freestyle Project eg. UserWorkspace1
* elete the existing build steps by clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**Create User Workspace (NetApp)**”
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the SVM and Aggregate name to create the clone from ( To view How to list all the available SVMs and Aggregates in the cluster from the Jenkins Dashboard, skip to step17))
* Enter the name of the Development branch to clone from.
* Enter the Checkpoint on the baseline to clone from (skip to step 16 to view how to list checkpoints and workspaces)
* Enter the name of the local workspace. Eg: ws1
* Enter the GID and UID for the clone volume (\*\*\* This Unix user and group ID has to be created and present on the ONTAP cluster)
* “Enter the name of the slave docker container” as “**devopsnetapp/netapp-jenkins\_slave**”
* Refer to the following screenshot to see how the configuration should look like
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process
* To verify the running of the Baseline container, access the host and give a “**docker ps**” command to view the running containers and check for the container’s name which was just created eg. “ws1” (make note of the container-id)
* For further verification, enter the context of this container using “**docker exec –it [container-id]or[container-name] /bin/bash**”and enter the command “**df**” to see if the Netapp volume is mounted at /workspace/UserWorkspace



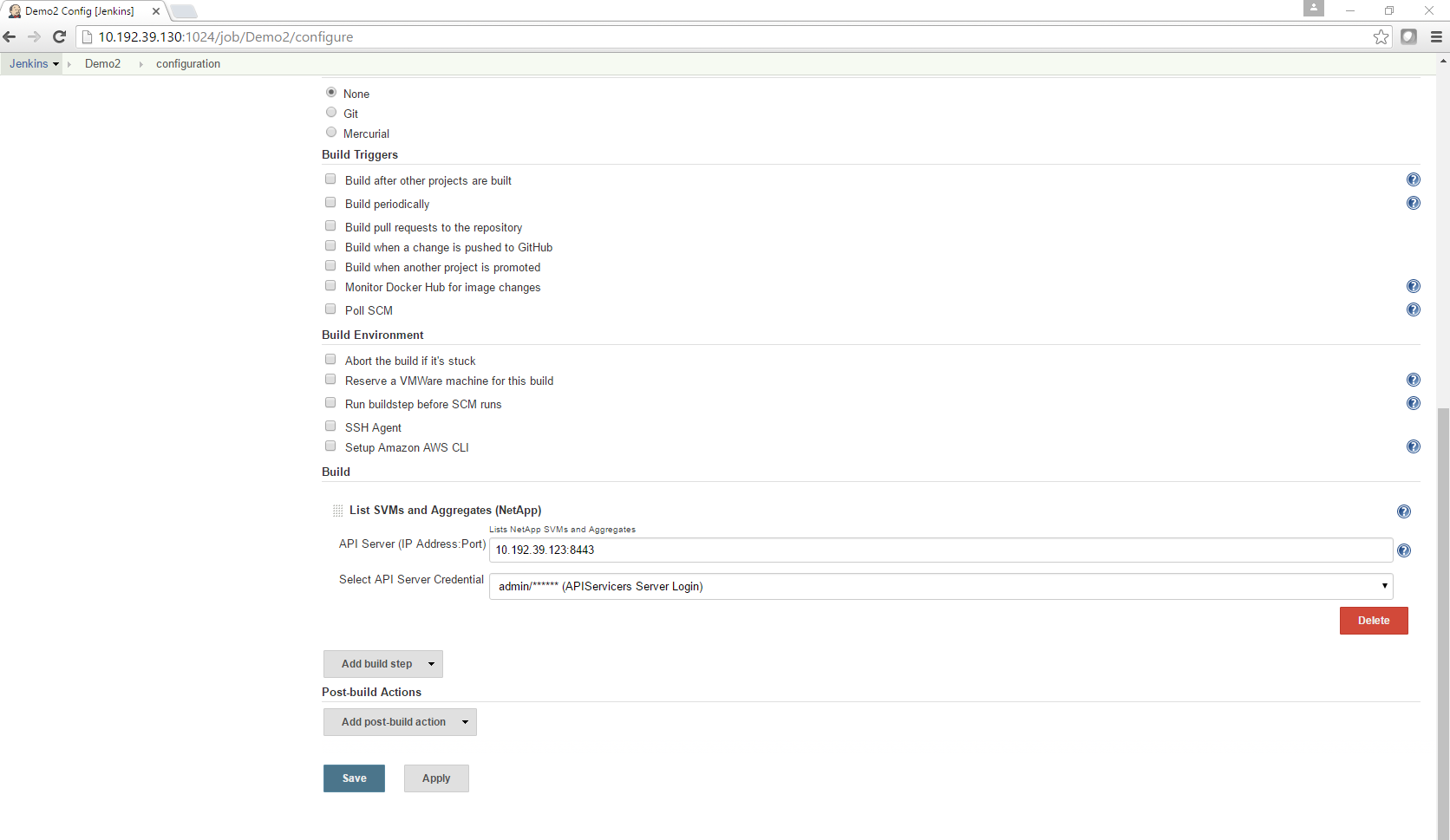
1. Listing Checkpoints and Workspaces

* On your Jenkins Dashboard click on the “Demo2”
* Delete the existing build steps by clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**List Checkpoint and Workspaces (NetApp)”**
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the name of the Development Branch baseline eg. **"baseline1”** for which the checkpoint and workspaces have to be listed
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



