Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

Aim:

Introduction

Bevywise IoT Simulator is a GUI based, data simulation tool which is used to load and test the MQTT/IoT Application to know the performance level of that Application. In IoT simulator, you can create virtual IoT networks and devices, publish events with normal or complex JSON payload, create subscriptions and more. IoT simulator help documentation will let you know how to simulate IoT networks & IoT devices. For reference, refer to our introduction video tutorial.

System requirements

OS – Windows 7 or Windows 8 & 8.1, Ubuntu 14.04 or higher.

CPU – Pentium 4 or higher.

Memory – 512 MB RAM or more.

Hard drive – 100 MB of free disk space

Installation

Windows Installation and setup

- Just Double click the downloaded“Bevywise_IoTSimulator_Win_64.exe” file to open the installation window.
- In Installation Window, give the installation path and install it.
- Next, open command prompt and go to the “bin” folder(< Installation path>/Bevywise/Iotsimulator/bin).
- Now type “runsimulator.bat” and hit enter to start the IoT Simulator. Once you start the IoT simulator user interface will open in your default web browser. If not you can directly go to your browser and navigate to http://localhost:9000/
- For reference, refer to our installation video.

Linux Installation

- Unzip the downloaded “Bevywise_IoTSimulator__Linux.zip” file and extract to the location you want, Now Bevywise folder will be created in the extracted location.
- Open the Linux terminal and go to the bin folder(< Installation path>/Bevywise/Iotsimulator/bin)
- To run the simulator, enter the following command : “sh runsimulator.sh”.
- Once you start the IoT simulator, user Interface will open in the Default web browser, in which you can see the sample IoT network called Health_care. If not you can directly go to your browser and navigate to http://localhost:9000/

```
$ sh runsimulator.sh
```

Default running configuration and port

- Default Data storage – SQLite
- Default user interface port – 9000.
- Websocket Port -12345.

Creating an IoT Network

- To create a Network, click the bar icon on the top bar and select ” New Network”
- Provide Network name & Description of the Network. The network name should be Alphanumeric and can have underscore.
- After providing the details, hit create. After creating a network, the Simulator Setting window will pop-up.

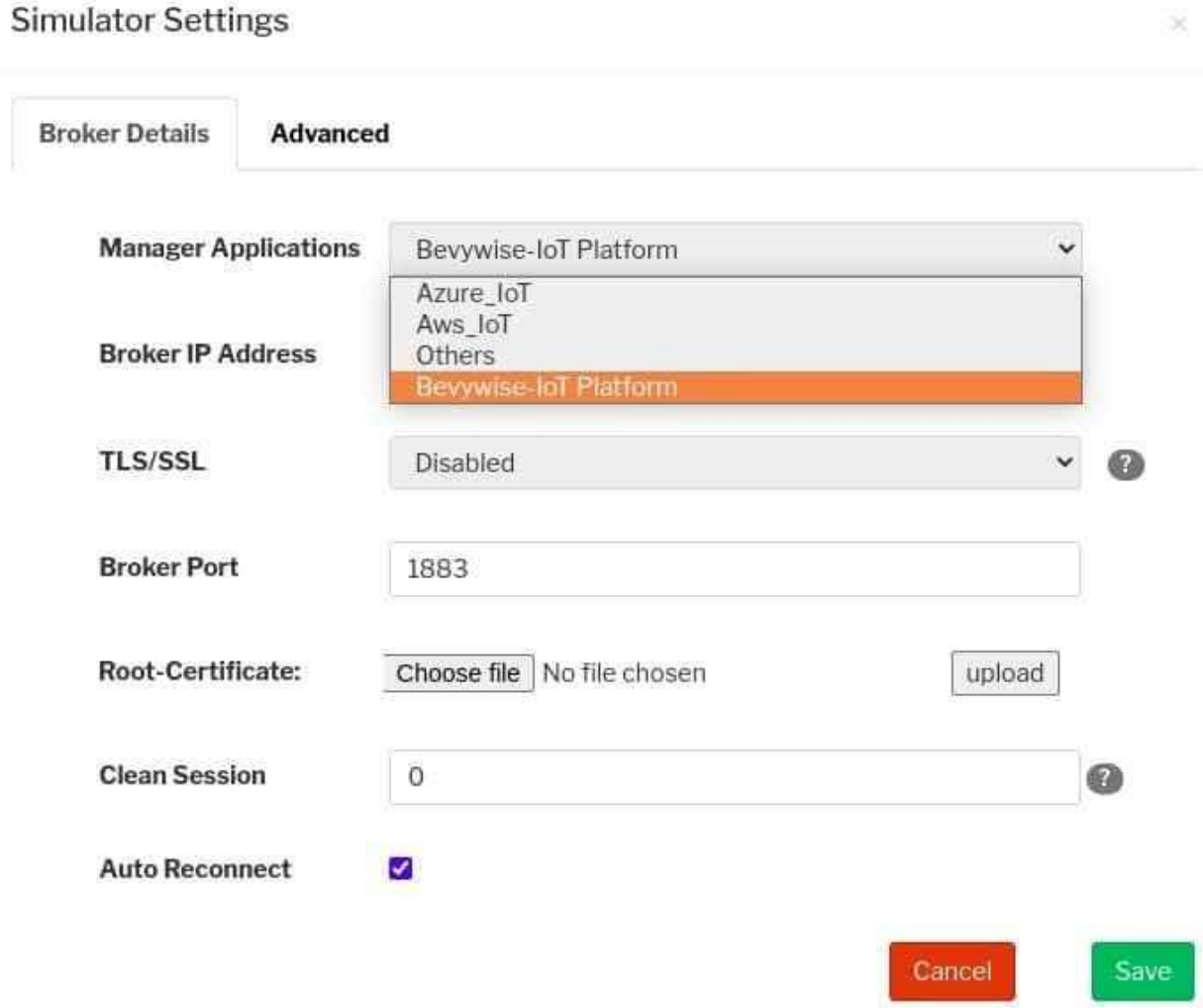
Simulator Settings

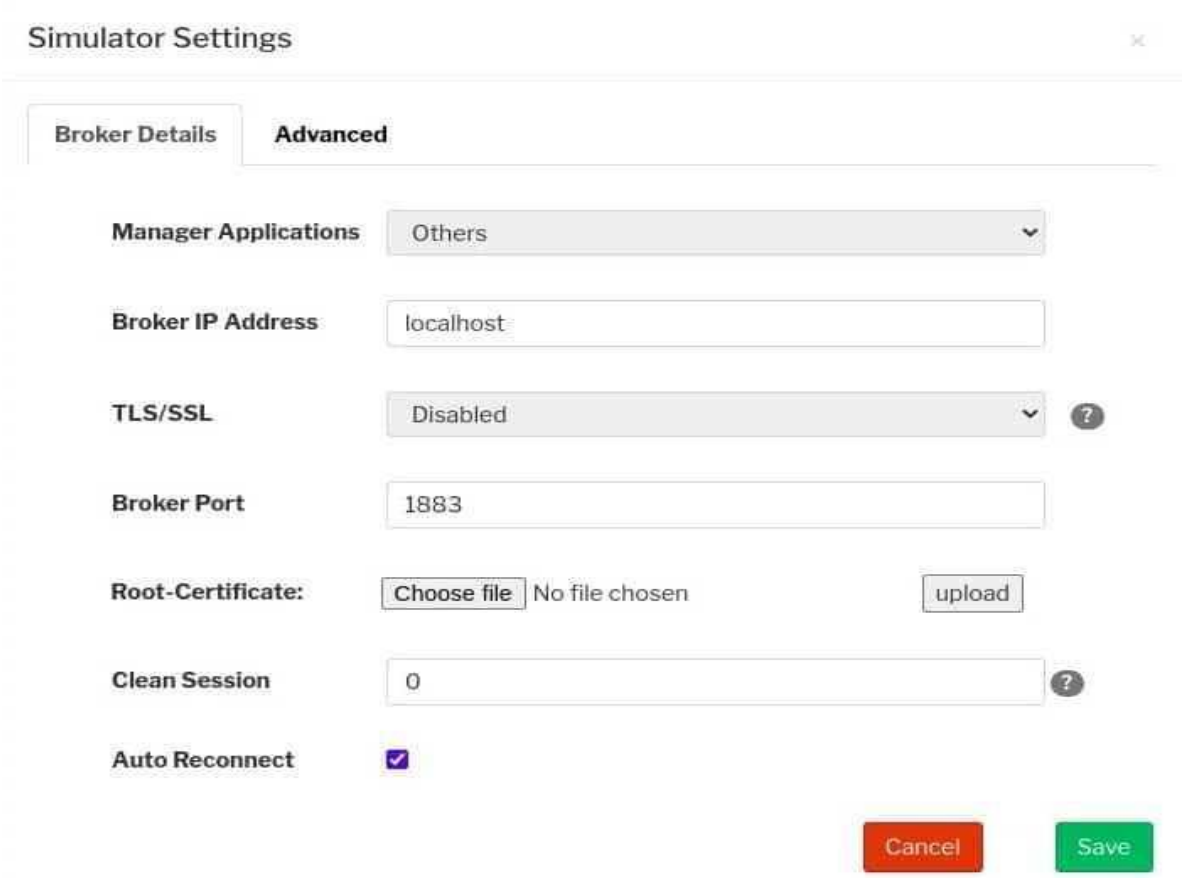
1. Broker Details settings

Manager Application:
Manager Application has four options, they are

- Azure: This option is used to connect the IoT simulator to Azure IoT Hub.

- AWS: This option is used to connect the IoT simulator to the AWS IoT core.
- Other: If you want to connect the IoT simulator to any other MQTT Application or Broker, choose this option.
- Bevywise IoT Platform: This option is used to connect the IoT simulator to the Bevywise IoT platform.


Simulator Settings
Broker Details
Advanced
Manager Applications
Bevywise-IoT Platform
Azure_IoT
Aws_IoT
Others
Bevywise-IoT Platform
Broker IP Address
TLS/SSL
Disabled
Broker Port
1883
Root-Certificate:
Choose file
No file chosen
upload
Clean Session
0
Auto Reconnect
Cancel
Save



Broker IP address

- Here you should specify the PC IP address where Broker is running or DNS/Hostname of the MQTT Application. For example, if you want to connect to AWS IoT core, you must specify the Host or DNS name of the AWS IoT Core and suppose if you want to connect to the MQTT broker which is running in your localhost or in any server, then you must give the IP address of the PC or server where MQTT broker is running. If IoT simulator and MQTT Broker are running in the same PC or server, then give "Broker IP address" as "localhost".

Note: The Manager Application and Broker IP address settings will not be modified after saving the Simulator settings.

TLS/SSL

- Enable this option if the Broker or your MQTT Application is running with TLS/SSL enabled.

Broker Ports

- Specify the Broker or MQTT Application port number. By default, if your Broker runs without TLS/SSL then the port will be 1883 and if TLS/SSL is enabled, then the port should be 8883.

Root Certificates

- Here you should upload the root certificate of the MQTT Application or MQTT Broker. This will enable only for TLS/SSL.

Clean Session

- If the clean session is enabled, the broker will delete all the details about the clients after the disconnection. As a result, when the client connects next time, it will look like a new device to the broker and the actions that are in the previous session won't affect the current section. By default, clean session will be 0.
- Clean Session = 0, means Clean Session is disabled.
- Clean Session = 1, means Clean Session is enabled.

Auto Reconnect

- If 'Auto Reconnect' is enabled, when a network gets disconnected due to any issues, it will automatically get connected.

2. Advanced Settings

Client IP address:

This feature will assign a separate IP address for each client in the Network. For example, if you create 5 devices, then by default the 5 devices will connect to the MQTT broker with the same IP address. But, when you enable "Client IP address" and give IP address range [like

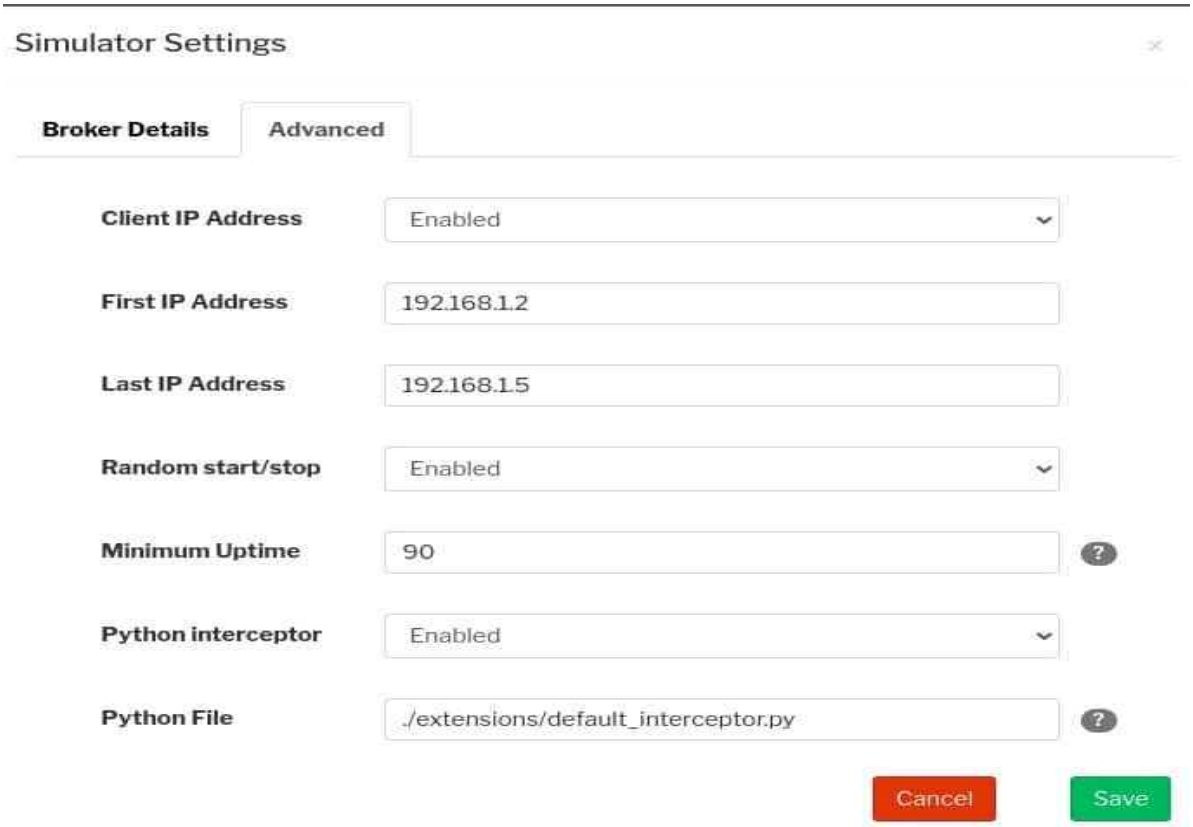
192.168.1.2 to 192.168.1.5] in First & Last IP address, then the IoT simulator will automatically assign separate IP address for each created client or device. Now the device will connect to the MQTT Application with a different IP address. To enable, follow the below steps

- Click the setting button on the top left corner.
- In the Settings window of Simulator, click Advanced tab and enable the Client IP Adrress.
- Next, specify your IP range in First IP address and Last IP address.[Support only IPV4]
- Now the simulator will assign separate IP address for each created device

Random Start/Stop:

Bevywise IoT Simulator will randomly stop and start the created IoT Devices based on the minimum uptime or runtime using the “Random Start/Stop” feature. For example, if you specify the minimum uptime as 50%, then the devices will randomly stop when it reaches 50% running or uptime. To enable, follow the below steps

- Click the setting button on the top left corner.
- In Simulator setting window click Advanced tab and enable the “Random Start/Stop”.
- Next, specify Minimum Uptime [range for Minimum Uptime should be 1 to 100 percent].
- Now the Simulator will randomly stop and start the created IoT devices.



Advanced Python Interceptor:

Bevywise IoT Simulator can be customized using a python-based interface called Python interceptor. This interceptor provides all the created clients' information and its received payload via python file called "default_interceptor.py". In that python file, you can write your own code based on the received payload and also it allows you to call API to simulate complex scenarios. To enable, please follow the below steps

- Click the setting button on the top left corner.
- In Settings window of Simulator, click Advanced tab and enable Python interceptor.
- Specify the default_interceptor.py file path in Python File. Default path should be [./extensions/default_interceptor.py]
- Start writing your code in default_interceptor.py file to customize the simulator based on received payload.
- For reference, refer to our Simulator settings video tutorial.

Creating an IoT Device

- Click the same Plus Symbol icon on the top left corner and select the “Blank Device” option
- In the following dialogue box, fill the Device name (Device name should be in Alphanumeric and in underscore) without any Space.
- Fill unique Device ID (maybe in Special Character, Alphanumeric), which will give an identity for your Devices.
- Fill Description or Function about the Device and hit the Save button.
- A Blank device will be created without any Events on the Corresponding Network.

Configuring Authentication

Edge device authentication will provide a secure connection between the MQTT application and device. Likewise, Bevywise IoT Simulator supports edge device authentication to connect the created device to the MQTT application securely. To enable Authentication,

- Select the Device from the Device list.
- Click the Red checkbox next to Authentication to enable the Authentication.
- Next, give the MQTT username in Access Key and MQTT password in Access Token.

IoT simulator has two types of Authentication which help you to connect to different MQTT/IoT applications. They're :

SAS : With a SAS token, you can grant created clients access to your MQTT Application. It will be useful when you connect to MQTT Applications like Azure.

Self Signed : If your MQTT application allows access to clients with client or device certificate and key, then you can select Self Signed. Here you should upload the client certificate and key which you have got from the MQTT Application.

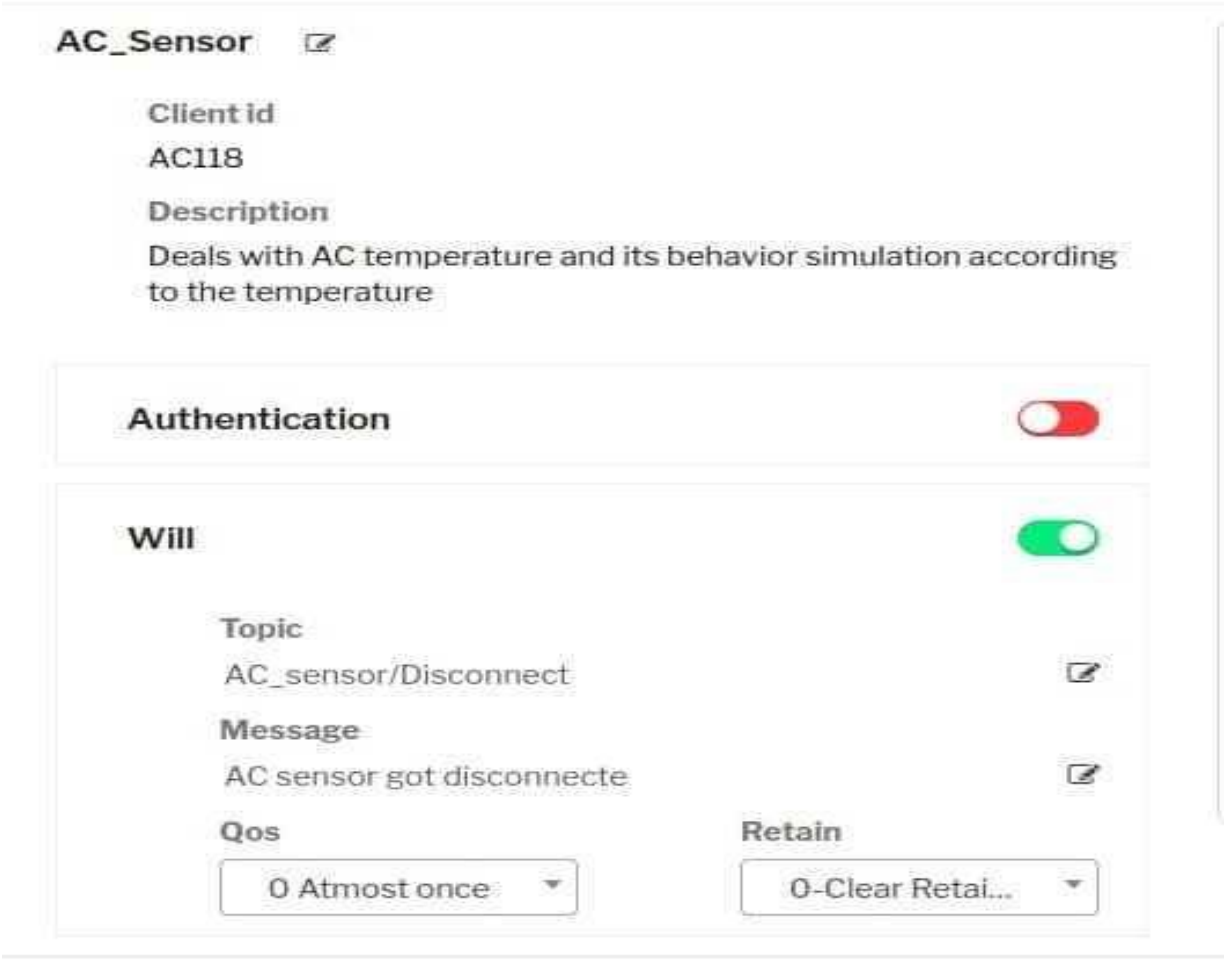
Note: This Authentication type will appear when you select Azure and AWS in Manager Application on Simulator Settings.

For other MQTT Application, you can directly copy & paste the client.certificate and key inside the client folder[./Bevywise/IotSimulator/Certificate]. After pasting the certificate and key, you must rename the certificate client.crt and client.key. For Root certificate, you can add it via simulator settings.

Configuration of MQTT Will Message

In case, if one of the IoT devices got disconnected due to some issue it can publish the events to the subscriber(who subscribes to the topic in that device) only when the WILL message is enabled. To enable the WILL message, select the WILL checkbox and give the topic, message, QoS, etc.

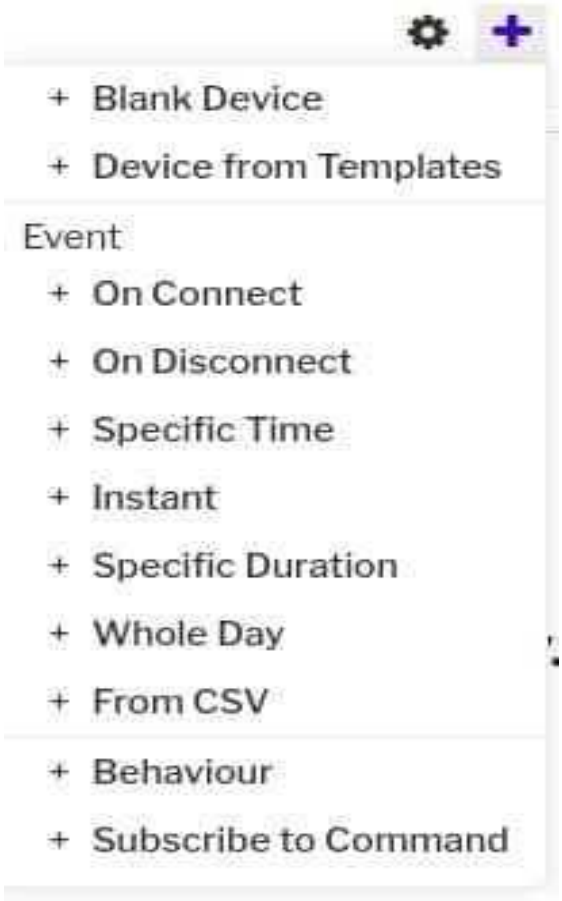
- To enable Will message, select the device from the Device list and click the Red Checkbox parallel to the Authentication tab, it will change to the Green Checkbox.
- Click the edit icon and give the Topic and Message.
- We can enable Qos and Retain for Will messaging like Device Events



Configuration of IoT Events

Events are used to create a time-based publishing message for the IoT devices. Based on the time-based publishing Message, Broker will collect the data from IoT Device and send that data to the subscriber.

- There are Six different types of events with Customized time-based publishing.
- To create Events, click the plus icon on the top right corner.
- Select any features which are below the Event title.
- We can create more than one Event for a Device.



Before Configuring the IoT Events you must know about the Following Parameters:

- Topic:It is a UTF-8 string format, which is used by the broker to filter messages for each connected client. A topic consists of one or more topic levels and each topic level is separated by a forward slash (topic level separator).
- QoS: It ensures that the events will reach the subscriber. For more information, refer to our Time based publish events video.
- Retain message: Basically retain means keep or hold something. Likewise, in events, if the retain flag is on, then the broker will hold or keep the corresponding details and message of those events. So it will help the newly subscribed clients to get the status update
- Events Type: Format of the events or messages to be published. Simulator support two types of events or message type, they are Text [Normal message] and JSON message.

- Variant: It defines how the events will be sent to the subscriber. It has four types of value. They are :

1. System variables – This value is used to publish the events with the System variables like $Client_ID, $Current_time, etc.
2. Random – This value is used to publish the events in random like when the system is ON|OFF
3. Range – This value is used to publish the events for a certain range of numbers like 1-100
4. Constant – This value is used to publish the events in Alphabet and numbers

- Message: The data which are written here will be published. It will published either in Text or JSON format, based on the message type you've selected.

JSON & Nested JSON

- By using JSON, we can easily read the Published message and enclose multiple data in a single message.
- In the Nested JSON, we can enclose multiple Object data in a single message. Refer the example below.
- To create JSON, click the Plus Symbol icon and just enter the Key & Value and Click "Add", JSON will be created automatically.
- To Create Nested JSON, click the Plus Symbol below the Message and Just enter the Key & Select Value as "Object" and Click "Add".
- An object will be added. Click the Plus Symbol icon below that Object and add the Keys and Values. Take the below example. "Accelerometer" is the Object and below that Keys and Values are nested by {}. You can add multiple objects like this.

JSON syntax

{ " KEY 1" : " VALUE1 " , "KEY2" : " VALUE2 " , " KEY 3" : "VALUE 3" }

Example For Nested JSON

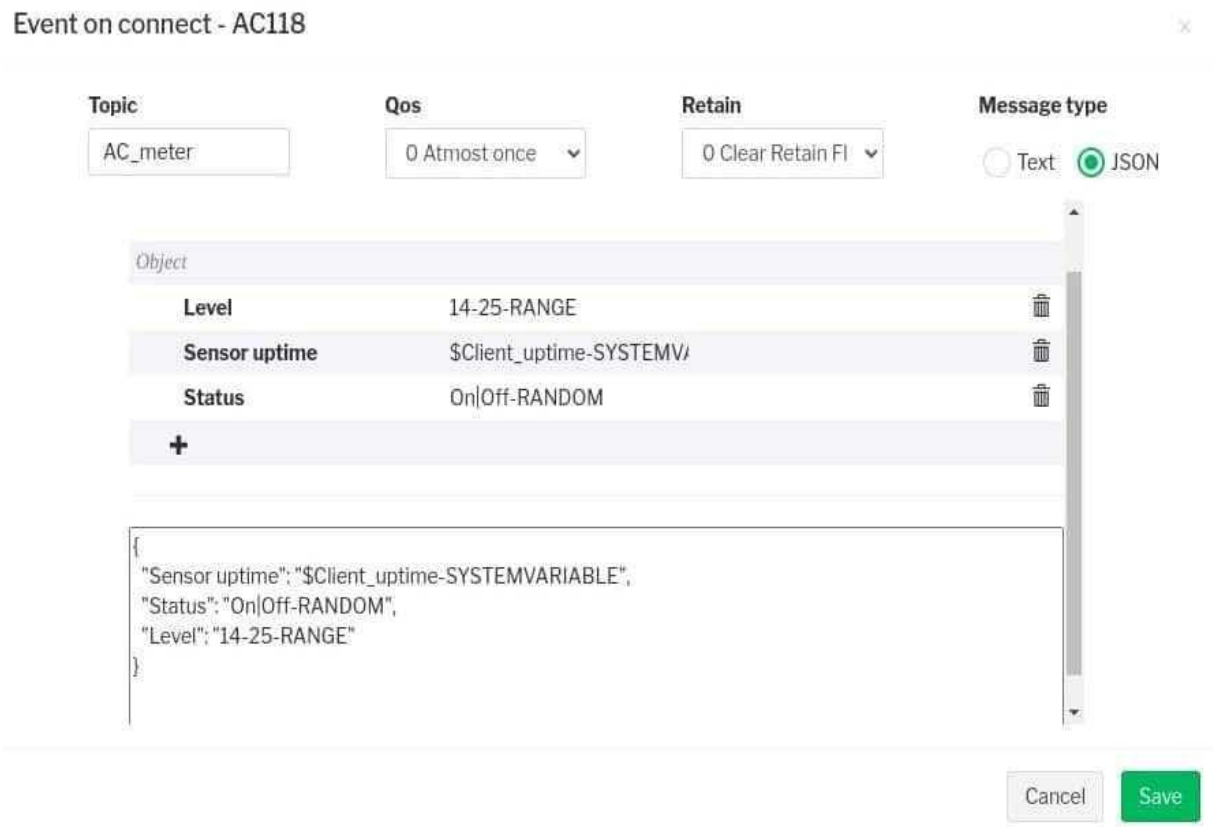
```
{
                                                    “Accelerometer”:
                                                                {
                                “Status”:                “On|Off-RANDOM”,
                                “Vibration_level”:                “50-55-RANGE”,
                        “Sensor_On_Time”:        “$Client_uptime-SYSTEMVARIABLE”
                                                                },
 “Gas_Sensor”
                                                                {
                                                “Status”:”On|Off-RANDOM”,
                                “Methane_level”:                “10-20-RANGE”,
                        “Sensor_On_Time”:        “$Client_uptime-SYSTEMVARIABLE”
                                                                }
}
```

1. On connect Events

- This Event will make the IoT Devices publish the data when it is connected to the Broker.
- If you want to know the status of the device when it is connected to the broker, you can use this Event.
- Select a Device from the Device list on the left side and click the Plus Symbol icon on the top left corner and Select “On Connect”.
- In the following Dialogue box, fill the Topic, QoS, Retain & select Text or JSON for Message type and save it.
- “On Connect” event will be created below the Event tab with the given Topic name, Date and Time. If you want to delete the events, click the Delete button.

Parameters

- Topic: AC_meter
- QoS level: 0 Atmost once
- Retain: 0 Set Retain Flag
- Events Type:JSON Format



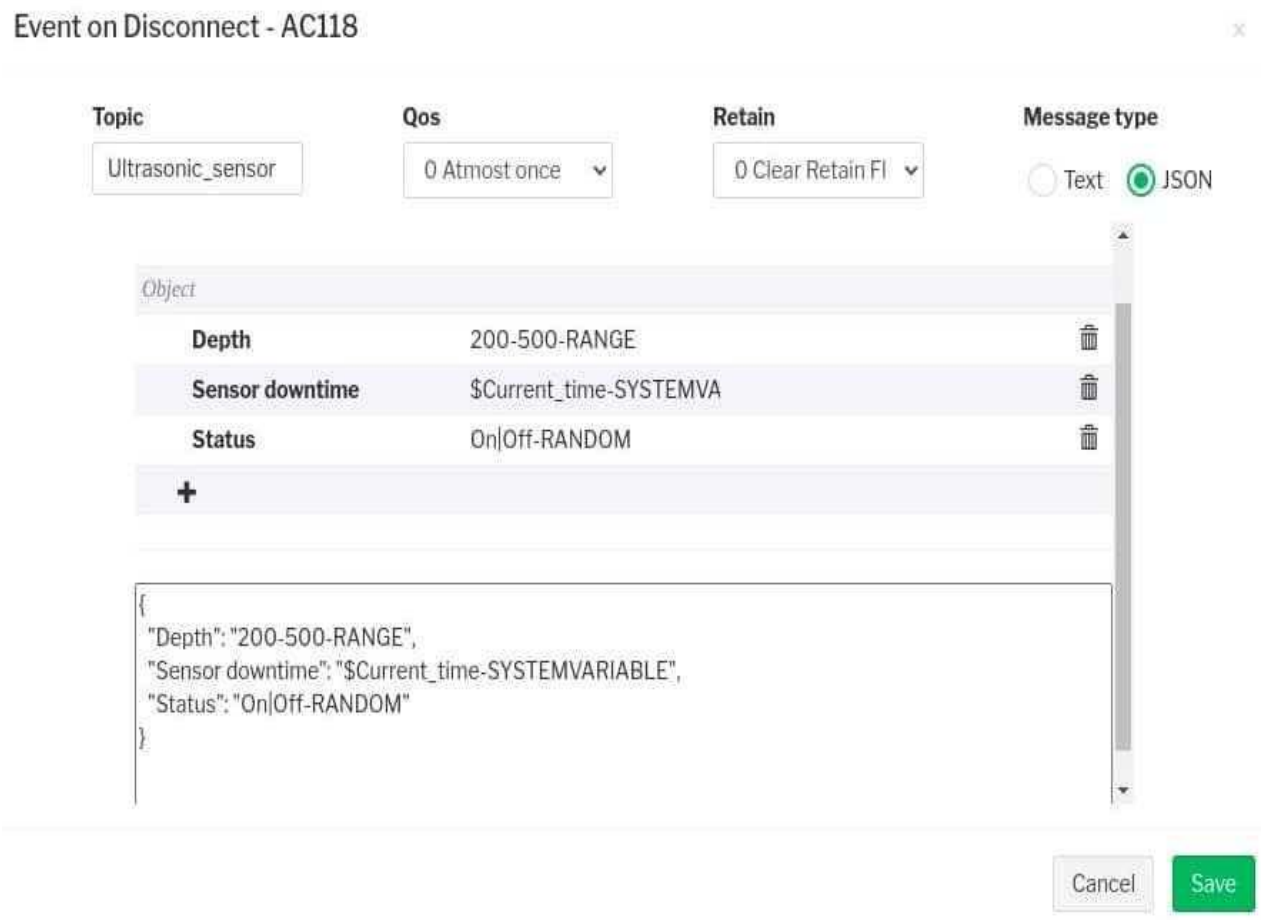
2. On Disconnect

- These Events will make the IoT Device to publish the data before disconnecting from the Broker
- If you want know the status of the Device when it is disconnected from the broker, you can use this Event
- Select a device from the Device list on the left side and click the add icon on the top right corner and select “On Disconnect”

- In the following Dialogue box, fill the Topic, QoS, Retain & select Text or JSON for Message type and save it.
- “On Disconnect” event will get created below the Event tab with the given Topic name, Data and Time. If you want to delete the events, click the Delete button.

Parameters

- Topic: Ultrasonic_sensor
- QoS level:0 Atmost once
- Retain: 0 Set Retain Flag
- Events Type: JSON Format



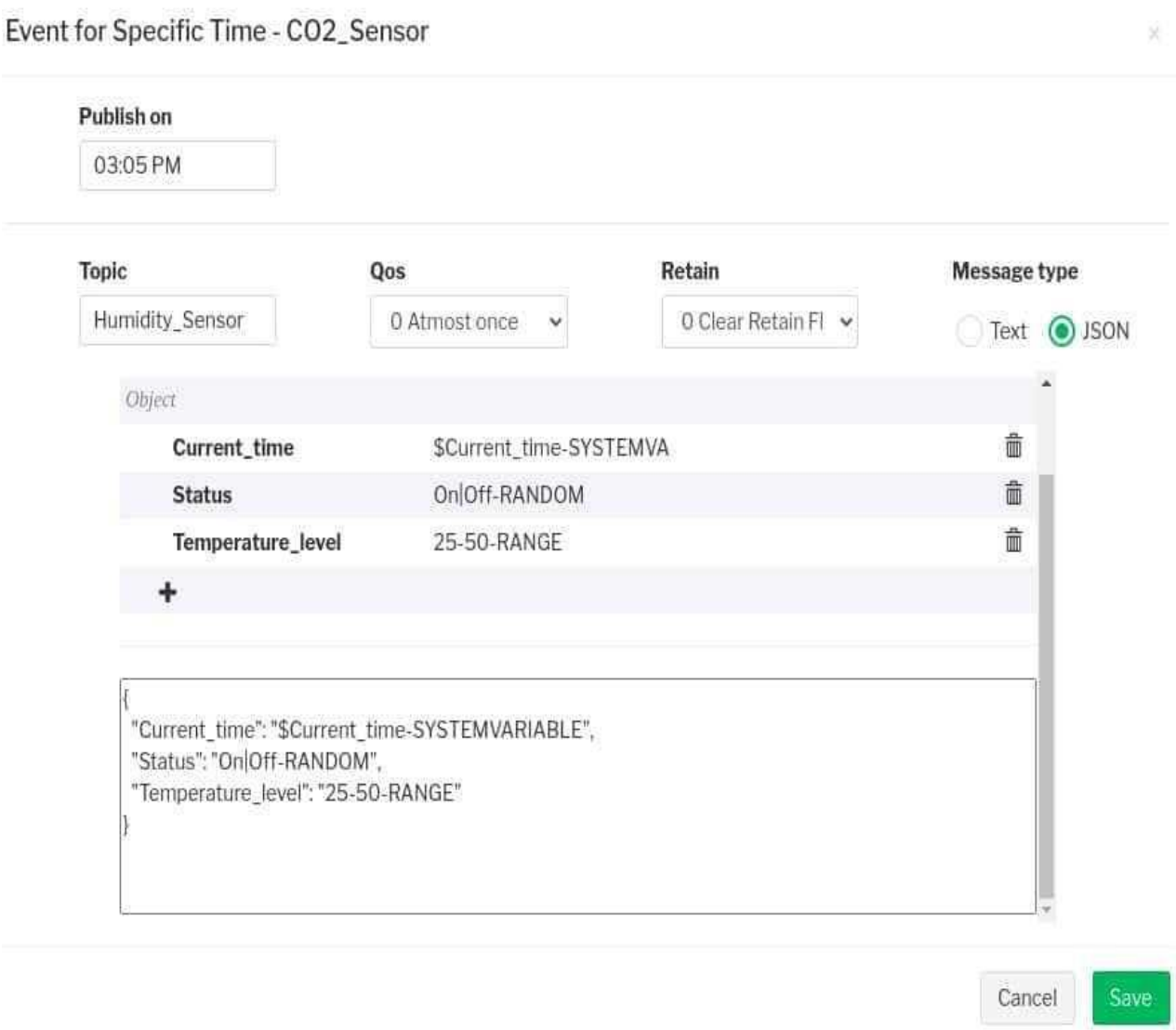
3. Specific Time

- This feature will make the IoT Device to publish the data for a specific time.

- You can set any time on AM & PM to publish the data in the “Publish On” text box.
- Select a device from the Device list on the left side and click the add icon on the top left corner and select “Specific Time”
- In the following Dialogue box, fill the Topic, QoS, Retain & select Text or JSON for Message type and save it.
- “Specific Time” event will be created below the Event tab with the given Topic name, Data and Time. If you want to delete the events, just click the Delete button.

