DENT Test Automation Guide

Version 0.1, 2021/11/04, Alisa Li, Bryan Lee, May Hsiang

Contents

Document History

Version	Description	Author	Review
V0.1	Initial version	Alisa Li, Bryan Lee, May Hsiang	Taskin Ucpinar

1. Introduction

This document guides to setup the environment to run the DENT test automatically.

2. Environment setup

2.1. An overview for the test environment

The below picture shows the environment for the test automation.

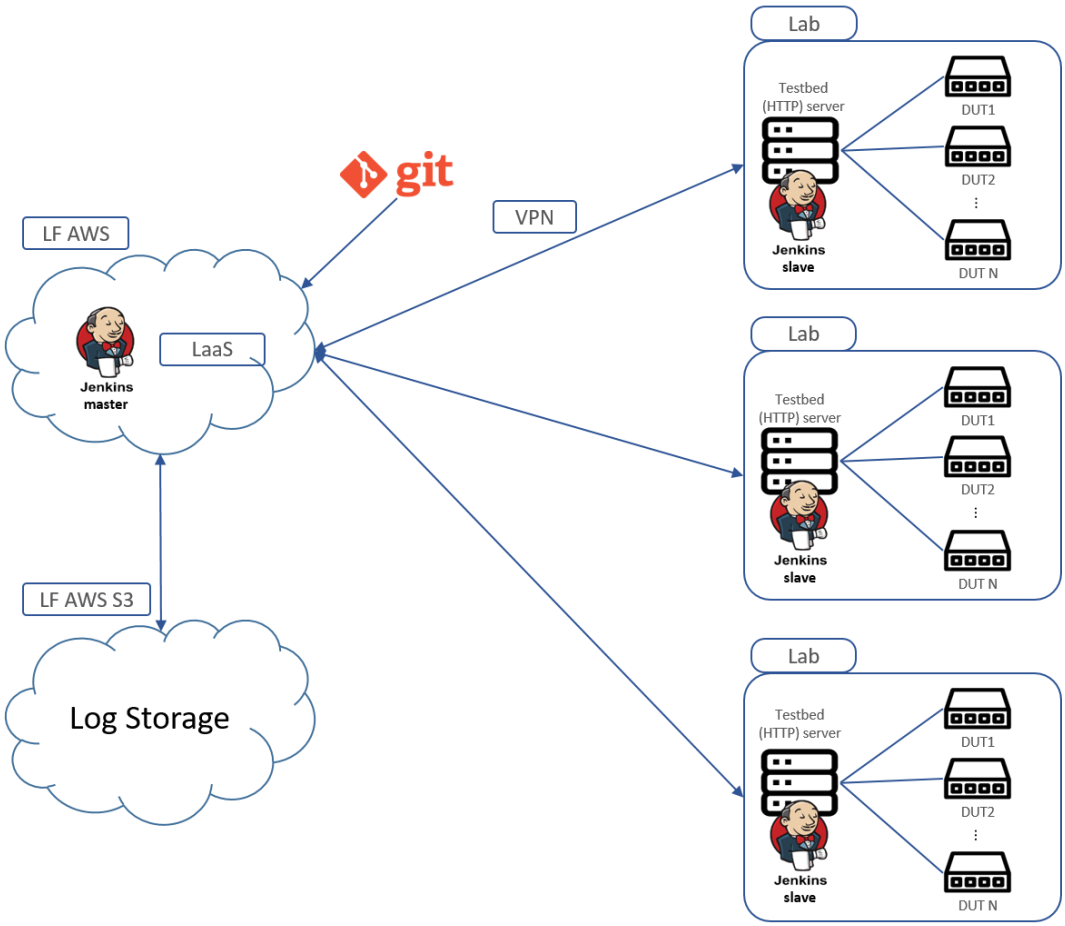
The LF AWS S3 bucket is where we put the logs after testing.

The LF AWS is a Jenkins master and a LaaS server. It can also be a testbed server.

The DENT AWS should be able to access the testbed and device in the lab.

The Lab includes a testbed and devices. The testbed also serves as a Jenkins slave and a HTTP server.

Currently, we have Git as SCM to put our Jenkins and python scripts to it. The testing begins with pulling the scripts from the Git (GitLab or GitHub) to make sure the scripts are up-to-date.



2.2. LaaS server setup

2.2.1. Generating self-signed certificates

```
$ openssl req -subj '/CN=*/' -x509 -nodes -days 365 -newkey rsa:2048 -
keyout ./server.key -out ./server.crt
```

2.2.2. Pre-requirements

The following software should be installed.

2.2.2.1. Docker container

Install docker for at least version 19.03.8.

There are two ways to install docker engine: install it manually or with script.

Please choose a suitable option of docker engine installation below:

2.2.2.2. Install docker engine manually

Remove old packages if any and update a cache:

```
$ sudo apt-get remove docker docker-engine docker.io containerd runc
$ sudo apt-get update
```

Install dependences:

```
$ sudo apt-get install \
apt-transport-https \
ca-certificates \
curl \
gnupg-agent \
software-properties-common -y
```

Download and install key:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Add registry:

```
$ sudo add-apt-repository \
 "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
$(lsb_release -cs) \
stable"
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io -y
```

2.2.2.3. Install docker engine with script

```
$ sudo apt-get update
```

Get the required installation script:

```
$ sudo curl -fsSL https://test.docker.com -o /tmp/test-docker.sh
```

Execute the script:

```
$ sudo sh /tmp/test-docker.sh
```

To use Docker as a non-root user, add your user to the "docker" group:

```
$ sudo usermod -aG docker ${USER}
```

Check if docker is installed successfully:

```
$ command docker
```

Clean up (optional):

```
$ sudo rm -rf /tmp/test-docker.sh
```

2.2.2.4. Docker loki plugin

Install Loki docker plugin:

```
$ sudo docker plugin install grafana/loki-docker-driver:latest --alias loki --
grant-all-permissions
```

Check if Loki plugin is installed successfully:

```
$ sudo docker plugin ls | grep loki
```

2.2.2.5. Docker-compose

Install docker-compose at least version 1.25.5.

Get all docker-compose installation files suitable to your system:

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.25.5/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
$ sudo chmod +x /usr/local/bin/docker-compose
```

Check if docker-compose is installed successfully:

```
$ command docker-compose
```

2.2.3. LaaS server installation

Please contact Edge-Core DevOps team *ec_devops_support@edge-core.com* to get the **laas_setup.run** installer file.

Change file mode to executable:

```
$ chmod +x laas_setup.run
```

Copy **laas_setup.run** file to the target machine and run as root:

```
$ sudo ./laas_setup.run
Please type y to accept, n otherwise: y
Do you wish to procceed with generated self-signed certificates? [y/N] y
INFO: Mail server specific settings.
Mail server host: <mail_server_host>
Mail server port: <mail_server_port>
Mail server SSL enabled [y/N]: y/N
Mail server TLS enabled [y/N]: y/N
INFO: It is required to provide system mail account credentials.
Email account: <email_address>
Email 'from' address: <email_address>
Email password/token: <email_password>
INFO: Do you wish to configure additional authentication module (AWS Cognito)?
[y/N] N
```

The final result:

```
laas_loki is up-to-date
Creating laas_redis ...
Creating laas_redis     ... done
Creating laas_promtail   ... done
Creating laas_scheduler  ... done
Creating laas_api        ... done
Creating laas_dispatcher ... done
Creating laas_nginx      ... done
INFO: The service is started
```

There are some containers running after executing the laas_setup.run file:

```
ubuntu@ip-172-31-12-250:~$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED       STATUS       PORTS
                    NAMES
12f76359beae   ubuntu-jenkins         "bash"                   12 days ago   Up 7 days    0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:8433->8433/tcp, :::8433->8433/tcp, 8443/tcp, 0.0.0.0:8822->22/t
cp, :::8822->22/tcp   jenkins
285057fa2c7c   nginx:1.17.9-alpine    "nginx -g 'daemon of…"   2 weeks ago   Up 2 weeks   0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp
                    laas_nginx
6da293b9b635   laas/api:latest        "server-entrypoint.s…"   2 weeks ago   Up 2 weeks
                    laas_api
e56cbf84ce81   laas/api:latest        "server-entrypoint.s…"   2 weeks ago   Up 2 weeks
                    laas_scheduler
9bc1025a9a24   laas/api:latest        "server-entrypoint.s…"   2 weeks ago   Up 2 weeks
                    laas_dispatcher
567982c7b4eb   grafana/grafana:6.5.0  "/run.sh"                2 weeks ago   Up 2 weeks   127.0.0.1:3000->3000/tcp
                    laas_grafana
083389f8da2c   grafana/promtail:1.4.1 "/usr/bin/promtail -…"   2 weeks ago   Up 2 weeks
                    laas_promtail
67f85cb35a56   redis:5.0.8-alpine     "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   6379/tcp
                    laas_redis
d94cace65f29   postgres:12.2-alpine   "docker-entrypoint.s…"   2 weeks ago   Up 2 weeks   5432/tcp
                    laas_database
4157c4246c11   grafana/loki:1.4.1     "/usr/bin/loki -conf…"   2 weeks ago   Up 2 weeks   80/tcp, 127.0.0.1:3100->3100/tcp
                    laas_loki
```

2.2.4. Connect to LaaS Server

2.2.4.1. CLI installation

Please contact Edge-Core DevOps team *ec_devops_support@edge-core.com* to get the **eclab** file.

Change mode to executable:

```
$ chmod +x eclab
```

Please note that bash autocompletion is available only for bash shell.

The command to check the current shell:

```
$ echo $SHELL
/bin/bash
```

The CLI includes bash autocompletion support.

Enable autocompletion for eclab:

```
$ sudo su
# apt install -y bash-completion
# ./eclab completion > /usr/share/bash-completion/completions/eclab
# exit
```

It is required to reapply the bashrc configuratoin file. The easiest way is to perform logout/login.

Help:

```
$ ./eclab -h

eclab is the solution which allows to easily organize access
to the set of devices for the big number of users.
For this purpose, solution introduces laboratory, grouping devices into domains,
users into user groups in context of specific domains.

Usage:
  eclab [command]

Available Commands:
  completion      Generates bash completion scripts
  config          Configures eclab
  device          Configures devices in the lab
  domain          Configures domain group
  forget-password Send verification code and resets the password for specified user.
  help            Help about any command
  log-severity    Configures severity level of the system log messages.
  login           Logins to server address specified with '--server' flag.
  reservation     Configures device reservation.
  switch          Configures switch in the lab.
  user            Executes user-related commands
  usergroup       Configures usergroup

Flags:
      --config-file string      Configuration file for one-shot command execution
      --help                    Displays this help text and exit
      --server-address string   Server address for one-shot command execution
      --version                 Displays version information and exit

Use "eclab [command] --help" for more information about a command.
```

2.2.4.2. CLI configuration

This section gives the examples to connect to LaaS Server by CLI with following conditions:

- Server is installed, Server IP – 127.0.0.1
- CLI client is run locally

Add server in the configuration:

```
$ ./eclab config server add

IP: 127.0.0.1

Username: admin

Status: New server successfully added.
```

Result: File is created **`{homedir}/.eclab/config.yaml`** with server configuration access info.