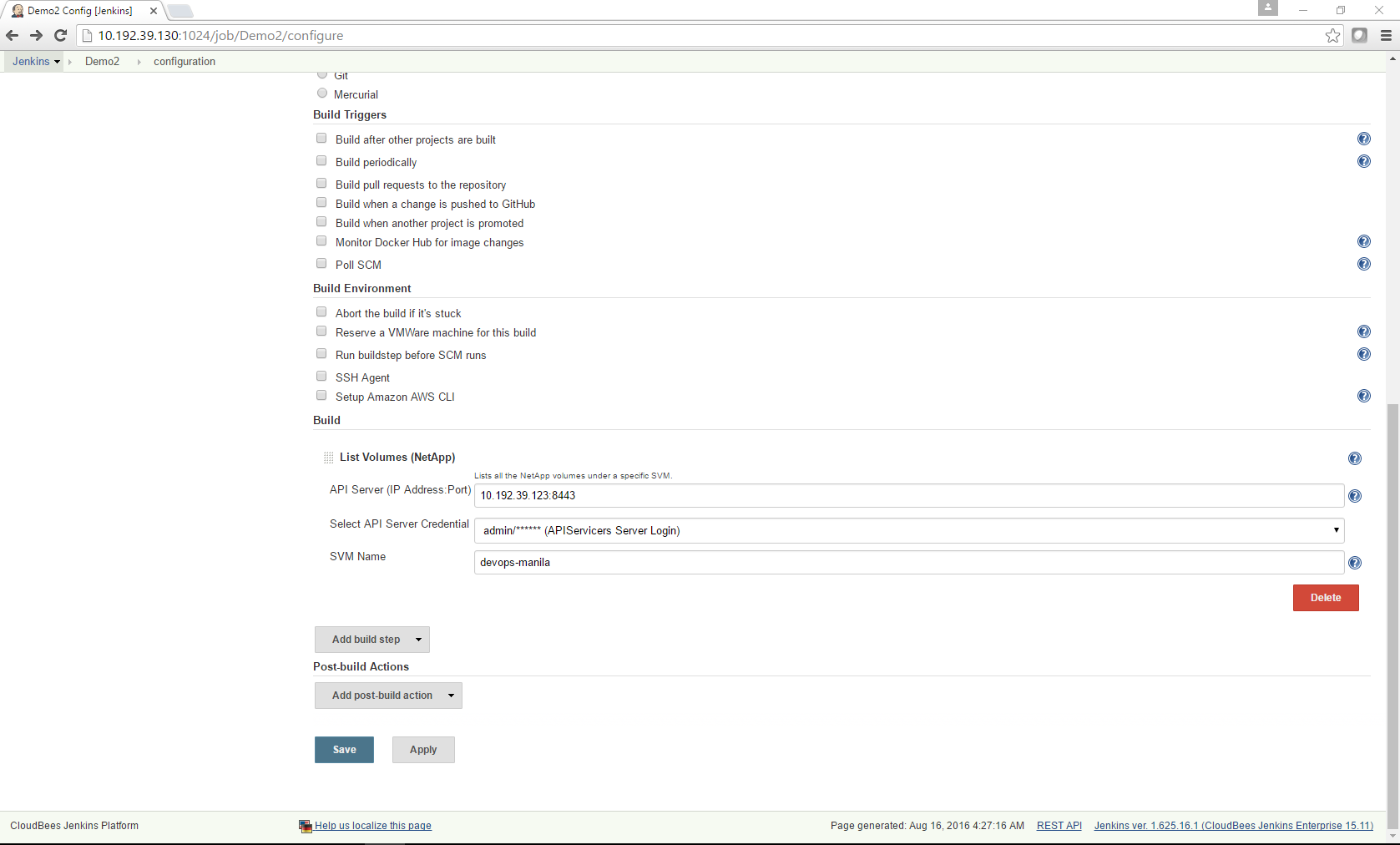
1. SVM and Aggregate listing

* On your Jenkins Dashboard click on the “Demo2” project ( You can create a New Item > Freestyle project too, or you can reuse any existing project also)
* Delete the existing build steps by clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step **“List SVMs and Aggregates (NetApp)”**
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* The example configuration is as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