Parameters

- Publish on:03:05 PM
- Topic: Humidity_Sensor
- QoS level: 0 At most once
- Retain: 0 Set Retain Flag
- Events Type: JSON Format



Conclussion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

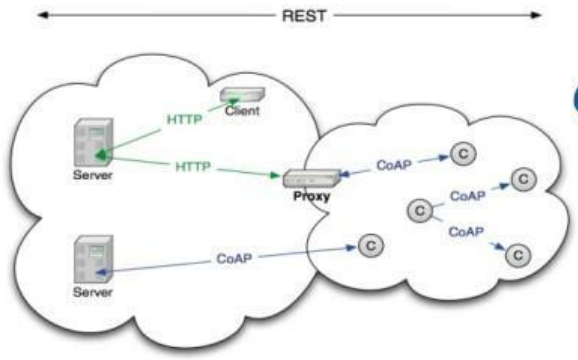
Department: CSE IOT And Cyber Security Including Blockchain
Subject: IoT Automation Lab
Semester: VIII
Name of Student:

Experiment No:

Aim: To study and simulate CoAP protocol in Contiki OS.

Constrained Application Protocol (CoAP) is a lightweight HTTP protocol that reads and controls the sensors deployed for IoT.It has actions like get, post, put, delete, observe, discover.So here is an example in contiki that shows how to use this CoAP based application using a border router.

How to Run CoAP Applications in Contiki Cooja Simulator?



How to Run CoAP Applications in Contiki Cooja Simulator?

S-Logix

Technology for the Next Generation

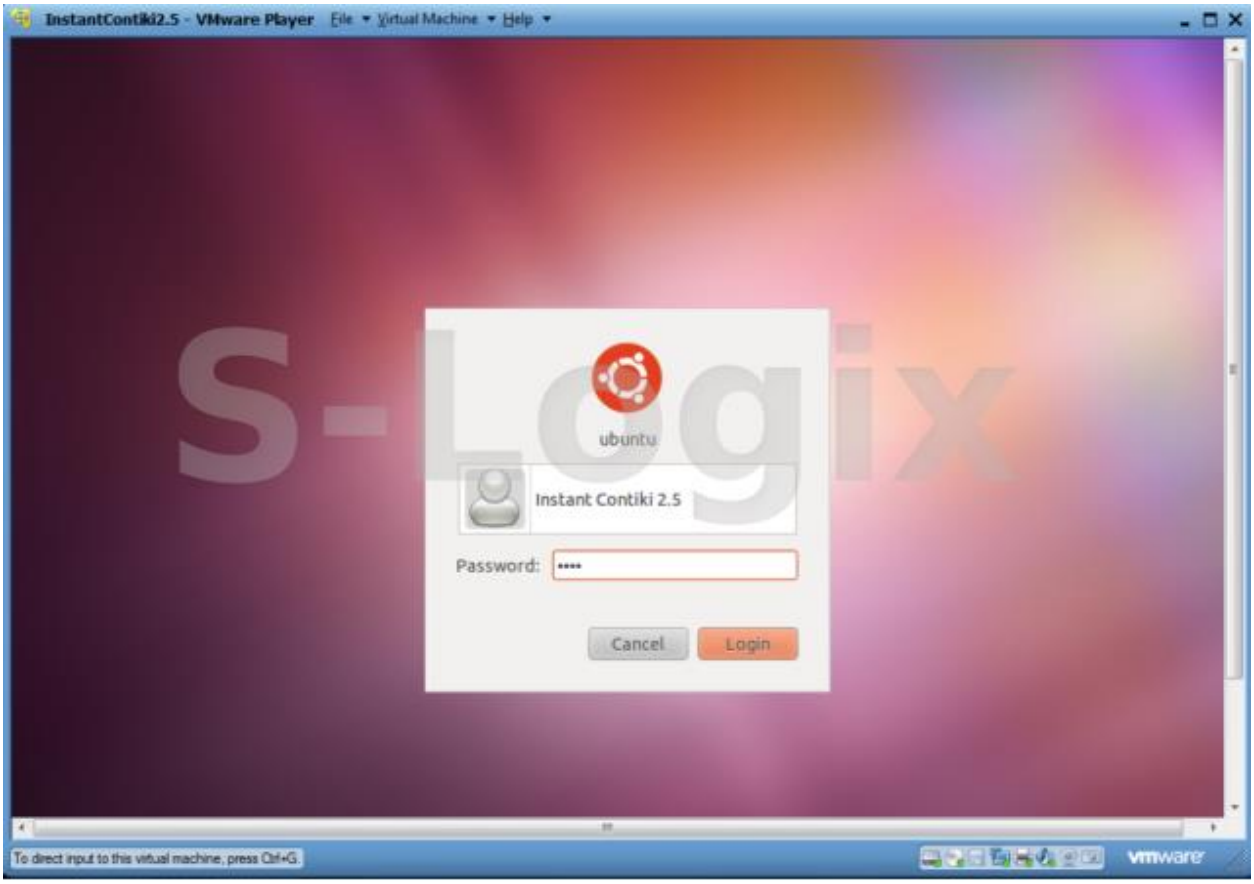
Execute CoAP Applications in Contiki Cooja Simulator

Step-1

Open the Contiki OS with Vmware workstation.

And login into contiki user

password: user.

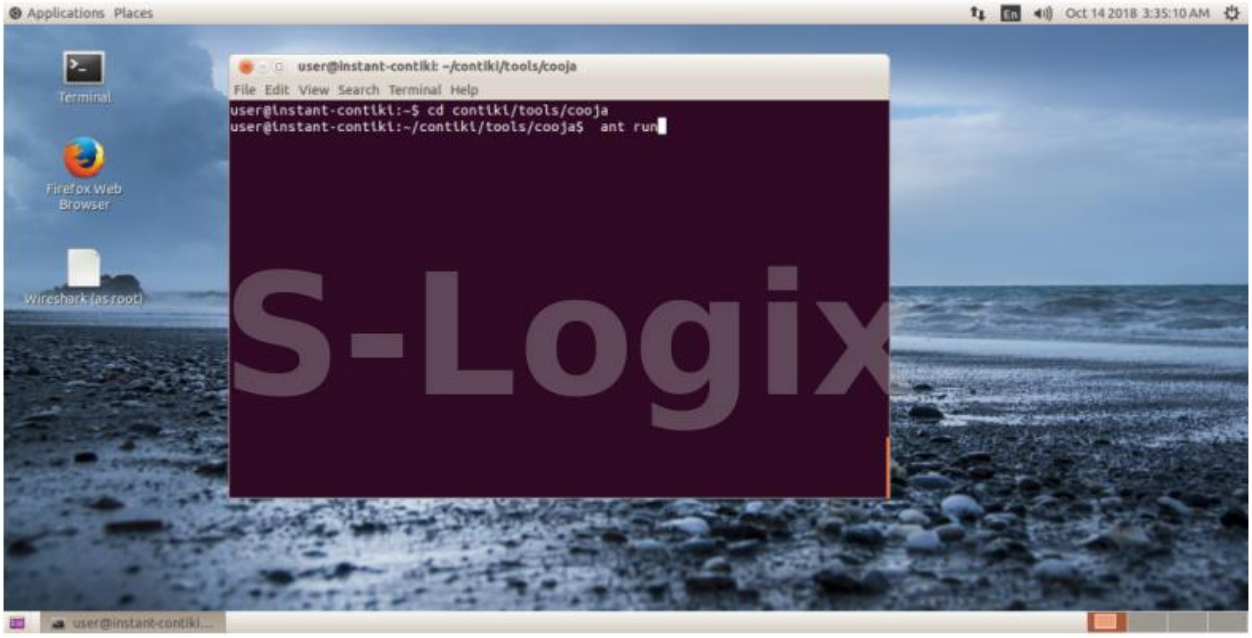


Step-2

Now open the terminal in contiki desktop and make the right directories to run the cooja simulator tools.

Cmd: cd contiki/ tools/ cooja --> press enter

ant run --> enter



After successful execution of above command. make file will build automatically and then Contiki Cooja Network simulator application tool will appear. It's a blue color terminal.

Step-3

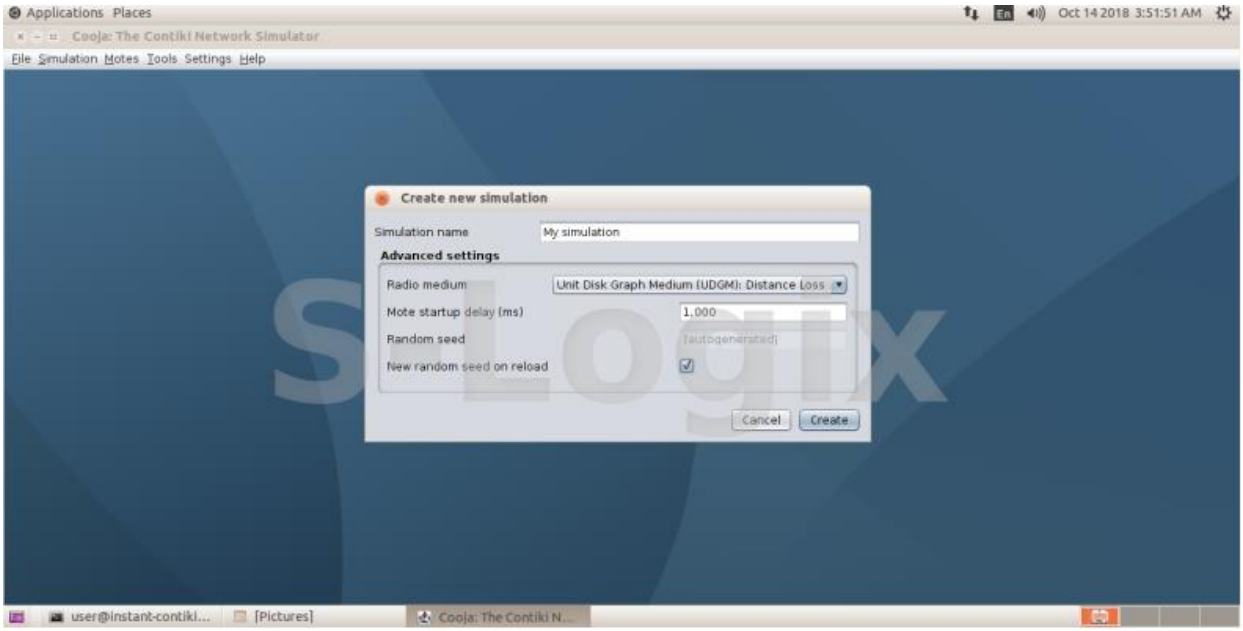
Open file menu on cooja and create new simulation with random seeding

It will auto-generate the seeds for every time reload the simulation.

File -new simulation (or) (ctrl+ n).

click -New random seed on reload option.

click -Create .



Step-4

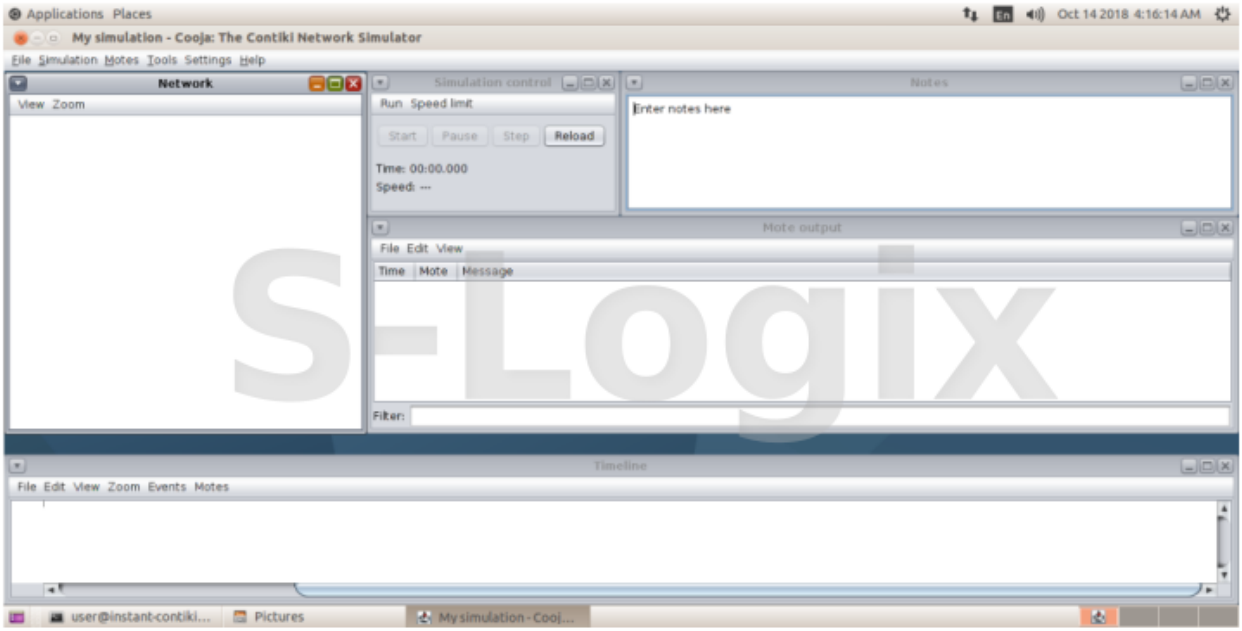
After click the create option number of tabs will appear in terminal it is basic necessary to simulate applications.Open motes menu >> add motes >> create new motes type >> sky

Three files are necessary to run CoAP applications. In order to create the motes,

i. border- router.

ii. er-example-server.c

iii. er-example-client.c

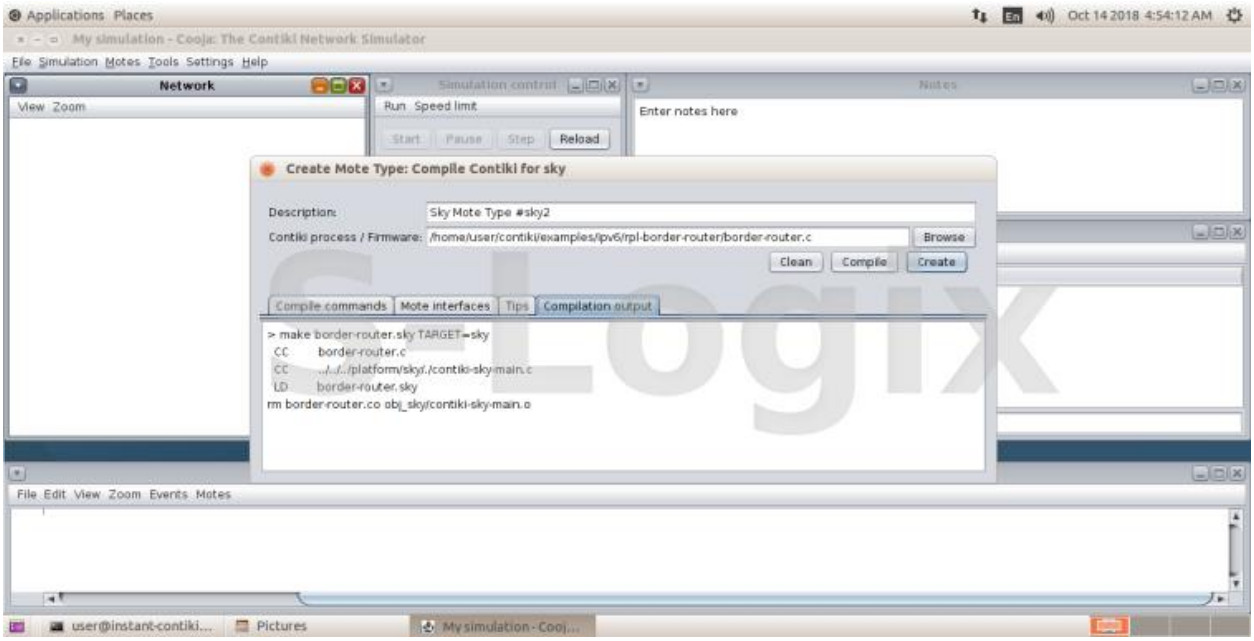


Step-5

To create border router motes

home/user/contiki/examples/ipv6/rpl-border-router/border-router.c

Choose the file in location >> compile >> create >> Add motes.



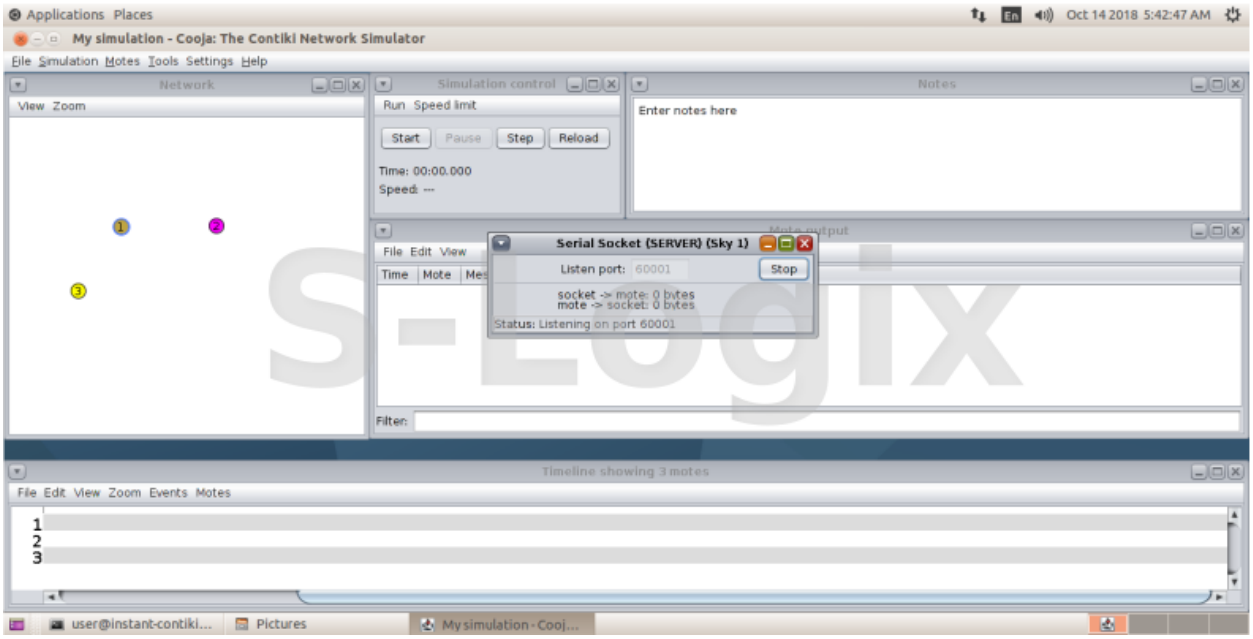
Step-6

To create server motes.

/home/user/contiki/examples/er-rest-example/er-example-server.c

Choose a file location >> compile>> create >> choose sever count(optional)>> Add motes.

Here,



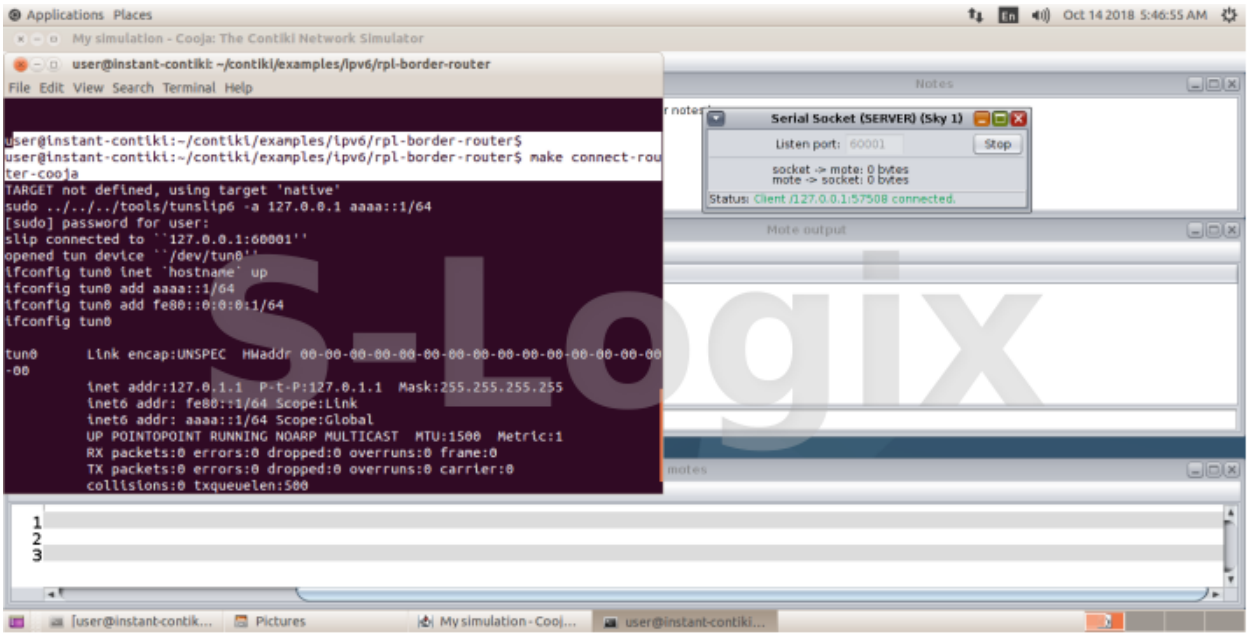
Step-7

To create client motes .

/home/user/contiki/examples/er-rest-example/er-example-client.c

Choose a file in location >> compile >> create >> choose client

count(optional) >> Add motes.



Step-8

Now connect the serial socket server in border router.

In cooja, network terminal .. View menu -choose the options you want,

For example: mote Type, mote ID's, radio traffic, ...

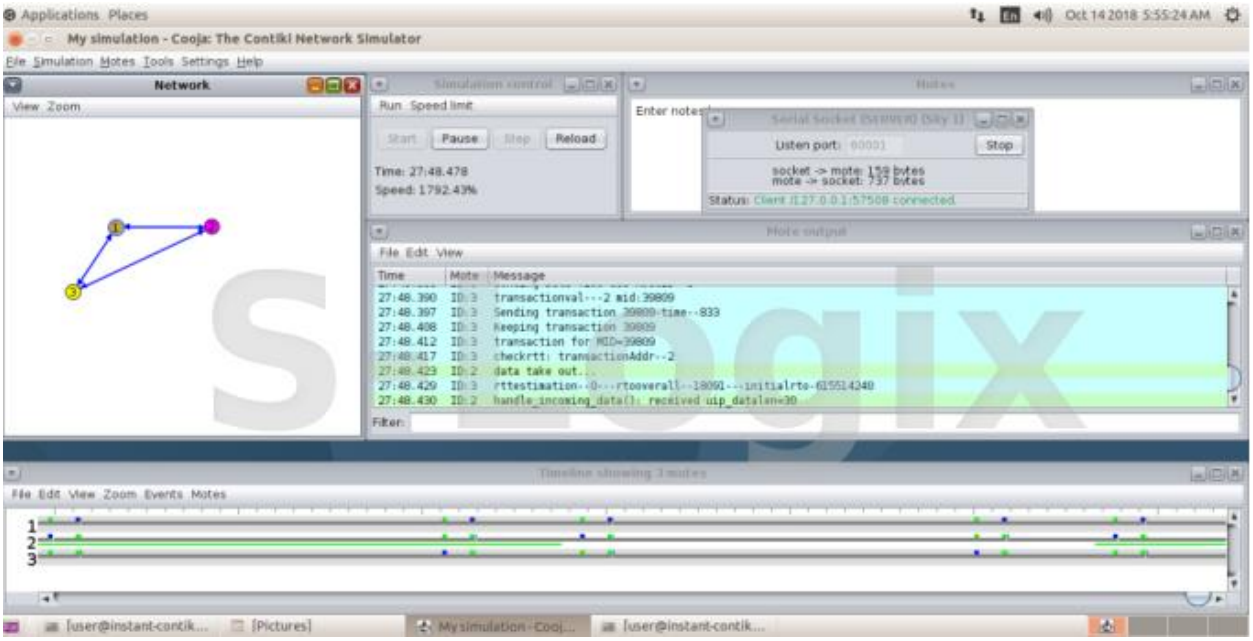
Left click on border router motes >> choose: mote tools for sky1 >> serial socket(SERVER).

New tab will appear click : start -> now the default port will listen.

open one more command terminal and set directories to this location:

Cd contiki/ examples/ ipv6/ rpl-border-router/

Make connect-router-cooja the server will be connected to router.



Conclusion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No. 3

Title: IOT NodeRed simulator

Aim: Real time data acquisition and transmission using NodeRed simulator.

Theory:

NodeRed:

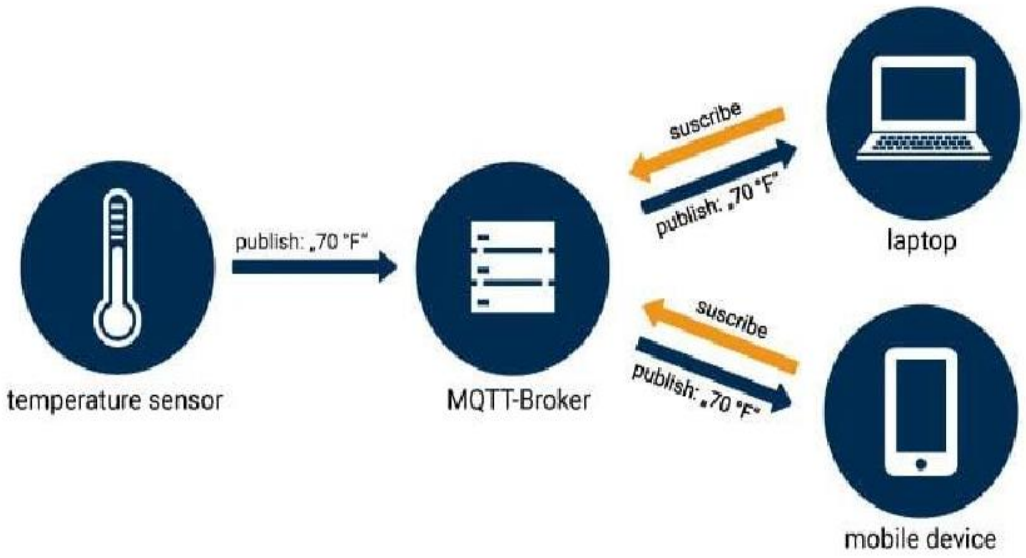
Node-RED is a flow-based, low-code development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things.

Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON. Since version 0.14, MQTT nodes can make properly configured TLS connections.

MQTT stands for Message Queuing Telemetry Transport.

MQTT stands for Message Queuing Telemetry Transport. It is an extremely simple and lightweight messaging protocol (subscribe and publish) designed for limited devices and networks with high latency, low bandwidth or unreliable networks. Its design principles are designed to reduce the network bandwidth and resource requirements of devices and ensure

security of supply. In addition, these principles are advantageous for M2M (machine-to-machine) or IoT devices because battery performance and bandwidth are very important.



The MQTT broker is the center of every Publish / Subscribe protocol. Depending on the implementation, a broker can manage up to thousands of simultaneously connected MQTT clients. The broker is responsible for receiving all messages, filtering the messages, determining who subscribed to each message and sending the message to those subscribed clients. The Broker also holds the sessions of all persistent clients, including subscriptions and missed messages. Another task of the Broker is the authentication and authorization of clients. Usually the broker is extensible, which facilitates custom authentication,
authorization and integration with backend systems. Integration is especially important, because the Broker is often the component directly exposed on the Internet, serves many clients and has to forward messages to downstream analysis and processing systems.

Features of MQTT

There are various features of MQTT which are as follows:

- It is a machine-to-machine protocol, i.e., it provides communication among the devices.
- It is produced as a simple and lightweight messaging protocol that supports a publish/subscribe system to transfer the data between the user and the server.
- It does not require that both the user and the server create a connection at an equal time.
- It provides quicker data transmission, like how Whatsapp/messenger supports a quicker delivery.
- It enables the users to subscribe to the definite selection of topics to receive the data they are viewing for.
- It can distribute data more effectively.

- It can increase scalability.
- It is used to lower network bandwidth consumption dramatically

Advantages Of MQTT Protocol

Following are the benefits or advantages of MQTT protocol:

The MQTT protocol payload can carry any type of data such as binary, ascii text etc. The receiver need to interpret and decode as per format used by the transmitter. Hence MQTT is packet agnostic.

- It uses packet of low size and hence can be used for low bandwidth applications.
- It offers lower battery power consumption.
- It is reliable protocol as it uses QoS options to provide guaranteed delivery.
- Due to its publish/subscribe model, It is scalable.
- It offers de-coupled design as it is easy to decouple the device and server.

Here's a general guide on how to simulate data or interact with simulated devices using Node- RED:

1. Install Node-RED:

- Ensure Node-RED is installed on your system. You can install it globally using npm:

  bashCopy code
  npm install -g node-red

2. Start Node-RED:

- Start Node-RED by running the following command:

  bashCopy
  code node-red

- Access the Node-RED editor by opening your web browser and navigating to **http://localhost:1880**.

3. Install Necessary Nodes:

- Depending on your simulation requirements, you might need to install additional nodes. For example, for MQTT communication or other IoT protocols.

4. Create a Flow:

- Design a flow in Node-RED by dragging and dropping nodes onto the workspace.

5. Simulate Data:

- Use inject nodes or function nodes to simulate data. Inject nodes allow you to inject messages at specific intervals.

6. Connect to External Simulators:

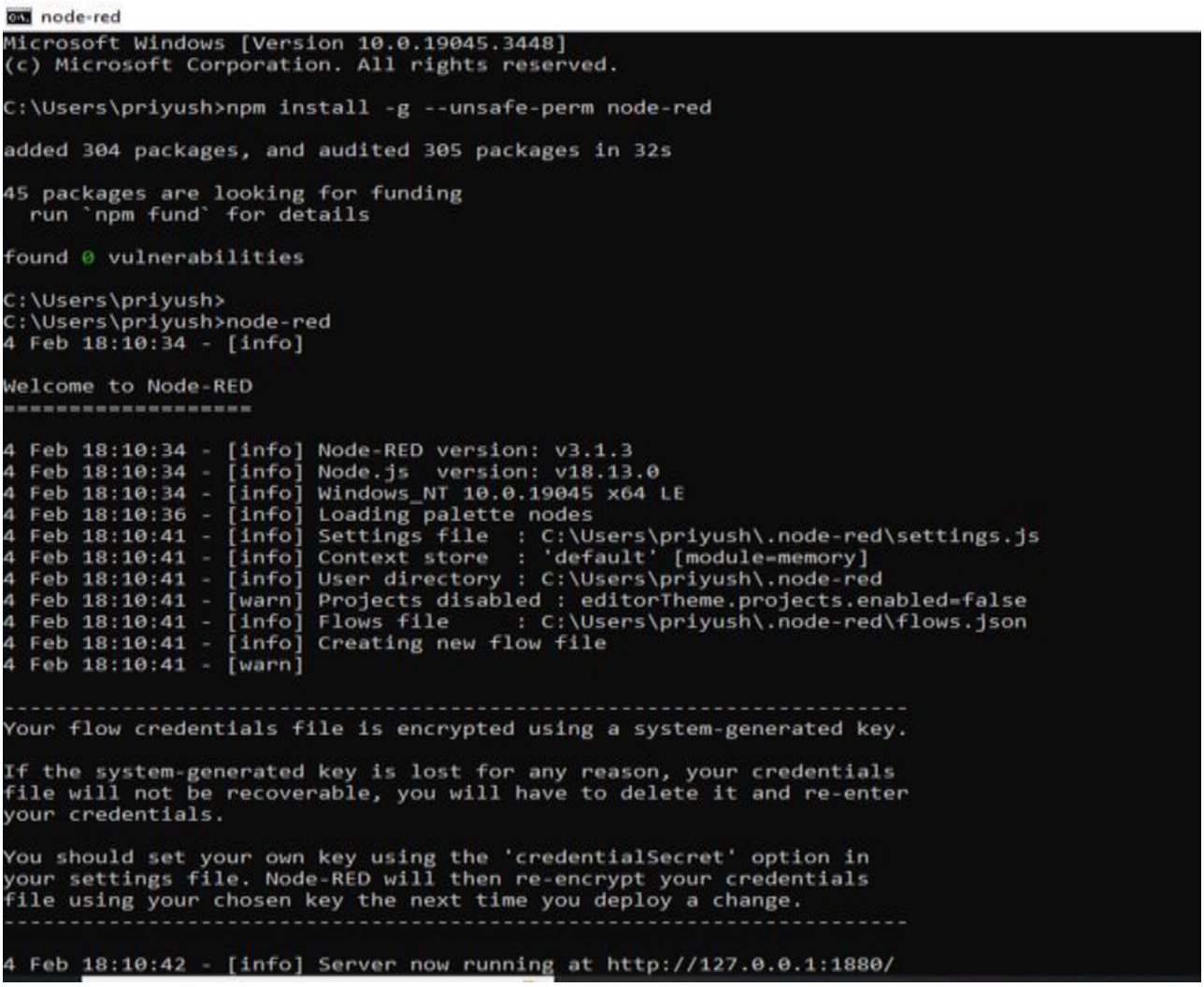
- If you're working with external simulators, connect Node-RED to them using appropriate nodes. For instance, MQTT nodes can connect to an MQTT broker that is part of your simulation.

7. Visualize Data (Optional):

- Use Dashboard nodes to create visualizations and dashboards for the simulated data.

8. Deploy and Monitor:

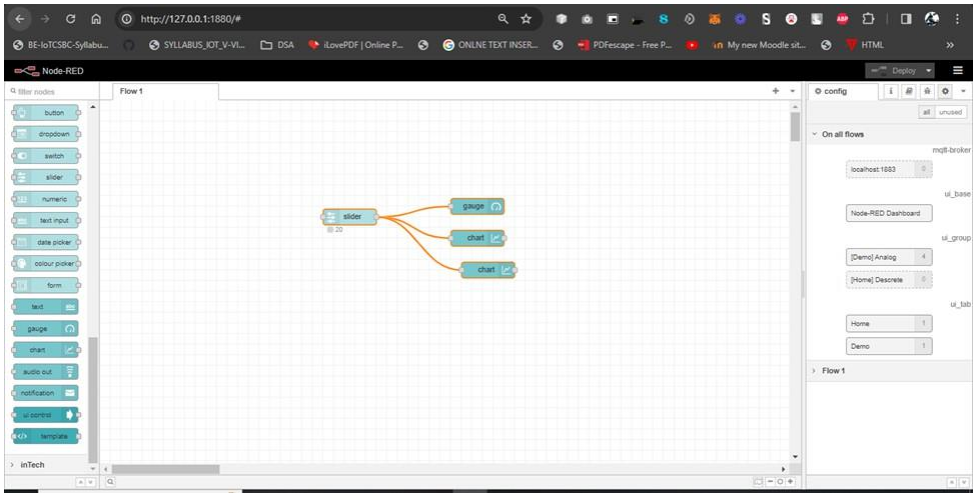
- Deploy your flow, and monitor the Node-RED debug console for any messages or errors.



<u>Step Real time data acquisition and transmission using NodeRed simulator.</u>

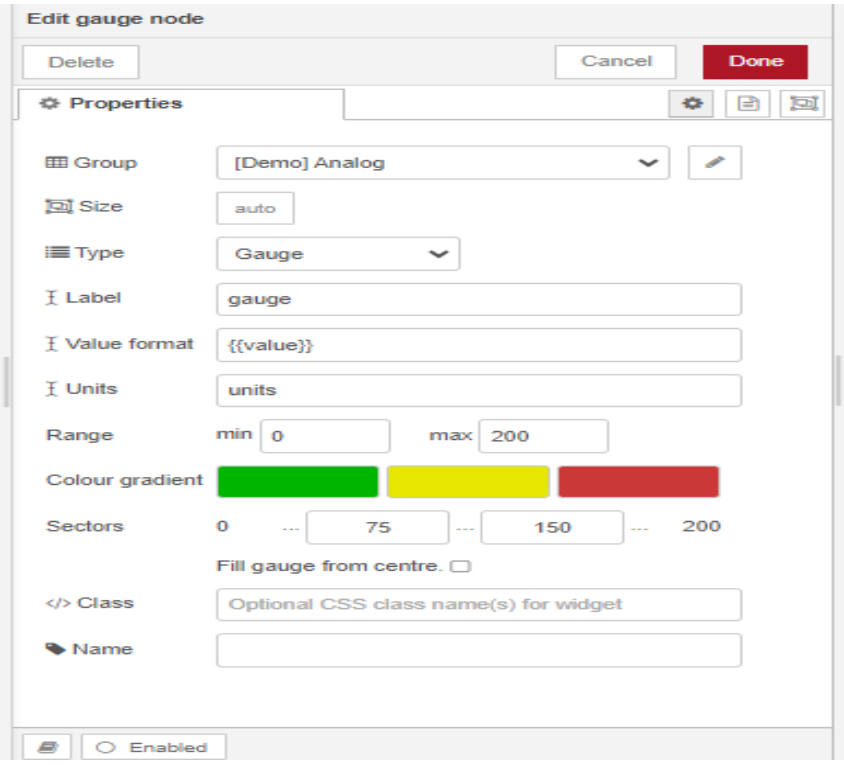
1. **Install Necessary Nodes:**
   - In Node-RED, install nodes that are necessary for your project, such as MQTT nodes for communication.
2. **Create Data Source:**
   - Simulate or connect a data source. This could be a simulated sensor generating data or an actual IoT device.
3. **Node-RED Flow Design:**
   - Design a flow in Node-RED that includes nodes for data acquisition, processing, and transmission.

4. Data Acquisition:



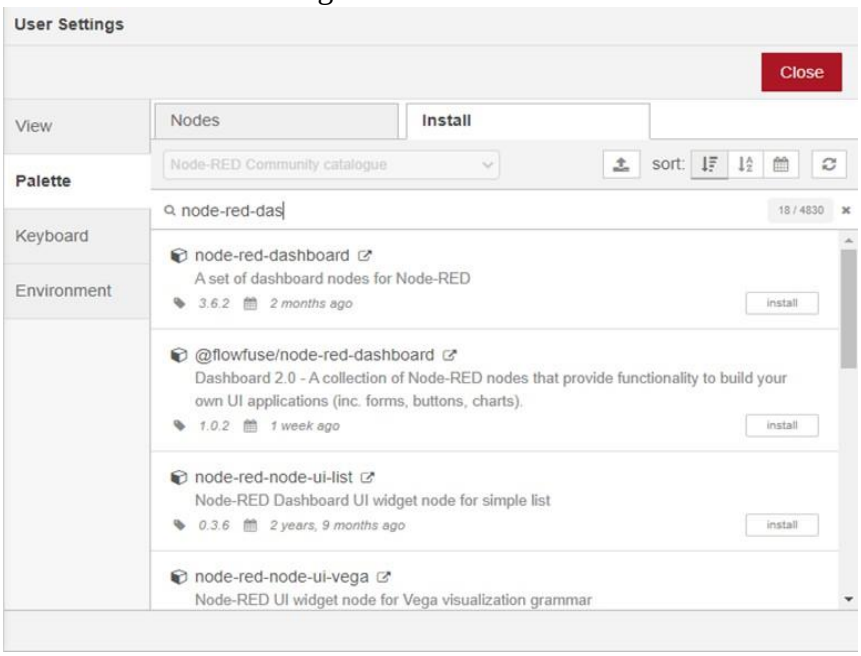
5.