```
$ cat ~/.eclab/config.yaml

- username: admin
  token: ""
  server: 127.0.0.1
  default: true
  response_wait_duration: 10
  timezone: UTC
```

Get the list of servers:

```
$ ./eclab config server list

+----------------+----------+-------+---------+----------+
| SERVER ADDRESS | USERNAME | TOKEN | DEFAULT | TIMEZONE |
+----------------+----------+-------+---------+----------+
| 127.0.0.1      | admin    | none  | yes     | UTC      |
+----------------+----------+-------+---------+----------+
```

2.2.4.3. Login

If use the self-signed certificates, please add the command **`export ECLAB_USE_INSECURE_CONNECTION=true`** into **`~/.bashrc`** to set the environment variable:

```
$ echo "export ECLAB_USE_INSECURE_CONNECTION=true" >> ~/.bashrc

$ source ~/.bashrc
```

Log in to the server (Use admin's default password: **fzS8G$5pwR** if you login first time):

```
$ ./eclab login (--server 127.0.0.1)

Enter Password: <admin_password>

Login Successful.
```

Result: User is logged in. In file config.yaml appears token value for this server.

If log in successfully, execute the command **`./eclab config server list`** and the value of "token" will be 'present':

```
$ ./eclab config server list

+----------------+----------+---------+---------+----------+
| SERVER ADDRESS | USERNAME |  TOKEN  | DEFAULT | TIMEZONE |
+----------------+----------+---------+---------+----------+
| 127.0.0.1      | admin    | present | yes     | UTC      |
+----------------+----------+---------+---------+----------+
```

2.2.4.4. Add/Delete/List users

Admin can add users to the Lab:

```
$ ./eclab user add regular <user_name> <user_email>
Password: <user_password>
Status: User successfully created.
```

The rules for password creation:

- Your password has to be at least 8 characters
- Must contain at least one lower case letter, one upper case letter, one digit, and one of these special characters ~!@#$%^&*()_+

Admin can also delete users from the Lab. Disable the user account before delete it:

```
$ ./eclab user disable <user_name>
Status: User successfully disabled.
$ ./eclab user delete <user_name>
Status: User successfully deleted.
```

There are two commands that we can use to get a list of users.

Use **user list** command:

```
$ ./eclab user list
Item count: 2
Current page: 1
Page count: 1
+--------------+
|   USERNAME   |
+--------------+
| admin        |
| jenkins_dent |
+--------------+
```

Result: Show the list of all users in the Lab (Columns: USERNAME)

Use **user show** command:

```
$ ./eclab user show

Item count: 2
Current page: 1         Page count: 1
+--------------+-----------------+-------+--------+--------------+-----------------+
|   USERNAME   |      EMAIL      | ADMIN | ACTIVE | DOMAINS LIST | USERGROUPS LIST |
+--------------+-----------------+-------+--------+--------------+-----------------+
| admin        | admin@no.where  | true  | true   | <empty>      | <empty>         |
+--------------+-----------------+-------+--------+--------------+-----------------+
| jenkins_dent | test@accton.com | false | true   | <empty>      | <empty>         |
+--------------+-----------------+-------+--------+--------------+-----------------+
```

```
$ ./eclab user show jenkins_dent

+--------------+-----------------+-------+--------+--------------+-----------------+
|   USERNAME   |      EMAIL      | ADMIN | ACTIVE | DOMAINS LIST | USERGROUPS LIST |
+--------------+-----------------+-------+--------+--------------+-----------------+
| jenkins_dent | test@accton.com | false | true   | <empty>      | <empty>         |
+--------------+-----------------+-------+--------+--------------+-----------------+
```

Result: Show the details of all (or a specific) user(s) in the Lab (Columns: USERNAME, EMAIL, ADMIN, ACTIVE, DOMAINS LIST, and USERGROUPS LIST)

2.2.4.5. Add/Delete/List devices

Admin can add devices to the Lab:

```
$ ./eclab device add <device_name>
Status: Device successfully added.
```

Enable devices that users can reserve them:

```
./eclab device enable <device_name>
Status: Device enabled.
```

Admin can also delete devices from the Lab. Disable the device before delete it:

```
$ ./eclab user disable <device_name>
Status: User successfully disabled.
$ ./eclab user delete <device_name>
Status: User successfully deleted.
```

There are two commands that we can use to get a list of devices.

Use **device list** command:

```
$ ./eclab device list

Item count: 2
Current page: 1         Page count: 1
+-------------------------+-----------+----------+-------------------------------+
|          NAME           |  STATUS   | USERNAME |  RESERVATION EXPIRATION DATE  |
|                         |           |          |          (UTC+00:00)          |
+-------------------------+-----------+----------+-------------------------------+
| /tainan/as4xxx/as4224-1 | available | <none>   | <none>                        |
| /tainan/as5xxx/as5114-1 | available | <none>   | <none>                        |
+-------------------------+-----------+----------+-------------------------------+
```

Result: Show the list of all devices in the Lab (Columns: NAME, STATUS, USERNAME, and RESERVATION EXPIRATION DATE)

Use **device show** command:

```
$ ./eclab device show

Item count: 2
Current page: 1         Page count: 1
+-------------------------+---------------------------------+
|       DEVICE NAME       |             DETAILS             |
+-------------------------+---------------------------------+
| /tainan/as4xxx/as4224-1 | Location: <undefined>           |
|                         | Management IP(s):               |
|                         |     first: 192.168.4.39         |
|                         | Power Strip Address: <undefined> |
|                         | Status: available               |
|                         | TFTP Server IP: <undefined>     |
|                         | Description: <undefined>        |
|                         | Chassis: <undefined>            |
|                         | Console Port IP(s):             |
|                         |     first: 192.168.4.100:5028   |
|                         | Domain Name: Dent               |
|                         | Username/Password:              |
|                         |     first: root / onl           |
|                         | Default-reset: false            |
|                         | Need-reset: false               |
|                         | Default-shutdown: false         |
|                         | Sub-status: ok                  |
+-------------------------+---------------------------------+
| /tainan/as5xxx/as5114-1 | Location: <undefined>           |
|                         | Management IP(s):               |
|                         |     first: 192.168.4.38         |
|                         | Power Strip Address: <undefined> |
|                         | Status: available               |
|                         | TFTP Server IP: <undefined>     |
|                         | Description: <undefined>        |
|                         | Chassis: <undefined>            |
|                         | Console Port IP(s):             |
|                         |     first: 192.168.4.100:5018   |
|                         | Domain Name: Dent               |
|                         | Username/Password:              |
|                         |     first: root / onl           |
|                         | Default-reset: false            |
|                         | Need-reset: false               |
|                         | Default-shutdown: false         |
|                         | Sub-status: ok                  |
+-------------------------+---------------------------------+
```

```
$ ./eclab device show /tainan/as4xxx/as4224-1

+-------------------------+---------------------------------+
|       DEVICE NAME       |             DETAILS             |
+-------------------------+---------------------------------+
| /tainan/as4xxx/as4224-1 | Location: <undefined>           |
|                         | Management IP(s):               |
|                         |     first: 192.168.4.39         |
|                         | Power Strip Address: <undefined> |
|                         | Status: available               |
|                         | TFTP Server IP: <undefined>     |
|                         | Description: <undefined>        |
|                         | Chassis: <undefined>            |
|                         | Console Port IP(s):             |
|                         |     first: 192.168.4.100:5028   |
|                         | Domain Name: Dent               |
|                         | Username/Password:              |
|                         |     first: root / onl           |
|                         | Default-reset: false            |
|                         | Need-reset: false               |
|                         | Default-shutdown: false         |
|                         | Sub-status: ok                  |
+-------------------------+---------------------------------+
```

Result: Show the details of all (or a specific) device(s) in the Lab (Columns: DEVICE NAME and DETAILS)

2.2.4.6. Devices configuration

The following are some example of devices configuration.

- **Management IP**:

Usage:

eclab device mgmt-ips <device name> [<mgmt-ips-list>]

The valid format of mgmt-ips is '<name of mgmt ip>:<ip address>'.

Separate each management port IP with ','(e.g., first:192.168.8.20,second:192.168.8.21).

To set the management port IP address to NULL, do not enter the value for <mgmt-ips-list>.

```
$ ./eclab device mgmt-ips /test/device first:192.168.8.20,second:192.168.8.21
Status: Management port IP successfully configured.
```

- **PDU**:

Usage:

eclab device power-strip <device name> [<strip-address-info>]

The valid type of mgmt-ips is a string.

To set the power strip address to NULL, do not enter the value for <strip-address-info>.

```
$ ./eclab device power-strip /test "127.0.0.1 #1"
Status: Power Strip successfully configured.
```

- **Console Port**:

Usage:

eclab device console-port-ips <device name> [<console-port-ips-list>]

The valid format of console-port-ips is '<name of console port ip>:<ip address>:<port number>'.

Separate each console port IP with ','(e.g., first:192.168.3.100:5030,second:192.168.8.130:5001).

To set the console port IPs to NULL, do not enter the value for <console-port-ips-list>.

```
$ ./eclab device console-port-ips /test first:192.168.3.100:5030,second:192.168.8.130:5001
Status: Console port IP(s) successfully configured.
```

- **Username / Password**:

Usage:

eclab device accounts <device name>

To set all accounts to NULL, enter 'y' in the first question "`Do you want to set accounts to NULL?`". Otherwies, enter 'n' to continue configuring device accounts.

To set multiple accounts, enter 'y' in the question "`Do you want to set another account?`", and vice versa.

To modify an existing account, enter the existing account name in the field "`Account Name:`", and enter the new username/password for this account.

```
$ ./eclab device accounts /test
Do you want to set accounts to NULL?(y/n): n
Account Name:
first
Username:
test_username
Password:
test_password
Do you want to set another account?(y/n): n
Abort.
Status: Account(s) of the device successfully configured.
```

2.2.4.7. Add/Delete/List domains

Create private domain that only the users in this domain can see the device in it:

```
$ ./eclab domain add private <domain_name>
Status: Domain successfully created.
```

Create a public domain that all users can see the device in it:

```
$ ./eclab domain add public <domain_name>
Status: Domain successfully created.
```

Delete a domain:

```
$ ./eclab domain delete <domain_name>
Status: Domain successfully deleted.
```

There are two commands that we can use to get a list of domains.

Use **domain list** command:

```
$ ./eclab domain list

Item count: 2
Current page: 1
Page count: 1
+------+
| NAME |
+------+
| Dent |
| Test |
+------+
```

Result: Show the list of all domains in the Lab (Columns: NAME)

Use **domain show** command:

```
$ ./eclab domain show

Item count: 2
Current page: 1         Page count: 1
+------+-------------+--------------------+------------------------+---------+
| NAME |  TIMEZONE   | RESERVATION PERIOD | MAX EXTENSION DURATION | IS OPEN |
+------+-------------+--------------------+------------------------+---------+
| Dent | <undefined> | <undefined>        | <undefined>            | public  |
| Test | <undefined> | <undefined>        | <undefined>            | private |
+------+-------------+--------------------+------------------------+---------+
```

```
$ ./eclab domain show Dent

+------+-------------+--------------------+------------------------+---------+
| NAME |  TIMEZONE   | RESERVATION PERIOD | MAX EXTENSION DURATION | IS OPEN |
+------+-------------+--------------------+------------------------+---------+
| Dent | <undefined> | <undefined>        | <undefined>            | public  |
+------+-------------+--------------------+------------------------+---------+
```

Result: Show the details of all (or a specific) user(s) in the Lab (Columns: USERNAME, EMAIL, ADMIN, ACTIVE, DOMAINS LIST, and USERGROUPS LIST)

2.2.4.8. Add/Delete/List users and devices in domains

Add a user to a domain:

```
$ ./eclab domain user add <user_name> <domain_name>

Status: User successfully added to domain.
```

Delete a user from a domain:

```
$ ./eclab domain user delete <user_name> <domain_name>
Status: User successfully deleted from domain.
```

Get the list of users in a domain:

```
$ ./eclab domain user list Dent
Item count: 1
Current page: 1
Page count: 1
+--------------+
|   USERNAME   |
+--------------+
| jenkins_dent |
+--------------+
```

Add a device to a domain:

```
$ ./eclab domain device add <domain_name> <device_name>
Status: Device successfully added to domain.
```

Delete a device from a domain:

```
$ ./eclab domain device delete <domain_name> <device_name>
Status: Device successfully deleted from domain.
```

Get the list of devices in a domain:

```
$ ./eclab domain device list Dent
Item count: 2
Current page: 1         Page count: 1
+-------------------------+-----------+----------+-------------------------------+
|          NAME           |  STATUS   | USERNAME |  RESERVATION EXPIRATION DATE  |
|                         |           |          |          (UTC+00:00)          |
+-------------------------+-----------+----------+-------------------------------+
| /tainan/as4xxx/as4224-1 | available | <none>   | <none>                        |
| /tainan/as5xxx/as5114-1 | available | <none>   | <none>                        |
+-------------------------+-----------+----------+-------------------------------+
```

2.3. Jenkins setup

2.3.1. Run a docker container

Run a docker container based on Ubuntu 18.04. We will install Jenkins in this container.

```
$ docker pull ubuntu:18.04
$ docker run –itd --name jenkins –p 8080:8080 –p 8443:8443 –p 8822:22 ubuntu:18.04
```

2.3.2. Install Jenkins in a docker container

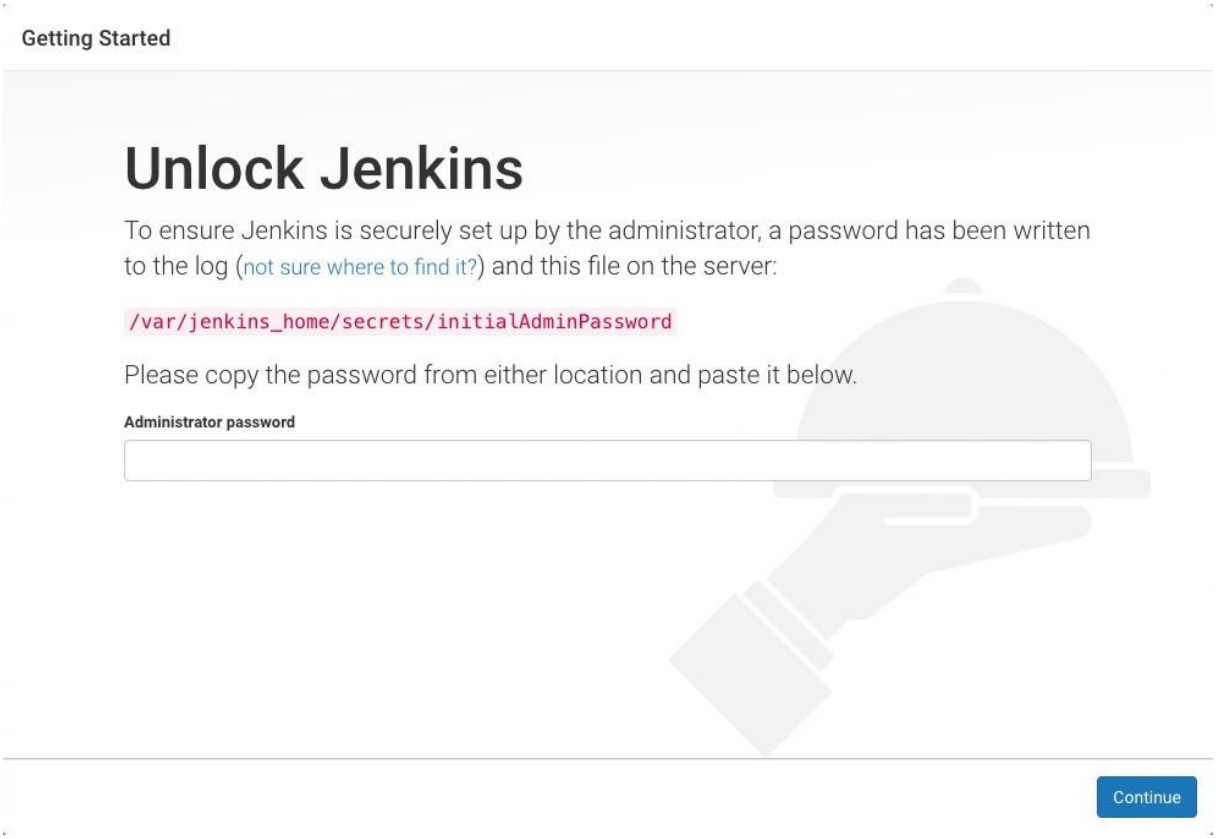
```
$ docker exec –it jenkins bash
```

In the Jenkins container, run the following command to install Jenkins server.

```
root@12f76359beae:/# apt-get update
root@12f76359beae:/# apt-get upgrade
root@12f76359beae:/# apt-get install -y curl openssh-server ca-certificates
openjdk-8-jre
root@12f76359beae:/# wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key |
sudo apt-key add –
root@12f76359beae:/# sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'
root@12f76359beae:/# sudo apt-get update
root@12f76359beae:/# sudo apt-get install jenkins
```

2.3.3. Browse and login Jenkins

Browse to http://**<instance IP address>**:8080 (or whichever port you configured for Jenkins when installing it) and wait until the Unlock Jenkins page appears.

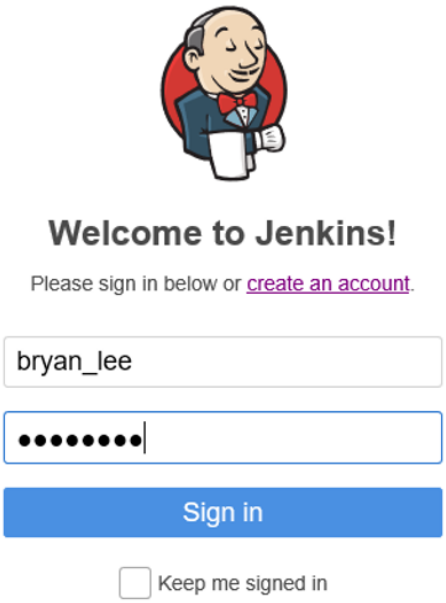


Go back to the Jenkins container console. Run the following command to get the administrator password