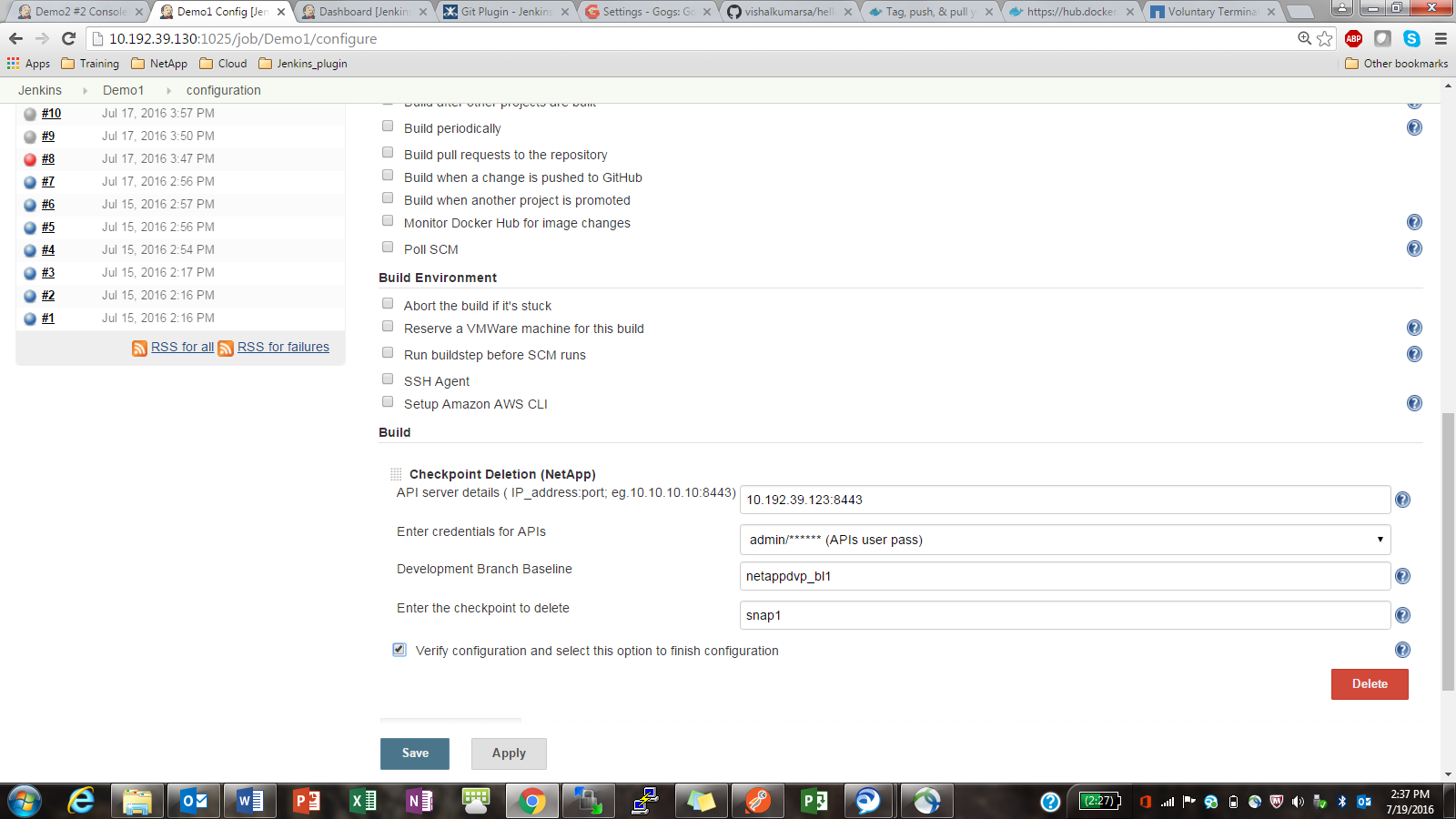
1. NetApp volume listing

* On your Jenkins Dashboard click on the “Demo2” project ( You can create a New Item > Freestyle project too, or you can reuse any existing project also)
* Delete the existing build stepsby clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step **“List Volumes(NetApp)”**
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* **Enter the name of the SVM whose volumes should be listed.**
* The example configuration is as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down optionClick on the drop down option and click on “Console output” to view the output of the process