- Use appropriate nodes (e.g., inject nodes, function nodes) to acquire real-time data from the source.



6. **Data Processing:**
   - Implement any required data processing within Node-RED. This could include filtering, formatting, or aggregating data.
7. **Real-time Transmission:**
   - Use nodes like MQTT or WebSocket nodes to transmit real-time data to a destination.

8. **Integration with External Services:**
   - If necessary, integrate Node-RED with external services or databases to store or further process the data.
9. **Dashboard (Optional):**
   - Create a real-time dashboard using Node-RED Dashboard nodes to visualize the acquired



   data.

10. **Testing:**

- Test the entire flow by deploying it in Node-RED and observing the real-time data acquisition and transmission.

Conclusion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

Aim:

A Raspberry Pi for use with AWS IoT. If you have another device that you'd like to connect, the instructions for the Raspberry Pi include references that can help you adapt these instructions to your device.

This normally takes about 20 minutes, but it can take longer if you have many system software upgrades to install.

In this tutorial, you'll:

- Set up your device
- Install the required tools and libraries for the AWS IoT Device SDK
- Install AWS IoT Device SDK
- Install and run the sample app
- View messages from the sample app in the AWS IoT console

Important

Adapting these instructions to other devices and operating systems can be challenging. You'll need to understand your device well enough to be able to interpret these instructions and apply them to your device.

If you encounter difficulties while configuring your device for AWS IoT, you might try one of the other device options as an alternative, such as Create a virtual device with Amazon EC2 or Use your Windows or Linux PC or Mac as an AWS IoT device.

Set up your device

The goal of this step is to collect what you'll need to configure your device so that it can start the operating system (OS), connect to the internet, and allow you to interact with it at a command line interface.

To complete this tutorial, you need the following:

- An AWS account. If you don't have one, complete the steps described in Set up your AWS account before you continue.
- A Raspberry Pi 3 Model B or more recent model. This might work on earlier versions of the Raspberry Pi, but they have not been tested.
- Raspberry Pi OS (32-bit) or later. We recommend using the latest version of the Raspberry Pi OS. Earlier versions of the OS might work, but they have not been tested.

  To run this example, you do not need to install the desktop with the graphical user interface (GUI); however, if you're new to the Raspberry Pi and your Raspberry Pi hardware supports it, using the desktop with the GUI might be easier.
- An Ethernet or WiFi connection.
- Keyboard, mouse, monitor, cables, power supplies, and other hardware required by your device.

Important

Before you continue to the next step, your device must have its operating system installed, configured, and running. The device must be connected to the internet and you will need to be able to access the device by using its command line interface. Command line access can be through a directly-connected keyboard, mouse, and monitor, or by using an SSH terminal remote interface.

If you are running an operating system on your Raspberry Pi that has a graphical user interface (GUI), open a terminal window on the device and perform the following instructions in that window. Otherwise, if you are connecting to your device by using a remote terminal, such as PuTTY, open a remote terminal to your device and use that.

Install the required tools and libraries for the AWS IoT Device SDK

Before you install the AWS IoT Device SDK and sample code, make sure your system is up to date and has the required tools and libraries to install the SDKs.

1. **Update the operating system and install required libraries**

   Before you install an AWS IoT Device SDK, run these commands in a terminal window on your device to update the operating system and install the required libraries.

   ```
   sudo apt-get update

   sudo apt-get upgrade

   sudo apt-get install cmake

   sudo apt-get install libssl-dev
   ```

2. **Install Git**

   If your device's operating system doesn't come with Git installed, you must install it to install the AWS IoT Device SDK for JavaScript.

   a. Test to see if Git is already installed by running this command.

      ```
      git --version
      ```

   b. If the previous command returns the Git version, Git is already installed and you can skip to Step 3.

c. If an error is displayed when you run the **git** command, install Git by running this command.

```
sudo apt-get install git
```

d. Test again to see if Git is installed by running this command.

```
git --version
```

e. If Git is installed, continue to the next section. If not, troubleshoot and correct the error before continuing. You need Git to install the AWS IoT Device SDK for JavaScript.

Install AWS IoT Device SDK

Install the AWS IoT Device SDK.

- Python
- JavaScript

In this section, you'll install Python, its development tools, and the AWS IoT Device SDK for Python on your device. These instructions are for a Raspberry Pi running the latest Raspberry Pi OS. If you have another device or are using another operating system, you might need to adapt these instructions for your device.

1. **Install Python and its development tools**

   The AWS IoT Device SDK for Python requires Python v3.5 or later to be installed on your Raspberry Pi.

   In a terminal window to your device, run these commands.

   a. Run this command to determine the version of Python installed on your device.

   ```
   python3 --version
   ```

   If Python is installed, it will display its version.

b. If the version displayed is Python 3.5 or greater, you can skip to Step 2.
c. If the version displayed is less than Python 3.5, you can install the correct version by running this command.

```
sudo apt install python3
```

d. Run this command to confirm that the correct version of Python is now installed.

```
python3 --version
```

2. Test for pip3

In a terminal window to your device, run these commands.

a. Run this command to see if **pip3** is installed.

```
pip3 --version
```

b. If the command returns a version number, **pip3** is installed and you can skip to Step 3.
c. If the previous command returns an error, run this command to install **pip3**.

```
sudo apt install python3-pip
```

d. Run this command to see if **pip3** is installed.

```
pip3 --version
```

3. Install the current AWS IoT Device SDK for Python

Install the AWS IoT Device SDK for Python and download the sample apps to your device.

On your device, run these commands.

```
cd ~

python3 -m pip install awsiotsdk

git clone https://github.com/aws/aws-iot-device-sdk-python-v2.git
```

Install and run the sample app

In this section, you'll install and run the pubsub sample app found in the AWS IoT Device SDK. This app shows how your device uses the MQTT library to publish and subscribe to MQTT messages. The sample app subscribes to a topic, topic_1, publishes 10 messages to that topic, and displays the messages as they're received from the message broker.

Install the certificate files

The sample app requires the certificate files that authenticate the device to be installed on the device.

To install the device certificate files for the sample app

1. Create a certs subdirectory in your *home* directory by running these commands.

2. cd ~

   ```
   mkdir certs
   ```

3. Into the ~/certs directory, copy the private key, device certificate, and root CA certificate that you created earlier in Create AWS IoT resources.

   How you copy the certificate files to your device depends on the device and operating system and isn't described here. However, if your device supports a graphical user interface (GUI) and has a web browser, you can perform the procedure described in Create AWS IoT resources from your device's web browser to download the resulting files directly to your device.

   The commands in the next section assume that your key and certificate files are stored on the device as shown in this table.

Certificate file names	
File	File path
Root CA certificate	~/certs/Amazon-root-CA-1.pem
Device certificate	~/certs/device.pem.crt
Private key	~/certs/private.pem.key

To run the sample app, you need the following information:

Application parameter values	
Parameter	Where to find the value
your-iot-endpoint	In the AWS IoT console, choose **All devices**, and then choose **Things**. On the **Settings** page in the AWS IoT menu. Your endpoint is displayed in the **Device data endpoint** section.

The *your-iot-endpoint* value has a format of: *endpoint_id*-ats.iot.*region*.amazonaws.com, for example, a3qj468EXAMPLE-ats.iot.us-west-2.amazonaws.com.

- Python
- JavaScript

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

To install and run the sample app

1. Navigate to the sample app directory.

   ```
   cd ~/aws-iot-device-sdk-python-v2/samples
   ```

2. In the command line window, replace *your-iot-endpoint* as indicated and run this command.

   ```
   python3 pubsub.py --topic topic_1 --ca_file ~/certs/Amazon-root-CA-1.pem --cert ~/certs/device.pem.crt --key ~/certs/private.pem.key --endpoint your-iot-endpoint
   ```

3. Observe that the sample app:
   a. Connects to the AWS IoT service for your account.
   b. Subscribes to the message topic, **topic_1**, and displays the messages it receives on that topic.
   c. Publishes 10 messages to the topic, **topic_1**.
   d. Displays output similar to the following:

4. Connecting to a3qEXAMPLEffp-ats.iot.us-west-2.amazonaws.com with client ID 'test-0c8ae2ff-cc87-49d2-a82a-ae7ba1d0ca5a'...
5. Connected!
6. Subscribing to topic 'topic_1'...
7. Subscribed with QoS.AT_LEAST_ONCE
8. Sending 10 message(s)
9. Publishing message to topic 'topic_1': Hello World! [1]
10. Received message from topic 'topic_1': b'Hello World! [1]'
11. Publishing message to topic 'topic_1': Hello World! [2]
12. Received message from topic 'topic_1': b'Hello World! [2]'
13. Publishing message to topic 'topic_1': Hello World! [3]

14. Received message from topic 'topic_1': b'Hello World! [3]'
15. Publishing message to topic 'topic_1': Hello World! [4]
16. Received message from topic 'topic_1': b'Hello World! [4]'
17. Publishing message to topic 'topic_1': Hello World! [5]
18. Received message from topic 'topic_1': b'Hello World! [5]'
19. Publishing message to topic 'topic_1': Hello World! [6]
20. Received message from topic 'topic_1': b'Hello World! [6]'
21. Publishing message to topic 'topic_1': Hello World! [7]
22. Received message from topic 'topic_1': b'Hello World! [7]'
23. Publishing message to topic 'topic_1': Hello World! [8]
24. Received message from topic 'topic_1': b'Hello World! [8]'
25. Publishing message to topic 'topic_1': Hello World! [9]
26. Received message from topic 'topic_1': b'Hello World! [9]'
27. Publishing message to topic 'topic_1': Hello World! [10]
28. Received message from topic 'topic_1': b'Hello World! [10]'
29. 10 message(s) received.
30. Disconnecting...

Disconnected!

If you're having trouble running the sample app, review Troubleshooting problems with the sample app.

You can also add the --verbosity Debug parameter to the command line so the sample app displays detailed messages about what it's doing. That information might provide you the help you need to correct the problem.

View messages from the sample app in the AWS IoT console

You can see the sample app's messages as they pass through the message broker by using the **MQTT test client** in the **AWS IoT console**.

To view the MQTT messages published by the sample app

1. Review View MQTT messages with the AWS IoT MQTT client. This helps you learn how to use the **MQTT test client** in the **AWS IoT console** to view MQTT messages as they pass through the message broker.
2. Open the **MQTT test client** in the **AWS IoT console**.
3. Subscribe to the topic, **topic_1**.
4. In your command line window, run the sample app again and watch the messages in the **MQTT client** in the **AWS IoT console**.
    - Python
    - JavaScript

```
5.cd ~/aws-iot-device-sdk-python-v2/samples

python3 pubsub.py --topic topic_1 --ca_file ~/certs/Am
```

Conclusion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

Aim:

Ladder Logic is one of the top 5 most popular types of PLC programming languages used in manufacturing environments. Before Programmable Logic Controllers, manufacturing plants employed relay-based circuitry to energize different loads based on how the relays were wired together. The circuit patterns used for these drawings are known as ladder diagrams. Relays were costly, required constant maintenance, and could not be easily reconfigured. As PLCs took over this process, it was essential to keep a similarity of the old system; thus, **ladder logic was created as the first PLC programming language**.

Ladder Logic is labeled as such because the software is laid out in the shape of a ladder. On the left side, ladder logic instructions are set as conditions, while the ones on the right side are instructions that are triggered if the conditions are met. Each rung of the ladder spans from left to right and is executed from top to bottom by the PLC.

As mentioned above, ladder logic is extremely popular among PLC programmers. It's *easy to learn, mimics electrical circuits, and is easy to troubleshoot once deployed*.

Learning ladder logic is typically the entry point into a career in control systems as a PLC programmer. In this post, we will go over ladder logic components, cover basic principles, and outline what it takes to master this programming language.

Ladder Logic Basics

Just like computers, PLCs operate with binary signals; each one can be set to zero or one. In the programming world, this data type is called a boolean. **A boolean takes a single bit in the memory, can be set to 0 or 1, and is used in most basic PLC instructions**.

The PLC executes the program loaded into it one rung at a time. As the PLC begins to process the rung, it reads the instructions on the left and determines if the Logic on that side of the rung is set to TRUE. The Logic evaluates to TRUE when a hypothetical current is able to pass through the instructions. Each instruction has a set of conditions that make it TRUE or FALSE.

For the purpose of this tutorial, we'll start with two of the most basic instructions in ladder logic plc programming: Examine if Closed and Output Energize.

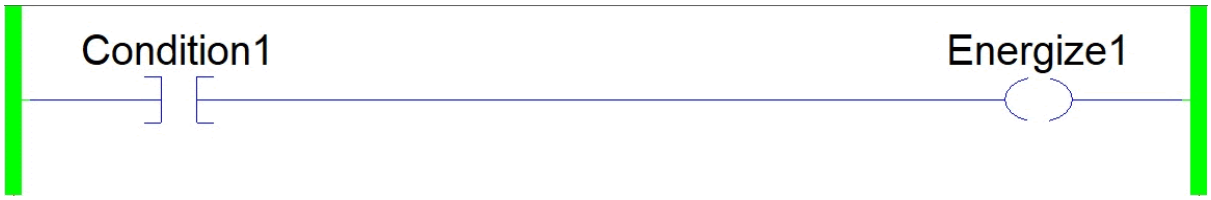
Examine If Closed [XIC] - This input instruction will look at the specified boolean bit and evaluate the condition to TRUE when the bit is set to 1 (or HIGH). While the bit is set to 0 (or LOW), the instruction will evaluate to FALSE.

Output Energize [OTE] - This output instruction will set the specified bit to 1 (or HIGH) if the input instruction conditions are TRUE. If they're FALSE, the Output Energize instruction will set the bit to 0 (or LOW).

Basic Ladder Logic Rung Analysis

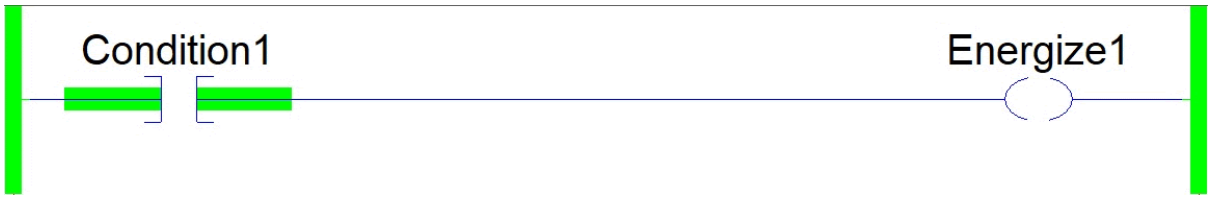
- **Step 1** - The hypothetical current starts moving from left to right.
- **Step 2** - When the hypothetical current encounters and XIC Instruction, it checks if the condition is TRUE or FALSE. If the XIC is False, the PLC aborts this rung.

- **Step 3** - The hypothetical current goes to the next instruction. Repeats Step 2 until the rung is completed.
- **Step 4** - The PLC moves to the rung below.



Ladder Logic PLC Programming XIC = OFF Example

In the example above, the XIC Instruction is tied to the bit “Condition1”. Since the bit is OFF (or 0), the hypothetical current stops at the instruction.

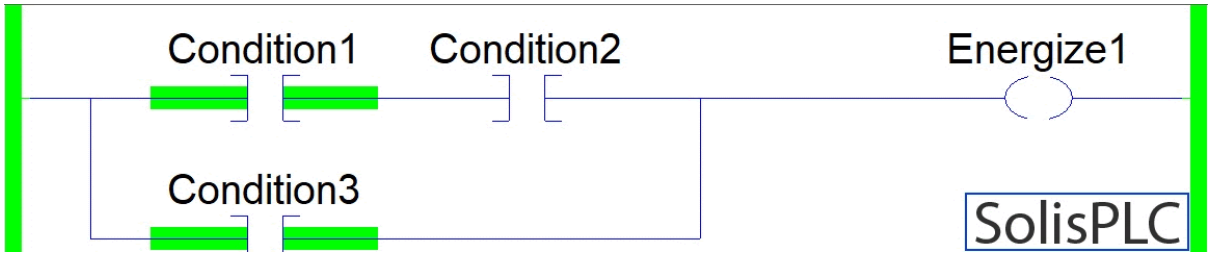


Ladder Logic PLC Programming XIC = ON Example

In the example above, the XIC Instruction is tied to the bit “Condition1”. Since the bit is ON (or 1), the hypothetical current is allowed to pass through and goes to the OTE Instruction. The OTE Instruction sets the “Energize1” bit to HIGH (or 1).

Ladder Logic Structure | Circuit Branches

Now that we’ve seen a basic example that illustrates how the execution of a single ladder logic rung is completed, it’s time to discuss circuit branches. Circuit branches create a way for the current to pass through a different path as the rung executes. The instructions are executed in the same way, but we now need to analyze different paths the current may take.

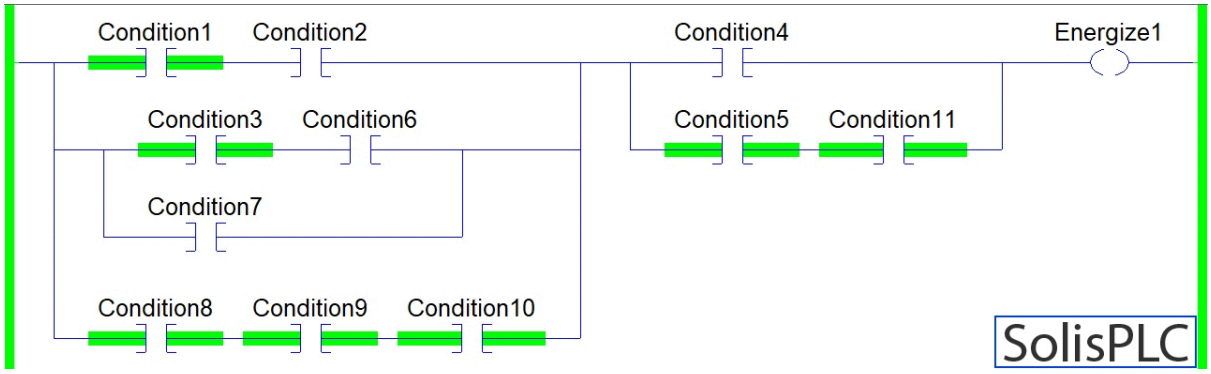


Ladder Logic PLC Programming Circuit Branch Example

The rung above has the primary rung and a branch that jumps the first two conditions with a 3rd one. Let's analyze what's happening with the execution of the Logic.

- **Step 1** - The hypothetical current starts on the main branch of the rung. As it reaches "Condition1", it evaluates the XIC Instruction. The XIC Instruction is TRUE and allows the current to proceed.
- **Step 2** - The hypothetical current flows to the next XIC Instruction and attempts to evaluate it. Since "Condition2" is set to 0, the XIC Instruction evaluates to FALSE. The current is stopped.
- **Step 3** - The hypothetical current goes back to the first branch. The XIC Instruction tied to bit "Condition3" is executed. Since the "Condition3" bit is HIGH, the XIC evaluates to TRUE. The current proceeds.
- **Step 4** - The current reaches the OTE Instruction and sets the "Energize1" bit to ON (or HIGH).

Here's a much more complex example for you to consider. It's not abnormal to find multiple branched circuits in ladder logic.



Ladder Logic PLC Programming Circuit Branch Advanced Example

Advanced Circuit Branching Ladder Logic Practice

Now that you're familiar with how circuit branches work in ladder logic, **it's important to practice tracing the logic** as you would in the field. Most of your work as a PLC programmer is going to be looking at rungs of logic and figuring out why the output is energized or what's preventing it from turning on.

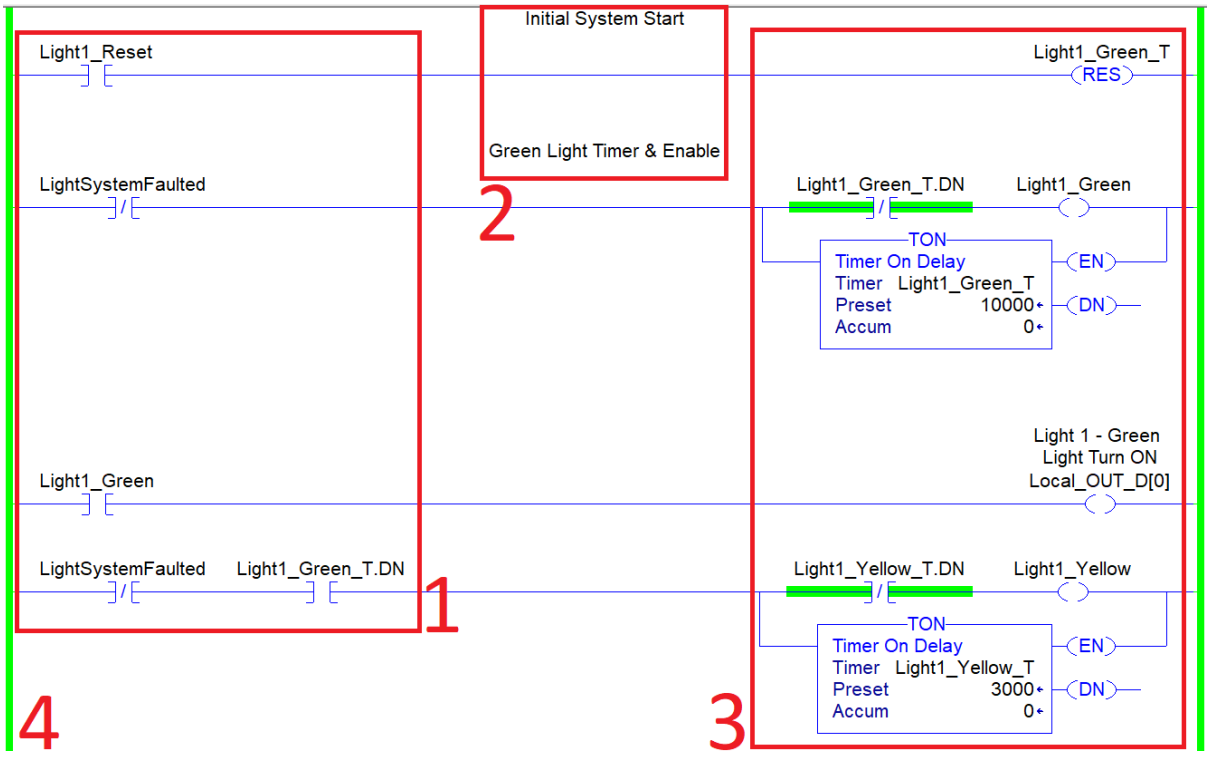
Consider the following situation: your supervisor calls you due to a problem on a production line. For some reason, the pump that's going to deliver raw materials to a specific tank isn't turning ON. As you show up to the operator station, he shows you

that when he pushes the button, the pump won't do anything.

Resolution: you look at the panel, press the button yourself, and confirm that it doesn't start. This pump worked in the past, so **you decide to see what's happening in the PLC logic**. As you trace the output tied to the pump, you notice a complex rung with multiple circuit branches. The reason is that there are numerous conditions for that pump to start. Since you're familiar with the approach above, you can quickly figure out that the pump wasn't able to start because one of the start conditions was that the tank must be empty. As you realized that the tank was, in fact, empty, the conclusion was that the level sensor was broken. You replaced the sensor, and the pump resumed regular operation.

Ladder Logic RSLogix 5000 Components

Now that we have some familiarity with how a basic rung structure is laid out, let's discuss other components of ladder logic.



Ladder Logic Components

1 - Ladder Logic Inputs

As we discussed above, the ladder logic instructions on the left side are called inputs. Their condition is evaluated on a true or false basis. If the evaluation is concluded with a

TRUE, the output of the ladder logic rung is executed. If it's evaluated to FALSE, the PLC goes to the following rung.

2 - Ladder Logic Rung Comments

Every programming language allows the user to add documentation to their software. In ladder logic, this opportunity comes with every rung, instruction and data structure. By adding a comment above the rung, you're making it easier for the person after you to understand your train of thought and troubleshoot the logic as needed. Furthermore, the comments may be used to indicate a change or temporary fix of a certain problem that was encountered by a PLC programmer.

3 - Ladder Logic Outputs

There are many instructions that will execute on the output side. In the example we covered above, our focus was on the OTE Instruction. However, the screenshot above also includes TON or Timer On Delay Instructions. As you gain experience as a PLC programmer, you'll encounter and master additional instructions.

4 - Ladder Logic Rails

Each rung of ladder logic lies between the two side rails (just like a regular ladder). These rails are what energizes each rung as they are executed. In the screenshot above, you can see two rails within the RSLogix / Studio 5000 environment. The rails remain grayed out until the main routine calls the program. In the screenshot, the rails are green, which means that this specific logic is being executed.

5 - Tag Names

Each instruction will be tied to one or more tags. Each tag requires a data structure element as well as a name or label. In the examples we looked at above, tags were labeled as "Condition1", "Condition2", "Condition3", etc. **In production circumstances, tags would typically reflect the physical element they control** or a set of PLC based tags. For example, tags that control motors may have the label of MTR1_Start, MTR2_Stop, MTR2_Status, etc. Furthermore, tags may also have a description that allows the user to give the tag a text-based description.

Ladder Logic Programming in RSLogix 5000 Basics

As you invest yourself in PLC Programming, you'll quickly realize that **the list of different instructions available to you is vast**. Furthermore, as you become advanced at the craft, you may find yourself creating your instructions through the use of an Add-On-Instruction or AOI. However, assuming that you're here for the basics, let's discuss the most useful instructions you should start working with as you tackle industrial automation.

Examine If Closed [XIC]

We've looked at these instructions at the start of the tutorial. It's the essential input check you can make on your data. In short, if the boolean assigned to the XIC is TRUE, the output will go through. If it's FALSE, it won't. Although it may seem that this would have limited utility, **many of the advanced constructs within PLCs have a boolean state**. For example, a Variable Frequency Drive may have an array of boolean structures that are tied to different faults. Therefore, you may create the same number of XIC instructions to verify which failure is present on the drive.

Examine If Open [XIO]

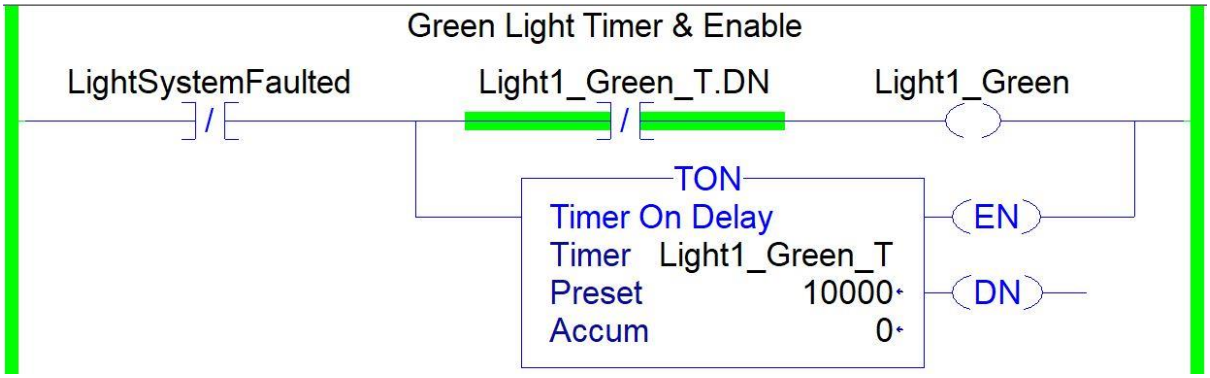
The XIO will energize the output if the exact opposite of the XIC is true. In other words, the output will energize if the boolean value is FALSE.

Output Energize [OTE]

The OTE is an output instruction and will set a boolean to TRUE if all the preceding conditions are TRUE leading to it. The OTE would also set the boolean to FALSE should there be no TRUE path of inputs leading to it. The Output Energize instruction is used to set digital outputs on field devices such as valves, motor contactors, relays, solenoids and more.

Timer ON [TON]

Timers are a basic data-structure of PLCs. They allow the user to create a condition that will start an internal timer and execute an action based on what the user has programmed.

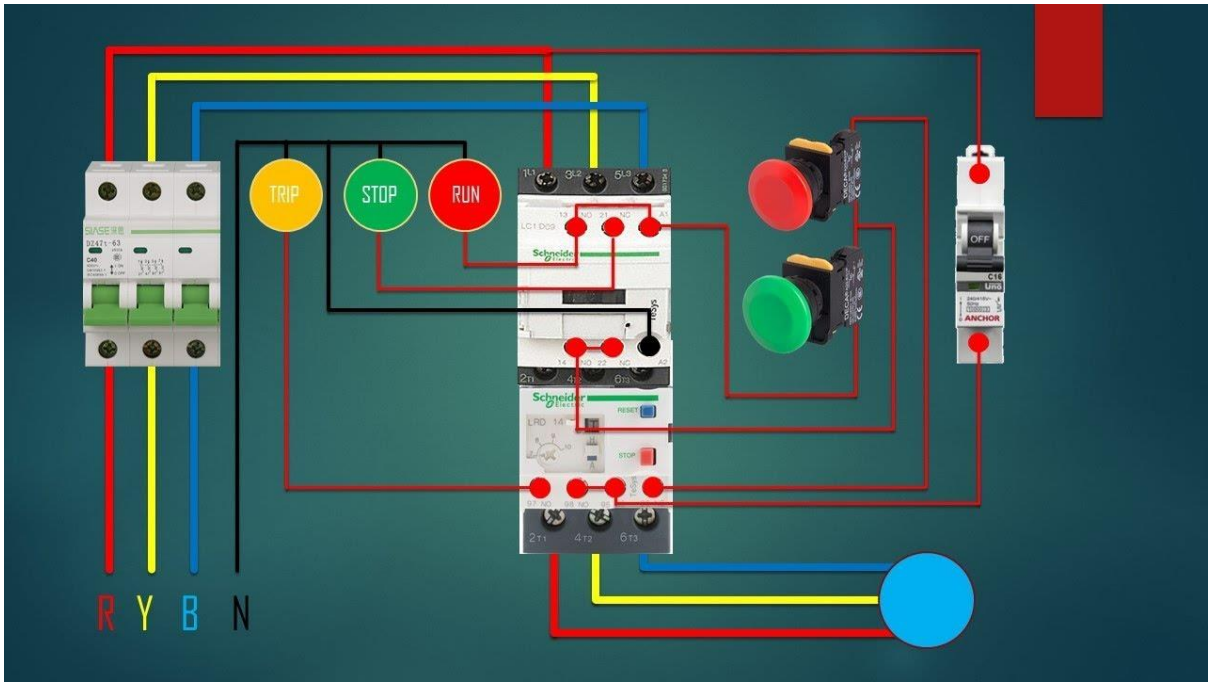


Timer TON Instruction in RSLogix 5000 Ladder Logic Example

The most basic timer instruction is Timer ON or TON. This instruction **will start counting as soon as the input is energized**. The timer will continue timing until it reaches a value specified by the user. A PLC programmer can use boolean status bits of the timer in order to execute logic based on the timer running, completed or not-running status bits. This instruction is fundamental in PLC programming and is often seen in basic sequences, de-bouncing logic and any other programs that require timed execution of ladder logic.

Basic Motor Control Circuit - PLC Programming Ladder Logic Example

One of the most iconic first circuits a PLC programmer must master is a motor starter. Although it is possible to wire motor starter to work without the use of a PLC. There are many benefits to wiring the inputs and outputs into the controller. Once the inputs and outputs are established, **a PLC programmer will create a ladder logic based routine** that would accomplish what the following circuit was intended to create in hardware.



Three Phase Motor Starter Circuit with Push Buttons and Contactor

Before we dive into programming, it is important to understand the operation of the circuit above. Here are the key components and stages of operation:

- **3 Phase Circuit Overload** - Each phase is protected by an overload that will trip as high current flows through due to a motor or circuit fault.
- **Motor Contactor** - The contactor acts as a bridge between the high voltage (motor) and low voltage (control (24VDC)) circuits. When the control circuit is energized, the power circuit is allowed to conduct the necessary current.
- **Start Push Button** - When pressed, the contactor is allowed to energize. The current flows through and the motor starts to operate.
- **Stop Push Button** - When pressed, the contactor is unlatched and stops conducting current which stops the motor.

Although the circuit is straightforward, there is one key feature: the momentary push buttons used for starting and stopping the motor latch in the contactor. In other words, once the momentary Start push button is pressed, the motor will continue to run until a Stop command is issued. In the industry, this is referred to as a latch circuit. **This scheme is used in many applications** including machinery starters, conveyors, process initiation and more.

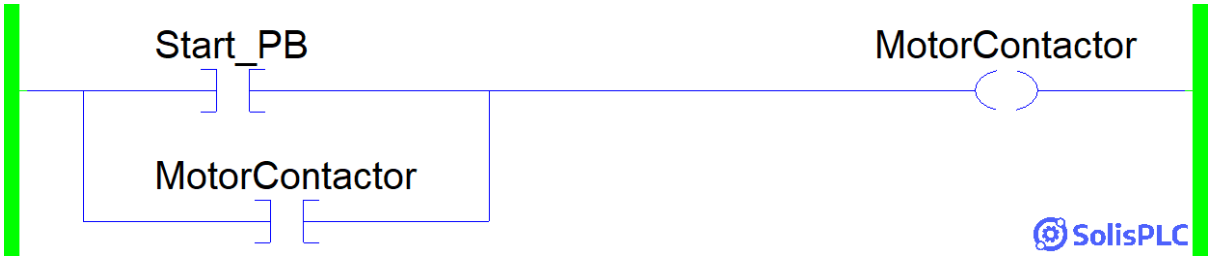
The momentary push buttons are wired into digital inputs of a PLC. When either button is pressed, the appropriate input bits are set to HIGH (1). When the buttons are released, the same input bits are set to LOW (0).

The motor is tied to an output when set to HIGH (1), energizes the coil in the contactor and allows the current to flow.

Based on the above, we start by building a circuit that will run the motor when the Start push button is pressed.

Ladder Logic Example - Motor Starter Part 1

The ladder logic above will take an input through an XIC instruction and energize an output over an OTE instruction. However, a major problem is the fact that the user must keep the button pressed for the motor to run. Typically, we'd want the motor to keep running after the button has been released. Let's look at the second iteration of our ladder logic circuit.



Ladder Logic Example - Motor Starter Part 2

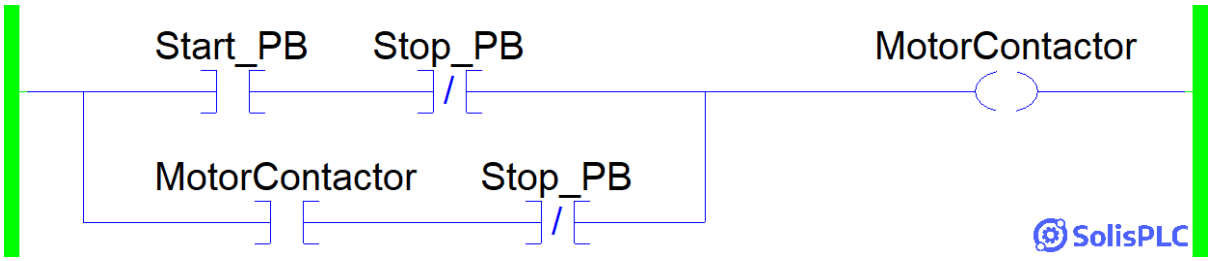
The second iteration of the ladder logic motor starter is able to start the motor and keep it running. However, we're now faced with another problem: there is no way to stop the motor.

The stop push button needs to be integrated into the logic. However, let's take a moment to understand the operation of this button. It has to have two functions:

1. The stop push button should prevent the motor from getting started.
2. The stop push button should stop the motor when it's running.

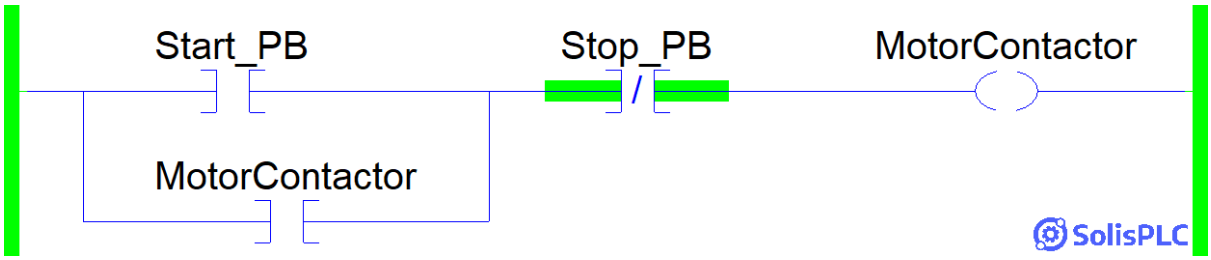
Based on the two requirements above, it's possible to add an XIO instruction into each branch of the circuit. However, ladder logic is such that the user can utilize a single instruction to cover both of those scenarios after the branch.

Example of a functionally sound rung based on the above requirements.



Ladder Logic Example - Motor Starter Part 3 (Not Optimized)

The rung above will operate as expected. However, it's important to create efficient ladder logic and utilize instructions in conjunction to the branch structures we've covered above.



Ladder Logic Example - Motor Starter Part 3

The rung above operates as follows:

Stage 1 - The Start_PB is pressed and the MotorContactor is Energized while Stop_PB is not pressed.

Stage 2 - The MotorContactor bit is used to keep the motor energized while the Start_PB is released.

Stage 3 - The MotorContactor is de-energized when the Stop_PB is pressed.

The motor starter ladder logic example is easy to follow. However, **as you expand your knowledge of ladder logic principles, you will create complex branches around similar circuits**. It's not uncommon to have multiple stop conditions that are set in series with the "Stop_PB" bit. Similarly, it's common to see other sources of starting the motor. For example, a sequence may be used to start a specific pump through software.