```
root@12f76359beae:/# cat /var/lib/jenkins/secrets/initialAdminPassword
```

On the Unlock Jenkins page, paste this password into the Administrator password field and click Continue.

Type the account & password that you set up. Then press Sign In button to login Jenkins.

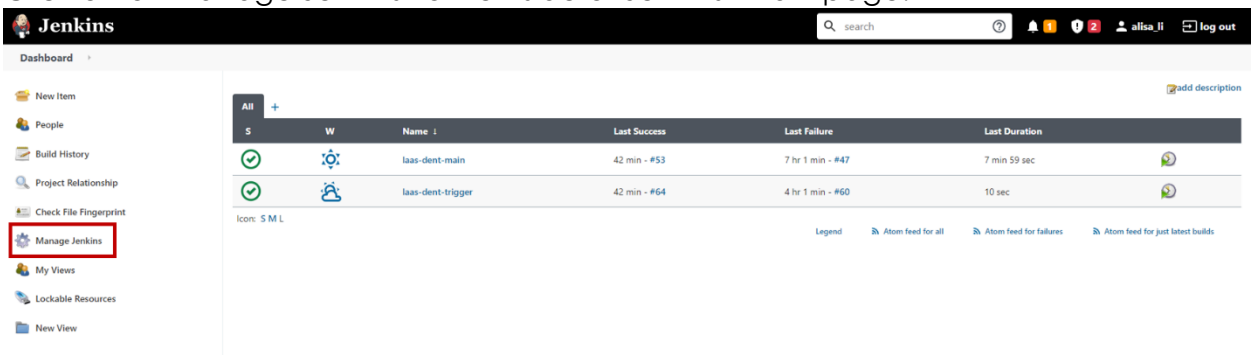


2.3.4. Install Jenkins plugins

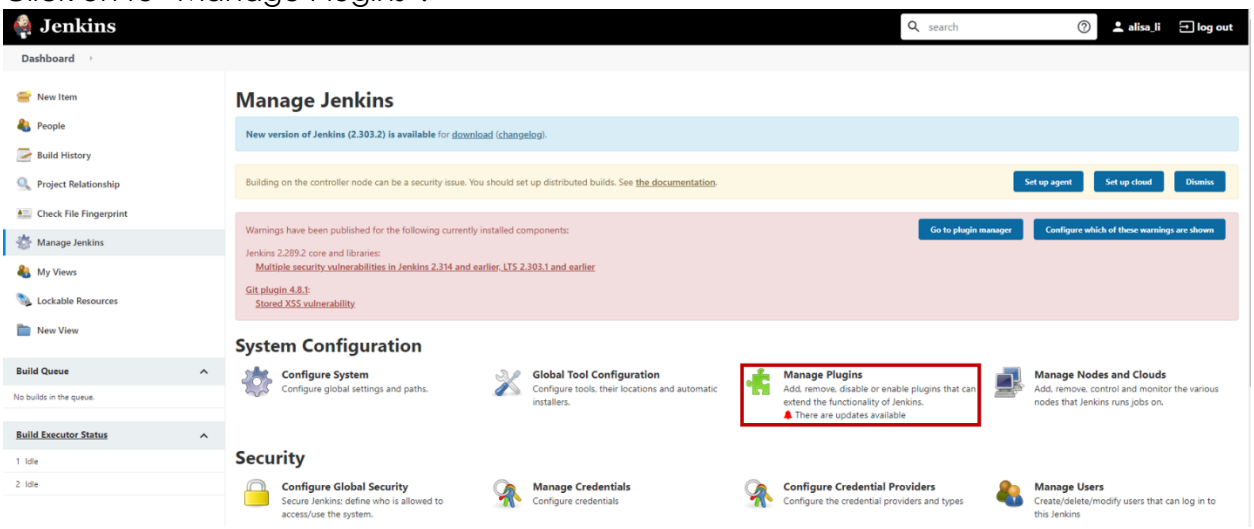
Please install the following Jenkins plugins:

- **Extended Choice Parameter**
- **Active choices**

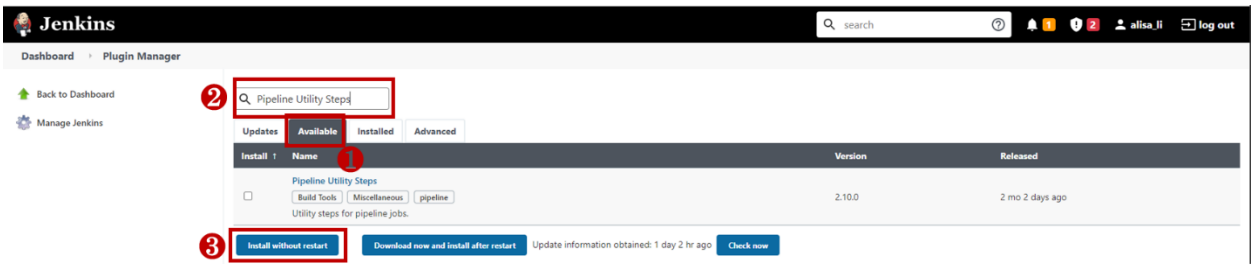
Click onto "Manage Jenkins" on left side of Jenkins' main page.



Click on to “Manage Plugins”.



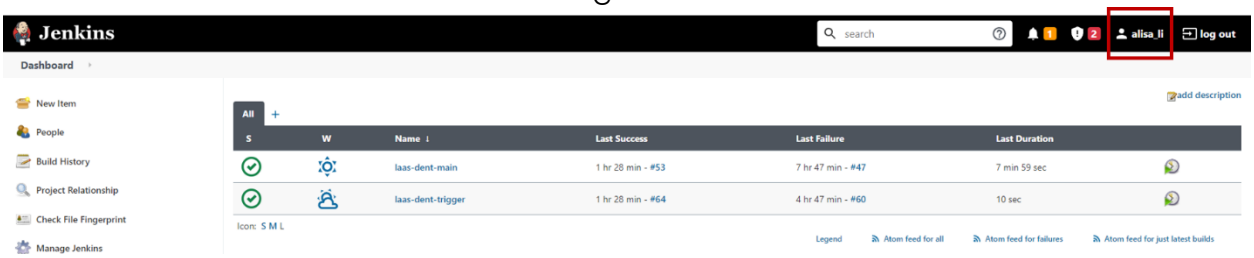
Click onto tab “Available” and search for the name of the plugin. Check for the plugin and click onto “Install without restart”.



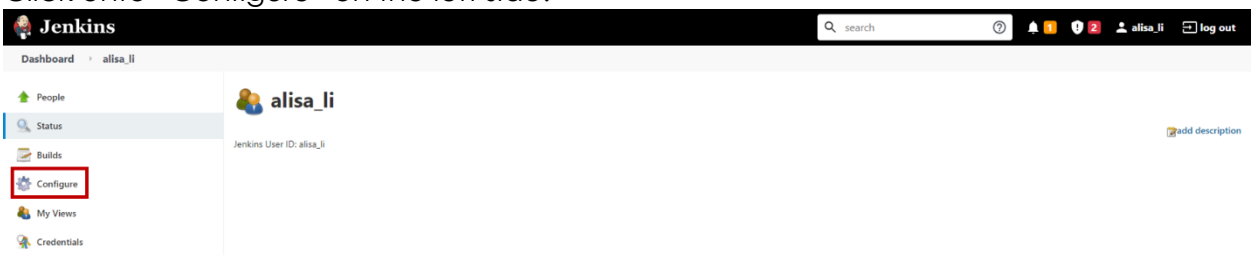
2.3.5. Generate a Jenkins token

In order to trigger the Jenkins job by LaaS server, we are going to generate a Jenkins token for the password. The token is going to be used in the next section, for the credential “jenkins-dent-trigger”.

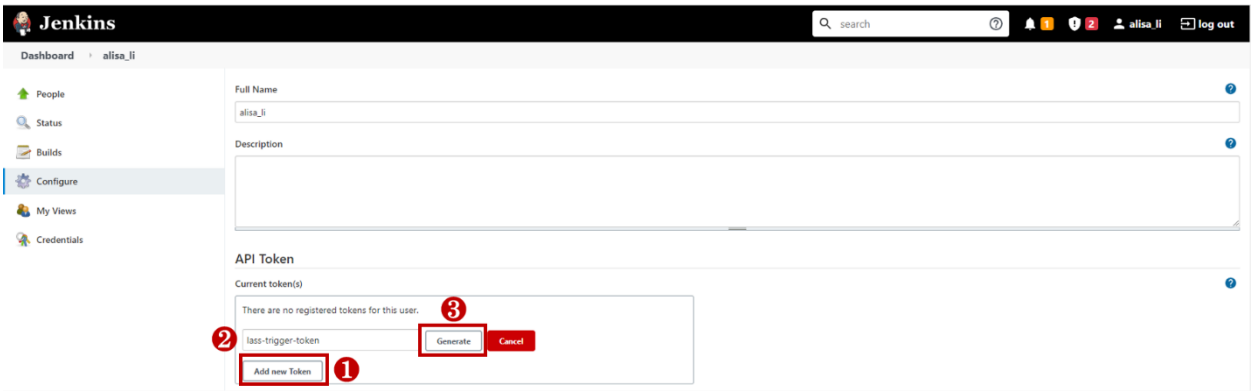
Click onto the “<user name>” on the navigation bar.



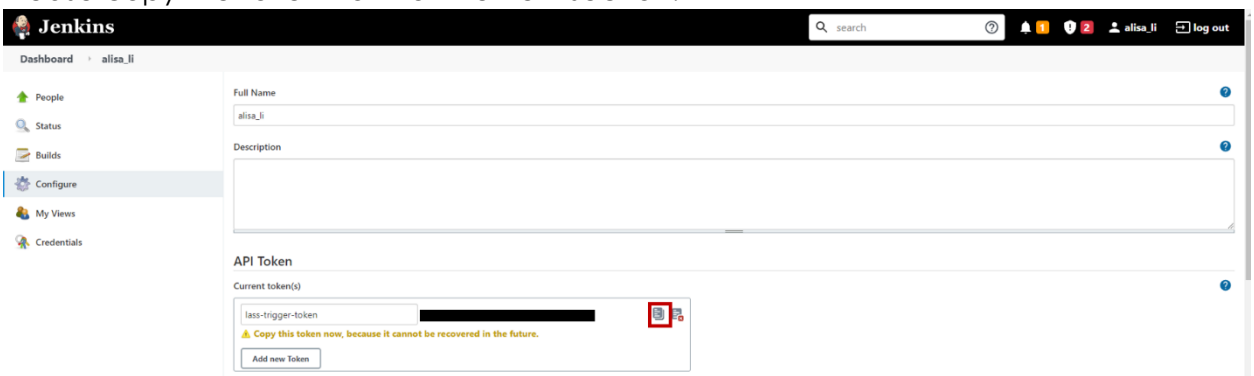
Click onto "Configure" on the left side.



Click onto "Add new Token", fill in the name of the token, and click onto "Generate".



Please copy the token now for the next section.



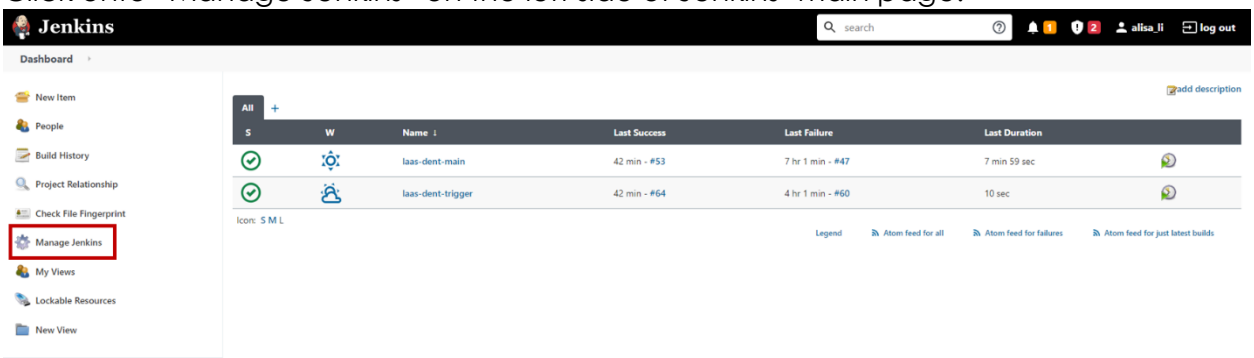
2.3.6. Configure for the Jenkins credential

During the testing, there are several credentials being used. In this section, we are going to guide through the setups for different credentials and their purpose.

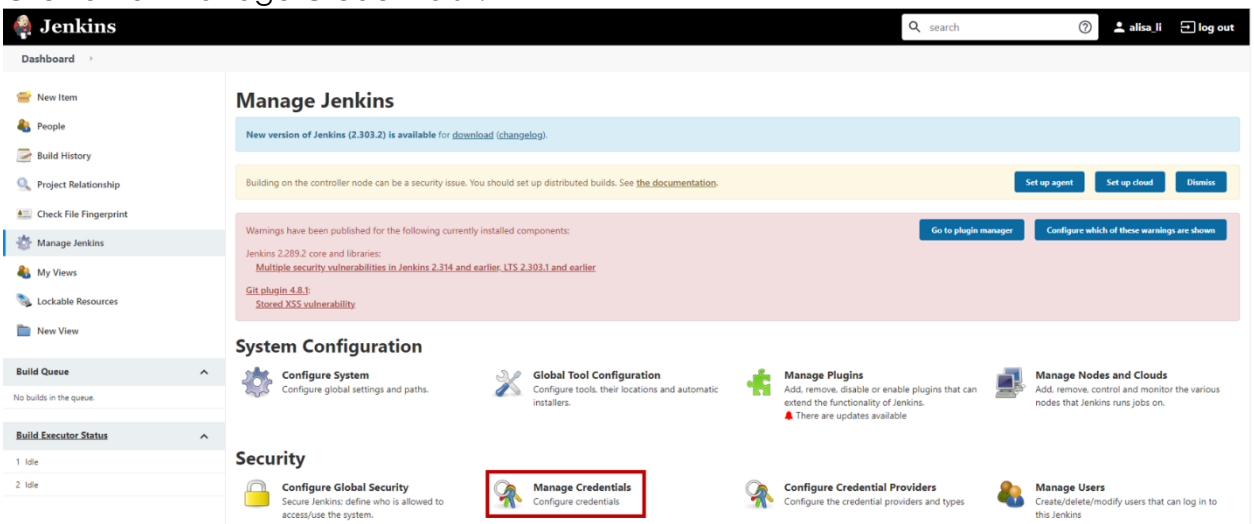
The followings are the credentials to be set up:

- **jenkins-dent-trigger**: the account used to trigger Jenkins job. The password for this credential is the token generated in the previous section
- **laas-dent-test**: a LaaS account for DENT test on Jenkins
- **dent-testbed-server**: the account used to connect to the testbed server
- **dent-image-server**: the account used to download image to HTTP (image) server
- **jenkins_gitlab**: a gitlab account for pulling the source code for testing

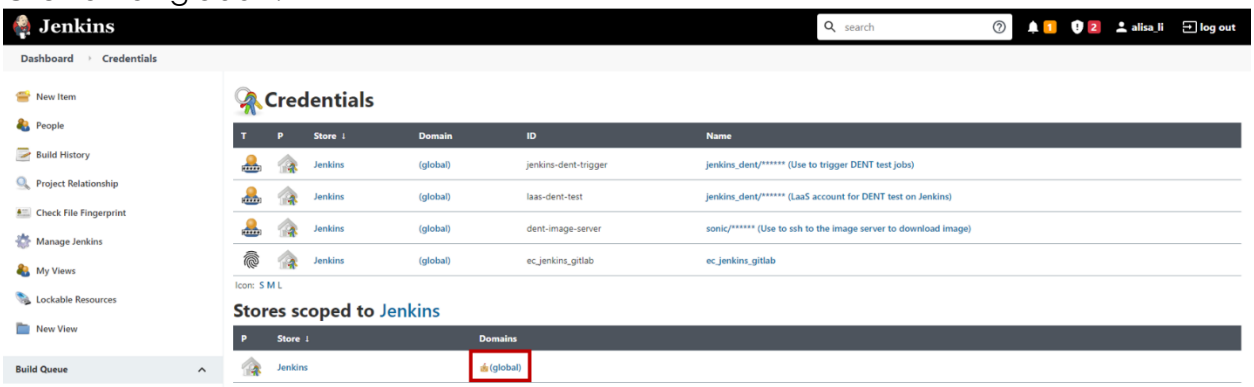
Click onto “Manage Jenkins” on the left side of Jenkins’ main page.



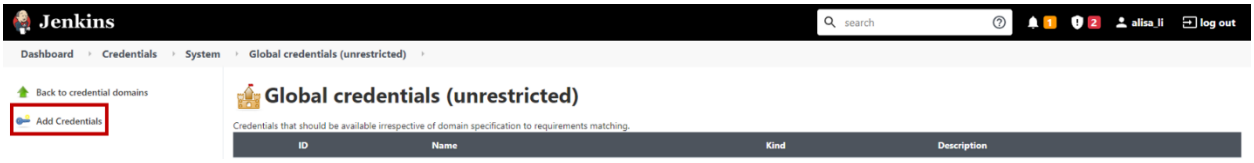
Click onto “Manage Credentials”.



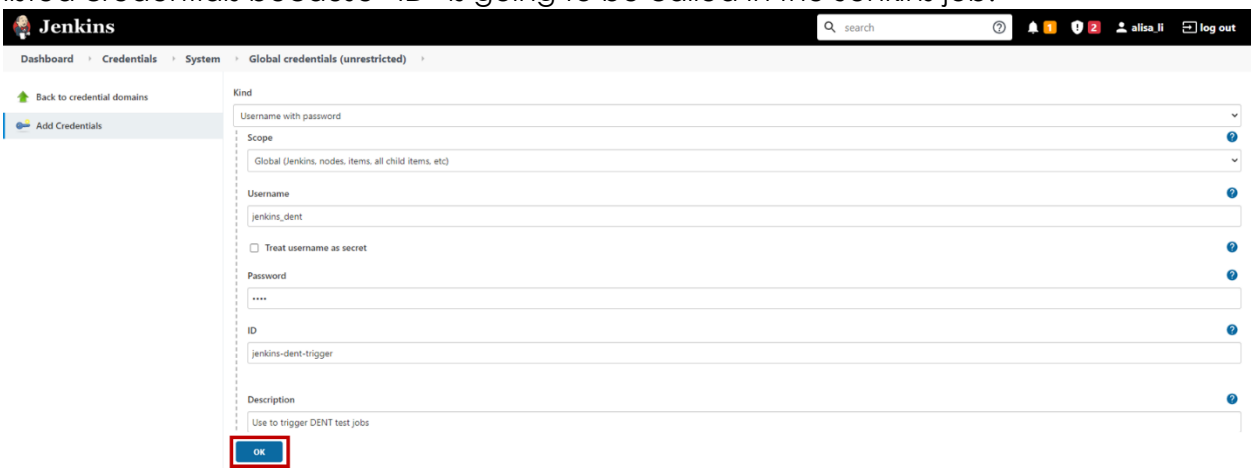
Click onto “global”.



Click onto “Add Credentials” on the left side.



Fill in the form and click onto "OK". Please make sure the "ID" is same as the above listed credentials because "ID" is going to be called in the Jenkins job.

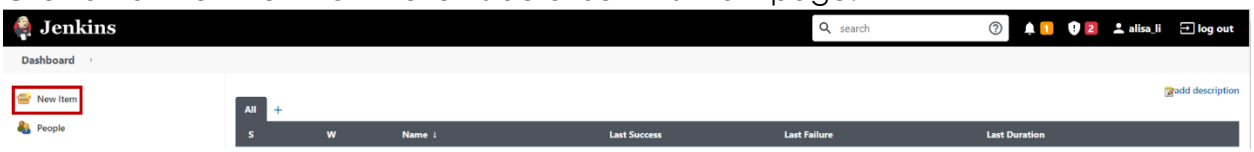


2.3.7. Create Jenkins job

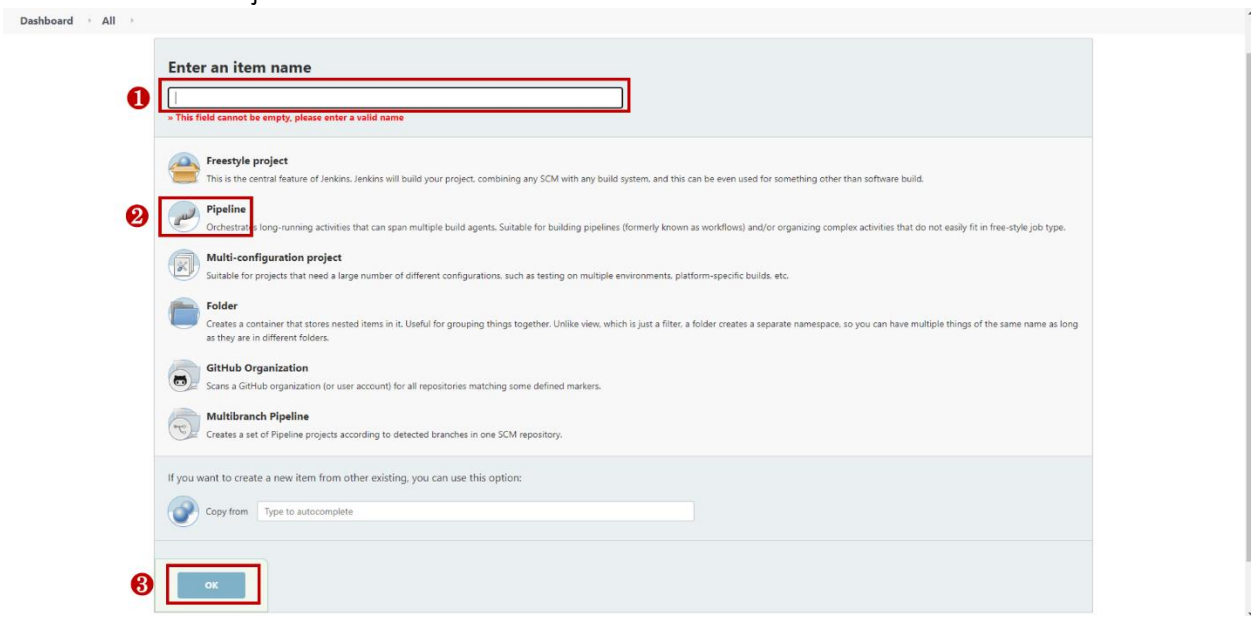
There are two Jenkins job for the DENT Auto testing:

- laas-dent-trigger
- laas-dent-main

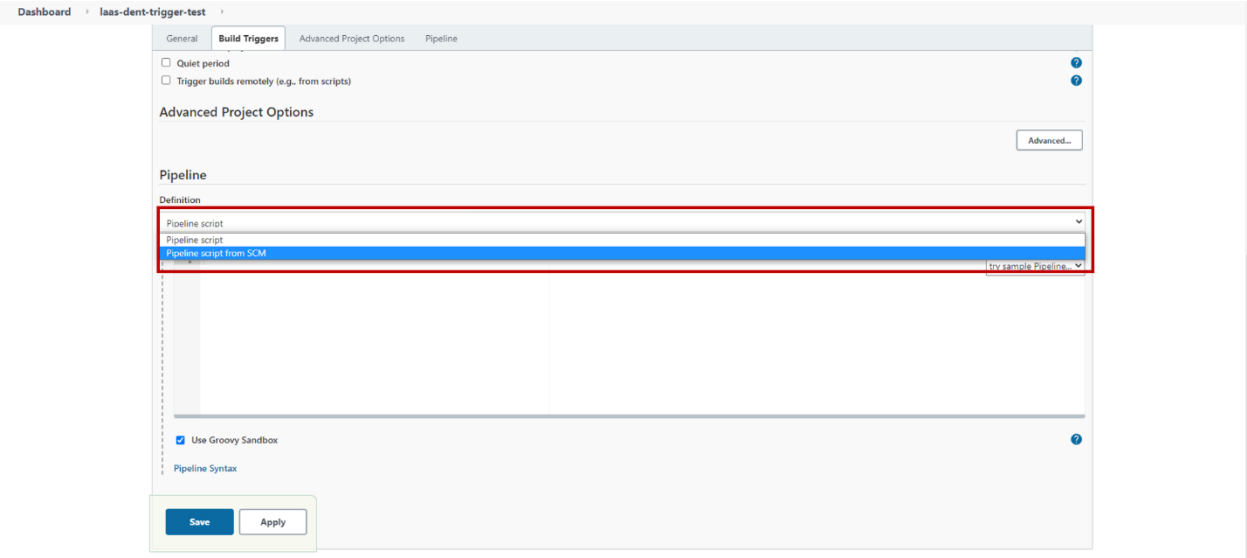
Click onto "New Item" on the left side of Jenkins main page.



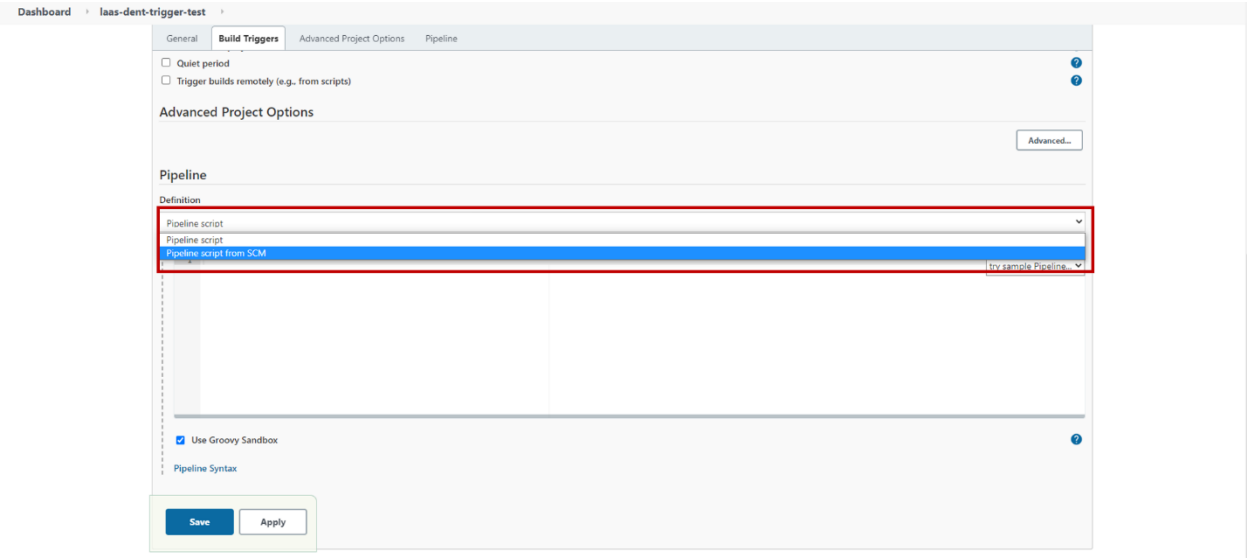
Enter the job name with the above listed job name, click onto "Pipeline" and click onto "OK" to create a job.



Scroll all the way down to the “Pipeline” section and select “Pipeline script from SCM” from the drop down menu.

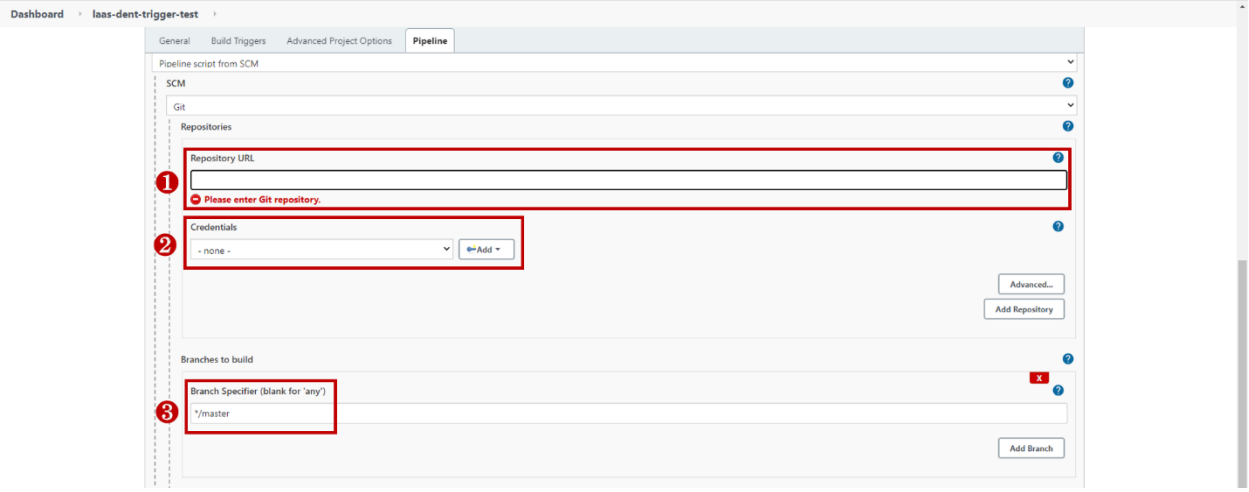


Select “Git” from the drop down menu of “SCM”.

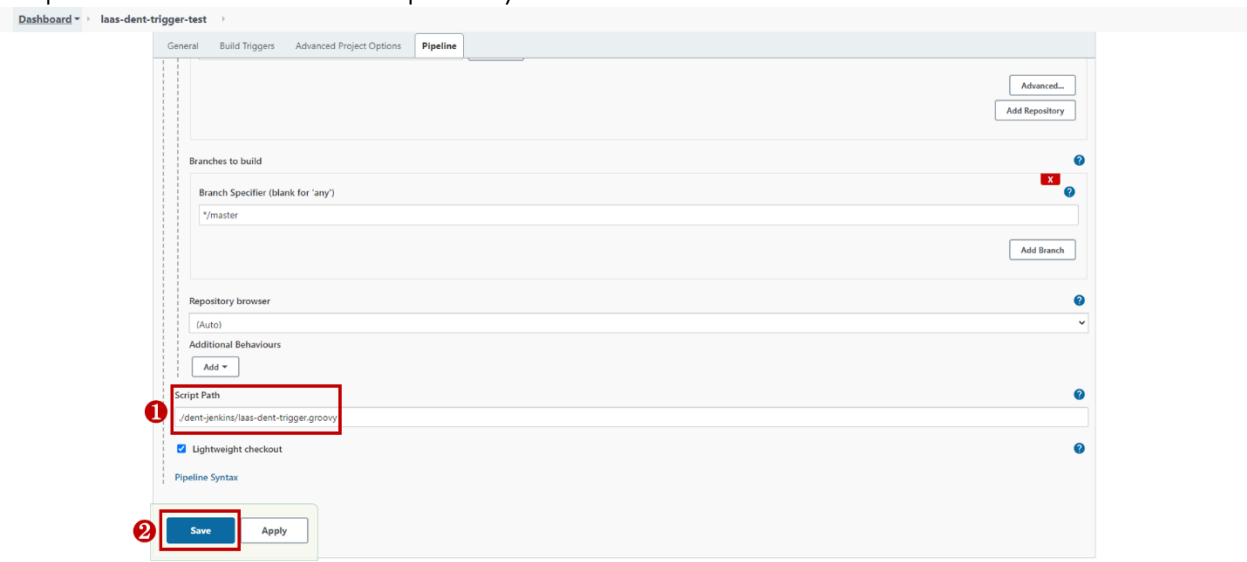


Fill in the “Repository URL”, select the credentials, and fill in the “Branch Specifies” with the branch which the code is at. For the credentials, it is able to select the credential

"jenkins_gitlab" created in the previous section.



Scroll down a little. Fill in the "Script Path" and click onto "Save". The path is where the script is located on the Git repository.



Please do the above steps for both of the Jenkins jobs.

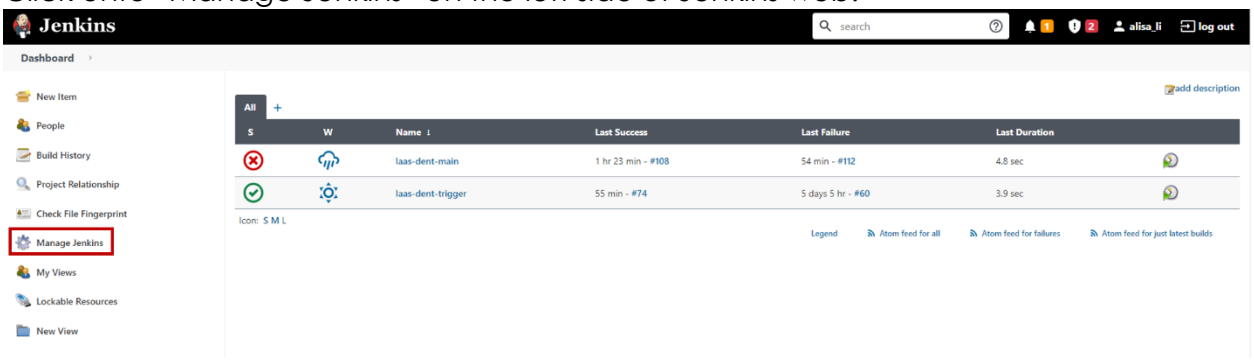
2.3.8. Create Jenkins slave node

If there are multiple labs located in different places, it is required to setup the Jenkins slave node to be able to run the test on the testbed in lab.

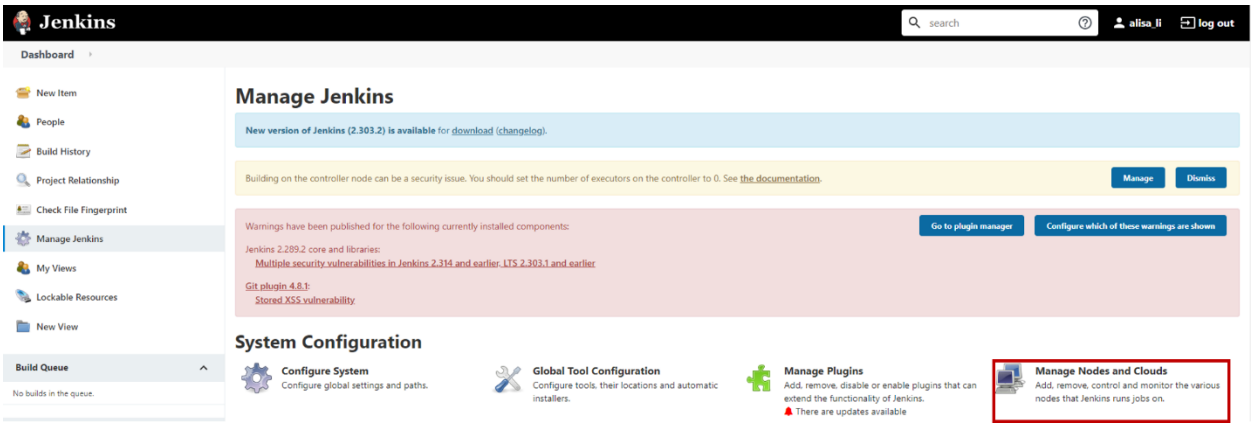
Install the following library in the testbed.

```
$ sudo apt install icedtea-netx
```

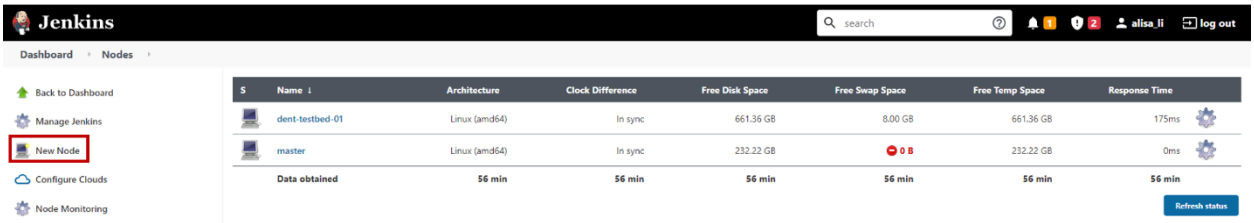
Click onto “Manage Jenkins” on the left side of Jenkins web.



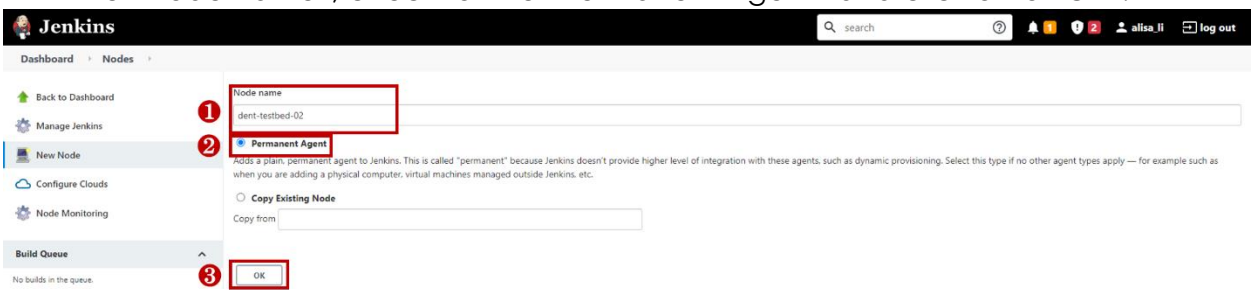
Click onto “Manage Nodes and Clouds”.



Click onto “New Node” on the left side.



Fill in the “Node name”, check for the “Permanent Agent” and click onto “OK”.

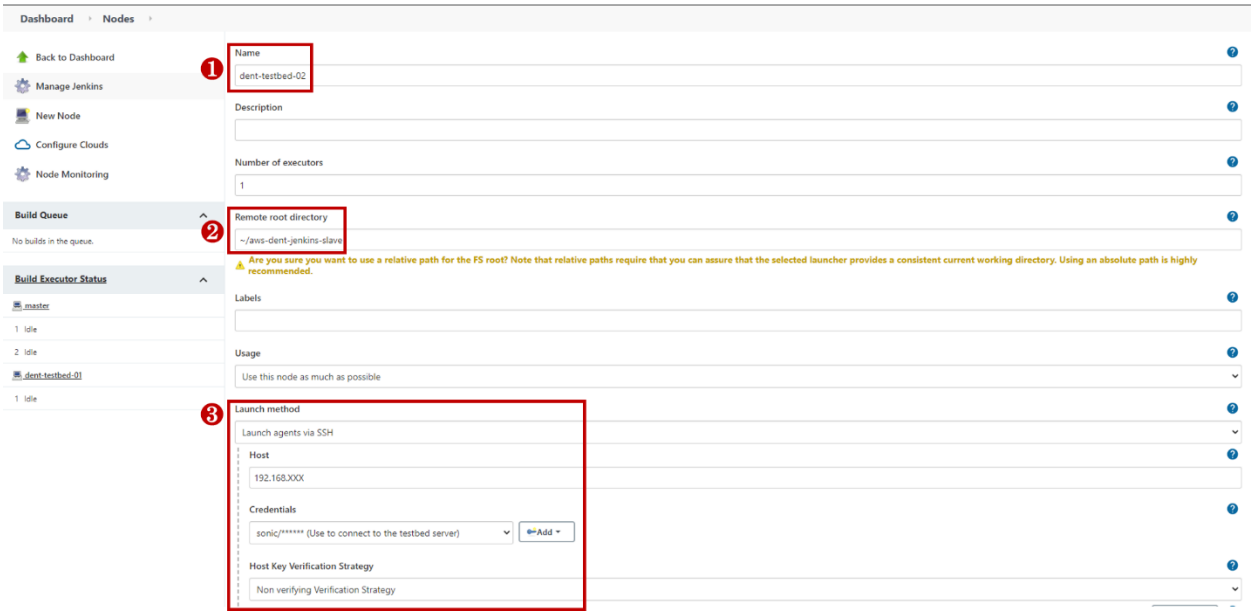


Fill in the node “Name” and “Remote root directory”.

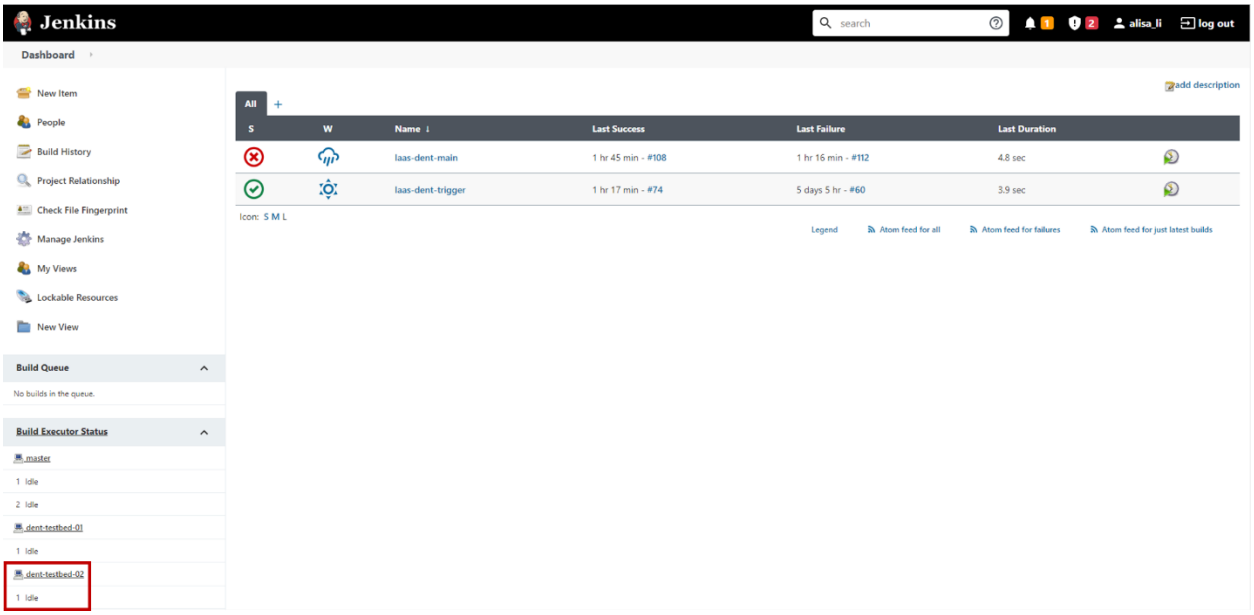
Please note that for the “Remote root directory”, it should be the path that Jenkins has the permission to access.

Choose “Launch agents via SSH” for the drop down menu of “Launch method”, fill the server IP for the “Host” and choose a credential.

Then, click onto “Save”.



The slave need to be launch for few minutes. It is able to check the status on the Jenkins main web.



2.4. Testbed setup

The following libraries or scripts is required to be installed on the testbed:

Please note that if the testbed is same as the Jenkins master, install the following libraries to the same place where Jenkins master is installed or install to the testbed (Jenkins slave).

2.4.1. Pre-required library

Install the following libraries in an Ubuntu18.04 environment where the testbed is located.

```
$ apt get install sshpass
$ apt get install telnet
$ apt install git-all
```

2.4.2. Python and Python library

The Amazon DENT testing framework is required to have python version above 3.6 and pip3 version above 21.1.2. The python library "virtual environment" is required by the LF tool, which will be cover in the following section.

```
python 3.6+
pip3==21.1.2+
virtualenv==20.8.1
pexpect==4.8.0
requests==2.18.4
```

2.4.3. DENT testing source code

The Amazon DENT testing code is required to be cloned on to testbed. For more detailed information, please refers to the [official document](#).

```
$ git clone https://github.com/dentproject/testing.git
$ cd ./testing/Amazon_Framework/DentOsTestbed
$ pip3 install -r Requirements.txt
$ pip3 install .
$ cd ../DentOsTestbedDiscovery
$ pip3 install .
$ cd ../DentOsTestbedLib
$ pip3 install -r Requirements.txt
$ pip3 install .
```

2.4.4. LF tool

The LF tool is used to deploy the test logs to a S3 bucket. The official document highly suggests to install LF tool in a python virtual environment, so our scripts for uploading the logs runs the LF tool in a python virtual environment.

```
$ virtualenv lftool-venv
$ source lftool-venv/bin/activate
(lftool-venv) $ pip3 install lftools
(lftool-venv) $ deactivate
```

2.4.5. AWS cli

The AWS cli is used to set up the configuration and credential for the S3 bucket. For a detailed information, please refer to the official document.

```
$ pip3 install --upgrade awscli
$ aws --version
```

Set up the configuration and credential:

```
$ aws configure
AWS Access Key ID [None]: <replace with your own value>
AWS Secret Access Key [None]: <replace with your own value>
Default region name [None]: <replace with your own value>
Default output format [None]: <replace with your own value>
```

Check the value is set correctly:

```
$ cat ~/.aws/credentials
$ cat ~/.aws/config
```

Please remember to put the credential file to the place where Jenkins can access. For example,

```
$ cat /etc/passwd
jenkins:x:105:106:Jenkins,,,:/var/lib/jenkins:/bin/bash
```

In the above case, please put `~/.aws/` under `/var/lib/jenkins/` to make sure Jenkins is able to access credential while uploading the log to S3 bucket.

```
$ cp -af /root/.aws /var/lib/Jenkins/
```

3. Modify script

Before we jump into the testing, there are several codes need to be modified because the environment may vary.

Please clone the scripts from the source.

3.1. Python script

3.1.1. Information for LaaS server

Change the IP address or domain name for the LaaS.

dent-auto-test/dent-script/library/LIB_LaaS.py

dent-auto-test > dent-script > library > LIB_LaaS.py

```python
import requests
import os, sys
from http import HTTPStatus
import urllib3

# To disable the ssl warning message on console
# InsecureRequestWarning: Unverified HTTPS request is being made.
# Adding certificate verification is strongly advised.
# See: https://urllib3.readthedocs.io/en/latest/advanced-usage.html#ssl-warnings
urllib3.disable_warnings(urllib3.exceptions.InsecureRequestWarning)
class LaaSApi():
    '''
        Class Name: LaaSApi
        Purpose:
            Methods relate to the LaaS Api
    '''
    def __init__(self):
        self.SERVER_URL = "https://[redacted]:443/api/1.0"
        self.MAX_RETRY_TIME = 20
        self.is_debug_mode = False
        self.using_ssl = False
        self.headers = dict()
```

3.1.2. Device gateway

Please do this modification if it is needed.

Change the IP address for “self.gateway” or leave it as “None”.

dent-auto-test/dent-script/library/LIB_Utils.py

dent-auto-test > dent-script > library > LIB_Utils.py

```python
import pexpect
import sys, time, re, json
import json

class UI():
    '''
        Class Name: UI
        Purpose:
            Methods relate to the DUT user interface or utility
    '''
    def __init__(self, prompt, server, port, username, password, ip, netmask):
        self.device = None
        self.prompt = prompt
        self.server = server
        self.port = port
        self.username = username
        self.password = password
        self.ip = ip
        self.netmask = netmask
        self.gateway = None
```

3.1.3. Mapping file of testbed and device

This mapping file is used for getting the testbed that the device is able to run on under a domain.

This information should be the same as the information we register into the LaaS server.

dent-auto-test/dent-script/testbed_device_mapping.json

The format should be like this:

```
{
    "<domain name>": {
        "testbed": "<testbed name>",
        "device": [
            "<device name 1>",
            "<device name 2>"
        ]
    }
}
```

Please note that the testbed should be the same as the Jenkins slave node name.

dent-auto-test > dent-script > { } testbed_device_mapping.json >

```
1  {
2    "Dent": {
3      "testbed": "dent-testbed-01",
4      "device": [
5        "/tainan/as4xxx/as4224-1",
6        "/tainan/as5xxx/as5114-1"
7      ]
8    }
9  }
```

3.2. Jenkins pipeline code

3.2.1. HTTP server information

dent-auto-test/dent-jenkins/laas-dent-main.groovy

Please search for the keyword "http_server" and change the IP address for its value.

dent-auto-test > dent-jenkins > laas-dent-main.groovy

```
88      string(name: 'reservation_duration', defaultValue: "600", description: 'The unit is se
89    ])
90  ])
91
92  pipeline {
93    agent {
94      label 'master'
95    }
96    environment {
97      repository = "https://nexus.dent.dev/content/repositories/snapshots/org/dent/dentos/"
98      http_server = "[redacted]"
```

3.2.2. Git information

Since the code may be put onto a private Git repository, please modify the related code in both of **dent-auto-test/dent-jenkins/laas-dent-trigger.groovy** and **dent-auto-test/dent-jenkins/laas-dent-main.groovy**.

Please search for keyword "dent-auto-test" and change for the desired value as below:

```
checkout([$class: 'GitSCM', branches: [[name: '*/<branch name>']],
doGenerateSubmoduleConfigurations: false, extensions: [[$class:
'SparseCheckoutPaths', sparseCheckoutPaths: [[path: 'dent-script/']]]],
```

```
submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'jenkins_gitlab', url: '<url
of repository>']]])
```

Example:

```
checkout([$class: 'GitSCM', branches: [[name: '*/master']],
doGenerateSubmoduleConfigurations: false, extensions: [[$class:
'SparseCheckoutPaths', sparseCheckoutPaths: [[path: 'dent-script/']]]],
submoduleCfg: [], userRemoteConfigs: [[credentialsId: 'jenkins_gitlab', url:
'git@gitlab.edge-core.com:root/dent-auto-test.git']]])
```

3.2.3. Modify for the Jenkins UI

dent-auto-test/dent-jenkins/laas-dent-trigger.groovy

Add the device in the LaaS to Jenkins UI so that it is able to be selected.

dent-auto-test > dent-jenkins > laas-dent-trigger.groovy

```
1  properties([
2    parameters([
3      extendedChoice(
4        name: 'device',
5        description: '',
6        type: 'PT_SINGLE_SELECT',
7        visibleItemCount: 5,
8        multiSelectDelimiter: ',',
9        value: 'as4224,as5114',
10       defaultValue: 'as4224'
11     ),
```

dent-auto-test/dent-jenkins/laas-dent-trigger.groovy

Change the script path of DENT source code to be able to get the list of dent suite groups.

Please make sure the [DENT community source code]() is cloned onto the Jenkins master.

```
dent-auto-test > dent-jenkins > laas-dent-trigger.groovy
58      [$class: 'CascadeChoiceParameter',
59        name: 'test_suite_group',
60        description: 'The DENT test suite group.',
61        choiceType: 'PT_CHECKBOX',
62        filterLength: 1,
63        filterable: true,
64        referencedParameters: 'select_all_suite_group',
65        script: [
66          $class: 'GroovyScript',
67          fallbackScript: [
68            classpath: [],
69            sandbox: false,
70            script:
71              'return[\'Could not get test_suite_group.\']'
72          ],
73          script: [
74            classpath: [],
75            sandbox: false,
76            script:
77              '''def list = []
78                  String fileContents = new File('home/dent_volume/testing/Amazon_Framework/DentOsTestbed/src/dent_os_testbed/constants.py').text
79
80                  fileContents.eachLine { line ->
81                    if (line =~ /^.+(suite_group_.+)":.*$/) {
82                      suite_group = (line =~ /^.+(suite_group_.+)":.*$/)[0][1]
83                      if (select_all_suite_group == 'yes') {
84                        list << suite_group.concat(":selected")
85                      } else {
86                        list << suite_group
87                      }
```

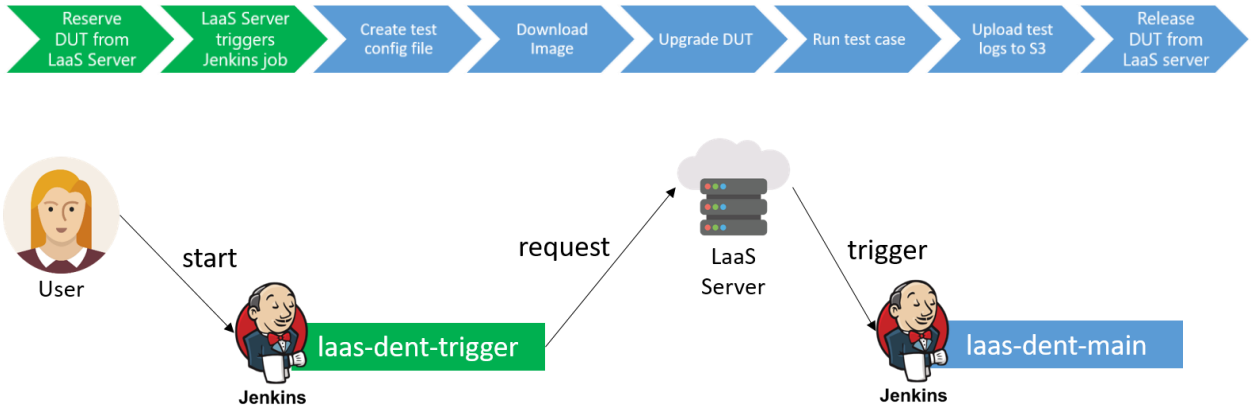
After all the modification is done, please push back the scripts to the Git to make sure the script is up-to-date and ready to be pulled by Jenkins job to run the test.

4. Run the test

In this section, we are going to guide through the brief introduction of the DENT test automation workflow to how to trigger the Jenkins job to start a test.

4.1. DENT test automation workflow

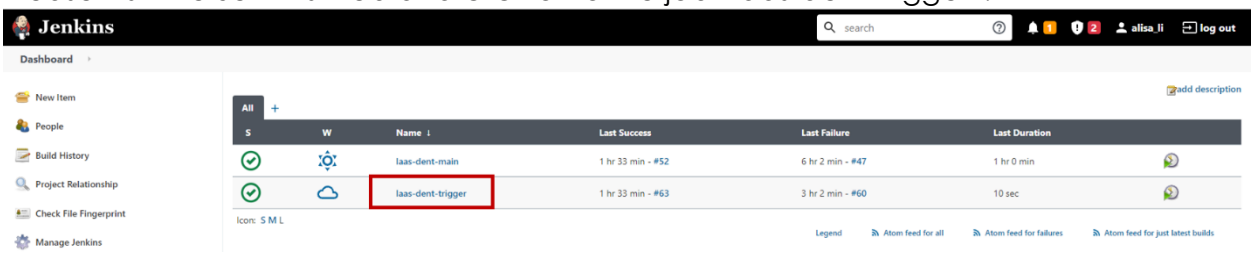
When a user starts a Jenkins job "laas-dent-trigger", this job will make a request to LaaS server to reserve a device. Once the device is available, the LaaS server will then trigger the Jenkins job "laas-dent-main". The job "laas-dent-main" includes downloading image from DENT image repository, upgrading DUT, running test case, uploading test logs to S3 bucket, and release DUT from LaaS server.



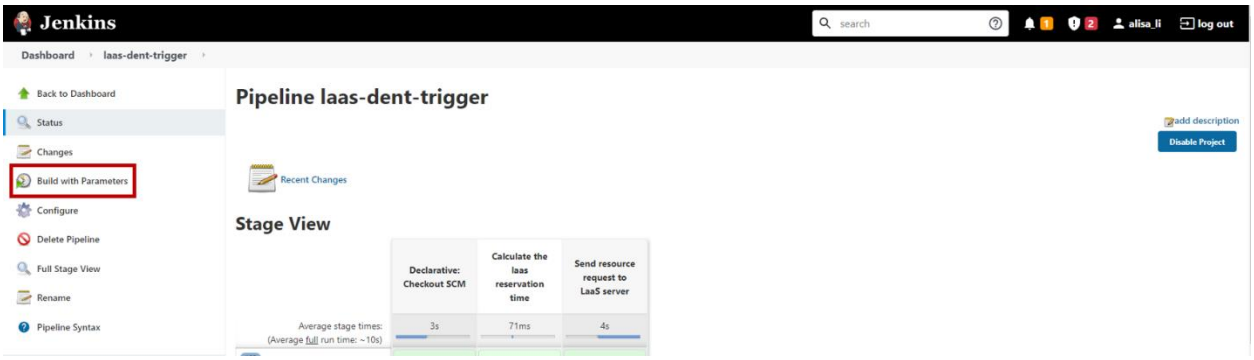
4.2. Steps to run the test

4.2.1. Start a Jenkins job

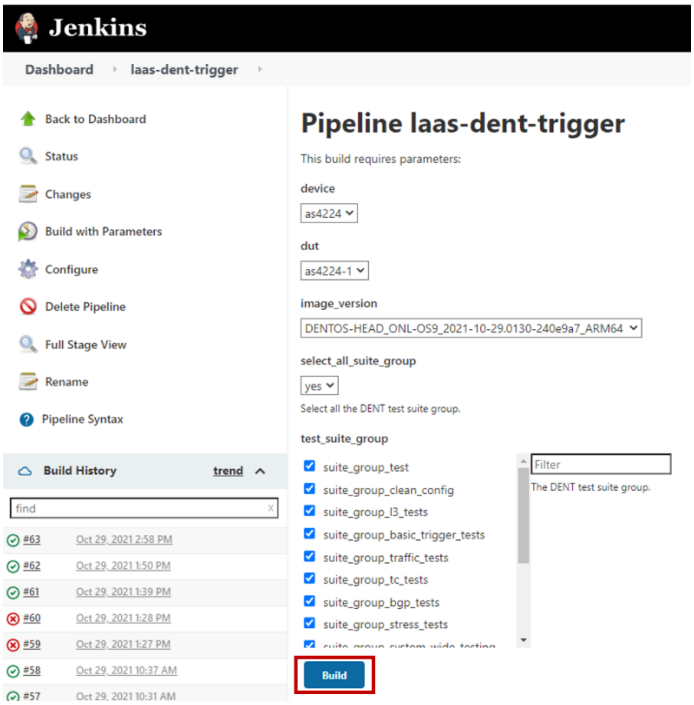
Please visit the Jenkins web and click onto the job "laas-dent-trigger".



Click onto "Build with Parameters" on the left bar.

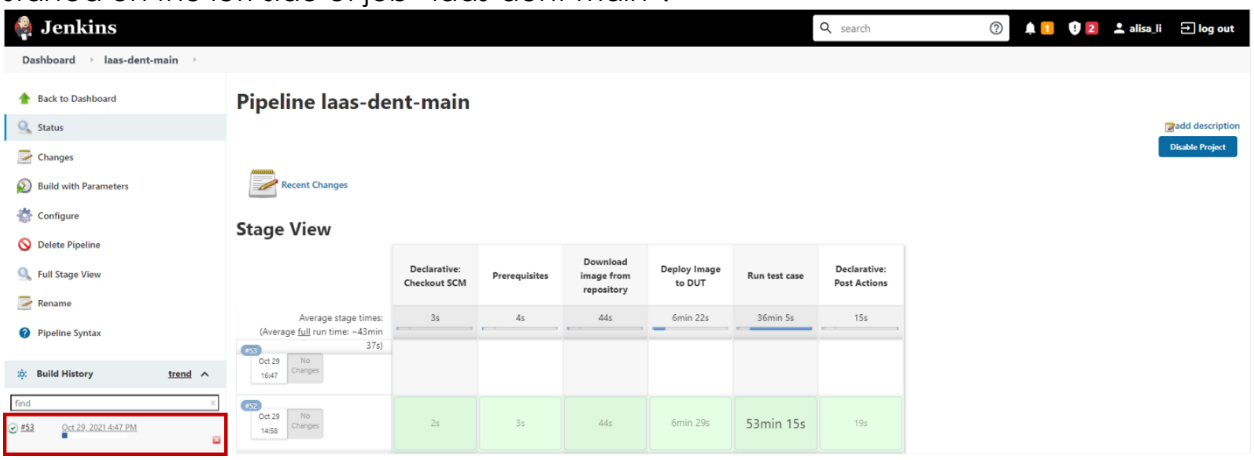


Choose the desired value for each parameter on the UI and click onto "Build" to start a Jenkins job.



4.2.2. Check for the testing job

After the device selected for "laas-dent-trigger" is ready, the LaaS server will trigger a downstream Jenkins job "laas-dent-main". We are able to check whether the job has started on the left side of job "laas-dent-main".



After the job finishes, we are able to access the logs by clicking onto "S3 Logs".