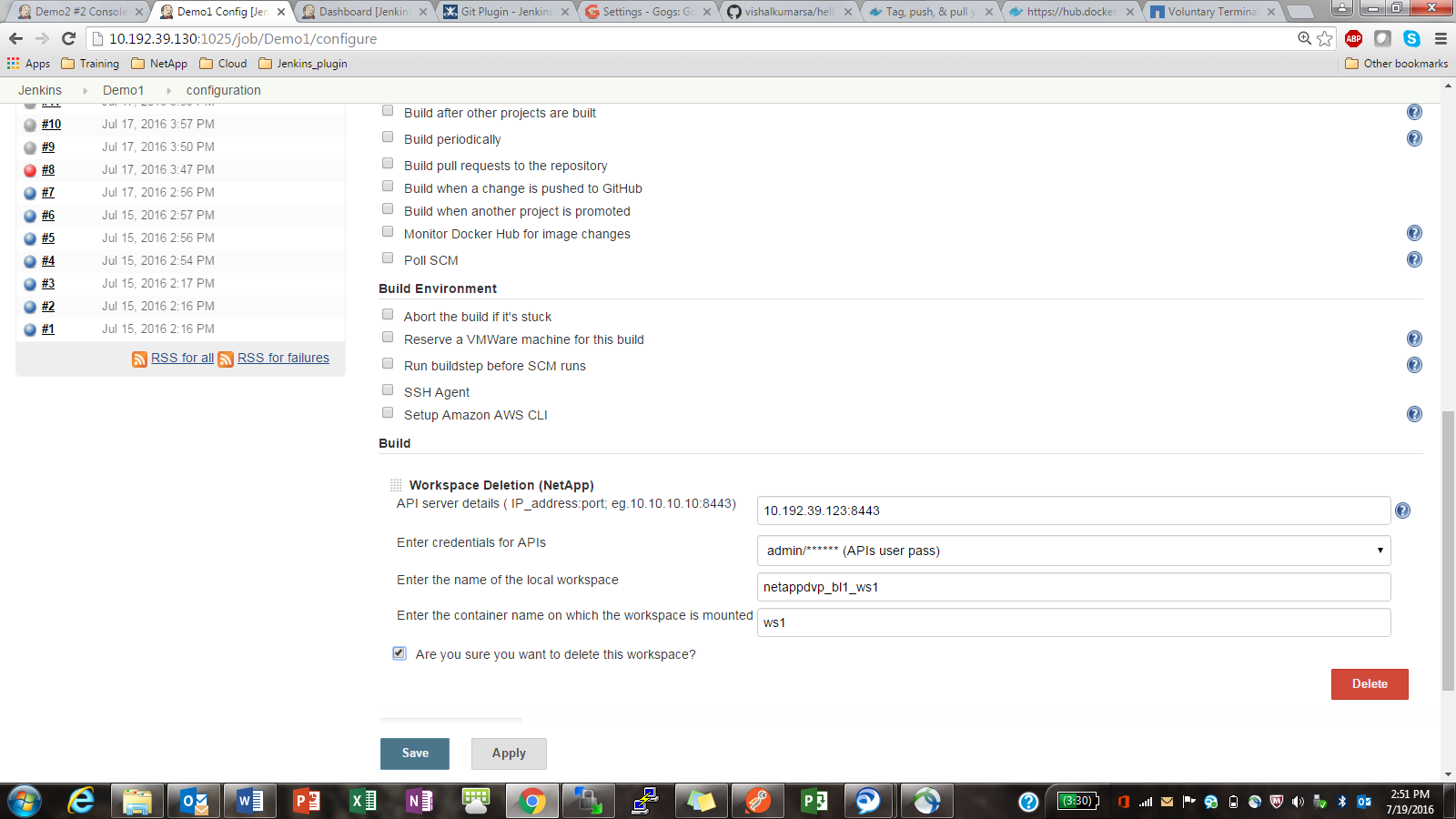
1. Checkpoint deletion

* On your Jenkins Dashboard click on the “Demo2” project ( You can create a New Item > Freestyle project too, or you can reuse any existing project also)
* Delete the existing build stepsby clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step **“Delete Checkpoint(NetApp)”**
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the name of the Development Branch baseline eg. **" bl1”** for which the checkpoint has to be deleted
* Enter the name of the checkpoint that has to be deleted **eg. snap1**
* Tick the final verify configuration option and save
* The example configuration is as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



1. Workspace Deletion

* On your Jenkins Dashboard click on the “Demo2” project ( You can create a New Item > Freestyle project too, or you can reuse any existing project also)
* Delete the existing build stepsby clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step **“Delete User Workspace (NetApp)”**
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the name of the local workspace **eg.” bl1\_ws1”**
* Enter the container name on which the workspace is mounted **eg.”ws1”**
* The example configuration is as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



1. Build Artifact Archival Setup (NetApp)

* On your Jenkins Dashboard click on the “Demo2” project ( You can create a New Item > Freestyle project too, or you can reuse any existing project also)
* Delete the existing build stepsby clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**Create Build Artifact (NetApp)**”
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Select the option “If build artifact volume isn't created yet, tick box to create new volume else continue with the next step” if the build artifact volume doesn’t already exist. If it already exists, do not choose the option, and just enter the name in the next line
* Enter the name of the build artifact archival volume. If the option above is ticked, then it’ll create this new volume. If not, then the existing volume will be mounted to the build archival container eg. **“ba\_1”**