Reading a Ladder Diagram

A Ladder Diagram is the representation of a circuit. It can differ in how it is drawn depending on the nature of the circuit, industry, and time it was drawn. We've seen a number of ladder logic examples above; a ladder diagram is used to represent the physical devices tied to the inputs and outputs in those software representations. In other words, a ladder logic diagram will contain an XIC (Examine if Closed) instruction that is tied to a physical push button; the ladder diagram will illustrate the contact of the push button.

Let's take a look at a few examples:

Ladder Diagram - Push Button & Lamp

Figure 5.1 - Ladder Diagram | Push Button & Lamp

The most fundamental example of a ladder diagram involves a push button and a lamp. You can think of the push button as the switch in your room. When it is pressed, the lamp will turn ON. Note that in the previous examples of ladder logic found in the PLC, we've covered XIC and OTE instructions. In this example of a ladder diagram, you can think of the push button as the XIC instruction and the lamp as the OTE instruction.

Ladder Diagram - Motor Starter & Indicator

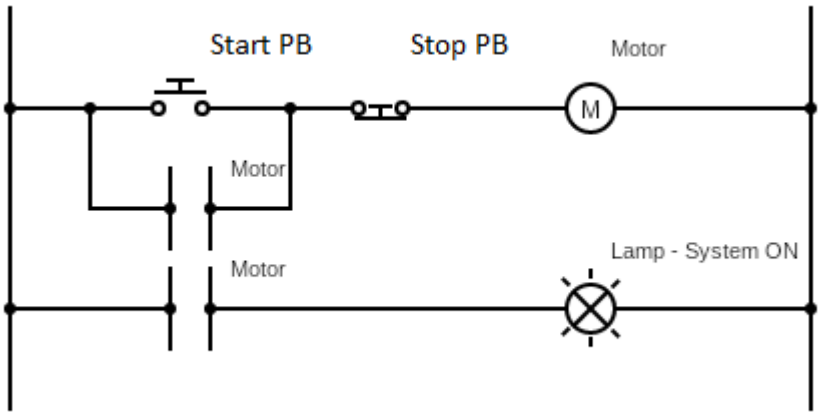


Figure 5.2 - Ladder Diagram | Motor Starter & Indicator

In the circuit diagram above, we've introduced a few new elements. In this scenario, we have a start push button, a stop push button, a motor, coils tied to the motor, and a lamp indicator. The operation of this ladder diagram is simple and has been covered in a few examples above. Once the start push button is pressed, the motor is energized. The first motor contact is used to check for the state and thus seals in the circuit. In other words, the momentary start push button allows the motor to run without needing to be pushed in. If the stop push button is pressed, the contact is broken and the motor is de-energized. The second level of the circuit illustrates a status light (typically green) that is enabled when the motor is running. This circuit is typical and notifies the operator of the state of a motor - pump, drive, etc.

Limit Switches and Ladder Logic Fundamentals

Limit switches are electromechanical devices that are used to detect the presence or absence of an object. They are commonly used in industrial automation to control machines and equipment. Limit switches are typically connected to a PLC (programmable logic controller), which is a computer that controls the machine or equipment. When a limit switch is activated, it sends a signal to the PLC. The PLC then executes the ladder logic program to determine what action to take. For example, a ladder logic program could be used to start a motor when a limit switch is activated, or to stop a motor when a limit switch is activated.

Here are some examples of how limit switches are used with PLC ladder logic:

- **Conveyor belt control:** A limit switch can be used to detect when a product reaches the end of a conveyor belt. The PLC can then use this information to stop the conveyor belt or to divert the product to another conveyor belt.
- **Machine safety:** Limit switches can be used to protect workers from injury. For example, a limit switch can be used to stop a machine if a worker's hand enters a dangerous area.
- **Product counting:** Limit switches can be used to count products as they move along a conveyor belt. The PLC can then use this information to track inventory or to control production.
- **Machine sequencing:** Limit switches can be used to sequence the operation of a machine. For example, a limit switch can be used to start a pump when a tank reaches a certain level, or to stop a conveyor belt when a product reaches a certain location.

Limit switches are a versatile and reliable way to control machines and equipment. By using limit switches with PLC ladder logic, engineers can create complex and sophisticated control systems.

Limit Switch | Examples of common industrial limit switches that are wired to a PLC and programmed using Ladder Logic

Ladder Diagram - Other

In modern electrical system designs, you'll often see ladder diagrams in industrial applications. This representation is easy to follow, lends itself to electrical equipment well, and is easily represented in software control systems - PLCs, DCS, SCADA, etc. However, you'll rarely find ladder diagrams to represent other electrical circuitry - Ex: embedded hardware. The reason is that these systems are difficult to illustrate in this manner and are rarely programmed through the same structure as PLCs.

Conclusion

3.2 Hardware Configuration Tool – HWCONFIG 3.0

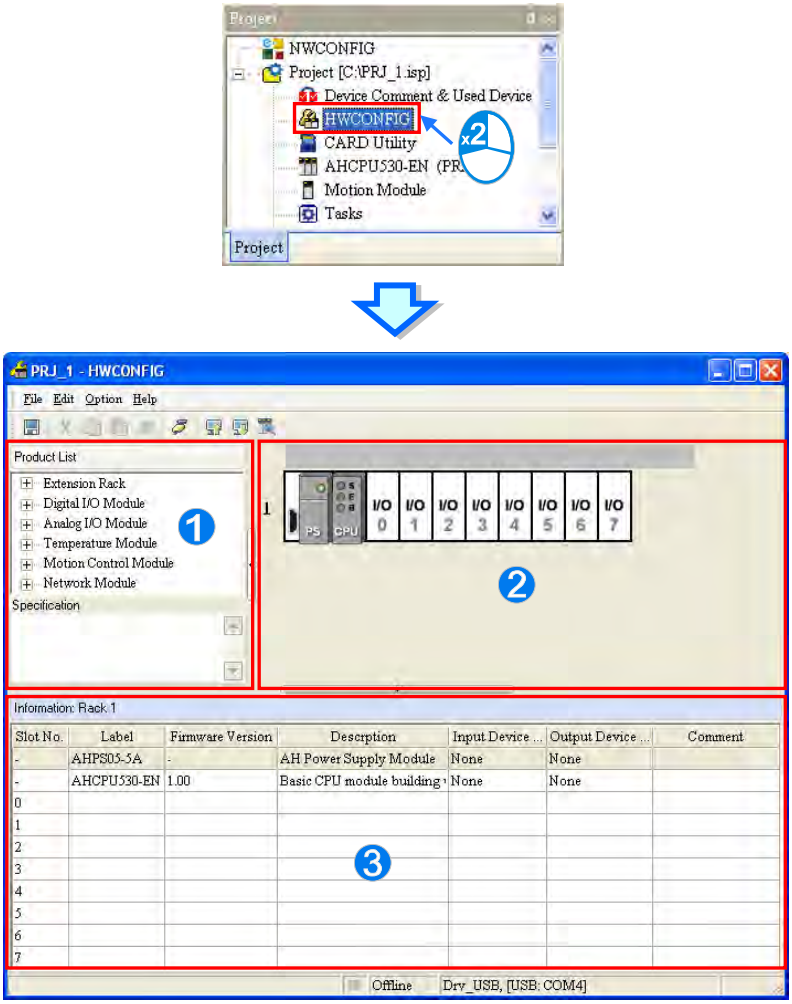
HWCONFIG is one of the tools in ISPSoft for hardware configuration. Its functions include configuration of module racks, parameter settings for modules, download/upload hardware parameters and simple on-line detection and diagnosis function. This section will introduce the functions mentioned above and point out some model types and its unique functions.

Attention! When parameter settings regarding hardware configuration are complete, the settings are effective once downloaded into PLC CPUs.

3

3.2.1 HWCONFIG Environment

Users can double-click **HWCONFIG** from the Project section to open the setting window.



❶ **Product list**: Available hardware are listed in the box.

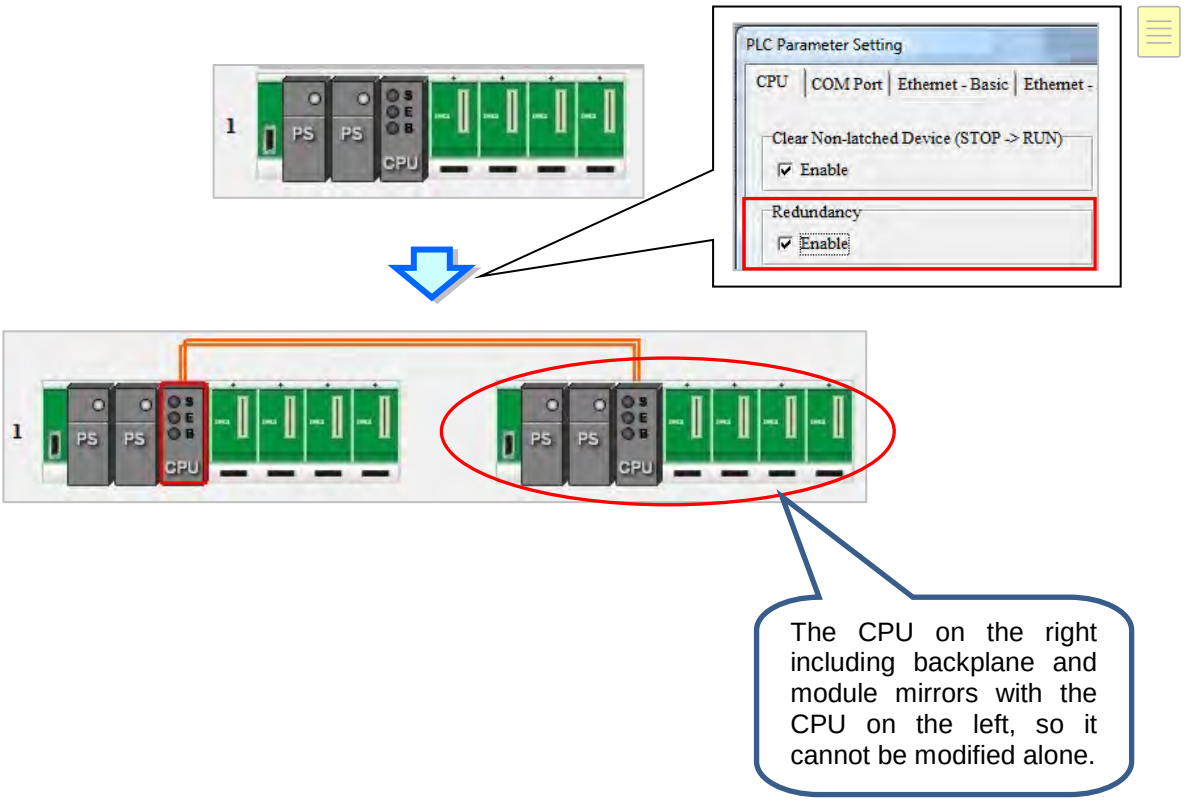
❷ **System configuration area:** This is the main work area for system configuration and settings.

❸ **Information:** Manages the present system configuration through a list.

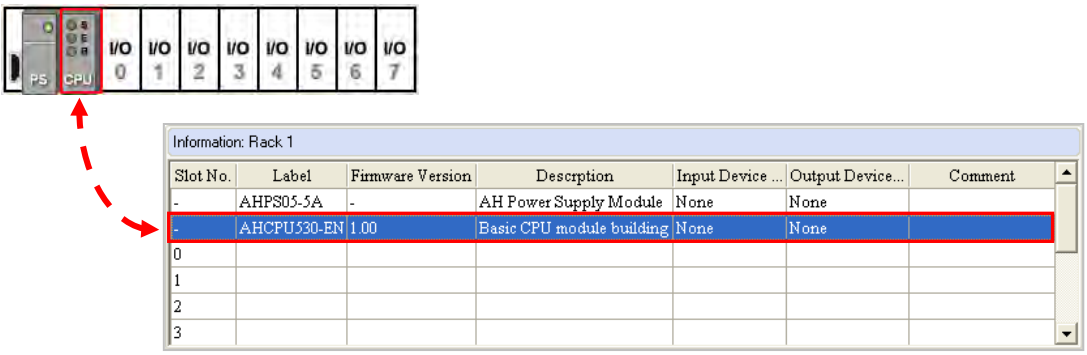
In **System Configuration**, the area uses graphics to demonstrate the current configuration. For AH series PLC, each module is represented by a graphic and the backplane number of a module is shown on the left of the graphic; when operating, use the mouse to click on the selected module in the graphic.

When selecting a backplane, click on the connected port on the left of the graphic.

When using AH560 redundant PLC series, double-click on the CPU and select **Enable** from the **PLC Parameter Setting** window where users will find a redundant CPU appear on the right with settings that mirror the CPU on the left, but cannot be modified alone.

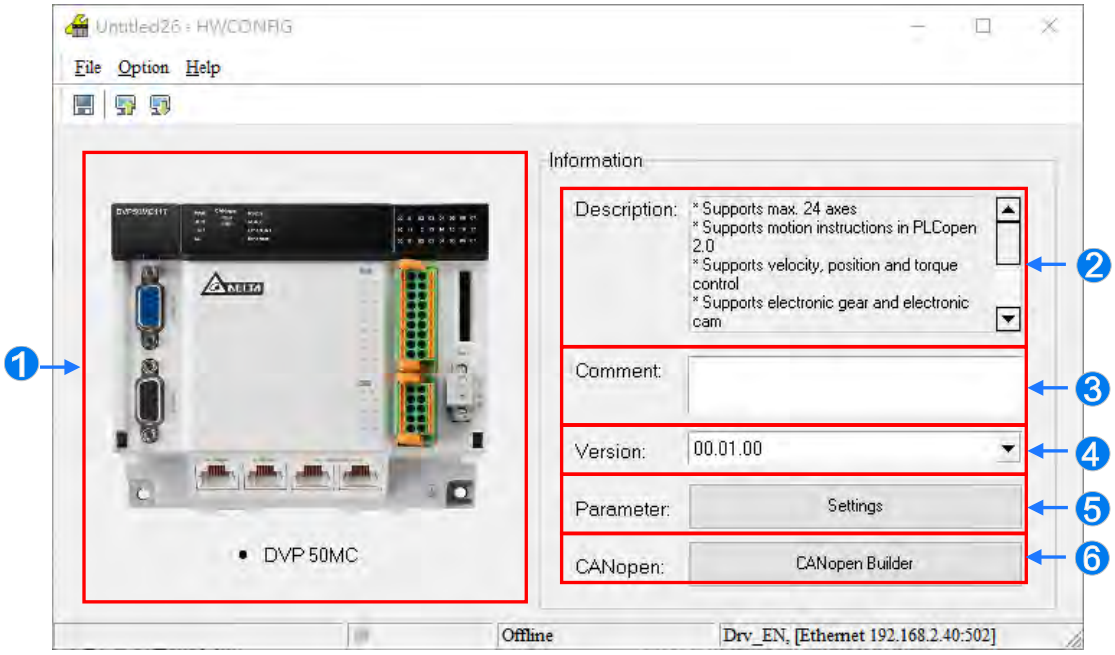


The system configuration area and Information section will update together in real-time. So, when a module is chosen from the system configuration area and Information section, the module is displayed as selected in both places.



3

When using DVPxxMC series, a HWCONFIG window is shown below.



❶ PLC CPU image.

❷ **Description:** Describes the PLC CPU functions.

❸ **Comment:** Comments concerning the PLC CPU.

❹ **Version:** DDF version.

❺ **Parameter:** Click **Settings** to enter PLC Parameter Settings

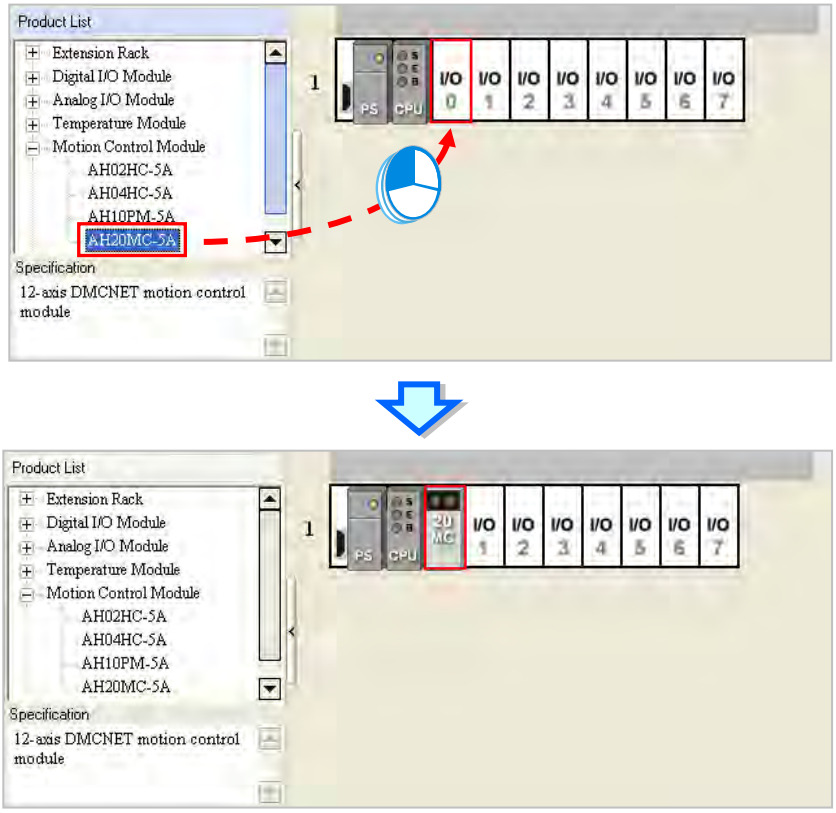
❻ **CANopen:** Click **CANopen Builder** to open the software.

3.2.2 Module Configuration

3.2.2.1 Add Module

- **Method 1**

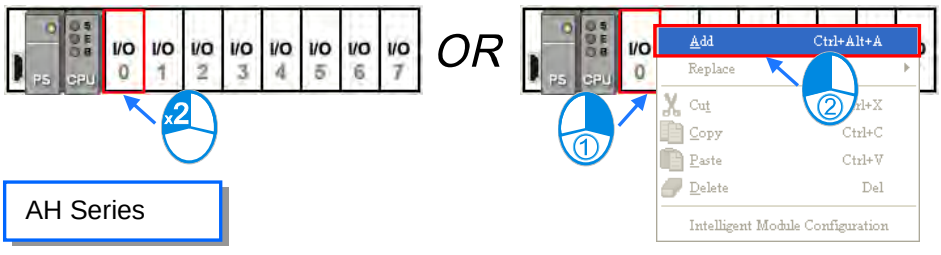
Select the desired module from the left Product List regarding AH series and drag that module to the desired vacant slot on the right.



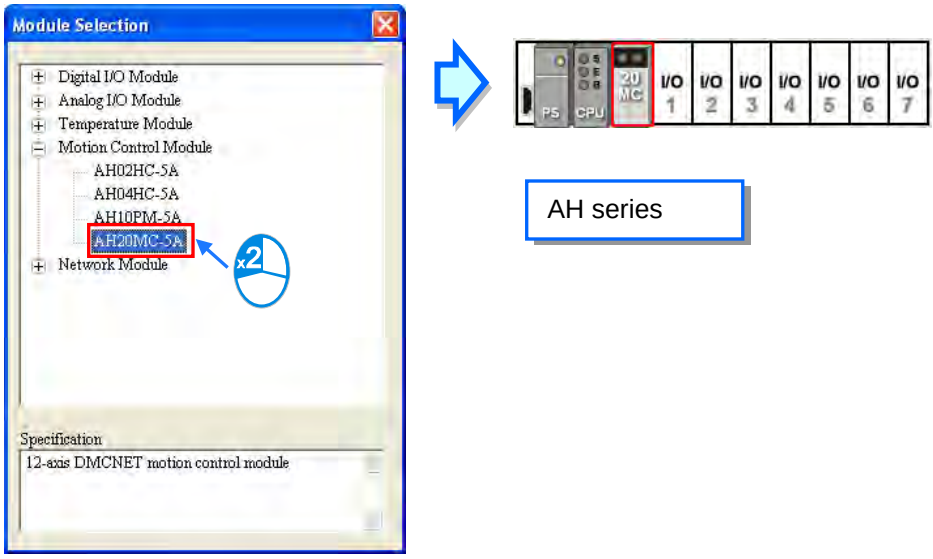
***. When a selected module is dragged to the system configuration area on the right, the module is placed in a vacant slot. The system does not allow the selected module be placed in an existed module slot.**

- **Method 2**

For AH series PLCs, double-click a vacant slot or right-click the slot and choose **Add**. For AS series PLCs, double-click on the + position or right-click the position and choose **Open**.



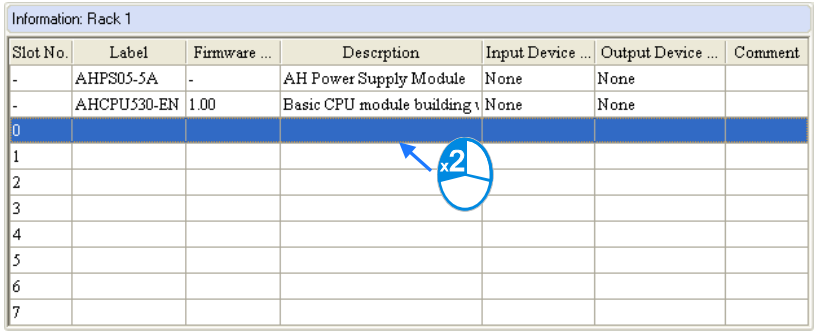
(2) Double-click on the selected module to add.



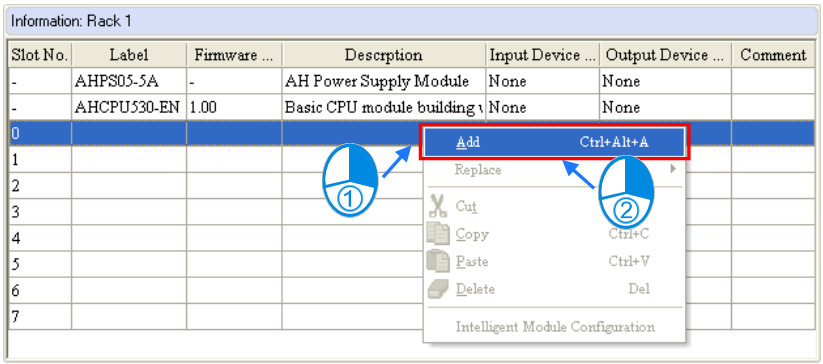
3

- **Method 3**

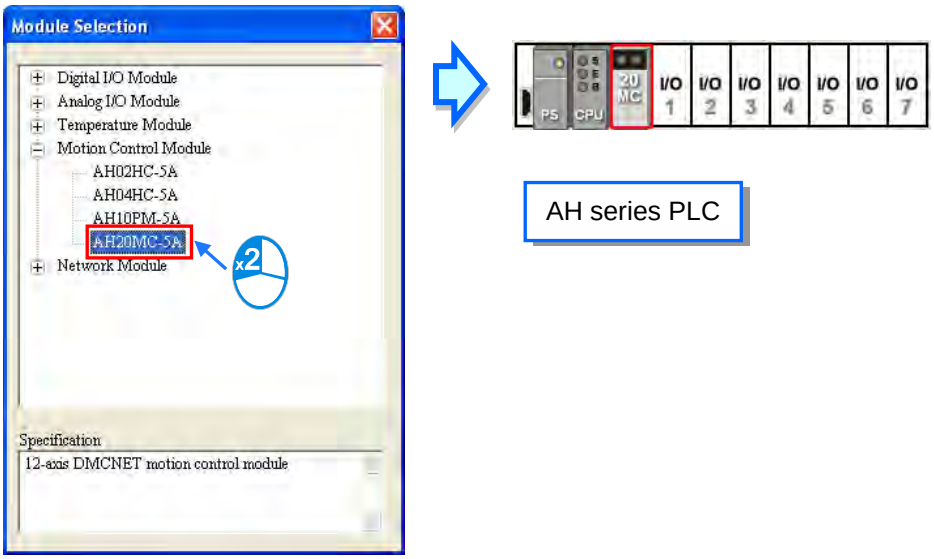
(1) In **Information**, double-click on the blank row of the corresponding slot or right-click and choose Add to see the Module Selection window.



OR

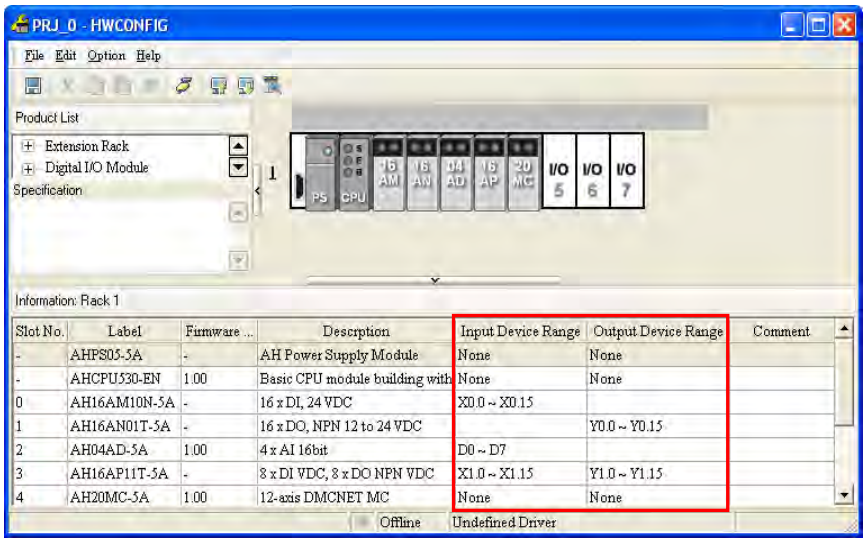


(2) By clicking on a selected module from the list, users can view the **Specification** containing the module information and double-click to add the module.



3.2.2.2 Address Assignment of Modules

When data updates are constantly required for AH series PLCs, for example, analog input modules constantly receives analog signals, the system will automatically assign a device address to save data. The assigned device address is shown in **Information** containing **Input Device Range and Output Device Range** (see below). For AS Series, the group PLC do not support manual settings for address assignment but provides auto configuration on fixed IP address according to the installation position of the modules. For RTU, users can only set up the starting device IP of the first SCM module then the following modules are assigned accordingly. Please refer to section 3.5.3 for AS series remote module configuration.



● **AH16AM10N-5A Module (slot 0)**

This is a 16-point digital input module. The system assigns X0.0~X0.15 that is 1 WORD device in total to store and input module status regarding all points.

● **AH16AN01T-5A Module (slot 1)**

This is a 16-point digital output module. The system assigns Y0.0~Y0.15 that is 1 WORD device in total to store and output module status regarding all points.

● **AH04AD-5A Module (slot 2)**

This is a 4-channel analog input module. The system assigns D0~D7 which is 8 WORD in total and the conversion of external analog input signal is stored in D0~D7.

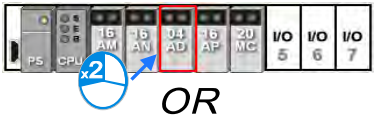
● **AH16AP11T-5A Module (slot 3)**

This is a 8-point digital input/output module. Since the assigned device requires at least 1 word unit, the system assigns X1.0~X1.15 (actual range of use: X1.0 ~ X1.7) device to store and input module status regarding all points; in addition, Y1.0 ~ Y1.15 (actual range of use: Y1.0 ~ Y1.7) is assigned to store and output module status regarding all points.

● **AH20MC-5A Module (slot 4)**

This is a motion control module, but since the module does not require constant data updates, therefore, the system does not assign device address. To obtain the corresponding functions of the assigned device address, double-click on the graphic module or the row on Information to open the Parameter Setting window for the module.

If users want to know functions to which devices assigned correspond, they can double-click the module or the information about the module on the information list to open the **Parameter Setting** window.

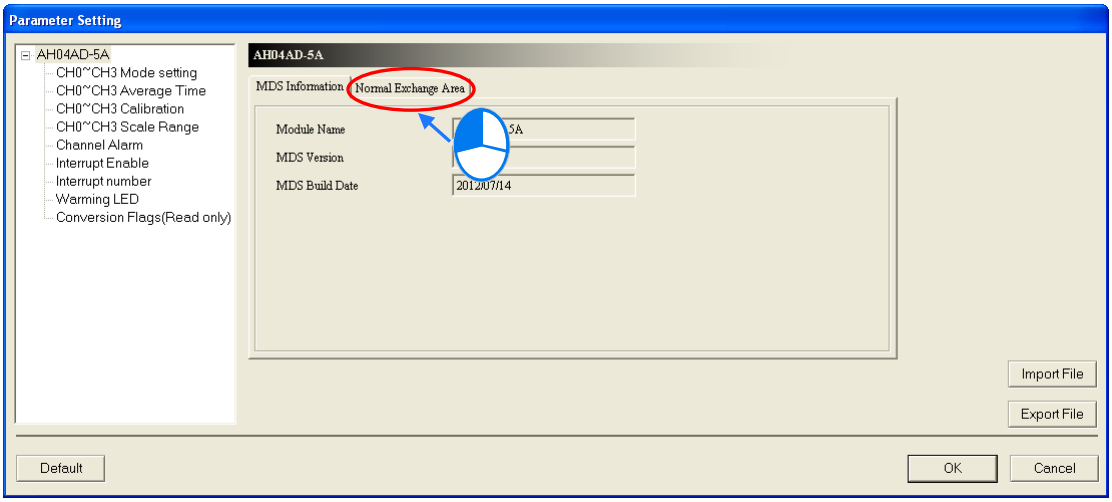


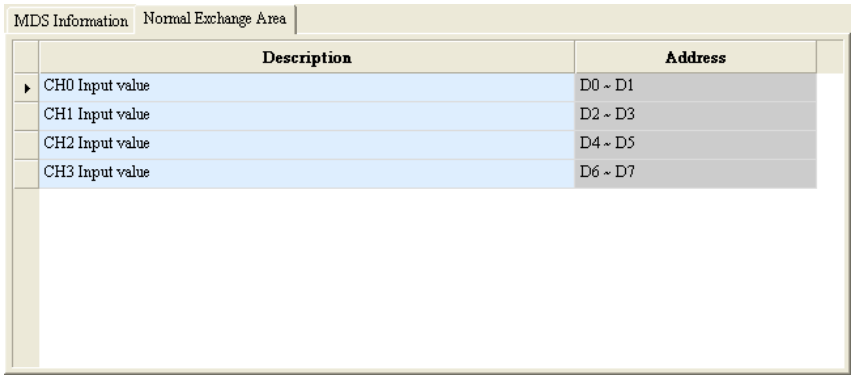
Information: Rack 1

Slot No.	Label	Firmware ...	Description	Input Device Range	Output Device Range	Comment
-	AHPS05-5A	-	AH Power Supply Module	None	None	
-	AHCPU530-EN	1.00	Basic CPU module building with ...	None	None	
0	AH16AM10N-5A	-	16 x DI, 24 VDC	X0.0 ~ X0.15		
1	AH16AN01T-5A	-	16 x DO, NPN 12 to 24 VDC		Y0.0 ~ Y0.15	
2	AH04AD-5A	1.00	4 x AI 16bit	D0 ~ D7		
3	AH16AP11T-5A	x2	8 x DI VDC, 8 x DO NPN VDC	X1.0 ~ X1.15	Y1.0 ~ Y1.15	
4	AH20MC-5A		12-axis DMCNET MC	None	None	
5						
6						
7						

***. When users double-clicks the row on Information box, please click on either the Slot No., Label, or Description cell to avoid columns for edit.**

Click the **Normal Exchange Area** tab in the **Parameter Setting** window to see a list regarding device address and corresponding descriptions.

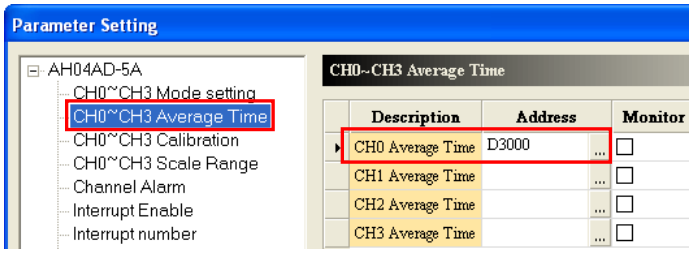


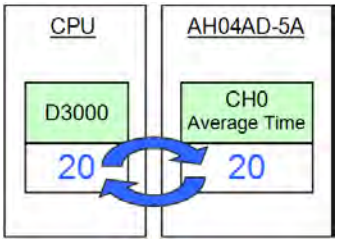


In order to update data constantly, the system uses auto-configuration for the device in storing data. As for internal parameters not updated frequently in modules, users can define the corresponding device D and when the system starts operating, the corresponding device D in the PLC CPU will align with internal parameters of the assigned module. As a result, users can indirectly save internal data of the module through corresponding device D. This is more effective than using the traditional **FROM / TO** command to save internal data of the module.

For example, the **AH04AD-5A** module parameter setting window has D3000 defined as the module parameter – **CH0 Average Time** of **corresponding device D**. Thus, after HWCONFIG parameters are downloaded to the PLC CPUs, users can directly modify D3000 content value in the PLC CPU and indirectly modify **CH0 Average Time** of **AH04AD-5A** module.

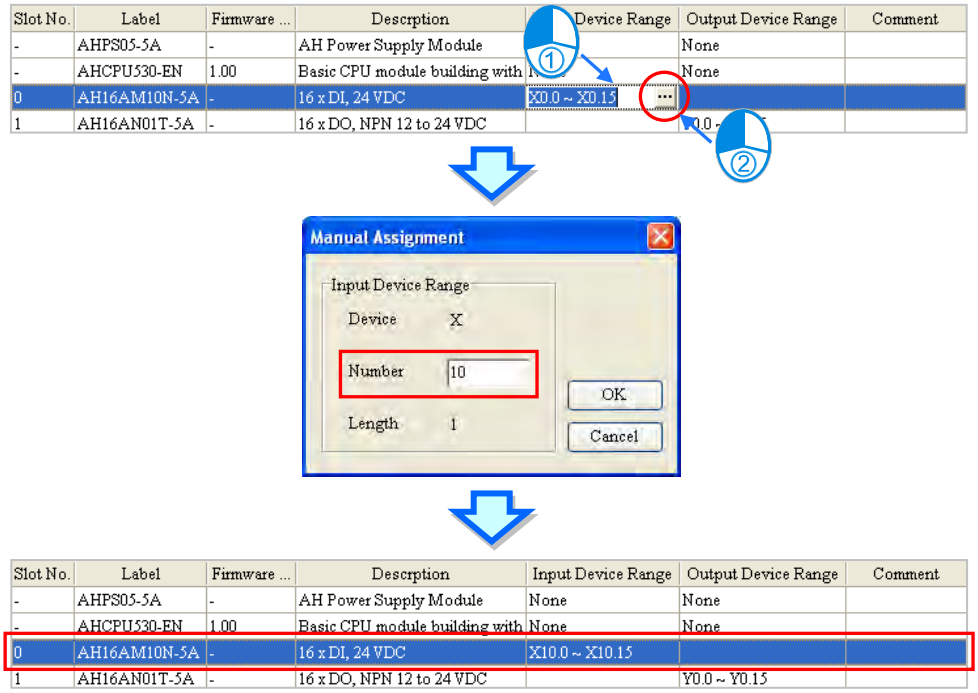
In addition, if modules parameters written in corresponding device D are not permitted, the system will restore the parameters to its original values.



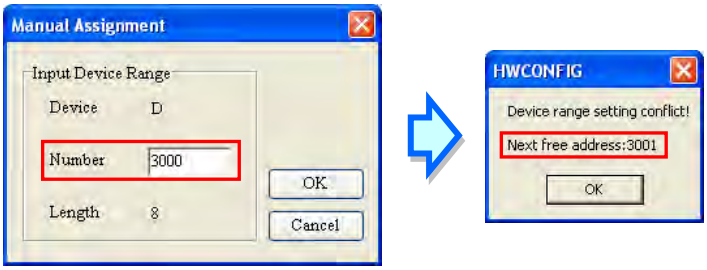


***. Please refer to section 3.7 for more information about internal parameter settings of modules; for FROM/TO command, please see the instruction manuals for any PLC types.**

3 The information in **Input and Output Device Range** is auto-configured by the system, but also user-defined. Users can choose the selected row listed in Information box and then click ···, or type the starting address to open the setting window and click **OK** once finish input the address.

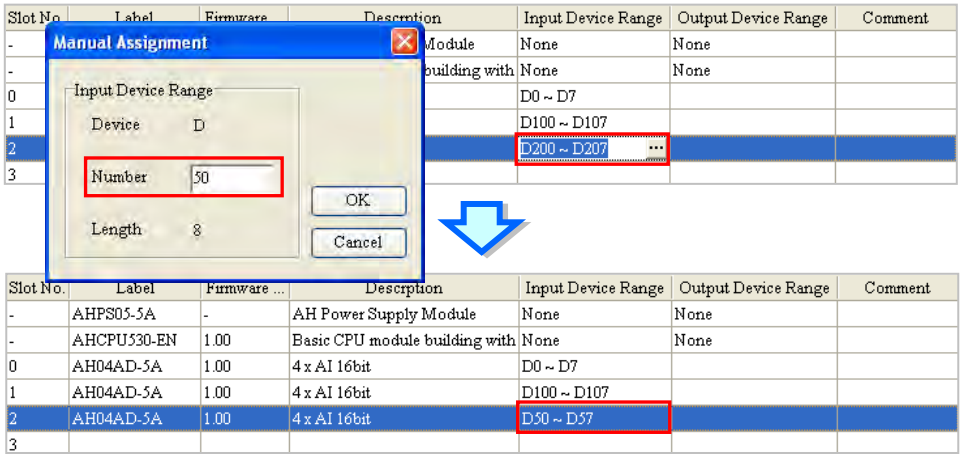


Whether in **Input and Output Device Range** from **Information** or **corresponding device D** for defining the internal parameters of the module, the input device range cannot overlap. When the setting address is in conflict, the system will auto-correct and suggest another available address based on the current input; for instance, when D3000 device is occupied by another module but is being assigned again, the system will automatically search for another available address from D3000.



When **Input and Output Device Range** is modified and modules need to be added, the default device address will be based on the last modified device address and skip the used ones for assignment; even though the number in the last modified address is not the biggest, the new module address assignment is still based on the last modification.

The following example shows **Input Device Range** is modified to D50~D57 for Slot No. 2.



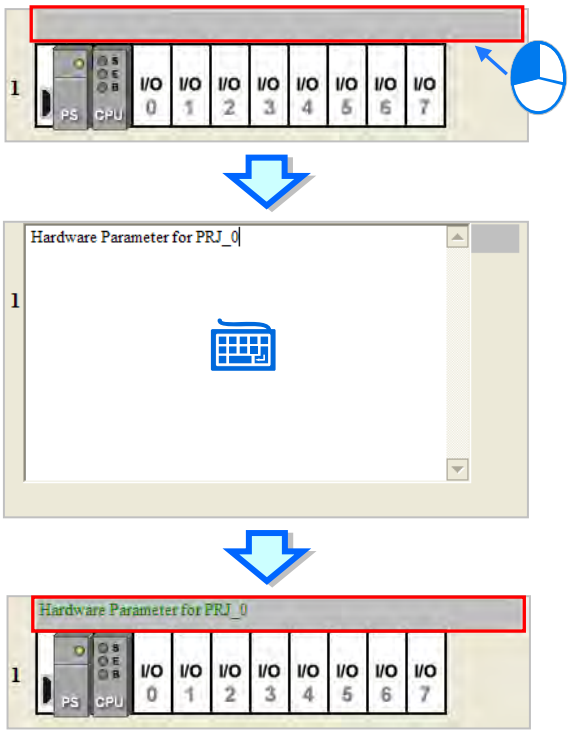
Slot No.	Label	Firmware ...	Description	Input Device Range	Output Device Range	Comment
-	AHPS05-5A	-	AH Power Supply Module	None	None	
-	AHCPU530-EN	1.00	Basic CPU module building with	None	None	
0	AH04AD-5A	1.00	4 x AI 16bit	D0 ~ D7		
1	AH04AD-5A	1.00	4 x AI 16bit	D100 ~ D107		
2	AH04AD-5A	1.00	4 x AI 16bit	D50 ~ D57		
3						

When a new module is added, the **Input Device Range** is automatically assigned from the last modified D50~D57 and not based from the system's biggest D100~D107. As a result, the **Input Device Range** of the module is assigned as D58~D65.

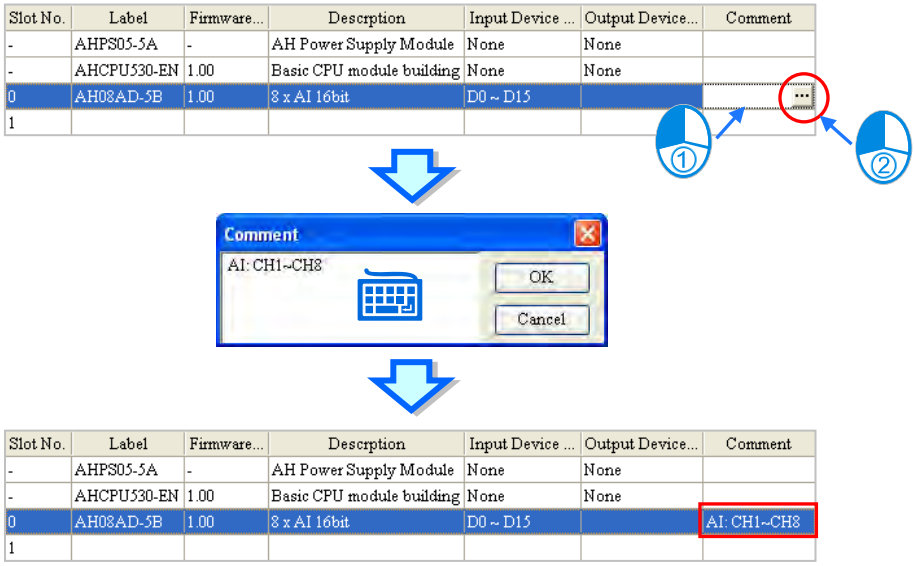
Slot No.	Label	Firmware ...	Description	Input Device Range	Output Device Range	Comment
-	AHPS05-5A	-	AH Power Supply Module	None	None	
-	AHCPU530-EN	1.00	Basic CPU module building with	None	None	
0	AH04AD-5A	1.00	4 x AI 16bit	D0 ~ D7		
1	AH04AD-5A	1.00	4 x AI 16bit	D100 ~ D107		
2	AH04AD-5A	1.00	4 x AI 16bit	D50 ~ D57		
3	AH04AD-5A	1.00	4 x AI 16bit	D58 ~ D65		
4						

3.2.2.3 Edit Comment

Click the gray area on the top of the **System Configuration** area and type comments for the hardware configuration regarding the Project. When typing the comments, press Shift + Enter to start a new line and when complete, click Enter.



For AH5x0, AH5x1 series and AH560 redundant series, users can select the **Comment** column and click on the keyboard icon in the Comment window or click ... to view a pop-up window and input the comment concerning the module.

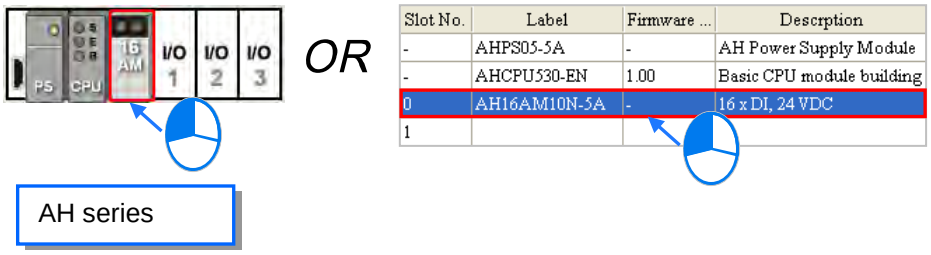


3.2.2.4 Delete Module

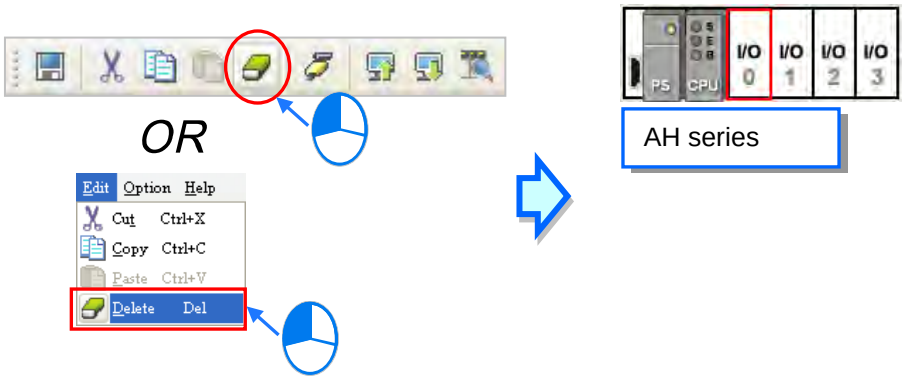
There are two ways to delete a module which has been configured. (The CPU module and AH-series power supply module cannot be deleted.)

- **Method 1**

(1) Select a module which will be deleted from the system configuration area or information list.

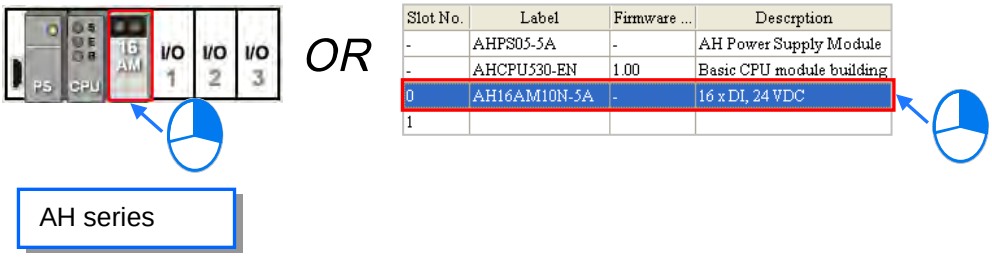


(2) Click **Delete** on the **Edit** menu, click on the toolbar, or press Delete on the keyboard.

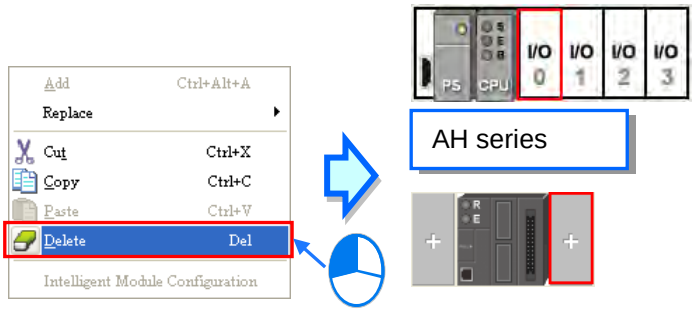


- **Method 2**

(1) Right-click a module which will be deleted from the system configuration area or information list.

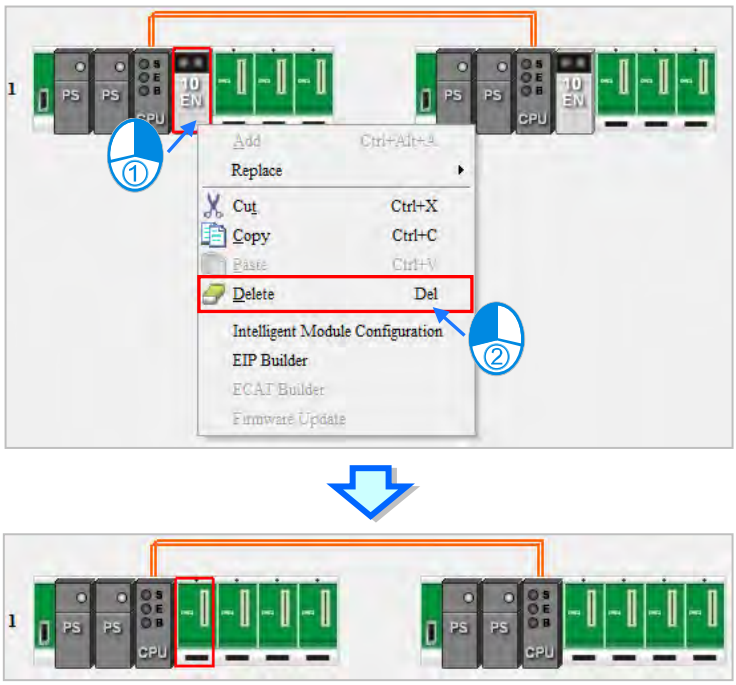


(2) Click **Delete** on the context menu.



Additional remark

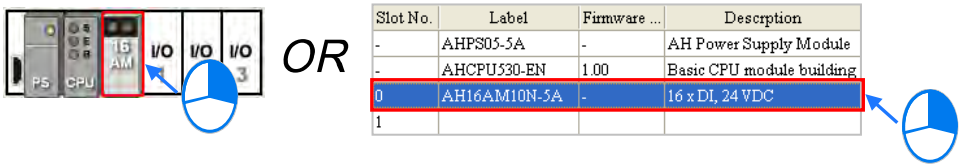
When users select AH560 redundant series and enables **Redundancy** function, the redundant backplane module can be deleted only from the left redundant backplane shown in the **System Configuration** area, while the redundant backplane configuration on the right mirrors the left so it cannot be modified.



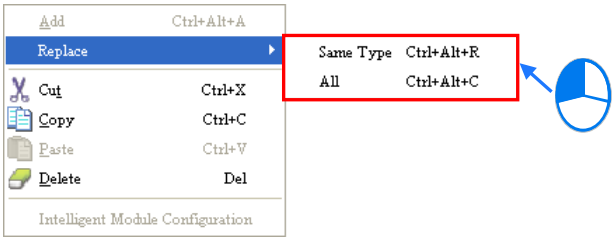
3.2.2.5 Replace Module

Users can take the following steps to modify AH series modules in System Configuration area:

(1) Select and right-click an AH series module for modification in **System Configuration** area or from **Information**. (CPU and power module cannot be modified.)



(2) Choose **Replace** on the quick menu and two ways are shown in the following.



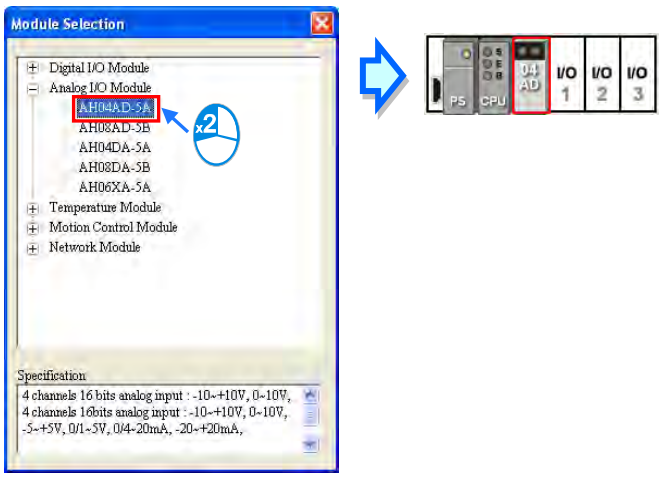
➢ **Same Type**

Replace only the selected module to be the same type module. The new module **Input/Output Device Range** will be the same, while other parameters may return to system defaults if it is not corresponding.

➢ **All**

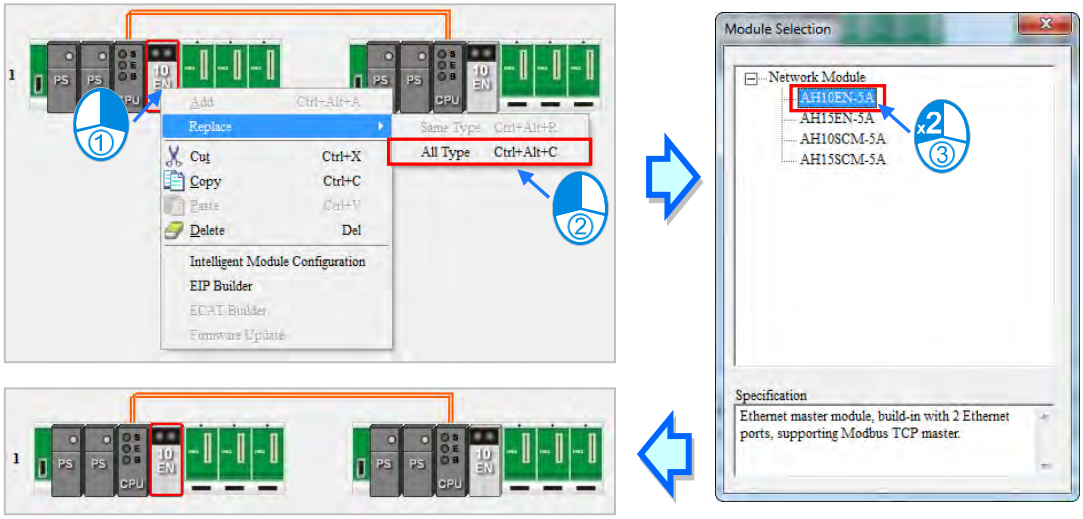
Replace selected module to be any type of module. The result is similar to deleting the original module by adding a new one, so the new module **Input/Output Device Range** will be re-configured and other parameter settings will also return to system defaults.

(3) When selecting on methods to Replace, the Module Selection window appears with modules available for the selected Replace method; when the replaced module is decided, you can double-click on that module.



Additional remark

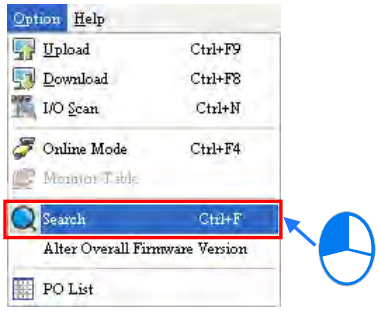
When users select AH560 redundant series and enables **Redundancy** function, the redundant backplane module can be deleted only from the left redundant backplane shown in the **System Configuration** area, while the redundant backplane configuration on the right mirrors the left so it cannot be modified.



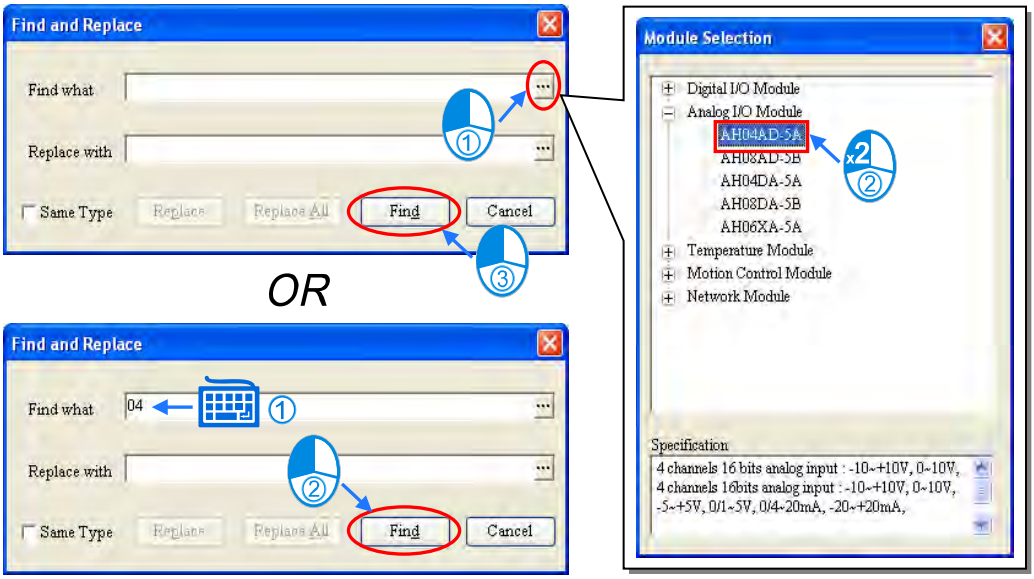
3.2.2.6 Search / Find and Replace

Users can take the following steps to search or find and replace AH series modules in System Configuration area:

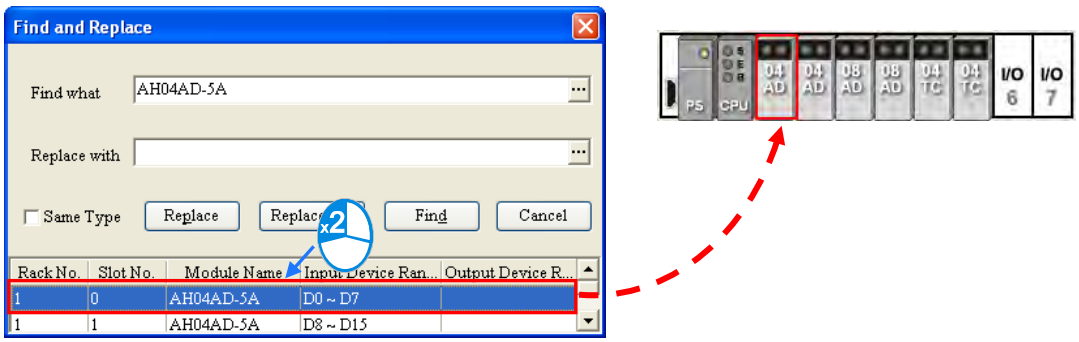
Click **Search** from the **Option** menu in the HWCONFIG toolbar and an operating window appears.



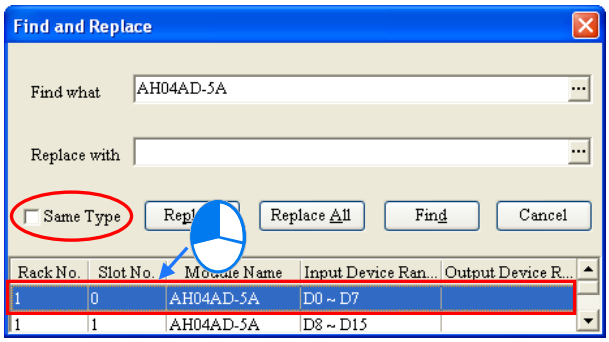
(1) Click ··· in the **Find and Replace** box to search the selected module type from the **Module Selection** window and double-click on the module. Or input keywords regarding the module type in the Find What box and click **Find**.



(2) When finding is complete, a list of modules that meet the search condition is shown below in the Find and Replace window; when double-click a selected item from the list, that module is also selected in the System Configuration area.



(3) To replace a module, please first select the item to be replaced from the list in the Find and Replace window (see below) and select the checkbox if to replace as the Same Type.



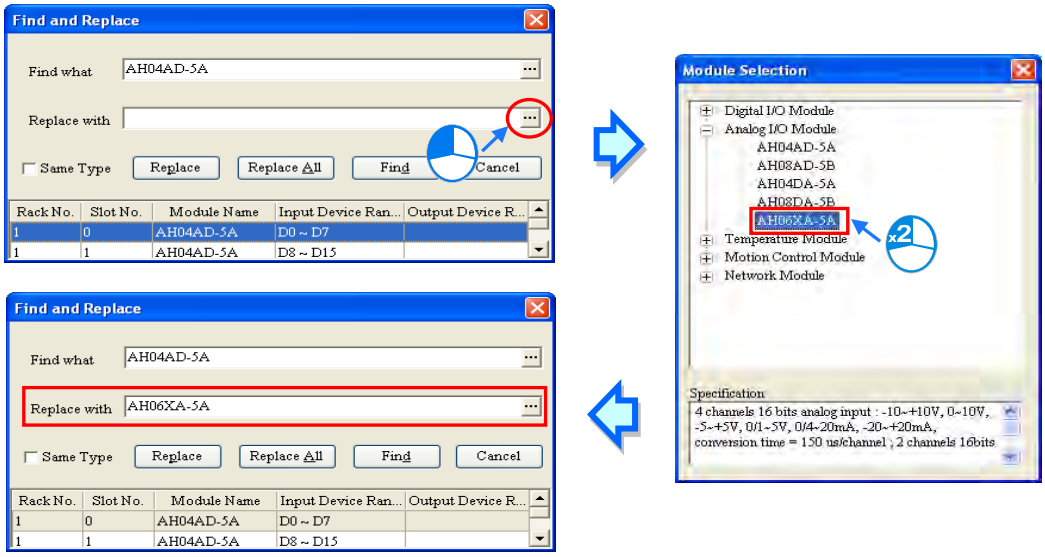
➢ **Same Type** Selected

A module selected is replaced by a same type of module. After the module is replaced, the input/output devices assigned to the new module are the same as the input/output devices assigned to the module replaced. Besides, if the parameters in the new module are not the same as the parameters in the module replaced, the setting of the parameters in the new module will be restored to the default values.

➢ **Same Type** Not Selected

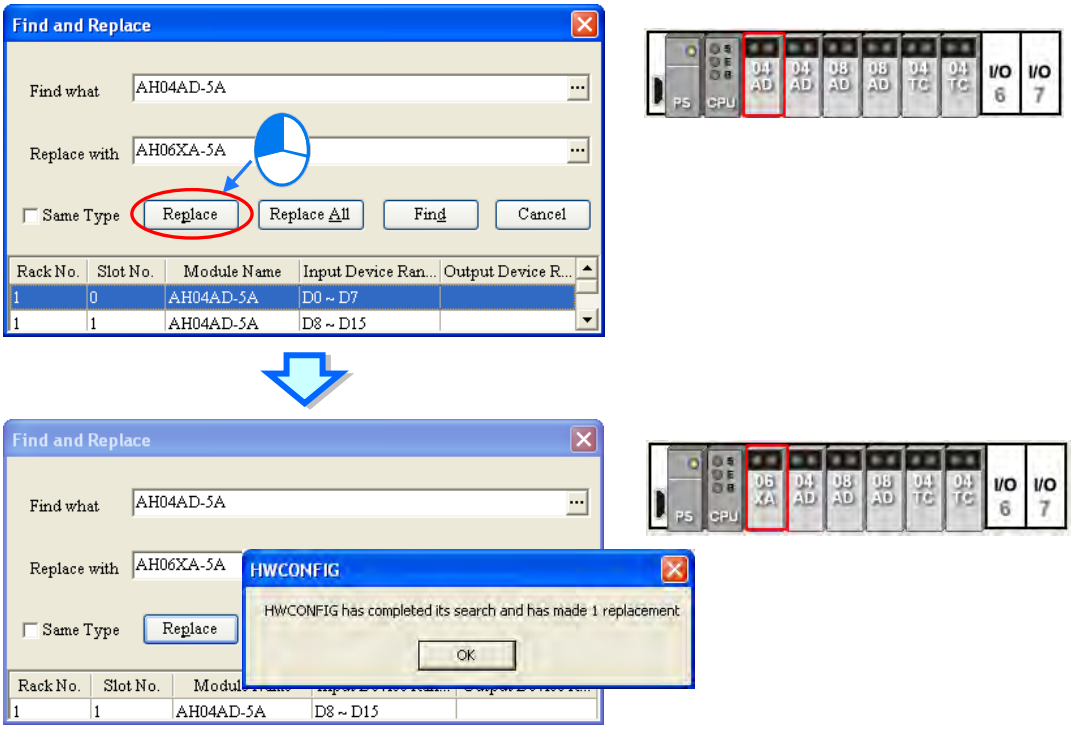
A module selected can be replaced by any type of module. After the module is replaced, the different input/output devices will be assigned to the new module, and the setting of the parameters in the new module will be restored to the default values.

(4) Click ··· in the **Replace with** box, select a module in the **Module Selection** window, and double-click the module. Owning to the fact that a module must be replaced by a specific module, typing a module model or part of a module model in the **Replace with** box is not allowed.

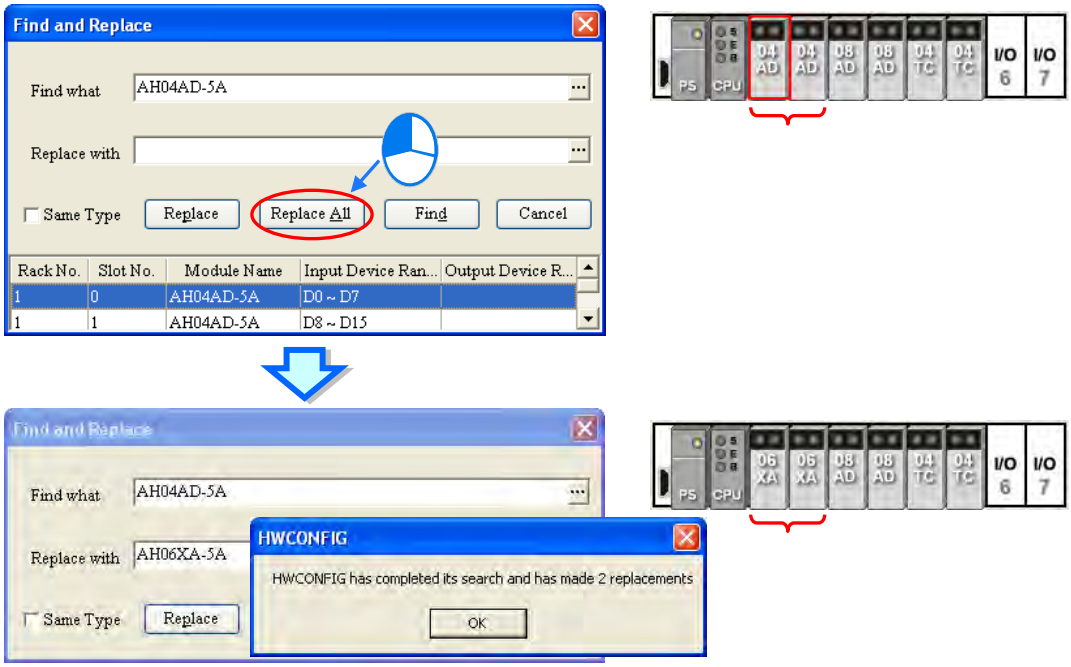


***. If the Replace With box is blank, the selected module is deleted once Replace function is executed.**

(5) When Find and Replace setting is complete, users can click Replace to replace the module selected with the new module. The search list is updated once Replace is complete and will automatically select the next item.



(6) After filling in the setups in the Find and Replace window, users can click **Replace All and** new modules based on the setting will replace the listed items shown in the window.



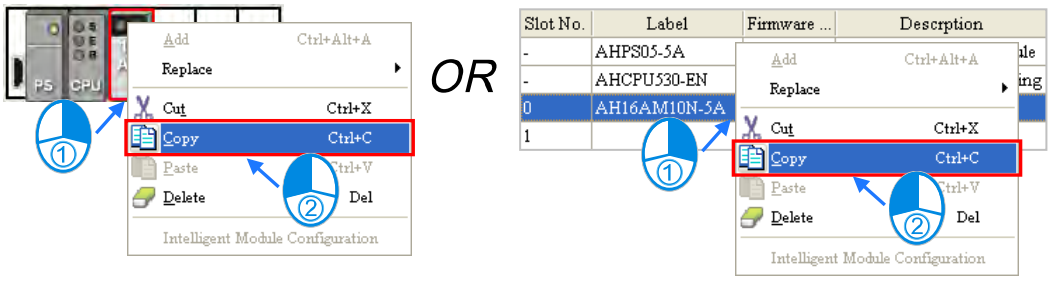
* When using AH560 redundant system with Redundancy function enabled, only the main redundant backplane on the left in System Configuration area can perform and replace modules, while the main redundant backplane on the right is configured to mirror the left and cannot be modified. To execute Find and Replace function, the configuration of the main redundant backplane on the right is not on the search list.

3.2.2.7 Copy / Paste

There are two ways to copy modules (CPU and power modules are not to copy/paste):

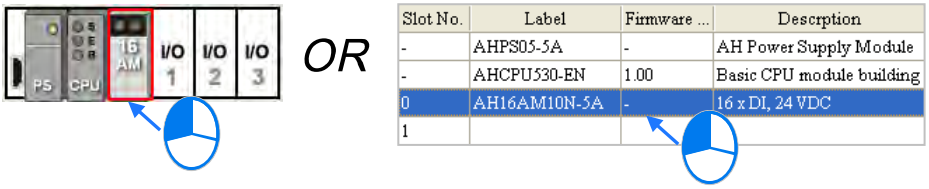
- **Method 1**

Right-click a selected module to copy in the **system configuration area** or **Information**, and choose **Copy** from the context menu.

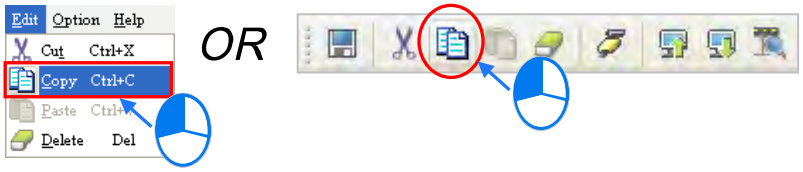


- **Method 2**

(1) Select a module to copy from **System Configuration** area or **Information.**



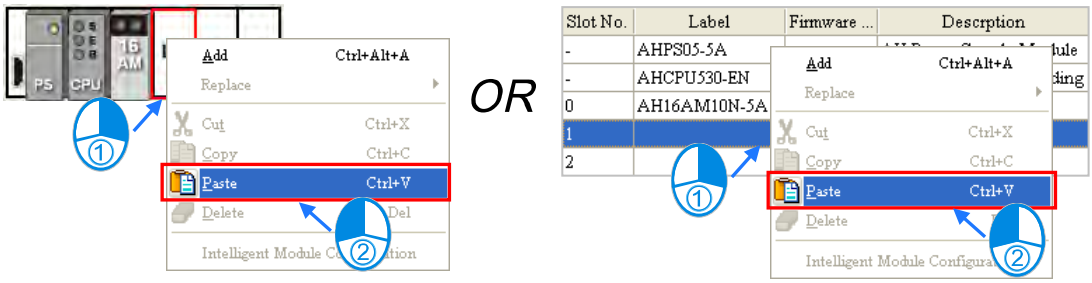
(2) Click **Copy** from the **Edit** menu, or  on the toolbar.



There are two ways to paste modules:

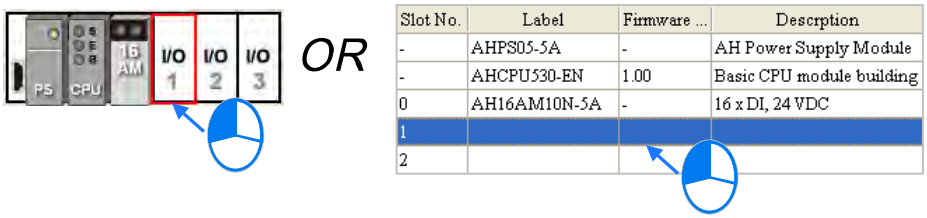
● **Method 1**

Right-click a slot to paste from **System Configuration** area or I**nformation** and choose **Paste** from the quick menu.



● **Method 2**

(1) Select a slot to paste from the **System Configuration** area or **Information.**



(2) Choose **Paste** from **Edit** in the HWCONFIG toolbar, or click  on the toolbar.

Additional remark

To copy/paste a module, users can view the following descriptions on module parameters.

● **Input/Output device range**: The paste module automatically skips to the available device range number.

● **Comment**: Comments are copied to the paste module.

● **Internal parameters**: Parameters are copied the paste module.

- **Corresponding device D**: An assigned column of the **Corresponding device D** for internal PLC paste module, which automatically skips to the available device range number.

- **Advanced parameters** for **intelligent module**: The parameters cannot be copied and is referred to as system defaults. Users will need to reset the parameters.

The following example shows that slot 0 is the copy module, while slot 1 is the paste module.

Description	Address	Monitor	Initial
CH0 Average Time	D3000	☑	30
CH1 Average Time		☐	10

Description	Address	Monitor	Initial
CH0 Average Time	D3001	☑	30
CH1 Average Time		☐	10

Slot No.	Label	Firmware ...	Descrption	Input Device ...	Output Device ...	Comment
-	AHPS05-5A	-	AH Power Supply Module	None	None	
-	AHCPU530-EN	1.00	Basic CPU module building	None	None	
0	AH04AD-5A	1.00	4 x AI 16bit	D0 ~ D7		AI
1	AH04AD-5A	1.00	4 x AI 16bit	D8 ~ D15		AI
2						

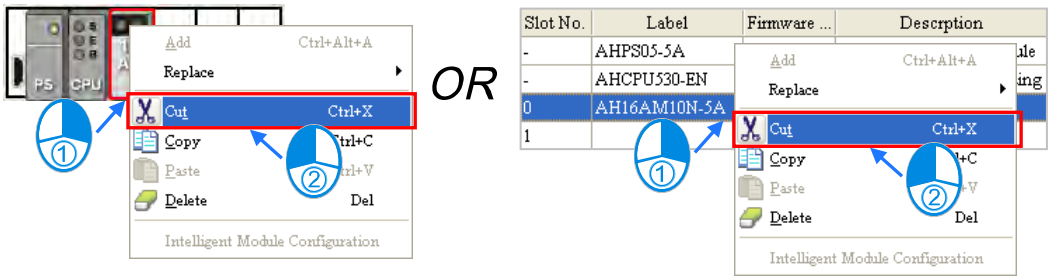
When using AH560 redundant system with Redundancy function enabled, only the main redundant backplane on the left in System Configuration area can copy/paste modules, while the main redundant backplane on the right is configured to mirror the left and cannot be modified.

3.2.2.8 Cut / Paste

There are two ways to cut modules (CPU and power modules are not to cut or paste):

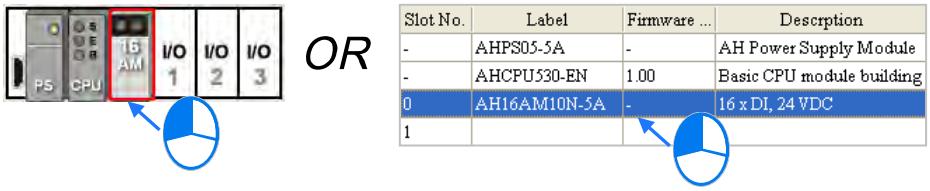
- **Method 1**

Right-click a selected module to cut in the **system configuration area** or **Information**, and choose **Cut** from the context menu.

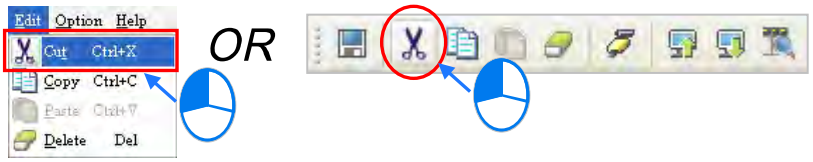


- **Method 2**

(1) Select a module to cut from **System Configuration area** or **Information**.



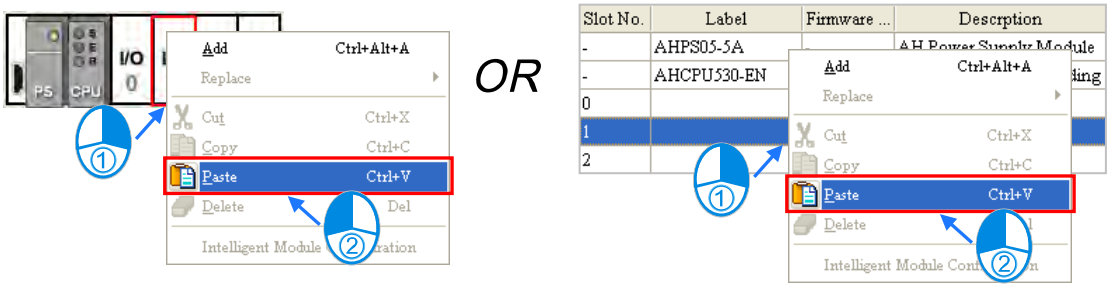
(2) Choose **Cut** from the **Edit** menu, or click on the toolbar.



There are two ways to paste modules:

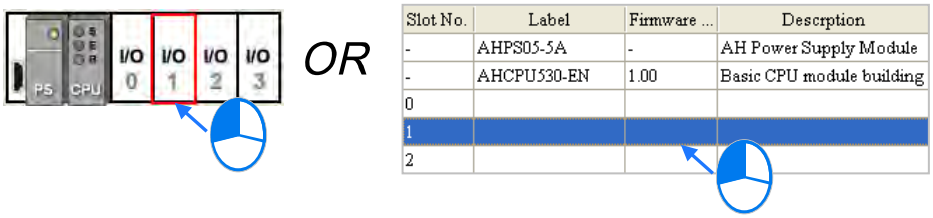
- **Method 1**

  Right-click on the selected slot to paste from **System Configuration** area or **Information**, and choose **Paste** in the context menu.

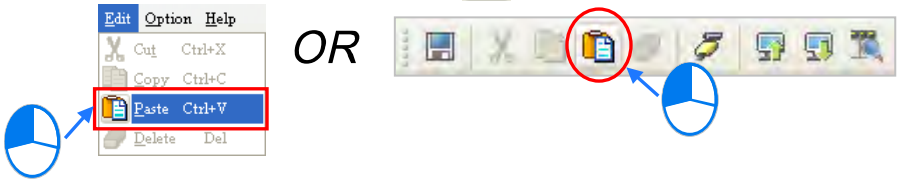


- **Method 2**

  (1) Select a slot to paste from **System Configuration** area or **Information.**



  (2) Choose **Paste** from the **Edit** menu, or click  on the toolbar.