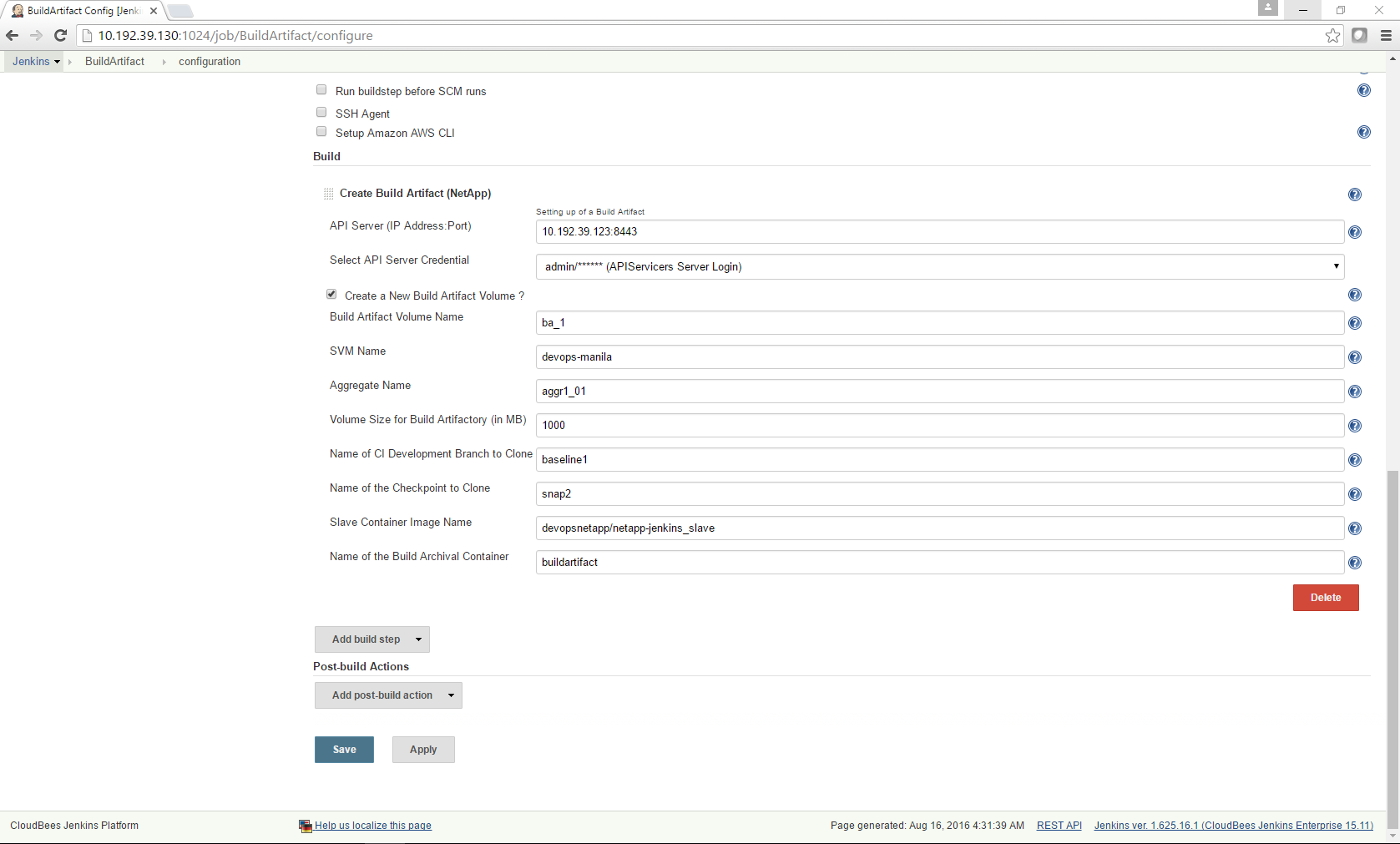
**In case the create base archival volume option is selected, fill out the following details.**

Enter the SVM and Aggregate name to create the volume from ( To view How to list all the available SVMs and Aggregates in the cluster from the Jenkins Dashboard, skip to step17))