Additional remark

When executing **Cut/Paste** function, modules that are cut can only be paste once. However, the **input/output device range, comment, internal parameters, corresponding device D** and **advanced parameters** for **intelligent modules** from modules that are cut is altogether transferred to the paste module.

When using AH560 redundant system with Redundancy function enabled, only the main redundant backplane on the left in System Configuration area can cut/paste modules, while the main redundant backplane on the right is configured to mirror the left and cannot be modified.

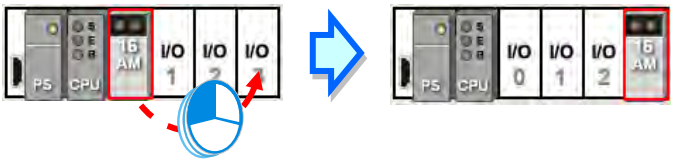
3.2.2.9 Drag and Drop

Besides CPU and power modules, all module graphics in System Configuration area can be dragged by the mouse .

For AH series, there are two dragging and dropping conditions:

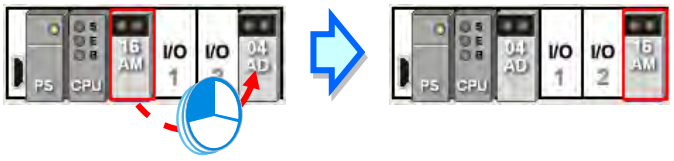
- **Condition 1**

  Select a module to drag and drop onto a vacant slot, then the will be moved to a new position.



- **Condition 2**

  Drag a selected module to a position occupied by another module, the position of the two modules are changed.



Additional remark

When using AH560 redundant system with Redundancy function enabled, only the main redundant backplane on the left in System Configuration area can drag and drop modules, while the main redundant backplane on the right is configured to mirror the left and cannot be modified.

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

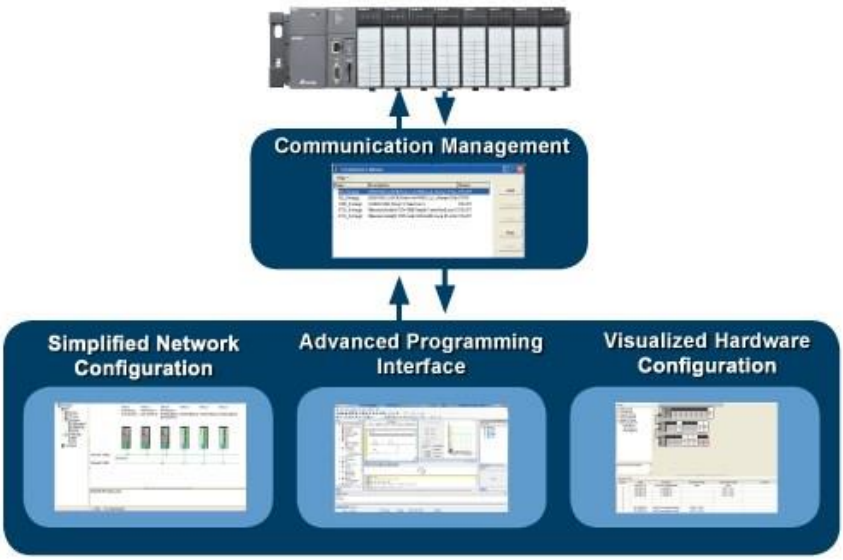
Name of Student:

Experiment No:

Aim:

Fully Integrated Interface

- Advanced Programming Interface + Visualized Hardware Configuration + Simplified Network Configuration + Data Tracer & Logger + Motion programming



- **Advanced Programming Interface**

- New Functions: Network configuration, hardware configuration and PLC card.

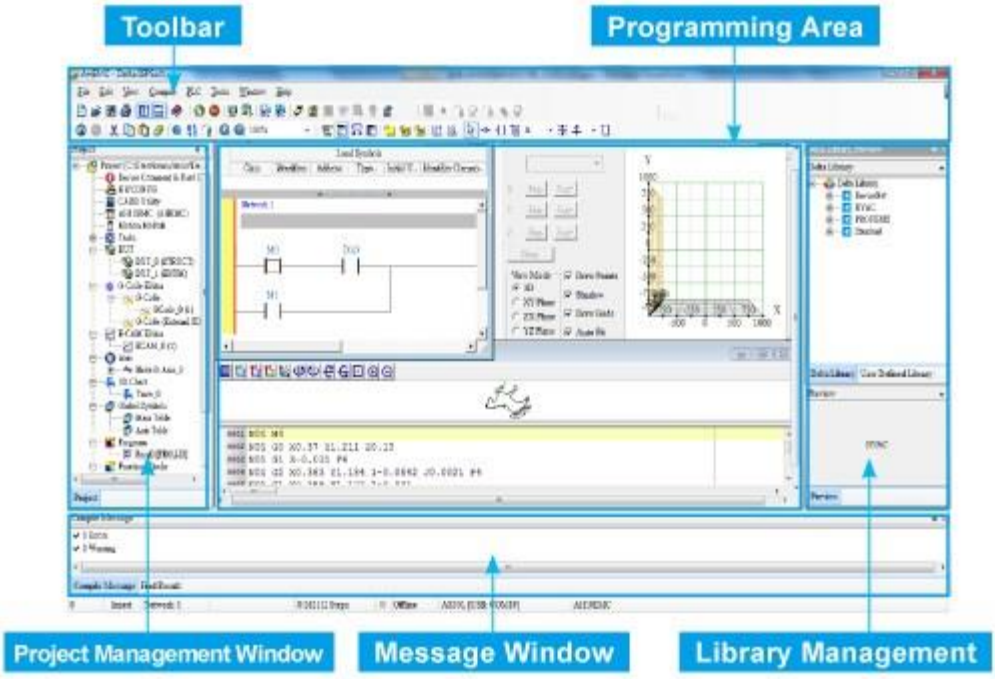
- Supports 5 programming languages (LD / FBD / SFC / IL / ST)

- Function Blocks: Symbols can be introduced in call-by-value or call-by-reference types. Function blocks can be called in function block for up to 32 levels.

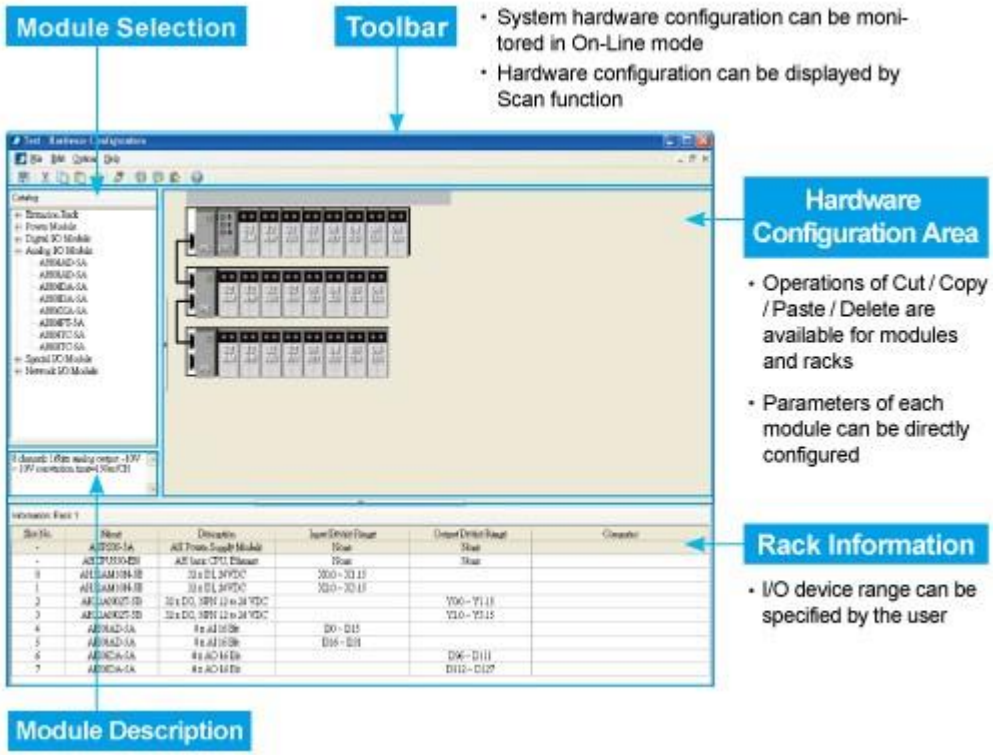
- Monitor Table: It can be stored and managed separately. Multiple monitor tables can be stored in a single project.

- User Library: Users can design frequently used instructions for specific applications in different industries.

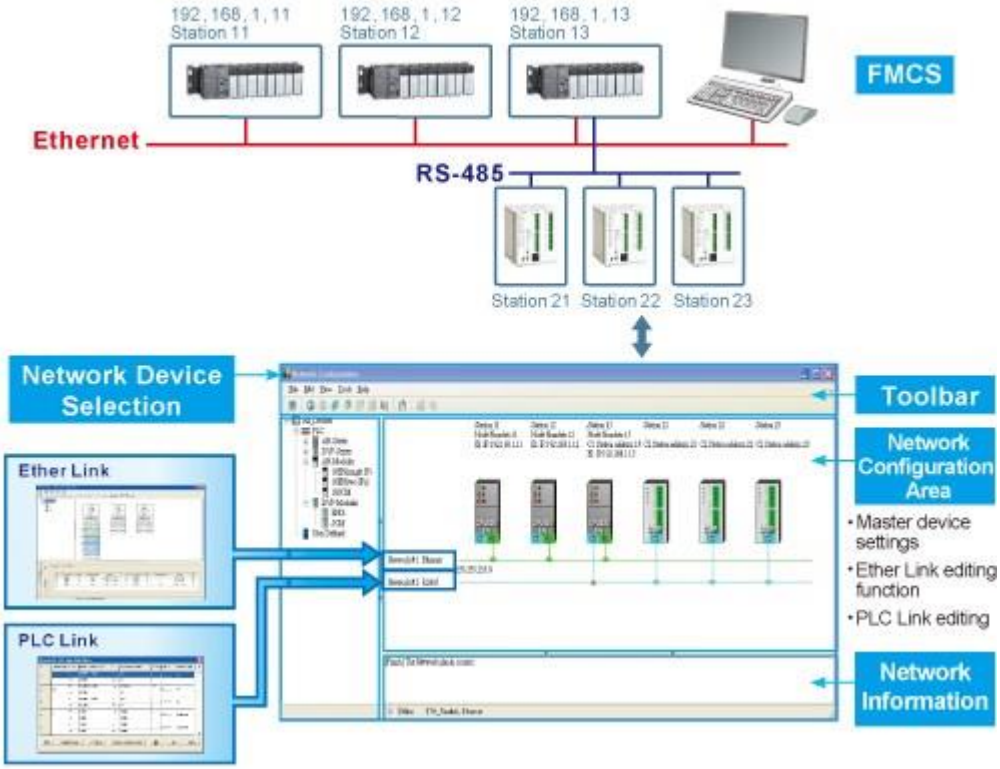
- Task: Supports cyclic, I/O interrupt, timer interrupt, external interrupt, and more. Software will provide usable tasks for different CPUs.



- **Visualized Hardware Configuration**



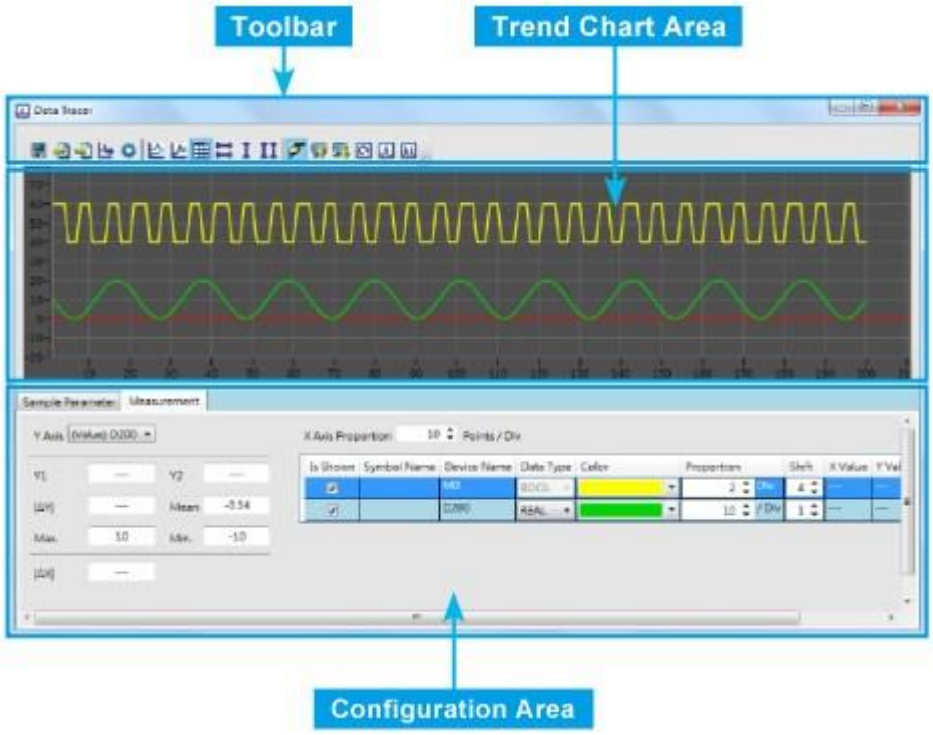
- **Simplified Network Configuration**



- **Convenient Wizards**

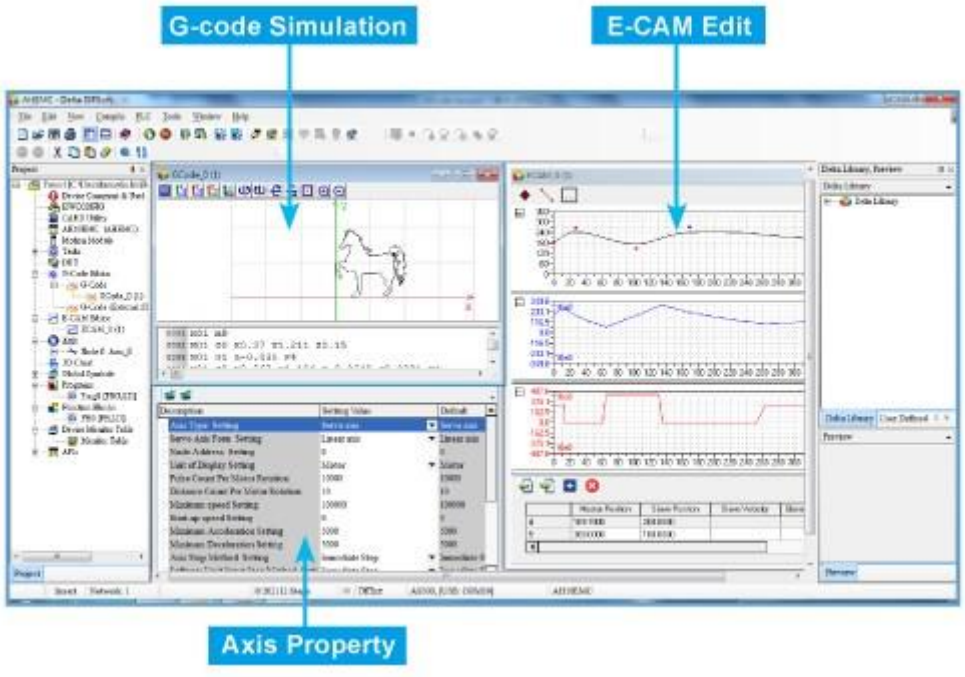
- Data Tracer: Provides high speed data log ability and the interval could be 1 CPU scan. Users can easily
analyze the program logic with this function.

- Data Logger: Provides big data log ability and adjustable intervals. Users can log critical system data and then
analyze the system operation status.



- **Integrated Motion Programming**

ISPSoft V3 integrates both logic and motion programming into one software, and provides PLCopen® function blocks, as well as Delta's function blocks with abundant applications for



customers

Applications

Programming, hardware configuration, network configuration and project management for Delta DVP, AS, AH series PLC.
Specifications

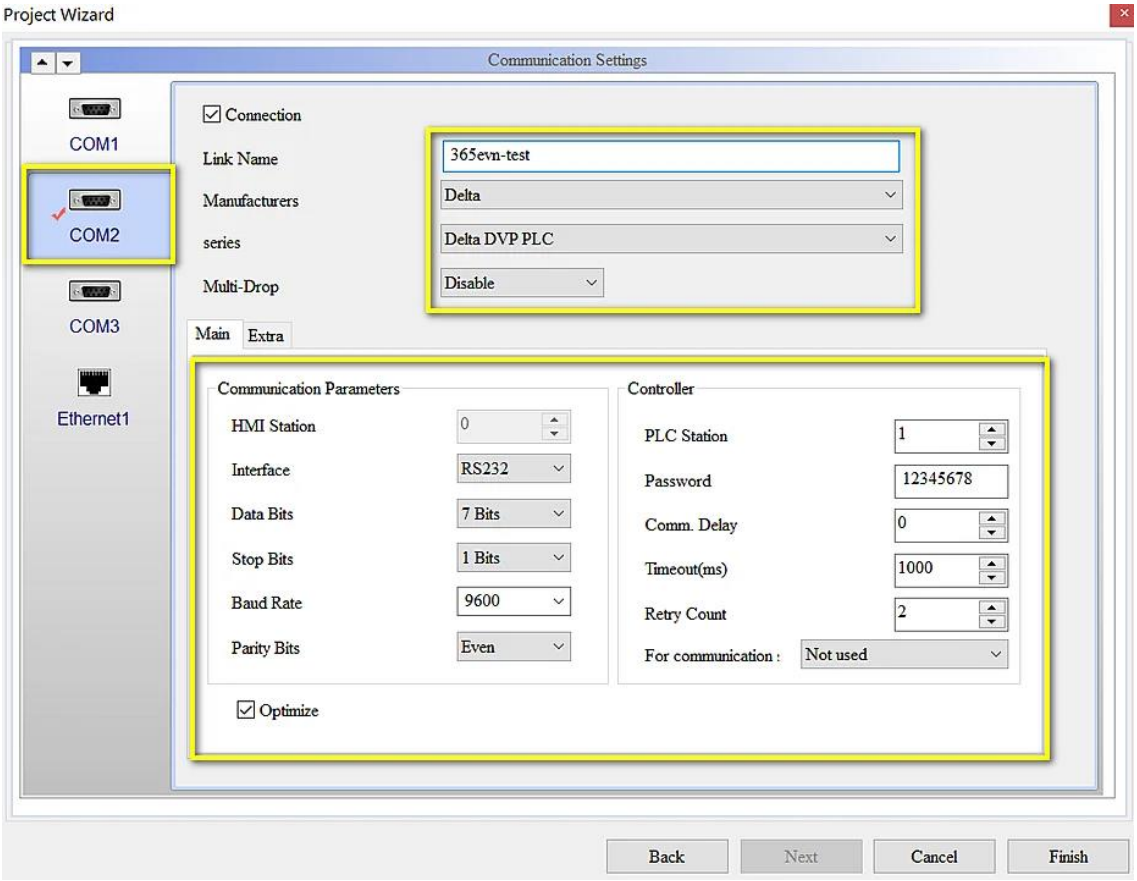
1. Supported Operation System: Windows® XP / Vista / 7 (32-bit / 64-bit) / 8 / 10 (64-bit)

Create a DOPSoft Tag Table – What's you need?

- The DOPSof v4.00 software for *Delta HMI Programming*. You can download DOPSoft v4.00.11 at the download center.
- A "*.csv" file is exported from the ISPSoft symbol table (or WPLSoft). Read more: Manage variables on ISPSoft 3.14

Step 1: Create a project on DOPSoft 4.00

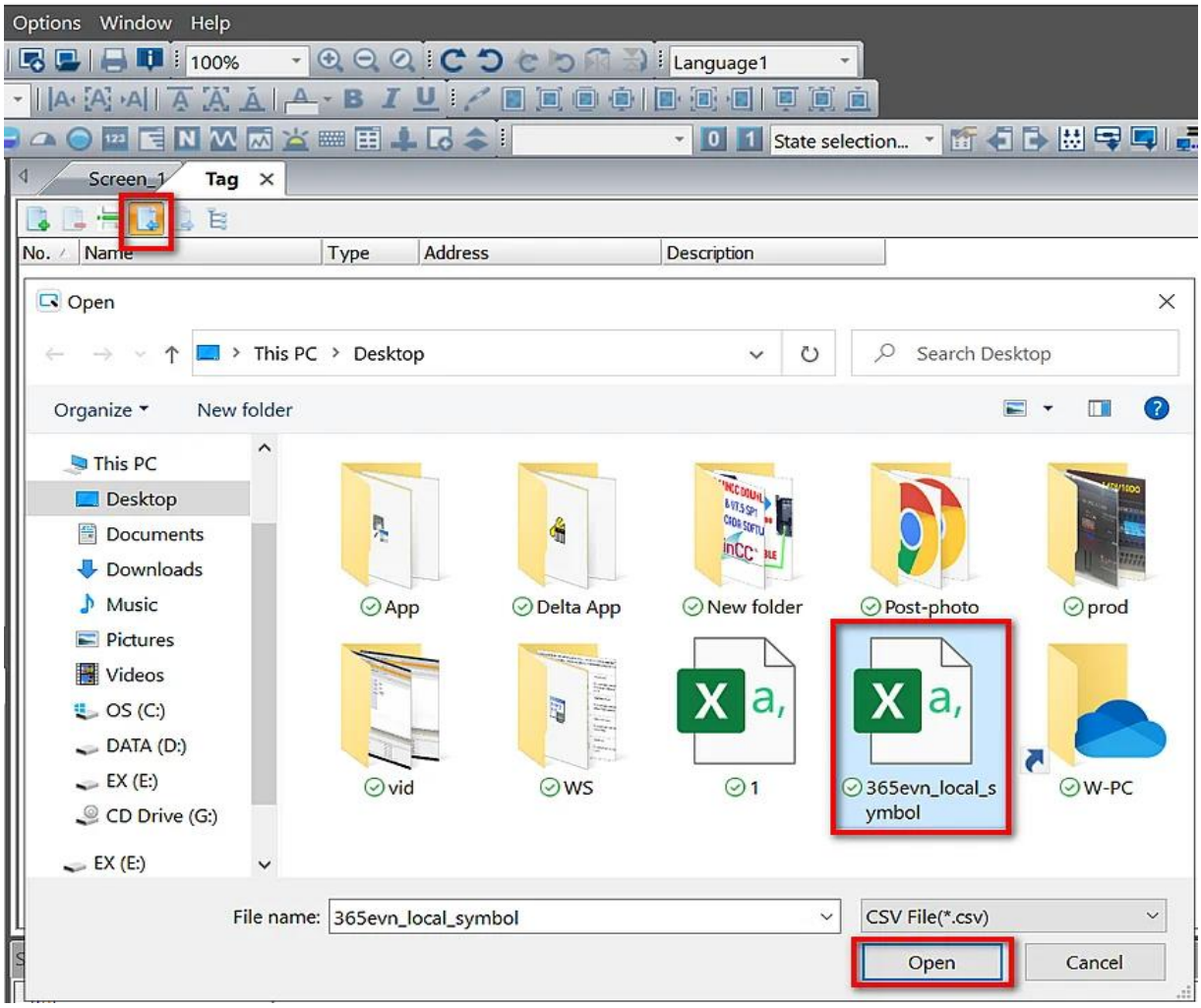
- Open the DOPSoft 4.00, select your device, and set the parameters for communication to PLC.



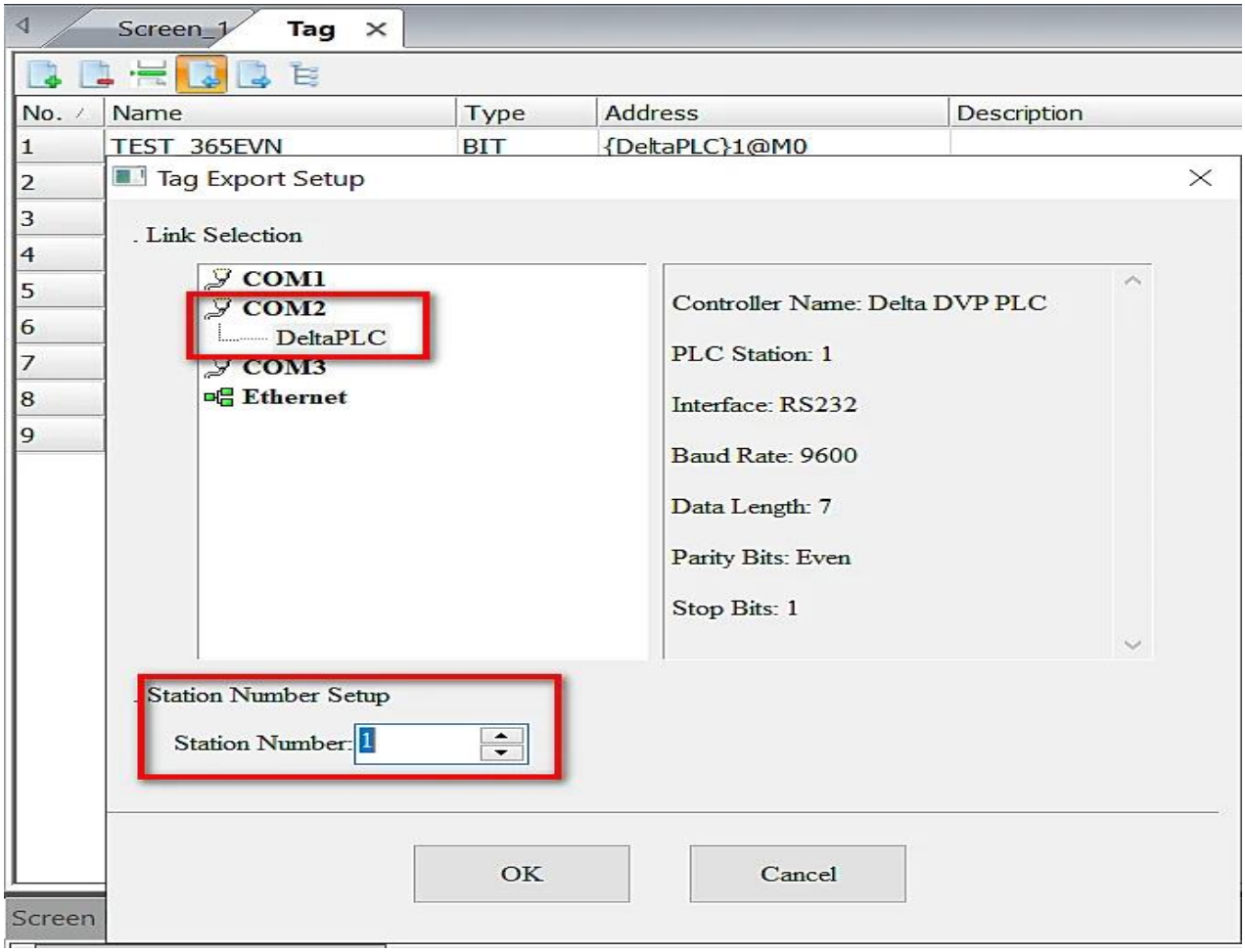
- lick "Finish" and open the editing windows.

Step 2: Create a tag table

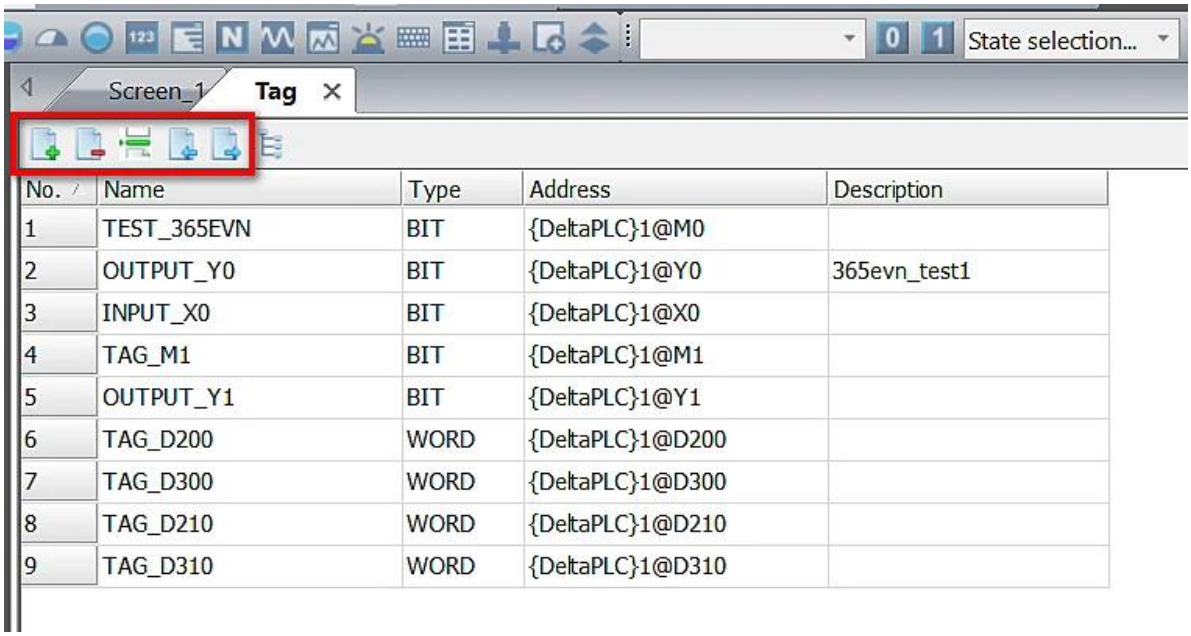
- Click “Options” -> “Tag Table” -> “Import label” and browse to the “*.csv” file which we exported from the ISPSoft symbol table. And then open it.



- Select the communication port (COM 1,2, 3 or ethernet) used for connecting to the Delta PLC and set the station number. The default station number is 0, but you can set the number as you like. Press “OK” to import the symbol table to DOPSoft 4.00.



- After pressing “OK”, the symbol Table of ISPSoft 3.14 (or WPLSoft) is imported to the DOPSoft tag table. The imported symbols are all recognized by *DOPSoft symbols* as upper case letters. For example, the lower case symbols of “tag_m1” edited in ISPSoft (or WPL) are recognized as upper case symbols of “TAG_M1”.



- On the Delta Dopsoft tag table, you can click the icon:

- Add label: to add a new tag data entry.
- Insert label: When you press it, the new data entry is inserted above the row of the selected row.
- Delete label: When you select a row of data entry, press it to delete the selected row.
- Export label: You can save the edited tag on the DOPSoft tag table as a "*.CSV" file, and use Microsoft Excel to edit the contents.

Conclusion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

Title: MySQL database on Raspberry Pi

Aim: To install MySQL database on Raspberry Pi and perform basic SQL queries.

Theory:

MySQL is a real-time relational database management system that organizes data into one or more data tables. The data stored in the tables may be related to each other, thus, ensuring an organized data structure. This database management system is mostly used in creating web servers or accessing another server and is an ideal choice for both small and large-scale applications. It can handle the most expensiveand powerful database packages and is simple to use on any system.

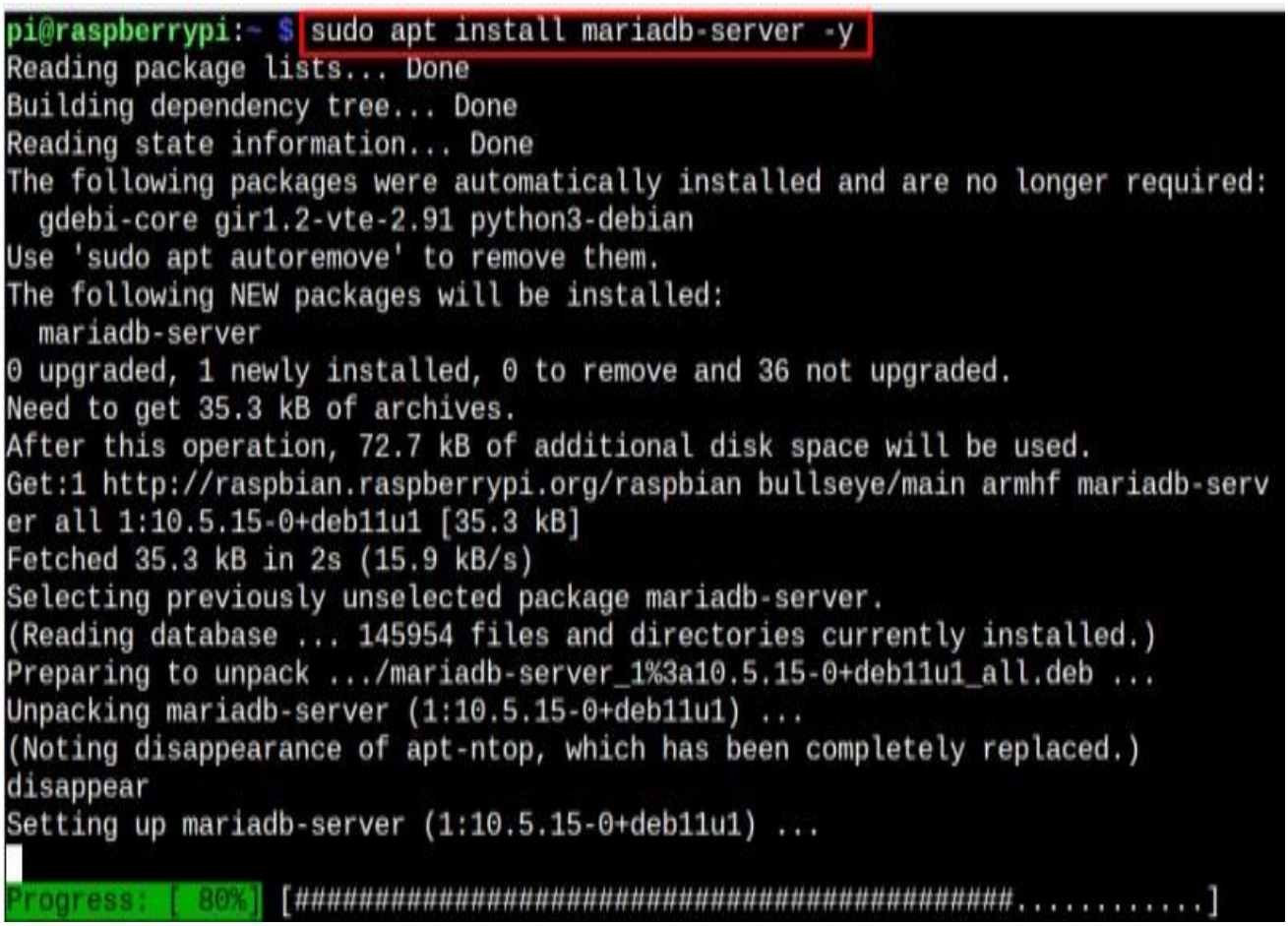
Procedure:

1) How to Setup MySQL Database on Raspberry Pi

The MariaDB server is introduced by MySQL developers as a drop-in replacement for the MySQL server.Therefore, the MySQL server is no longer available in the Raspberry Pi repository.

1.1) To install and setup MySQL (MariaDB) database, follow the below-mentioned steps:

```
$ sudo apt install mariadb-server -y
```



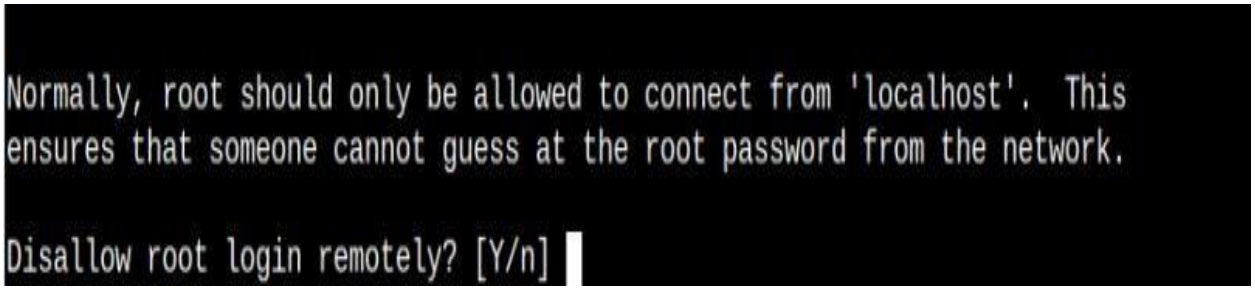
1.2) After completing the installation, you must secure your **MySQL** database from the following command:$ sudo mysql_secure_installation

1.3) Enter your system password to configure MySQL database setup. Then choose the default option “n”to stay with the root account instead of switching to unix_socket.

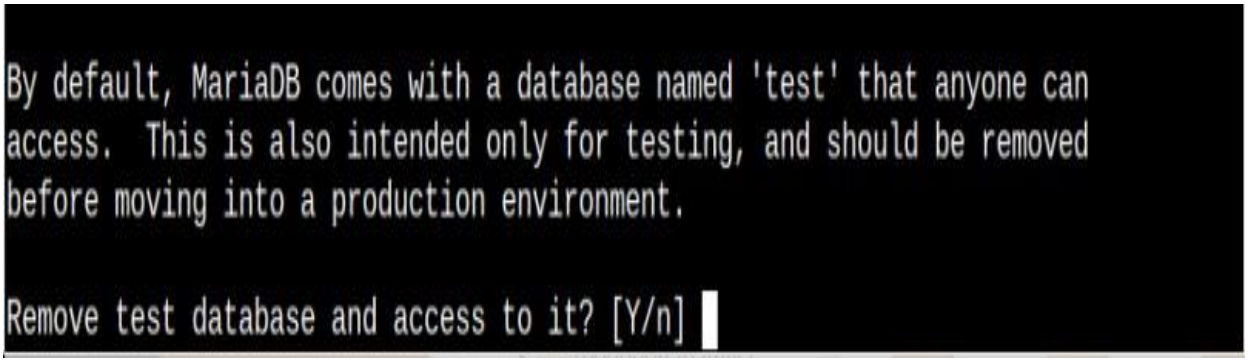
1.4) Enter “n” to stick with the default root password.

1.5) Remove the anonymous users by entering “Y”.

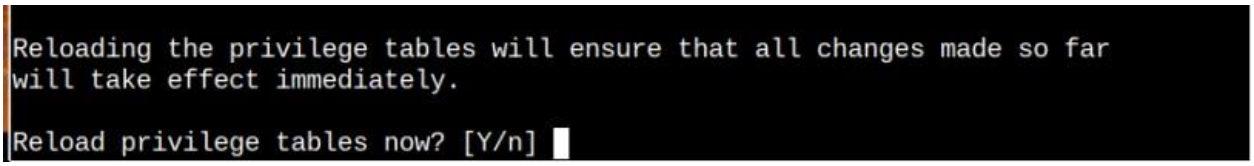
You can disable the root login remotely or enable it according to your choice. If write "n".



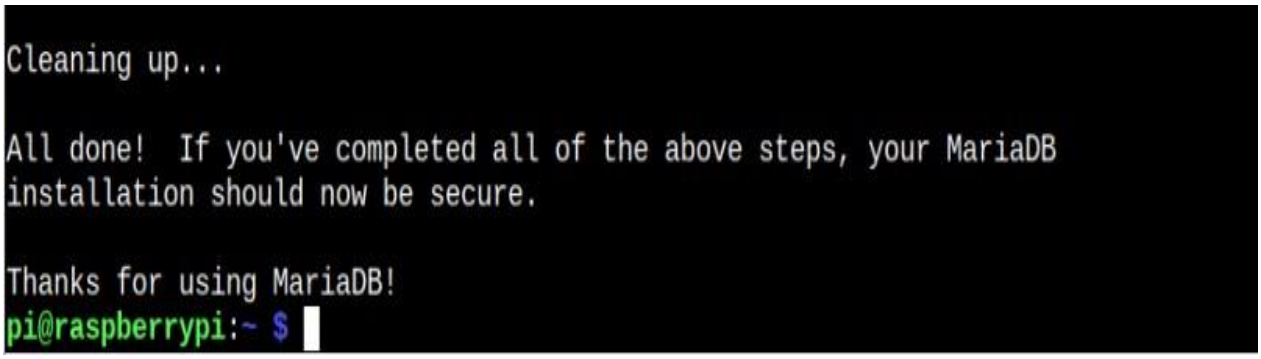
1.6) Enter "Y" to remove the test database or you can reply with "n" too.



1.7) Reload the privilege tables by entering **"Y"** to make the changes.



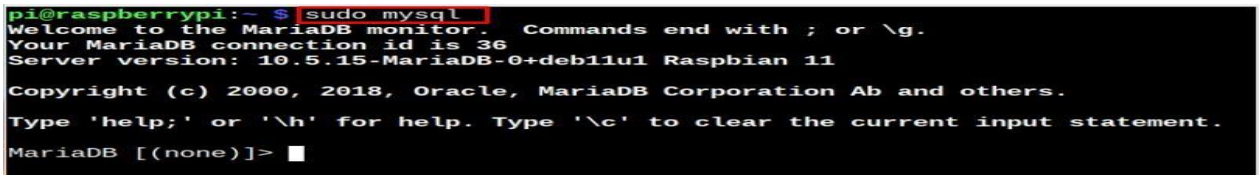
This completes the MySQL database configuration.



2) Create a Database Using MySQL

2.1) Now, to start creating your first database through MySQL, first, execute the following command tolog in to MySQL.

$ sudo mysql



2.2) Then create a database using the following syntax:CREATE DATABASE <database_name>;
For example, the name of the database is "sqldatabase":CREATE DATABASE sqldatabase;

```
MariaDB [(none)]> CREATE DATABASE sqldatabase;
Query OK, 1 row affected (0.001 sec)

MariaDB [(none)]>
```

2.3) Create a username and password for your database and then grant all privileges to your databaseusing the following syntax:

GRANT ALL PRIVILEGES ON <database_name>.* TO '<username>'@'localhost' IDENTIFIED BY '<password>';

```
MariaDB [(none)]> GRANT ALL PRIVILEGES ON sqldatabase.* TO 'linuxhint'@'localho
st' IDENTIFIED BY '[redacted]';
Query OK, 0 rows affected (0.006 sec)

MariaDB [(none)]>
```

2.4) Replace the <username> and <password> in the above command.

Finalize the changes by flushing the privilege table through the following command:
FLUSH PRIVILEGES;

```
MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.003 sec)
```

2.5) Exit to the database through the “exit” command.

```
MariaDB [(none)]> exit
Bye
pi@raspberrypi:~ $
```

This completes the MySQL database creation on the Raspberry Pi
system.
Result & Conclusion

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

It is the most popular, open-source **Continuous Integration / Continuous Delivery (CI/CD)** tool written in Java with built in plugins for almost any combination of languages and source code repositories using pipelines. It builds and tests our software projects that make it easier for developers to incorporate improvements to the project and make it easier for users to get fresh construction.

What is Continuous Integration and Continuous delivery?

Continuous Integration (CI): Continuous Integration (CI) is a practice of software development, where developers often integrate their work with the Integration branch of the project and create a build.

Continuous integration is a major step in the DevOps lifecycle, it makes the development testing and deployment of applications easier and faster. There are a number of continuous integration tools like below

- Bamboo
- Apache
- Travis CI
- Jenkins
- Buddy
- TeamCity

Among them **Jenkins** is the most popular CI tool.

Smt. Indira Gandhi College of Engineering, Navi Mumbai

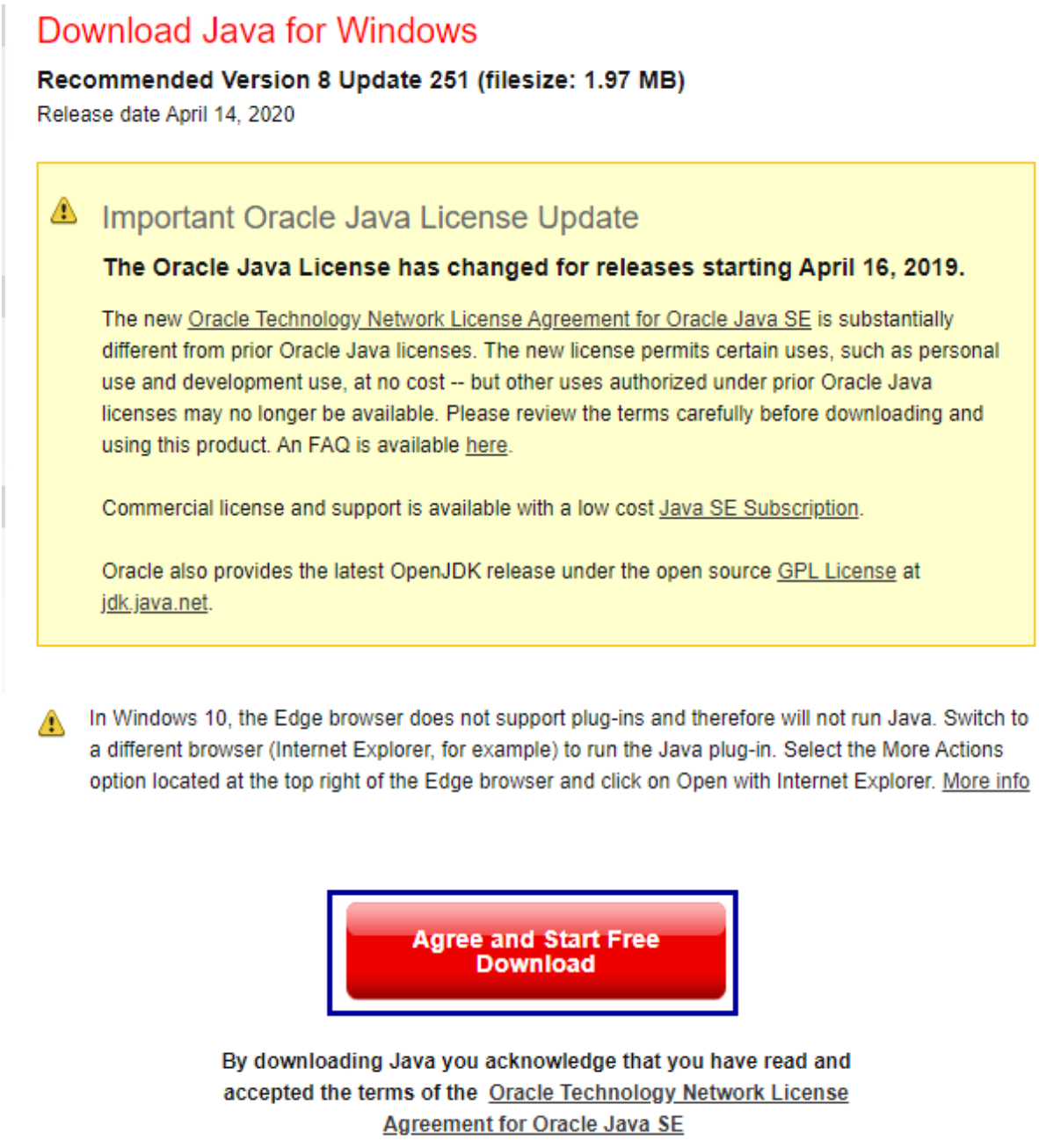
Continuous Delivery (CD): Continuous Delivery (CD) is an extension of continuous integration to ensure you can easily and sustainably bring new updates to your customers or software.

---

1. Installing Jenkins:

Firstly before installing Jenkins we need to download Java, as it is a Java based program. If you have already JAVA make sure by typing "JAVA" in command prompt and skip here.
To install Java click here

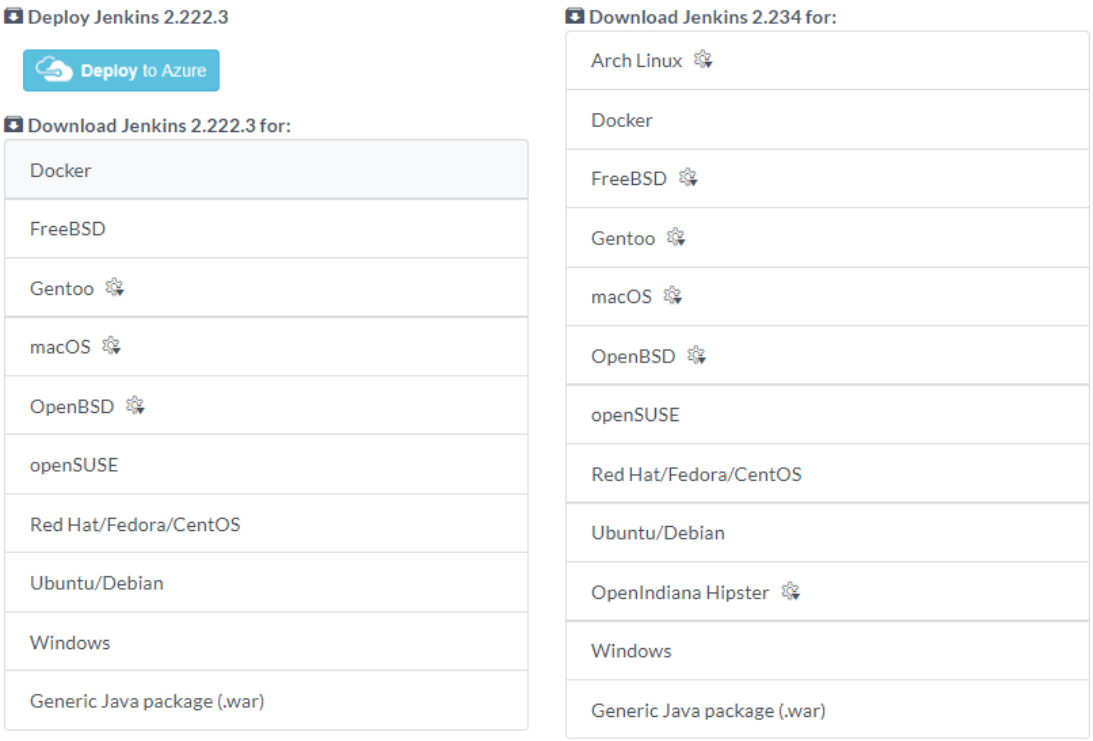
Official Website of Java Download
Click "Java Download"



Click "Agree and Start Free Download". After downloading, install the Java software.

Smt. Indira Gandhi College of Engineering, Navi Mumbai

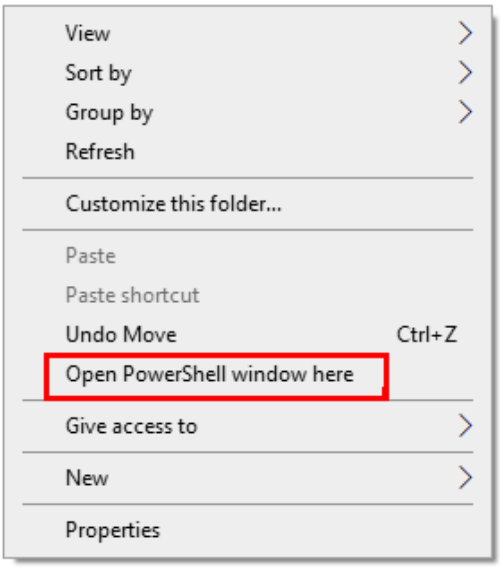
Downloading Jenkins:



Here is the official website of Jenkins [Download]

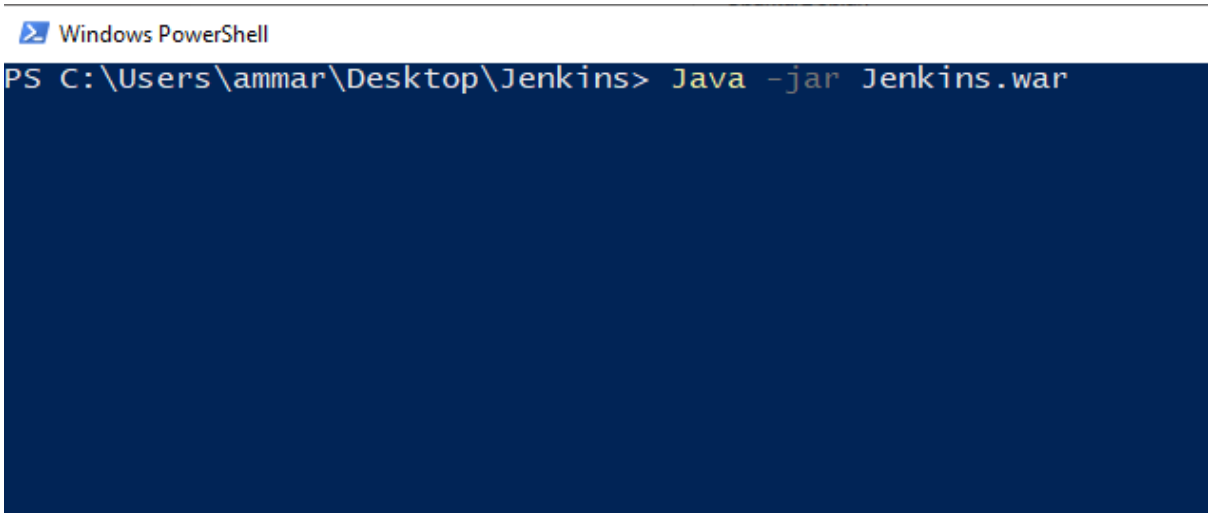
Click the "**Generic Java package(.war)**", this is best to install on any operating system

Once you downloaded the file, go to the folder where you downloaded the file and "**Hold Shift + Right Click on Mouse**"

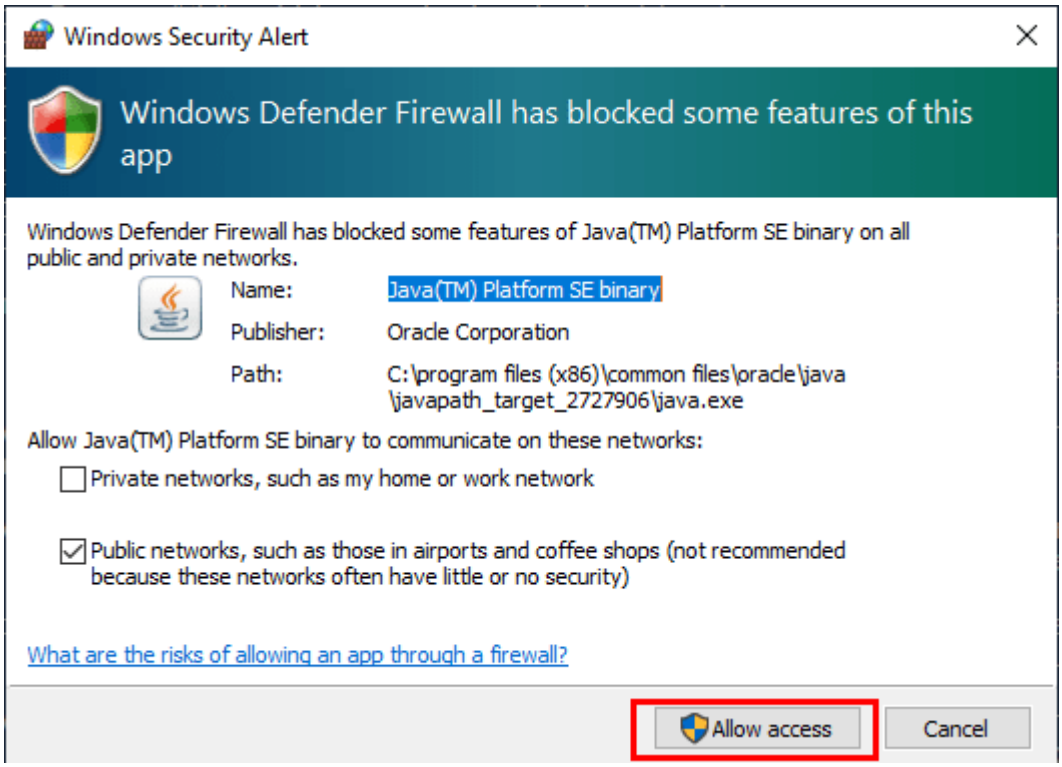


In this, click the option "**Open PowerShell window here**".

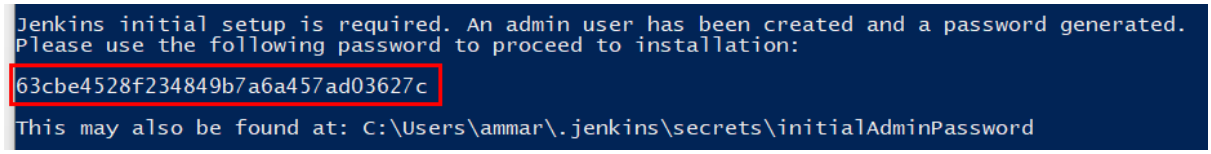
Once you clicked it, it will display the Windows PowerShell window. Now copy the code **Java –jar Jenkins.war** to the Power Shell



Now it will ask access for the software, click Allow access and give access.



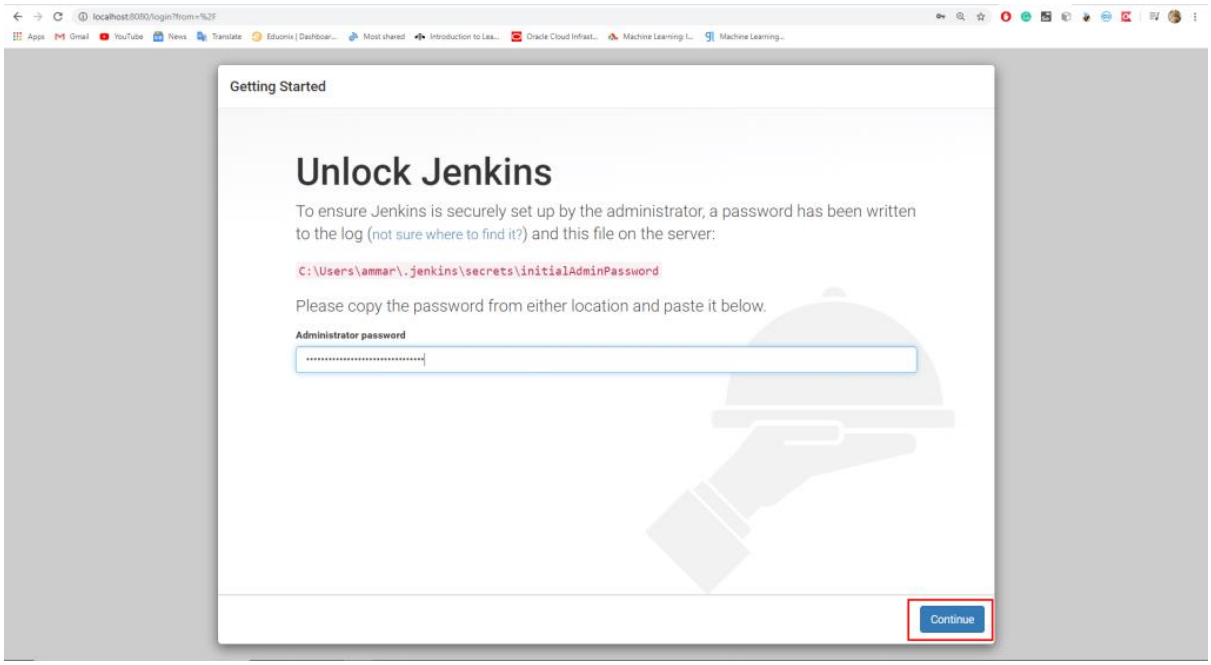
After clicking allow access, the powershell will install Jenkins to your system



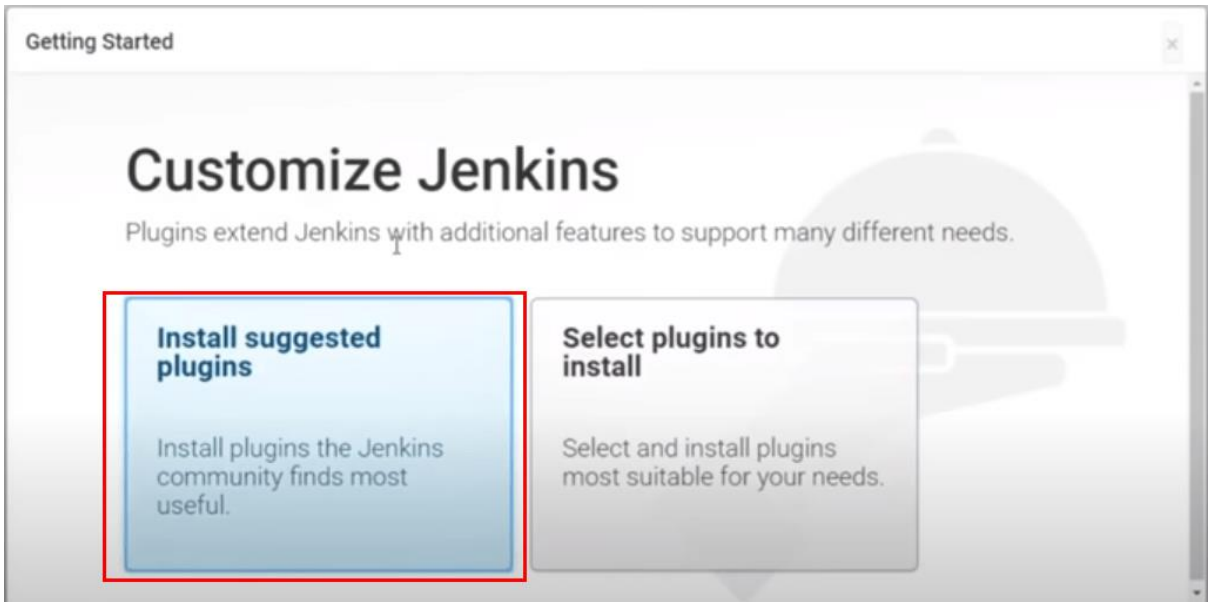
Once installed, copy the generated password from powershell like the above.

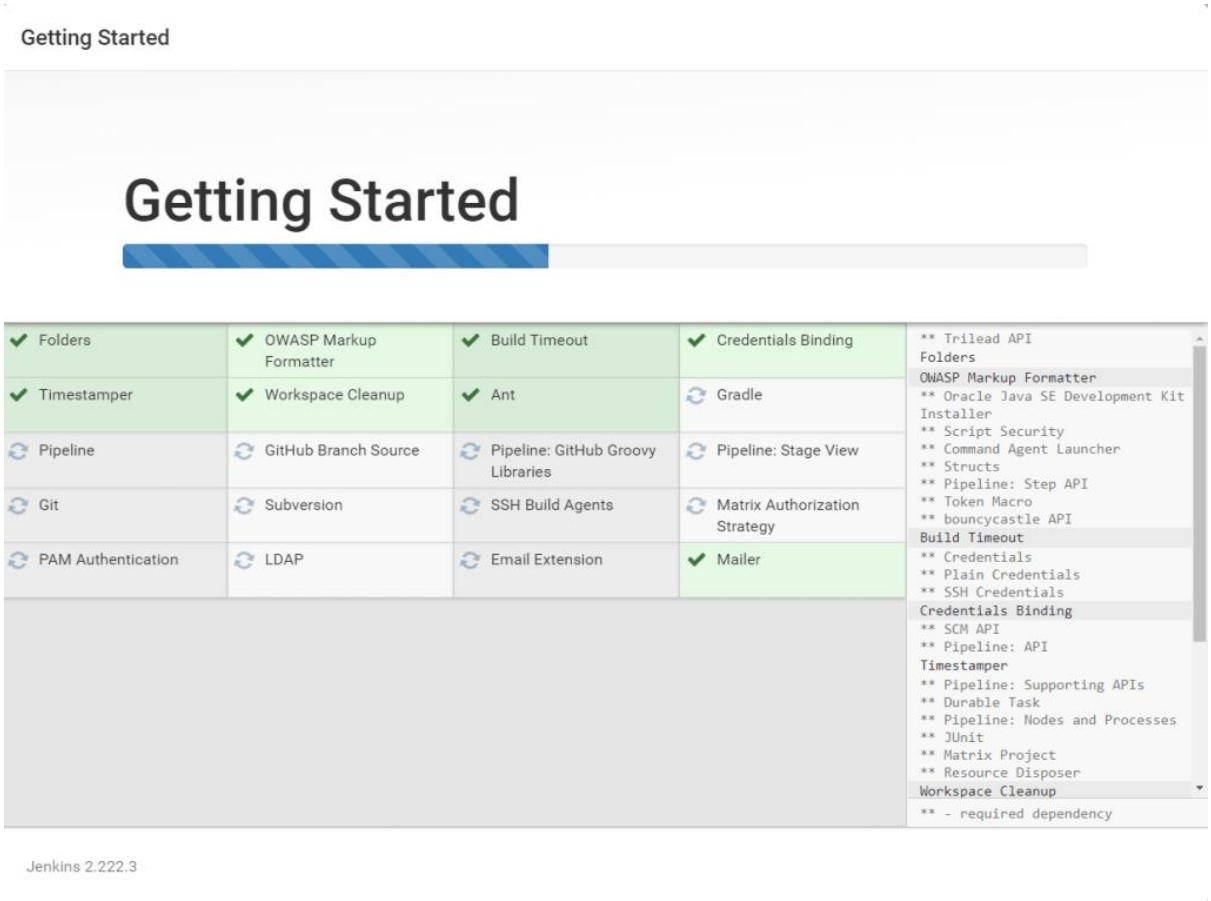
Now, Browse here

Unlock Jenkins using the password copied from PowerShell and click “Continue”

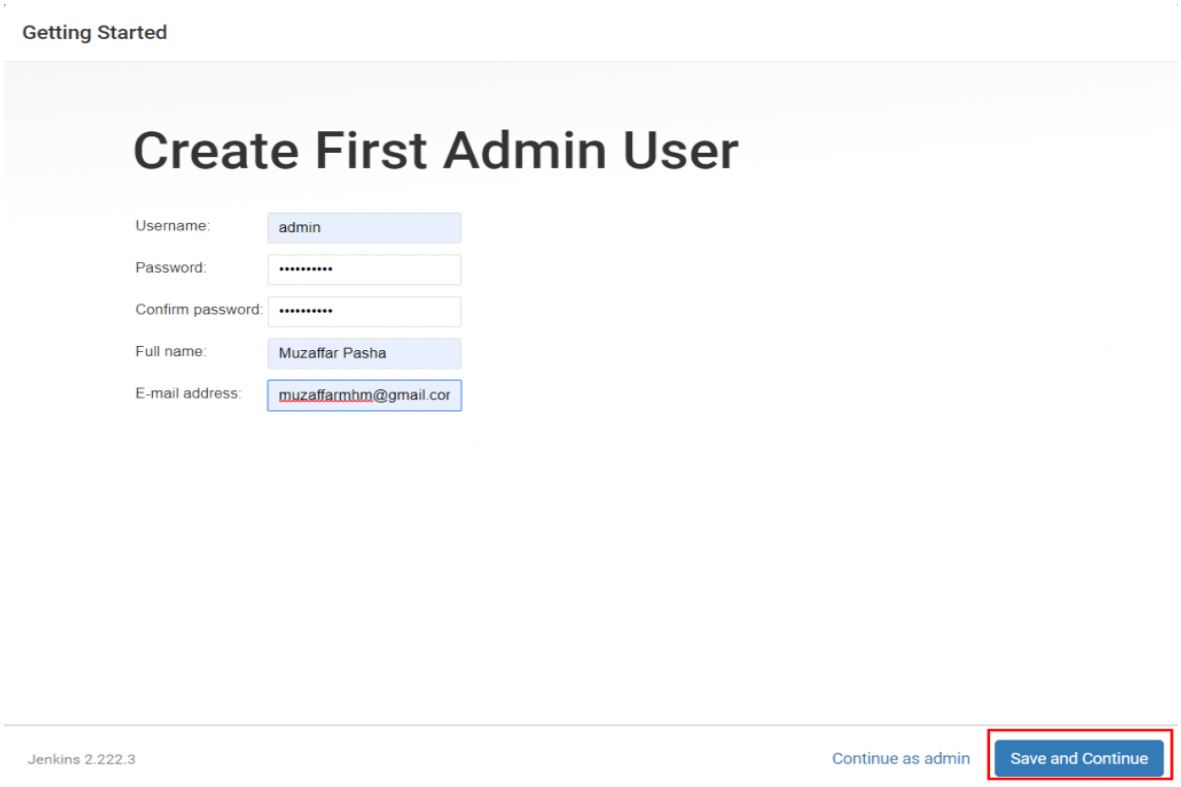


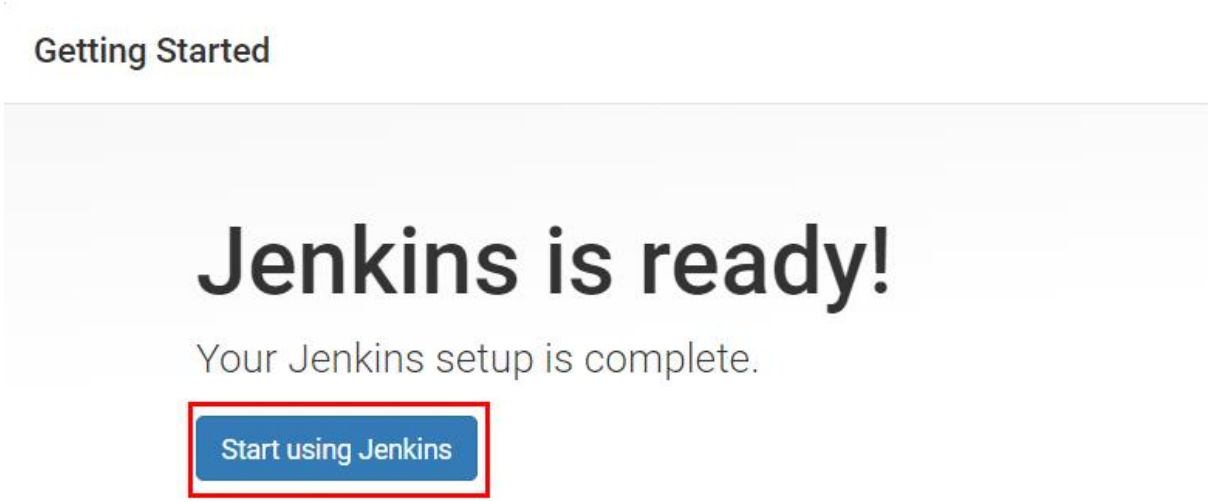
Here select the "**Install suggested plugins**", It will automatically install most popular and most used plugins and wait till the installation is finished.



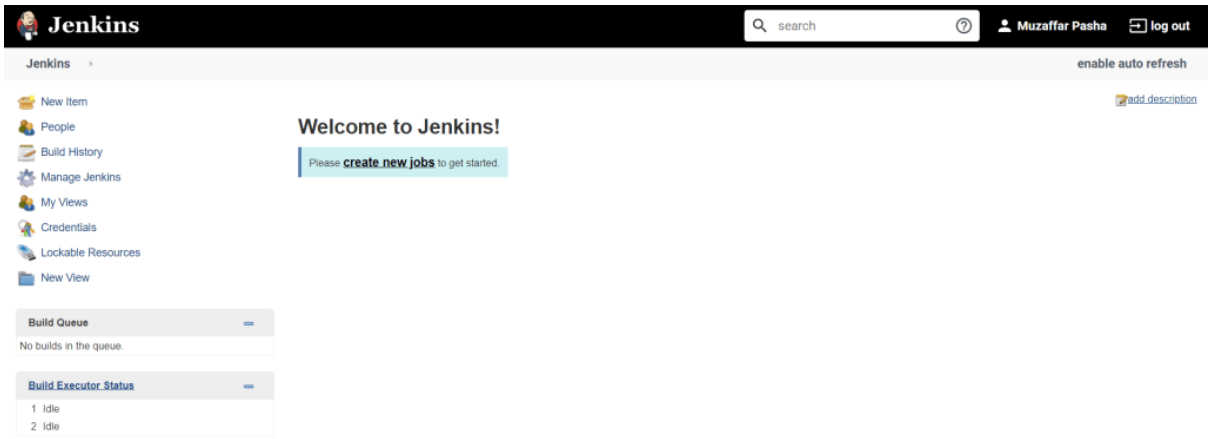


After installation, enter the details asked and click "**Save&Continue**". Make sure to remember username and password





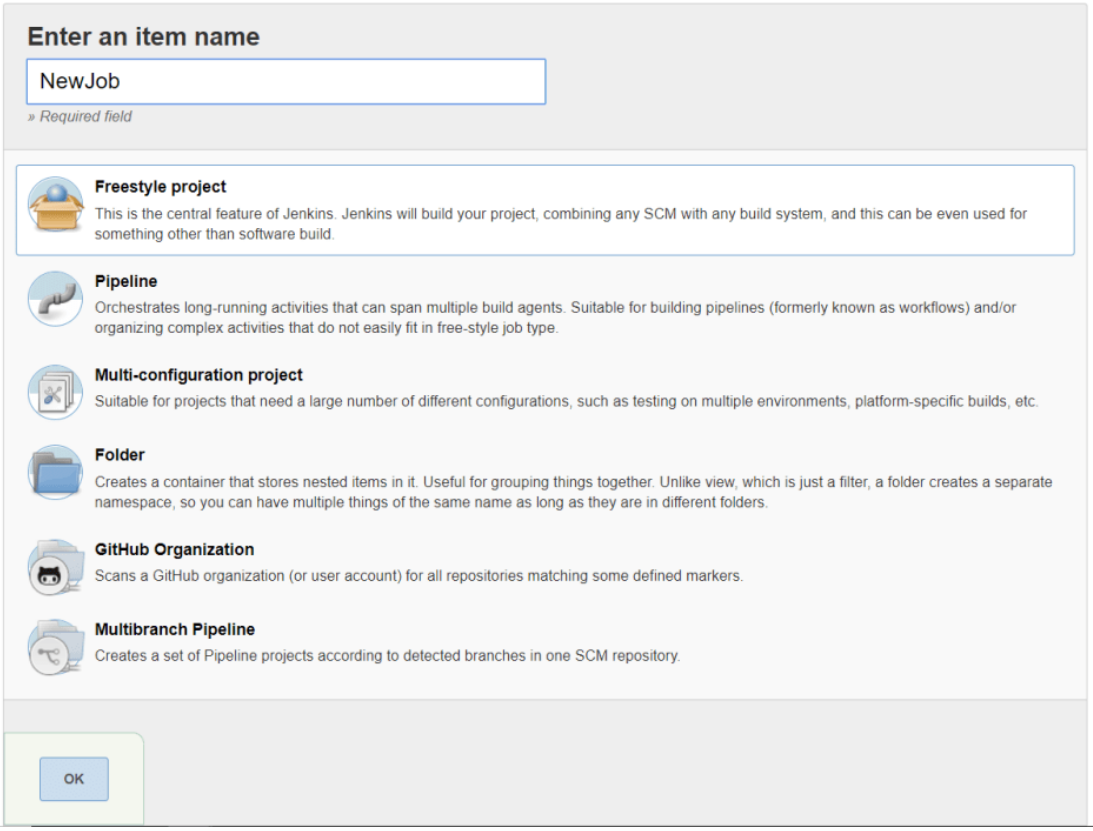
Click "**Start using Jenkins**". Now installation has been successfully completed



---

2. Creating a New Job

Click New Items in the menu to create a new job,



In here type the name of the project and select the type of project. It supports six types of projects.

Freestyle project

Freestyle means unregulated or improvised. A Jenkins freestyle project is a project covering several operations. It can be a build, a running script or even a pipeline.

Pipeline project

The pipeline project is a plugin suite that supports the implementation and integration of continuous delivery pipelines into Jenkins.