* Enter the size of the new volume to be created in MBs
* Enter the baseline volume to clone from eg.” Baseline1”
* Enter the snapshot to clone from eg. “snap1”
* Enter the name of the clone whose contents will be zipped and archived. This is the temporary clone which will be created from a specified snapshot to archive the build artifacts
* Enter the GID and UID for the clone (\*\*\* This Unix user and group has to be created in advance on the ONTAP controller)
* Enter the Jenkins Master IP (Refer Appendix)
* “Enter the name of the slave docker container” as “devopsnetapp/netapp-jenkins\_slave”
* It’s a best practice to have the same simple name for “Enter the name of the slave container which will be created for archiving” & “Enter the Label for the slave build artifact archival container” **eg. “buildartifact”**

**\*\*\* Remember the label of the container, this will be required later in the flow**

* The configuration is filled out as shown in the 2 screenshots below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process

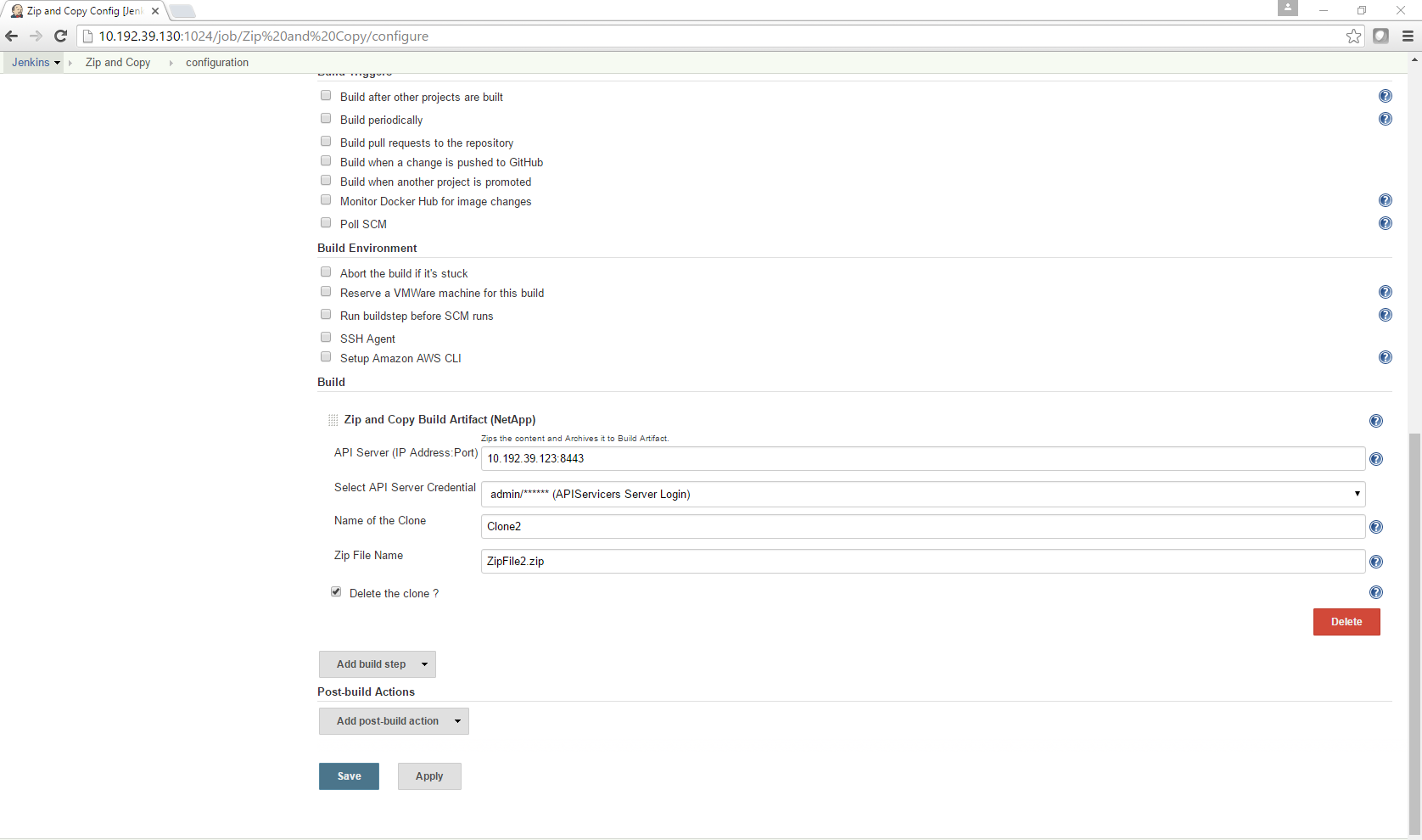


1. Build Artifact Archival Execution (NetApp)

* Create a new job by clicking on new item on the Jenkins dashboard
* Enter item name eg. “**Zip and Copy**” and Select the first option > Freestyle project and click on >OK
* For this project select the option “**Restrict where this project can be run**” and give the label expression which was assigned to the build archival container while creating it as part of the setup phase (xiii) **eg. “buildartifact” (seen in figure below)**
* Now add build step “**Build Artifact Archival Execution (NetApp)**”
* Enter the API server details- IP address:port
* Select the APIs user/pass from the dropdown list in the credentials section
* Enter the name of the zipfile to be created and archived (eg abc.zip)
* “Delete temperory clone created to zip contents of baseline and move to base archival volume” option can be ticked or left alone. **It’s a best practice to delete the clone and immediately delete the container as part of the “Build Archival Container Deletion” step**

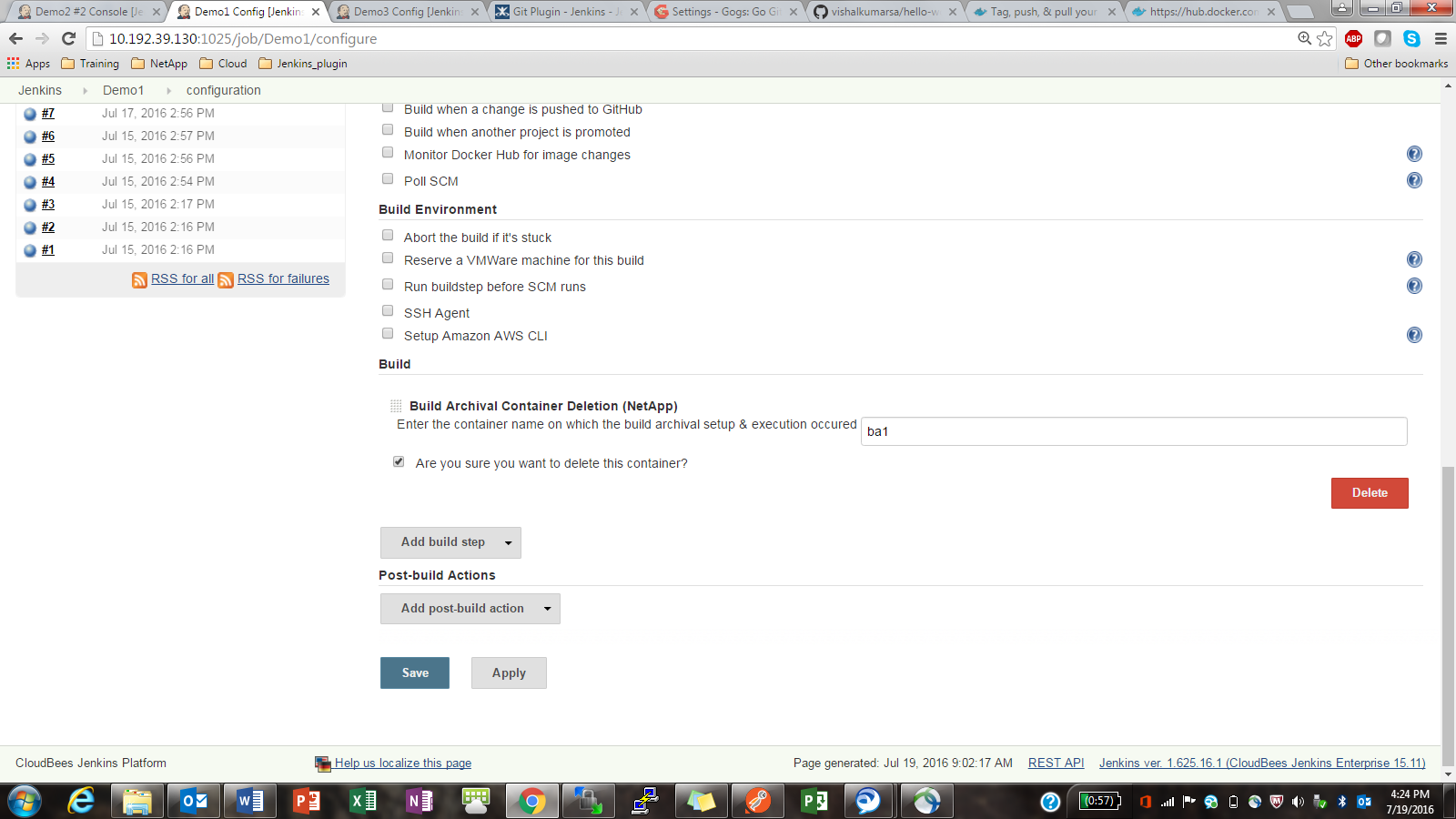
**If the container is not deleted immediately after deleting the clone, stale file handles will exist and prevent the entire flow from working!!!**

* The configuration is filled out as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



1. Build Artifact Container Deletion (NetApp)

* On your Jenkins Dashboard click on the “Demo1” project which was created as part of step1) and scroll down to build step
* Delete the existing build stepsby clicking on the delete button at the bottom right corner of each tab if they exist
* Now add build step “**Build Artifact Container Deletion (NetApp)**”
* Enter the container name on which the build execution occurred **eg. “ba1”**
* The configuration is filled out as shown in the screenshot below
* Once you click on save, the project page appears and on the left hand side, click on the “build now” option
* The project building is triggered and by the side of this running build in the build history section on the left side is a drop down option
* Click on the drop down option and click on “Console output” to view the output of the process



**Appendix**

**Known Issues**

* If a NetApp volume or clone is deleted and the container is still running, then it results in **stale file handles** on the host. This in turn leads to Netappdvp malfunctioning and throwing errors as it is not designed to handle this exception.
* Netappdvp has to be run on the host, if not inconsistency errors are thrown. Run a “**ps ax | grep netappdvp**” and you should see 2 instances of Netappdvp running. One on the Jenkins master container and one on the host. If you see only one instance running, then run the command “**netappdvp --config=/etc/netappdvp/ontap-nas.json &**” on the host. A longterm solution is to add this command to the crontab and make sure this process is start on-boot.

**Developer Guide**

1. Git clone from the location [Location to be decided] using the command “**git clone [URL]**”
2. In the current directory on your host where the cloning occurred, there will be a directory named NetApp\_Jenkins. This will contain 3 more directories, Jenkins\_Master, Jenkins\_Slave and SCM\_on\_Docker.
3. In order to make any modifications to the scripts for the Jenkins master docker images, access the Netapp\_Jenkins/Jenkins\_Master directory and look into the .py files
4. In order to make any configuration changes to the Jenkins builder templates, the best practice is to run the current version of the docker image for the Jenkins master by pulling it from the devopsnetapp docker hub repository and running it (Follow 7. And 8. above)
5. Setup the Jenkins master following 10. Listed above
6. Now access each template from the GUI and make the required changes to the attributes or transformer
7. Once all the required changes are made, copy the new version of the template by first accessing the container from the CLI of the host using “**docker exec –it Jenkins\_master /bin/bash**”
8. Here navigate to “**cd /var/jenkins\_home/jobs**”
9. Do listing using the “**ls**” command and “**exit**” the context of the container
10. Copy only the template that you made modifications to on the Jenkins Dashboard, to the Netapp-jenkins directory on your host using command “**docker cp [container-ID]:/var/Jenkins\_home/jobs/[template-name] /path-to-netapp\_jenkins**”
11. Changes can also be made to the Dockerfile as per the requirements in the Netapp\_Jenkins/Jenkins\_Master or Netapp\_Jenkins/Jenkins\_Slave directory
12. Run commands “**chmod +x plugins.sh**” and “**chmod +x jenkins.sh**”
13. Now assuming that all the required modifications to the python scripts, Dockerfile and the builder templates are made, build the new Jenkins-master docker image from within Netapp-jenkins directory using the command “**docker build –t [new-image-name] .**” (dot indicates current directory)
14. If this image has to be pushed to a docker hub repository, follow the steps listed here: <https://docs.docker.com/engine/getstarted/step_six/>
15. Similarly, Access the Netapp\_Jenkins/Jenkins\_Slave directory and make modifications to the scripts and/or Dockerfile and build the new image as per step 12).
16. Docker-Engine version 1.11.2 is needed for this to run smoothly. Ensure you have the exact version of Docker engine.

Installing a specific version of Docker on RHEL distro:

**yum install docker-engine-<version number>**

For example:

yum install docker-engine-1.11.2

or use

**wget -qO- https://get.docker.com/ | sed 's/lxc-docker/lxc-docker-1.11.2/' | sh**

**Commonly faced issues:**

* Error while starting Docker:

Job for docker.service failed because the control process exited with error code . See "systemctl status docker.service" and "journalctl -xe" for details.

>> For this error you need to check whether your docker socker file is present and configured properly:

usr/lib/systemd/system/docker.socket

The docker.socket file should look like :

[Unit]

Description=Docker Socket for the API

PartOf=docker.service

[Socket]

ListenStream=/var/run/docker.sock

SocketMode=0660

SocketUser=root

SocketGroup=docker

[Install]

WantedBy=sockets.target

* Docker Engine failing to resolve dependencies:

If your docker containers are failing to resolve Linux dependencies while building images then make sure you have configured your docker.conf file **(/etc/systemd/system/docker.service.d/docker.conf).**

This issue usually occurs when there is packet fragmentation in the network.

The file should have following content:

[Service]

ExecStart=

ExecStart=/usr/bin/docker daemon -H fd:// --mtu 1400 --exec-opt native.cgroupdriver=system

Version History

| Version | Date | Document Version History |
| --- | --- | --- |
| Version 1.0 | September 2016 | Initial release |