Multi Configuration project

The multi Configuration project just like a working template, it begins multiple sub jobs depending on the configuration, aggregates all the test results and shows them in the status page

Folder

A folder project allows users with configuration permission to define properties for a folder that can then be used by any jobs it contains or by any of its subfolders

GitHub Organization

To take the project from the GitHub organisation

Multi branch Pipeline

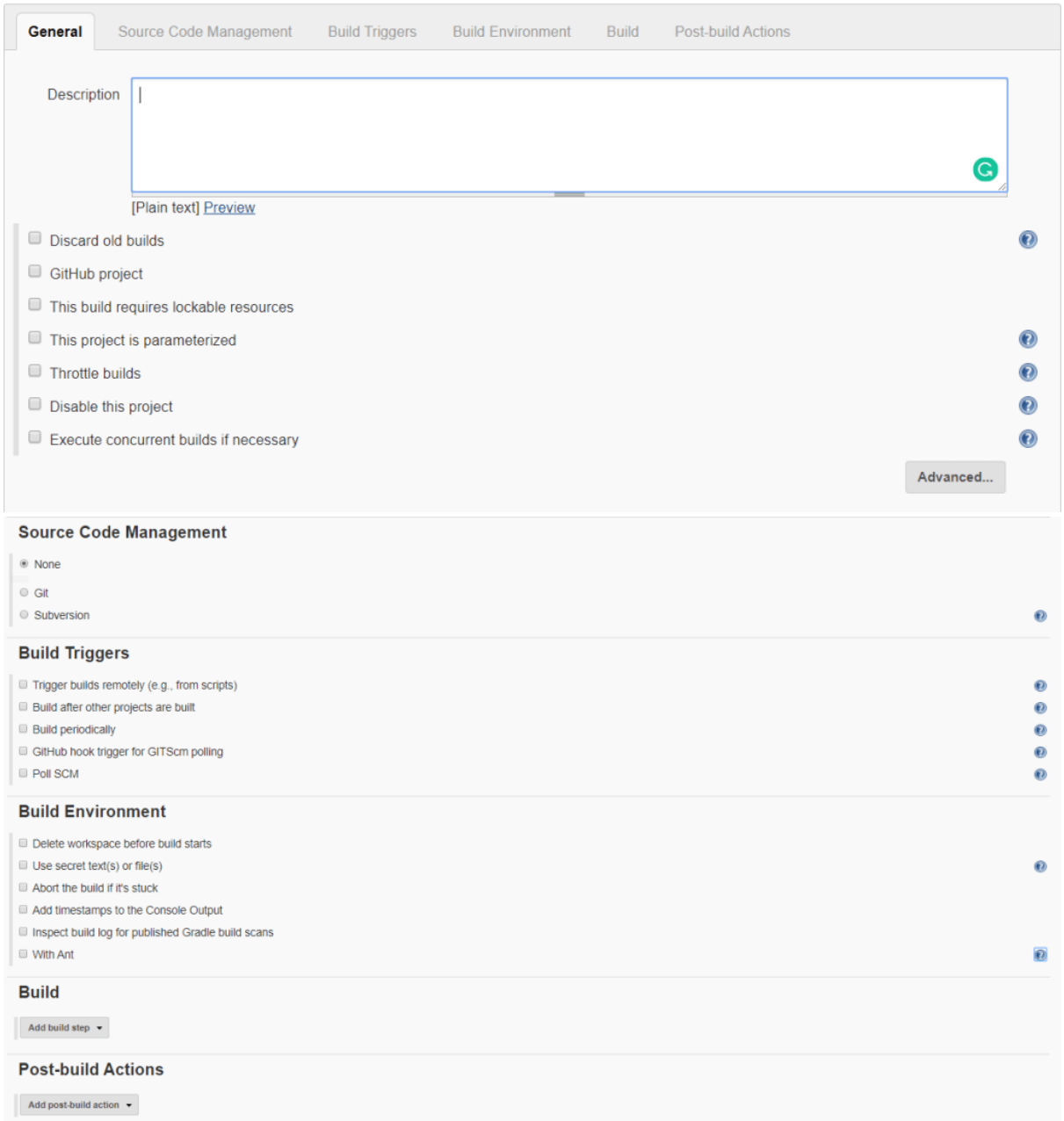
Multi branch pipeline helps you to apply numerous Jenkins files to different divisions of the same project. Jenkins automatically identifies, manages, and executes Pipelines for branches containing a Jenkins file in source control in a Multi branch Pipeline Plan.

---

3. Configuring New Jobs

1. General:

- **Description:** Description of the Project
- **Discard old builds:** This dictates when, if ever, the record builds should be discarded for this project.
- **Github Project:** To choose the project if it's from Github
- **This project is parameterized:** Parameters allow you to prompt users for one or more inputs going into a project. You may have a project, for example, that runs tests on demand by allowing users to upload a binary zip file to be tested.
- **Execute concurrent builds if necessary**: When this option is checked, multiple builds of this project may be executed in parallel.



2. Source Code Management:

- **None:** No source code management
- **Git:** Repositories management like GitHub

3. Build Triggers:

- **Trigger builds remotely (e.g., from scripts)**: Enable this option if you wish to activate new builds by accessing a certain predefined URL
- **Build after other projects are built:** Set up a mechanism so that a new build is planned for this project when any other projects finish building.
- **Build periodically**: Provides a time based like feature to periodically execute this project.

MINUTE	Minutes within the hour (0–59)
HOUR	The hour of the day (0–23)
DOM	The day of the month (1–31)
MONTH	The month (1–12)
DOW	The day of the week (0–7) where 0 and 7 are Sunday.

- **Poll SCM**: Configure Jenkins to poll changes in SCM.

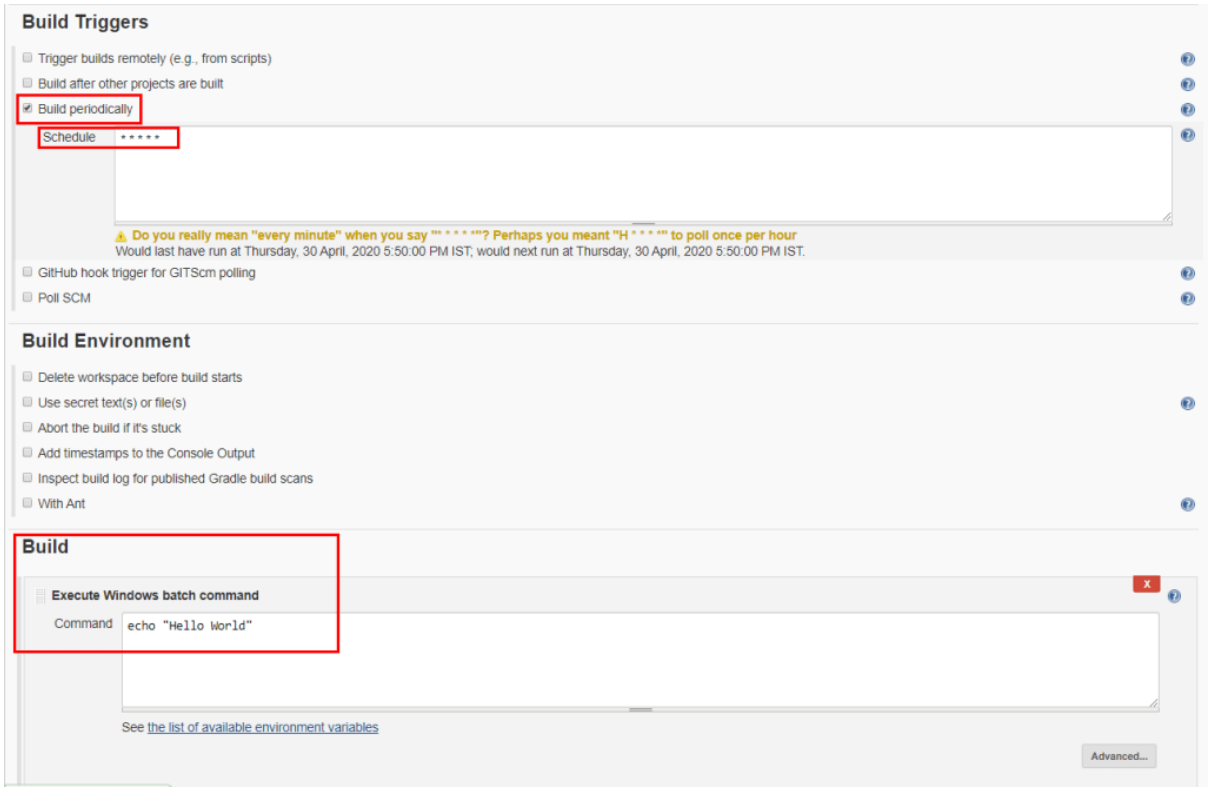
4. Build Environment:

- **Delete workspace before build starts**: Deletes the workspace from it before the build starts.
- **Use secret text(s) or file(s)**: Allows you to take credentials of various kinds and use them from measures in shell building.
- **With Ant**: Prepares an environment for it to run builds using Apache Ant.
- **Build**: To select what type of build.
- **Add post build action:** To add an extra feature to the project.

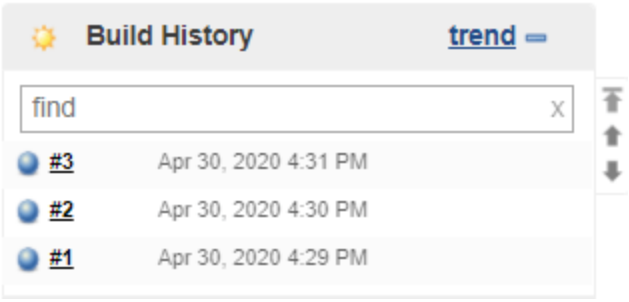
---

4. Checking the builds

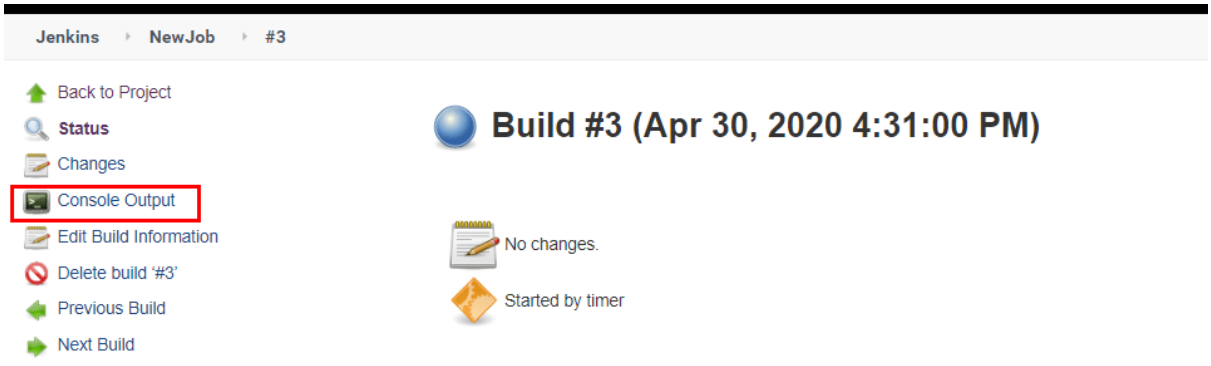
So, I configured a project to run a command every minute.
In the Build Trigger Configuration, I made it to "**Build Periodically**" and used "*** * * * ***" as my period which indicates one minute [Format: MINUTE HOUR DOM MONTH DOW]
In the Build Command, I typed echo "Hello World", to run the build and command "Hello World" every minute.

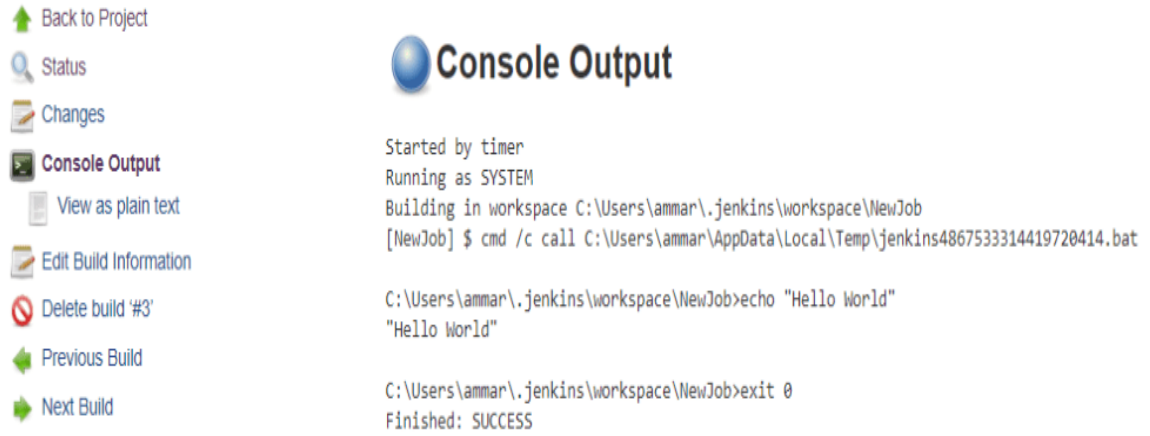


After saving it. It will build the job every minute as shown below screenshot



To check the output select any build, (In my case I selected the 3rd build) and click Console Output

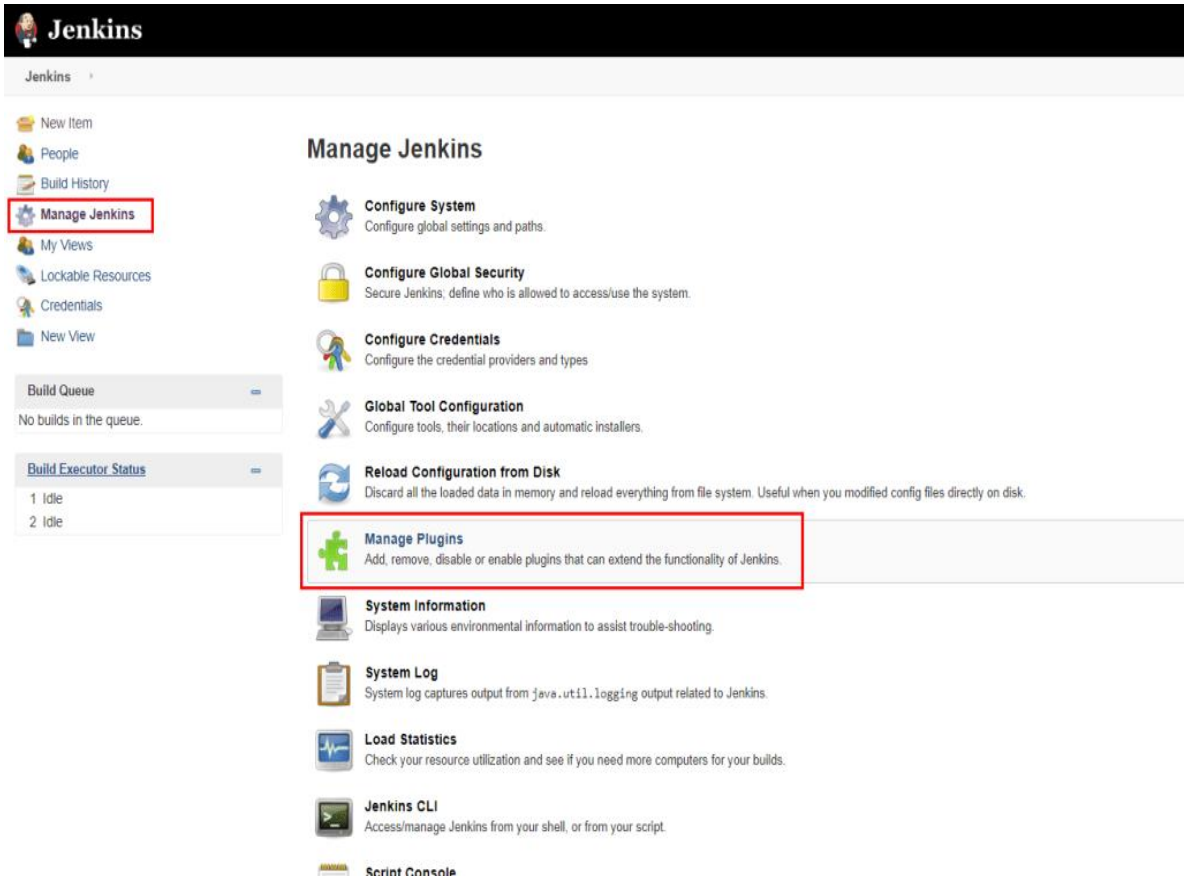




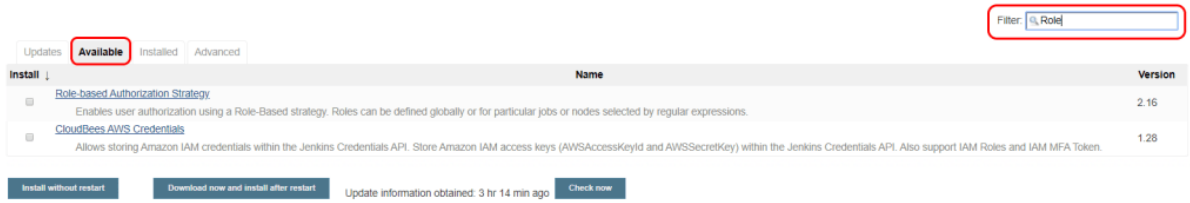
As you can see, it is displaying “Hello World” in the console.
You can also configure your project later using Configure in the menu

5. Adding Plugins

In Jenkins, you can add extra plugins from the software online. To add plugins Go to the main menu, click “**Manage Jenkins”** and select “**Manage plugins**“



Now click the available tab, and search for the plugin you need. I need to install the role based Authorization Strategy plugin I have searched for “**role**“.



After selecting the plugin, click “Download now and install after restart” which is recommended for plugin installation.

Conclusion:

Smt. Indira Gandhi College of Engineering, Navi Mumbai

Department: CSE IOT And Cyber Security Including Blockchain

Subject: IoT Automation Lab

Semester: VIII

Name of Student:

Experiment No:

Aim:

Ansible is a software tool that enables cross-platform automation and orchestration at scale and has become over the years the standard choice among enterprise automation solutions.

It’s mostly addressed to IT operators, administrators & decision-makers helping them to achieve operational excellence across their entire infrastructure ecosystem.

Backed by RedHat and a loyal open source community, it is considered an excellent option for configuration management, infrastructure provisioning, and application deployment use cases.

Its automation opportunities are endless across hybrid clouds, on-prem infrastructure, and IoT and it's an engine that can greatly improve the efficiency and consistency of your IT environments.

Ready to automate everything? Let's go!

How does Ansible work?

Ansible uses the concepts of control and managed nodes. It connects from the **control node**, any machine with Ansible installed, to the **managed nodes** sending commands and instructions to them.

The units of code that Ansible executes on the managed nodes are called **modules**. Each module is invoked by a **task**, and an ordered list of tasks together forms a **playbook.** Users write playbooks with tasks and modules to define the desired state of the system.

The managed machines are represented in a simplistic **inventory** file that groups all the nodes into different categories.

Ansible leverages a very simple language, YAML, to define playbooks in a human-readable data format that is really easy to understand from day one.

Even more, Ansible doesn't require the installation of any extra agents on the managed nodes so it's simple to start using it.

Typically, the only thing a user needs is a terminal to execute Ansible commands and a text editor to define the configuration files.

Benefits of using Ansible

- A free and open-source community project with a huge audience.
- Battle-tested over many years as the preferred tool of IT wizards.
- Easy to start and use from day one, without the need for any special coding skills.
- Simple deployment workflow without any extra agents.

- Includes some sophisticated features around modularity and reusability that come in handy as users become more proficient.
- Extensive and comprehensive official documentation that is complemented by a plethora of online material produced by its community.

To sum up, Ansible is simple yet powerful, agentless, community-powered, predictable, and secure.

Basic Concepts & Terms

Host: A remote machine managed by Ansible.

Group: Several hosts grouped together that share a common attribute.

Inventory: A collection of all the hosts and groups that Ansible manages. Could be a static file in the simple cases or we can pull the inventory from remote sources, such as cloud providers.

Modules: Units of code that Ansible sends to the remote nodes for execution.

Tasks: Units of action that combine a module and its arguments along with some other parameters.

Playbooks: An ordered list of tasks along with its necessary parameters that define a recipe to configure a system.

Roles: Redistributable units of organization that allow users to share automation code easier.

YAML: A popular and simple data format that is very clean and understandable by humans.

How to Install Ansible

To start using Ansible, you will need to install it on a control node, this could be your laptop for example. From this control node, Ansible will connect and manage other machines and orchestrate different tasks.

Installation Requirements

Your control node can be any machine with Python 3.8 or newer, but Windows is not supported.

For the managed nodes, Ansible needs to communicate with them over SSH and SFTP (this can also be switched to SCP via the ansible.cfg file) or WinRM for Windows hosts. The managed nodes also need Python 2 (version 2.6 or later) or Python (version 3.5 or later) and in the case of Windows nodes PowerShell 3.0 or later and at least .NET 4.0 installed.

The exact installation procedure depends on your machine and operating system but the most common way would be to use **pip**.

To install pip, in case it's not already available on your system:

```
$ curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py
$ python get-pip.py --user
```

After pip is installed:

```
$ python -m pip install --user ansible
```

Since there are different ways to install it for every operating system you can also have a look here to find the official suggested way for your environment. Check this guide on installing Ansible on Ubuntu, RHEL, macOS, CentOS, and Windows.

You can test on your terminal if it's successfully installed by running:

```
$ ansible --version
```

Demo requirements

Now that we have Ansible installed, we are going to create our first demo setup. I am going to use my personal laptop as the control node and Vagrant along with VirtualBox to generate locally 2 Ubuntu machines that we will manage with Ansible.

If you wish to follow along, install Vagrant and VirtualBox. You can also find all the files that we are going to be using in this demo in this GitHub repository.

From the top of this GitHub repository execute:

```
$ vagrant up
```

This command will spin up 2 Ubuntu hosts in VirtualBox so that we can use them as our managed hosts in this demo exercise. You can also open VirtualBox to verify the existence of your 2 virtual machines.

Finally, to get the info necessary to build our hosts file run the vagrant ssh-config command:

```
$ vagrant ssh-config
Host host1
  HostName 127.0.0.1
  User vagrant
  Port 2222
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile
/Users/ioannis/Desktop/blog/ansible_intro/.vagrant/machines/host1/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL

Host host2
  HostName 127.0.0.1
  User vagrant
  Port 2200
  UserKnownHostsFile /dev/null
  StrictHostKeyChecking no
  PasswordAuthentication no
  IdentityFile
/Users/ioannis/Desktop/blog/ansible_intro/.vagrant/machines/host2/virtualbox/private_key
  IdentitiesOnly yes
  LogLevel FATAL
```

Ansible Inventory

As previously mentioned, the inventory is the collection of the machines that we would like to manage. Usually, the default location for inventory is /etc/ansible/hosts but we can also define a custom one in any directory.

In the GitHub repository you will see a file named **hosts** that looks like this:

```
host1 ansible_host=127.0.0.1 ansible_user=vagrant ansible_port=2222
ansible_ssh_private_key_file=./.vagrant/machines/host1/virtualbox/private_key


host2 ansible_host=127.0.0.1 ansible_user=vagrant ansible_port=2200
ansible_ssh_private_key_file=./.vagrant/machines/host2/virtualbox/private_key
```

We used the information acquired by Vagrant to populate our hosts file. Currently, our hosts file contains only 2 entries for the hosts that we want to manage.

The **host1** and **host2** are the aliases we used to name them.

We specified some variables, such as the **host**, **user**, and **SSH connection parameters** necessary to connect to our managed nodes. Here's a full list of inventory parameters that we can configure per host.

We will keep this inventory simple for the time being, but check out this guide to explore other inventory options such as creating host groups, adding ranges of hosts, and grouping variables.

For example, we can define groups of hosts like this:

```
[webservers]
webserver1.example.com
webserver2.example.com
webserver3.example.com
192.0.6.45

[databases]
database1.example.com
```

```
database2.example.com
```

In the above example, we defined two groups of hosts, **webservers** and **databases**.

Two special groups always exist by default; **all** that includes every host and **ungrouped** that includes all the hosts that aren't in any groups.

💡 **You might also like:**

- 10 Ways to Improve Your Infrastructure as Code
- Common Infrastructure Challenges and How to Solve Them
- 44 Ansible Best Practices to Follow

Ansible ad hoc Commands

Using ad hoc commands is a quick way to run a single task on one or more managed nodes.

Some examples of valid use cases are rebooting servers, copying files, checking connection status, managing packages, gathering facts, etc.

The pattern for ad hoc commands looks like this:

```
$ ansible [host-pattern] -m [module] -a "[module options]"
```

host-pattern: the managed hosts to run against

-m: the module to run

-a: the list of arguments required by the module

This is a good opportunity to use our first Ansible ad hoc command and at the same time validate that our inventory is configured as expected. Let's go ahead and execute a ping command against all our hosts:

```
$ ansible -i hosts all -m ping

host1 | SUCCESS => {
```

```
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}

host2 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

Nice, seems like we can successfully ping the 2 hosts that we have defined in our hosts file.

Next, run a live command only to the host2 node by using the **–limit** flag

```
$ ansible all -i hosts --limit host2 -a "/bin/echo hello"

host2 | CHANGED | rc=0 >>
hello
```

Another example would be to copy a file to our remote nodes:

```
$ ansible all -i hosts -m ansible.builtin.copy -a "src=./hosts dest=/tmp/hosts"
host1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "0bd8efb12ac716fdddf6dd8feedb750a7fc8c370",
    "dest": "/tmp/hosts",
    "gid": 1000,
    "group": "vagrant",
    "md5sum": "f425732ff83fe576b00f37dd63d94544",
```

```
    "mode": "0664",
    "owner": "vagrant",
    "size": 291,
    "src": "/home/vagrant/.ansible/tmp/ansible-tmp-1645033325.338188-14454-
35489936437202/source",
    "state": "file",
    "uid": 1000
}
host2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": true,
    "checksum": "0bd8efb12ac716fdddf6dd8feedb750a7fc8c370",
    "dest": "/tmp/hosts",
    "gid": 1000,
    "group": "vagrant",
    "md5sum": "f425732ff83fe576b00f37dd63d94544",
    "mode": "0664",
    "owner": "vagrant",
    "size": 291,
    "src": "/home/vagrant/.ansible/tmp/ansible-tmp-1645033325.356349-14456-
242443746329447/source",
    "state": "file",
    "uid": 1000
}
```

Perfect, the files have been copied! We can verify this ourselves by “sshing” into one of the managed nodes:

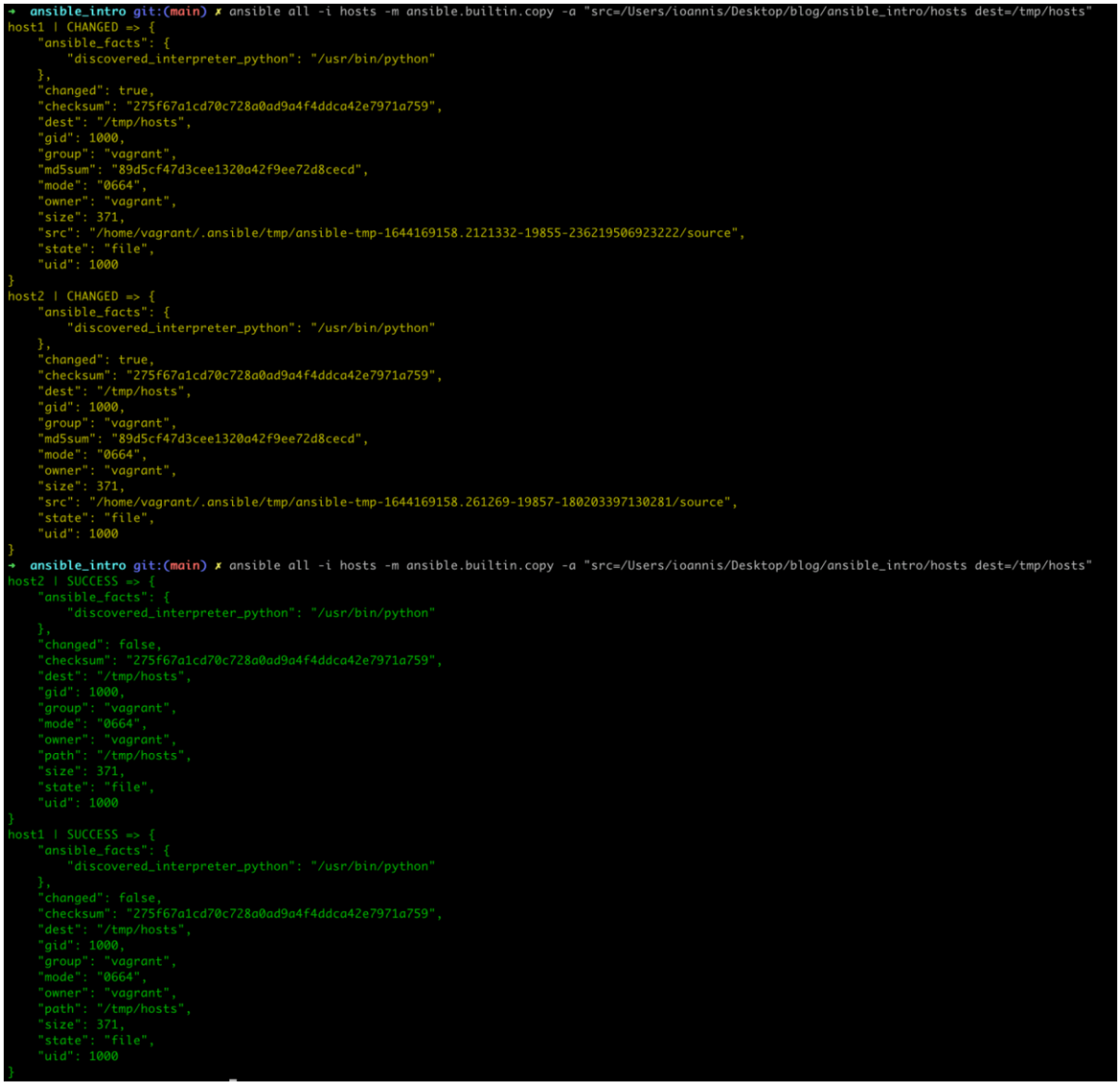
```
$ vagrant ssh host1
vagrant@vagrant:~$ cat /tmp/hosts


host1 ansible_host=127.0.0.1 ansible_user=vagrant ansible_port=2222
ansible_ssh_private_key_file=./.vagrant/machines/host1/virtualbox/private_key
host2 ansible_host=127.0.0.1 ansible_user=vagrant ansible_port=2200
ansible_ssh_private_key_file=./.vagrant/machines/host2/virtualbox/private_key
```

Most modules of Ansible are idempotent, which implies that the changes are applied only if needed.

If we try to run a command with an idempotent module, such as copy, we will see that the second time since the file already exists the tasks succeed without performing any actions.

Notice the different colors (green indicates no actions) of the command output after we execute the same command a second time:



If you would like to get more information about Ansible ad hoc commands, check out the Intro to ad hoc commands official user guide.

Intro to Ansible Playbooks

Playbooks are the simplest way in Ansible to automate repeating tasks in the form of reusable and consistent configuration files. Playbooks are scripts defined in YAML files and contain any ordered set of steps to be executed on our managed nodes.

As mentioned, tasks in a playbook are executed from top to bottom. At a minimum, a playbook should define the managed nodes to target and some tasks to run against them.

In playbooks, data elements at the same level must share the same indentation while items that are children of other items must be indented more than their parents.

Let’s look at a simple playbook to get an idea of how that looks in practice.

For the needs of this demo, we will use a simple playbook that runs against all hosts and copies a file, creates a user, and upgrades all apt packages on the remote machines.

```
---
- name: Intro to Ansible Playbooks
  hosts: all

  tasks:
  - name: Copy file hosts with permissions
    ansible.builtin.copy:
      src: ./hosts
      dest: /tmp/hosts_backup
      mode: '0644'
  - name: Add the user 'bob'
    ansible.builtin.user:
      name: bob
    become: yes
    become_method: sudo
  - name: Upgrade all apt packages
    apt:
      force_apt_get: yes
      upgrade: dist
    become: yes
```

On the top section, we define the group of hosts on which to run the playbook and its name. After that, we define a list of tasks. Each of the tasks contains some information about the task and the module to be executed along with the necessary arguments.

To avoid specifying the location of our inventory file every time we can define this via a configuration file (**ansible.cfg**). To find out more about Ansible configuration options check here.

```
[defaults]
inventory=./hosts
```

You can validate that this works as expected by running the ansible-inventory command.

```
ansible-inventory --list
{
    "_meta": {
        "hostvars": {
            "host1": {
                "ansible_host": "127.0.0.1",
                "ansible_port": 2222,
                "ansible_ssh_private_key_file":
"./.vagrant/machines/host1/virtualbox/private_key",
                "ansible_user": "vagrant"
            },
            "host2": {
                "ansible_host": "127.0.0.1",
                "ansible_port": 2200,
                "ansible_ssh_private_key_file":
"./.vagrant/machines/host2/virtualbox/private_key",
                "ansible_user": "vagrant"
            }
        }
    },
    "all": {
        "children": [
            "ungrouped"
        ]
```

```
    },
    "ungrouped": {
        "hosts": [
            "host1",
            "host2"
        ]
    }
}
```

Now we are ready to run our first playbook using the ansible-playbook command.

```
$ ansible-playbook intro_playbook.yml
 ____________________________________
< PLAY [Intro to Ansible Playbooks] >
 ------------------------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||


 ____________________________
< TASK [Gathering Facts] >
 ----------------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

ok: [host2]
ok: [host1]
 ____________________________________________
< TASK [Copy file with owner and permissions] >
 --------------------------------------------
        \   ^__^
```

```
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

changed: [host1]
changed: [host2]
 _____________________________
< TASK [Add the user 'bob'] >
 ---------------------------
        \   ^__^
         \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

changed: [host1]
changed: [host2]
 _____________________________________
< TASK [Upgrade all apt packages] >
 ---------------------------------
        \   ^__^
         \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

[WARNING]: Updating cache and auto-installing missing dependency: python-apt
changed: [host2]
changed: [host1]
 _____________
< PLAY RECAP >
 -------------
        \   ^__^
         \  (oo)\_______
```

```
          (__)\       )\/\
              ||----w |
              ||     ||


host1                  : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
host2                  : ok=4    changed=3    unreachable=0    failed=0    skipped=0    rescued=0
ignored=0
```

Our playbook has been executed successfully and we can follow the ordered execution of the tasks per host by checking the command output. At the bottom, a summary of the playbook execution is provided by Ansible.

Something that you might find useful at times, is to validate the syntax of a playbook with the flag **–syntax-check**.

Another handy option is to use the **-C** flag to perform a dry run of the playbook's execution. This option doesn't actually make any changes but it just reports the changes that will happen during a real run.

The official Working with playbooks user guide includes many more details and options about playbooks so make sure to read it when you start moving into more advanced use cases.

Using Variables in Playbooks

Variables can be defined in Ansible at more than one level and Ansible chooses the variable to use based on variable precedence.

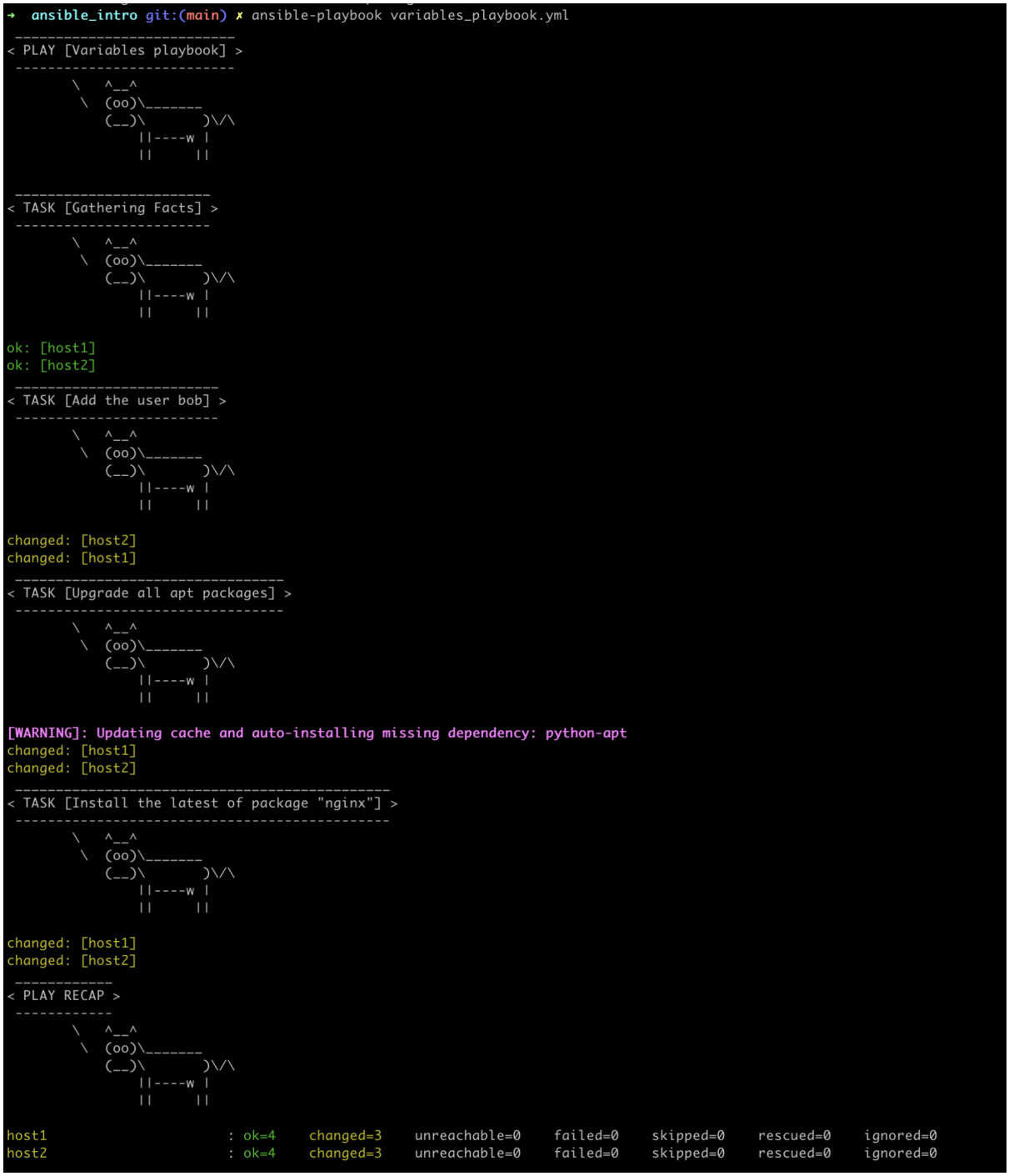
Let's see how we can use variables at the playbook level.

The most common method is to use a **vars** block at the beginning of each playbook. After declaring them, we can use them in tasks. Use **{{ variable_name }}** to reference a variable in a task.

```
---
- name: Variables playbook
  hosts: all
  vars:
     state: latest
     user: bob
  tasks:
  - name: Add the user {{ user }}
    ansible.builtin.user:
     name: "{{ user }}"
  - name: Upgrade all apt packages
    apt:
     force_apt_get: yes
     upgrade: dist
  - name: Install the {{ state }} of package "nginx"
    apt:
     name: "nginx"
     state: "{{ state }}"
```

In the above example, we have used the variables **user** and **state**. When referencing a variable as another variable's value, we must add quotes around the value as shown in our example.

During the playbook run below, we see that the variable's substitution happens successfully.



Take a look at the Using Variables official user guide to learn more about advanced use cases of their usage in Ansible.

Ansible Roles

Ansible roles take your automations to the next level of abstraction.

They are ideal for sharing functionality between different teams, environments, or projects, making your code DRY and enabling streamlined management of your IaC. With roles, you get a standardized structure for bundling related tasks, variables, templates, handlers, and files, and this enhances reusability.

Conclusion: