“SlackUp Jenkins”

To create a Slack channel, receive Jenkins build notifications and use it to invoke Jenkins jobs remotely by configuring slash commands in Slack.

Marco Da Pieve

Contents

1 Project Scope

To create a Slack channel and receive Jenkins build notifications and use it to invoke Jenkins jobs remotely by configuring the slash commands in Slack

2 Procedure

2.1 Requirements

This step is required to setup the environment that will be used during the project implementation

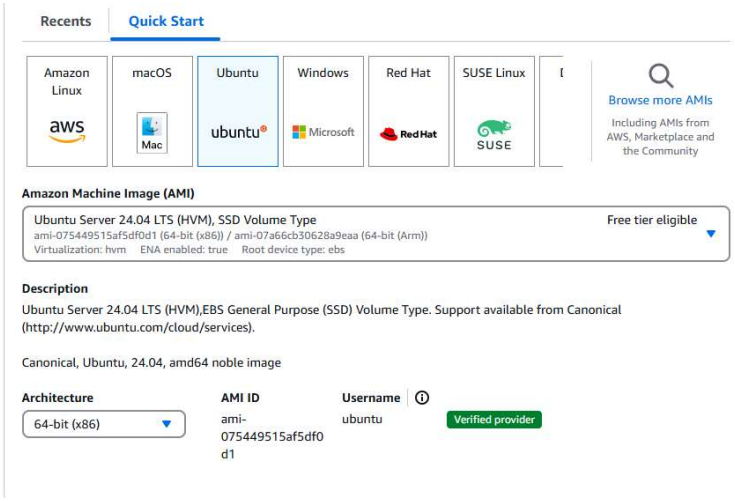
2.1.1 Platform Setup AWS EC2 instance

In order to prepare a setup an AWS EC2 instance running Ubuntu 24.04 has been used. This allows reachability of the services running on this instance via a public IP address which is necessary for interaction between Slack and Jenkins.

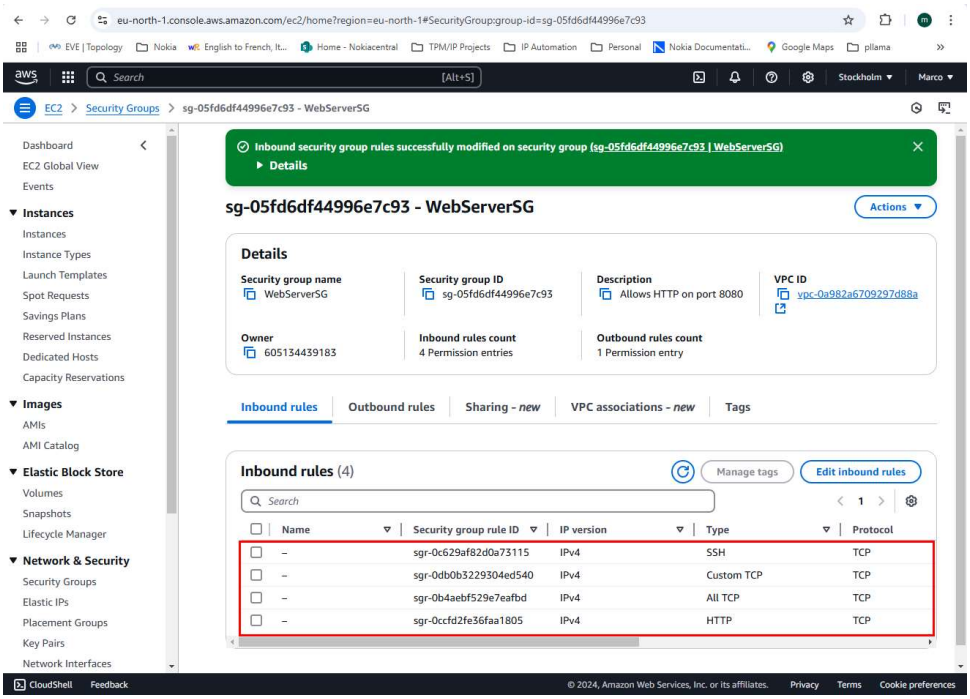
In case you are using AWS, as in this case, please complete section 1, 2, 3 and 4 at the following link.

https://www.jenkins.io/doc/tutorials/tutorial-for-installing-jenkins-on-AWS/

Note: at step 4 in section 4, please select “ubuntu” as this is the instance used in this project



The steps at the link above allows the reachability of the Jenkins web service running on the AWS EC2 instance from remote. A reference of the security group that needs to be created to allow inbound traffic is shown below:



This is required when connecting via SSH from the local laptop or when Slack needs to invoke the Jenkin's job.

SSH to the AWS EC2 instance, to verify the instance of the underlying host machine, the following command can be used:

```
ubuntu@ip-172-31-42-71:~$ cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=24.04
DISTRIB_CODENAME=noble
DISTRIB_DESCRIPTION="Ubuntu 24.04.1 LTS"
```

To prepare the environment, let's update the apt repositories:

```
ubuntu@ip-172-31-42-71:~$ sudo apt update
```

if any package is found to be upgradable, proceed with the upgrade:

```
ubuntu@ip-172-31-42-71:~$ sudo apt -y upgrade
```

2.1.2 Java

Java is a prerequisite for Jenkins. To verify whether Java is already installed, please run the following command:

```
ubuntu@ip-172-31-42-71:~$ java -version
Command 'java' not found, but can be installed with:
sudo apt install openjdk-17-jre-headless  # version 17.0.12+7-1ubuntu2~24.04, or
sudo apt install openjdk-21-jre-headless  # version 21.0.4+7-1ubuntu2~24.04
sudo apt install default-jre              # version 2:1.17-75
sudo apt install openjdk-11-jre-headless  # version 11.0.24+8-1ubuntu3~24.04.1
sudo apt install openjdk-8-jre-headless   # version 8u422-b05-1~24.04
sudo apt install openjdk-19-jre-headless  # version 19.0.2+7-4
sudo apt install openjdk-20-jre-headless  # version 20.0.2+9-1
sudo apt install openjdk-22-jre-headless  # version 22~22ea-1
ubuntu@ip-172-31-42-71:~$
```

if the output is similar to the one above, then Java is not installed on the system and needs to be installed.

```
ubuntu@ip-172-31-42-71:~$ sudo apt -y install fontconfig openjdk-17-jre
```

After installing Java, repeat the “java -version” command to verify that installation was successful:

```
ubuntu@ip-172-31-42-71:~$ java -version
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Ubuntu-2ubuntu124.04)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Ubuntu-2ubuntu124.04, mixed mode, sharing)
ubuntu@ip-172-31-42-71:~$
```

2.1.3 Jenkins

Now, Jenkins can be installed. To do this, the repositories necessary for Jenkins installation need to be added:

```
ubuntu@ip-172-31-42-71:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
>   https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2024-12-17 08:58:35--  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 151.101.86.133, 2a04:4e42:14::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|151.101.86.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: ‘/usr/share/keyrings/jenkins-keyring.asc’

/usr/share/keyrings/jenkins-keyrin
100%[=====================================================================>]   3.10K  --.-KB/s    in
0s

2024-12-17 08:58:35 (34.5 MB/s) - ‘/usr/share/keyrings/jenkins-keyring.asc’ saved
[3175/3175]

ubuntu@ip-172-31-42-71:~$
ubuntu@ip-172-31-42-71:~$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]"
\
>   https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
enkins.l>   /etc/apt/sources.list.d/jenkins.list > /dev/null
ubuntu@ip-172-31-42-71:~$
```

Let’s run the apt-get update command once again to update the repository list:

```
ubuntu@ip-172-31-42-71:~$ sudo apt-get update
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu noble-security InRelease
Ign:5 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
```

```
Get:7 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [28.2 kB]
Fetched 31.1 kB in 1s (47.5 kB/s)
Reading package lists... Done
ubuntu@ip-172-31-42-71:~$
```

Finally, install Jenkins:

```
ubuntu@ip-172-31-42-71:~$ sudo apt-get -y install Jenkins
```

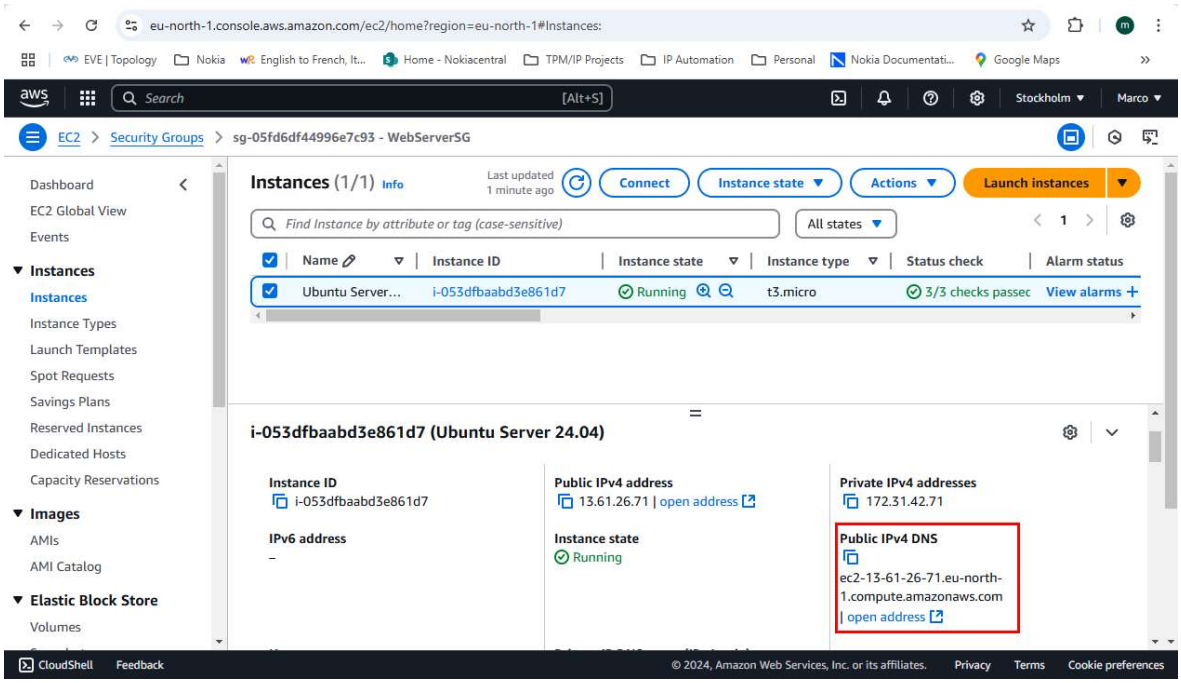
After installation, verify that Jenkins is running with the following command:

```
ubuntu@ip-172-31-42-71:~$ systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Tue 2024-12-17 09:05:00 UTC; 1min 49s ago
```

If a service is already running on the system by using port 8080, Jenkins listening port can be changed by editing the /lib/systemd/system/jenkins.service file at the following line:

```
Environment="JENKINS_PORT=8080"
```

Now to connect to the Jenkins web page, the Public IPv4 DNS needs to be used:



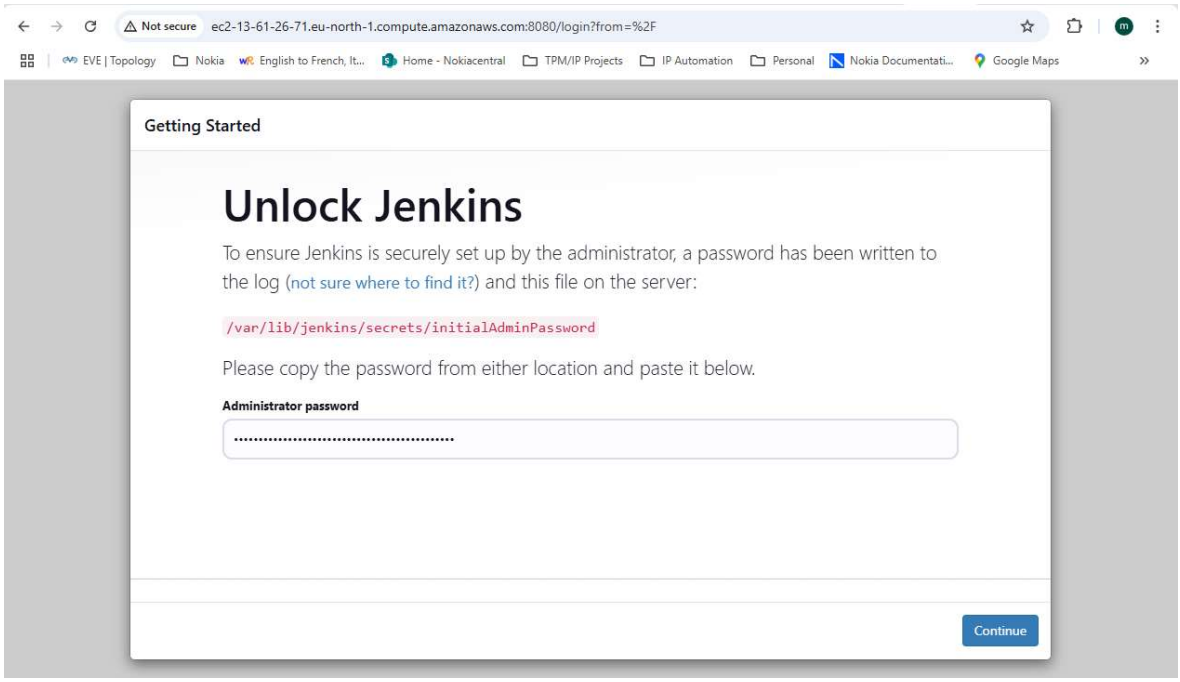
Open a web browser and enter the following url

http://ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080

For your own environment please replace the public IPv4 DNS part of the URL above (before the ":8080").

2.1.4 Jenkins First Setup

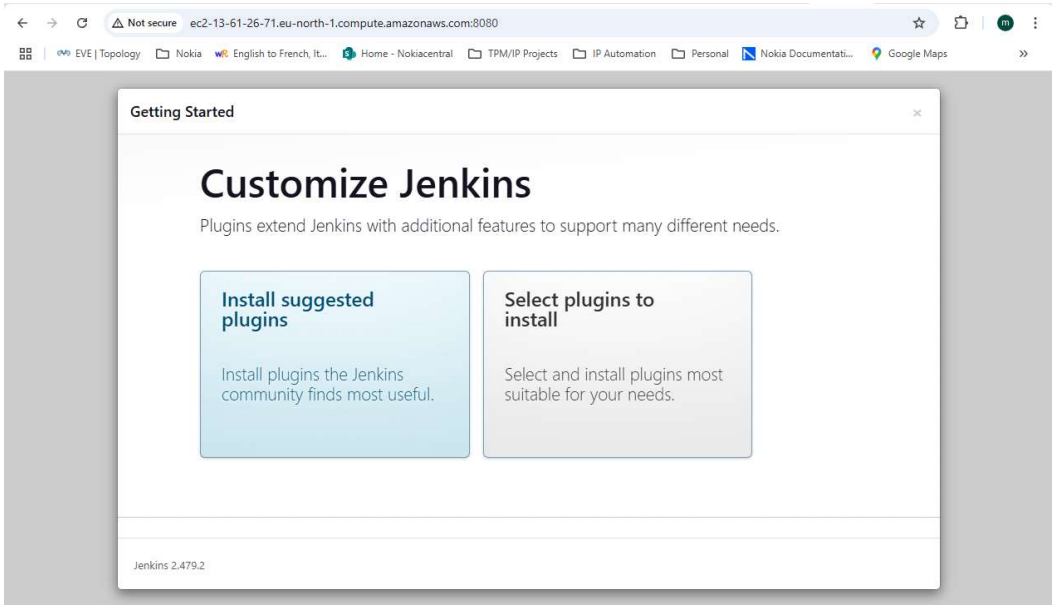
The URL above lands to the following page:



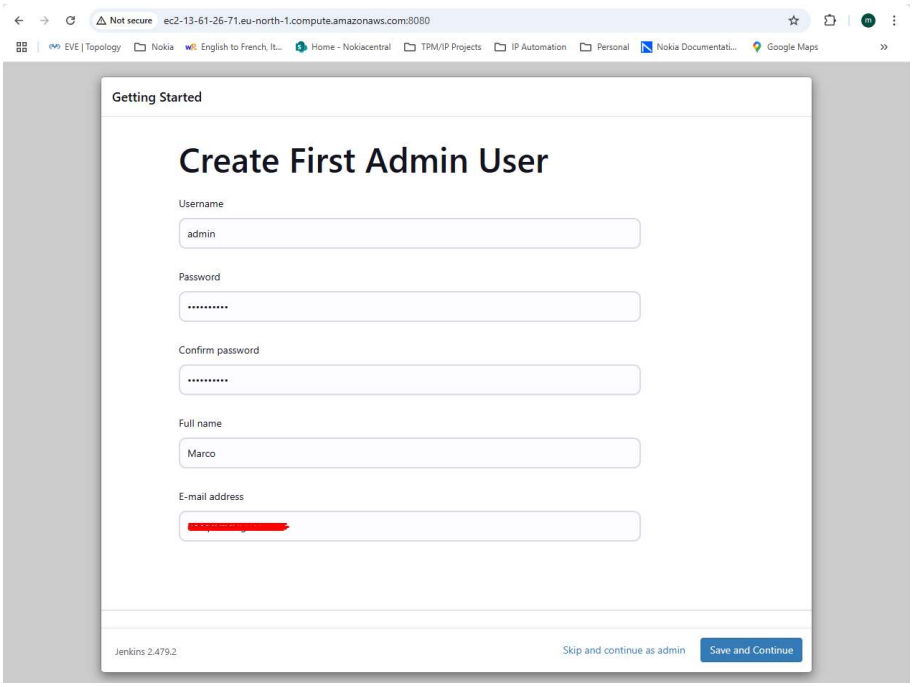
1. Please verify the password in the location specified in the page above and click on "Continue":

```
ubuntu@ip-172-31-42-71:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```

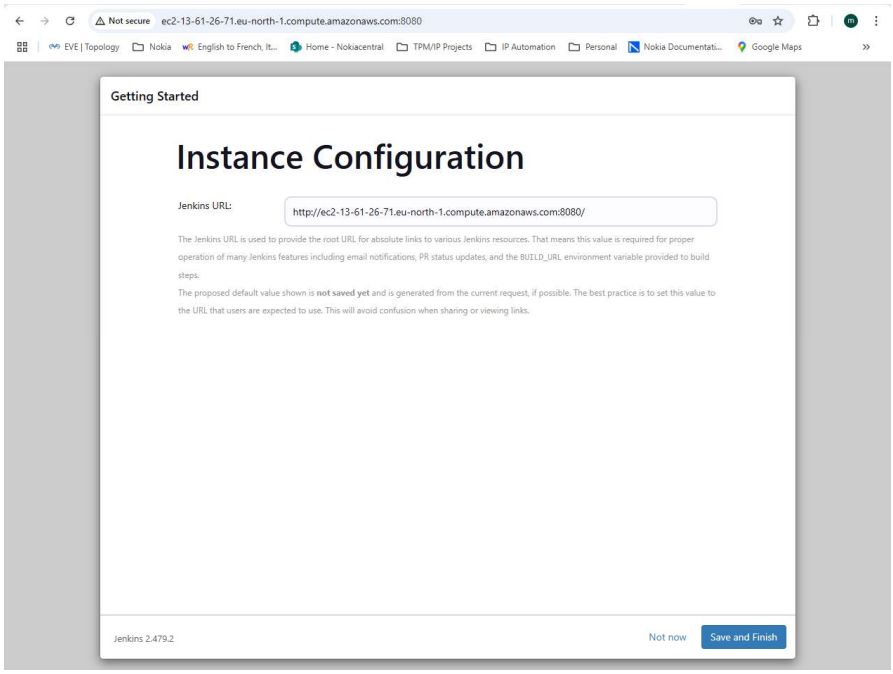
2. Proceed by installing the suggested plugins:



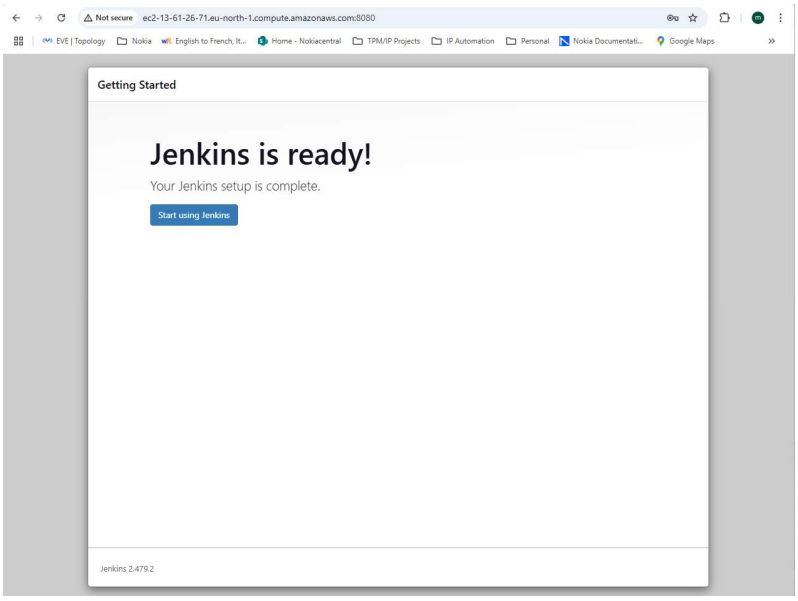
3. Ater the installation of the suggested plugins, create the first admin user:



4. The next page is to specify the URL through which the Jenkins web page is accessible from remote. The one displayed can be used. Please click on “Save and Finish”:

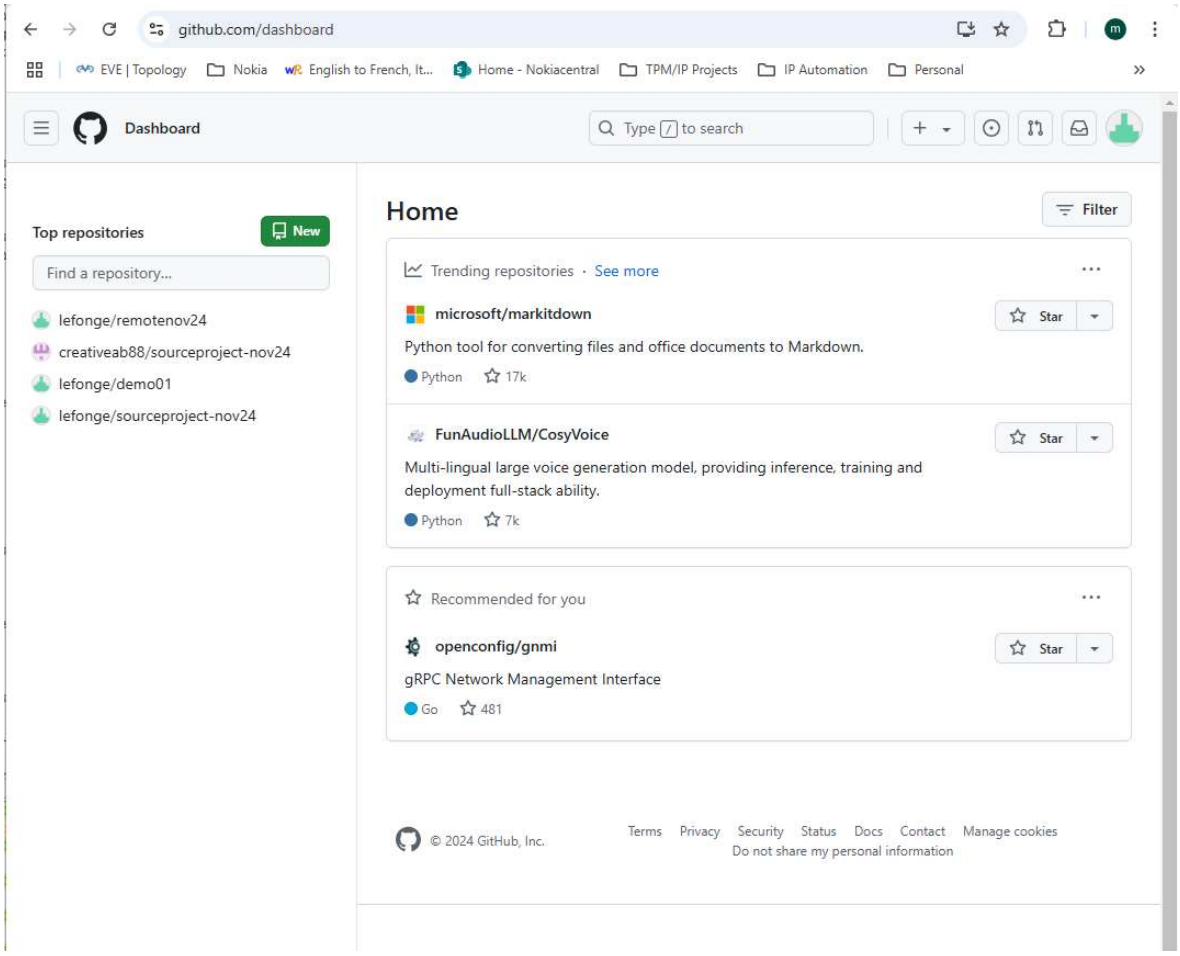


5. Finally, click on “Start using Jenkins”:

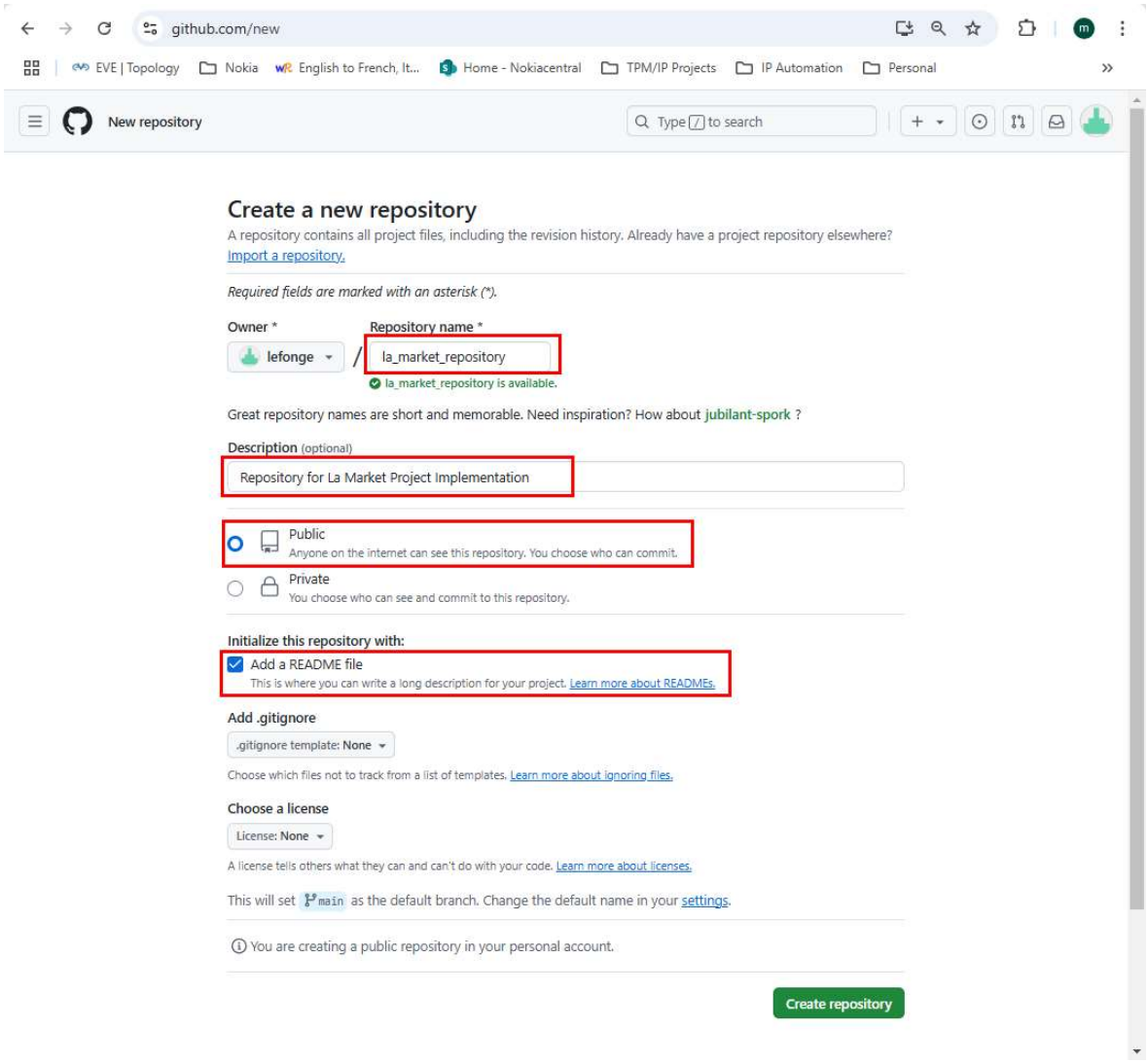


2.2 Create a GitHub Repository

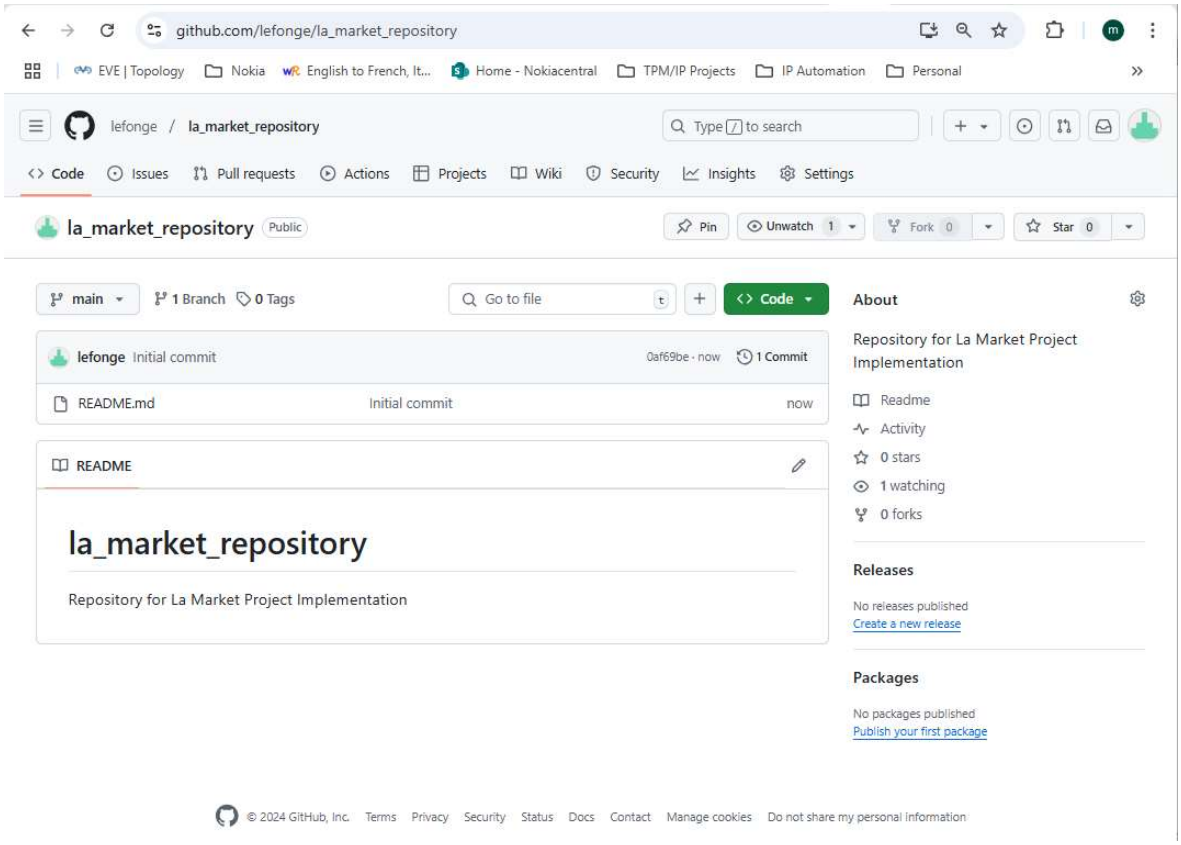
For this step, an account on github.com is required. To create a new repository login and access your github home page:



1. Click on the “New” button at the top-left corner of the page. The following page loads and fill with the required/optional data:



2. After creating the repository the following page loads and a README.md file is also available. The branch name for this repository was left as the default “main” name.



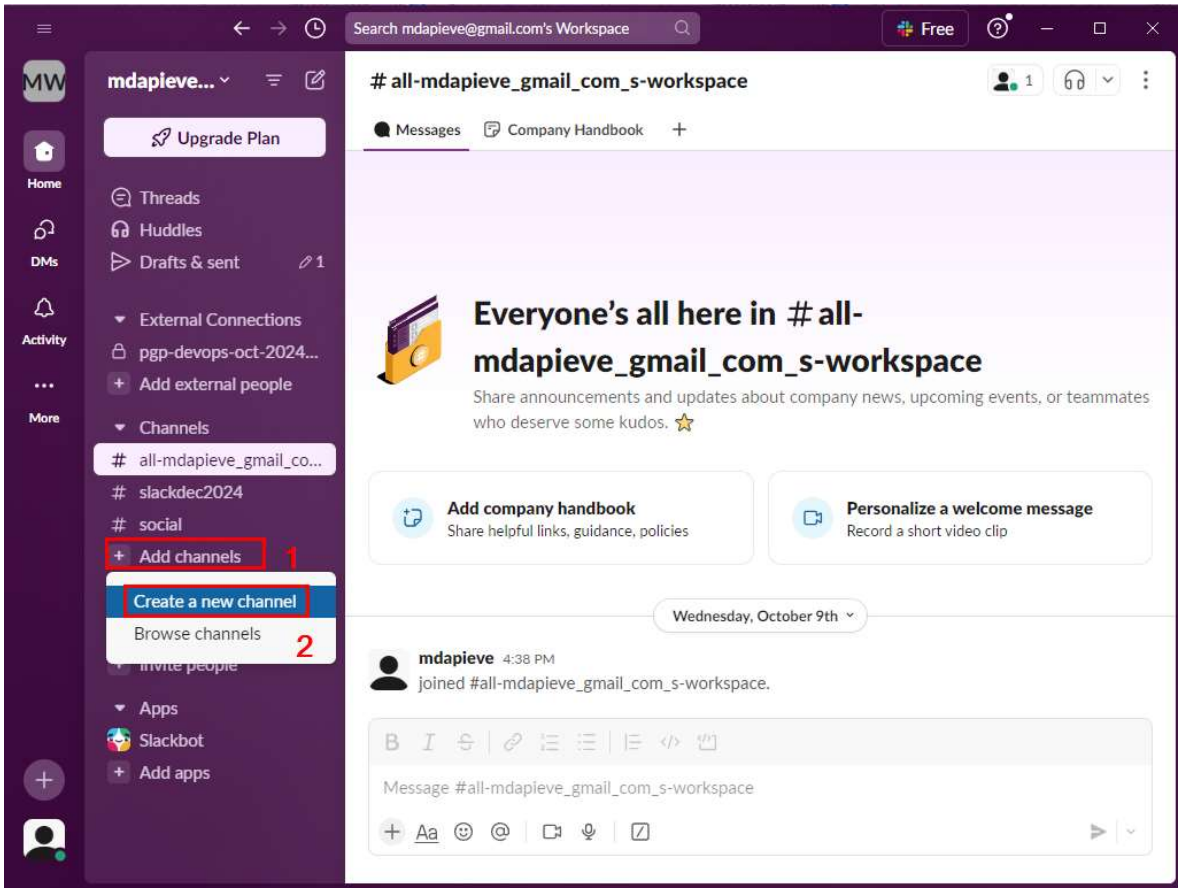
2.3 Slack

Please install Slack on your local machine

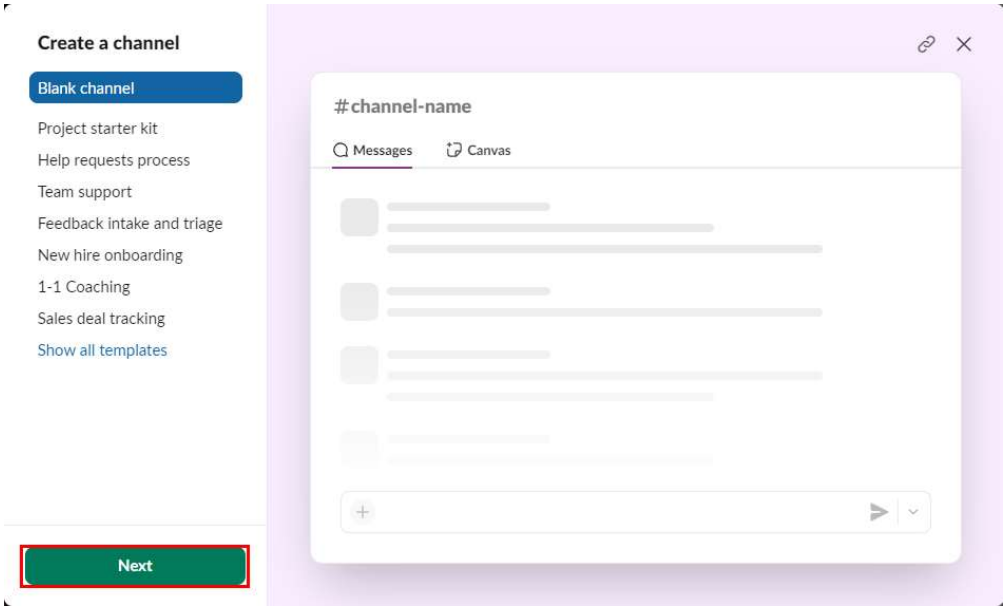
2.3.1 Create a Slack Channel

For this step, an account on SLACK is required.

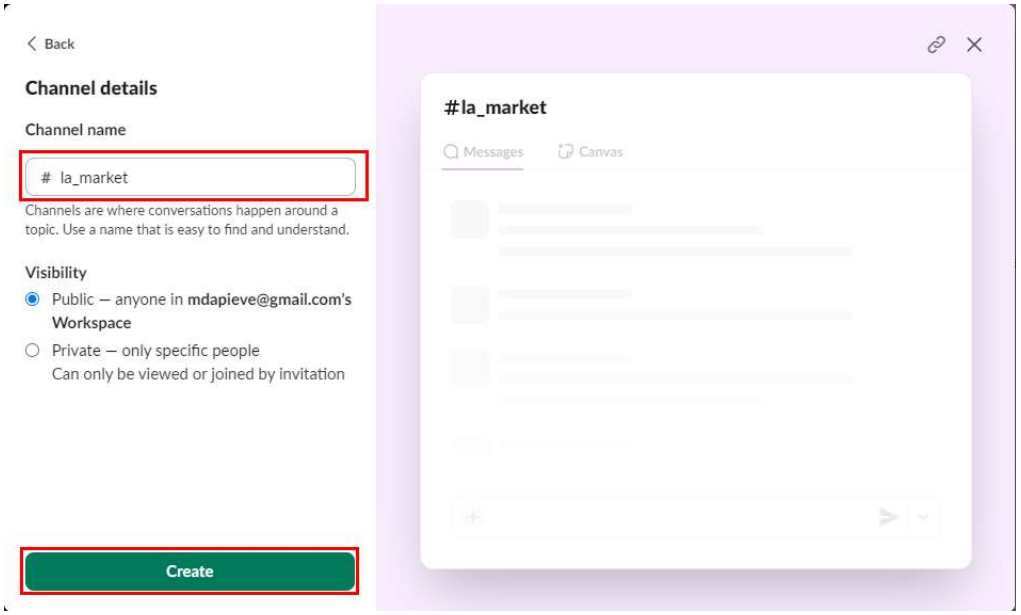
1. To create a new slack channel, click on add channel and crate a new channel



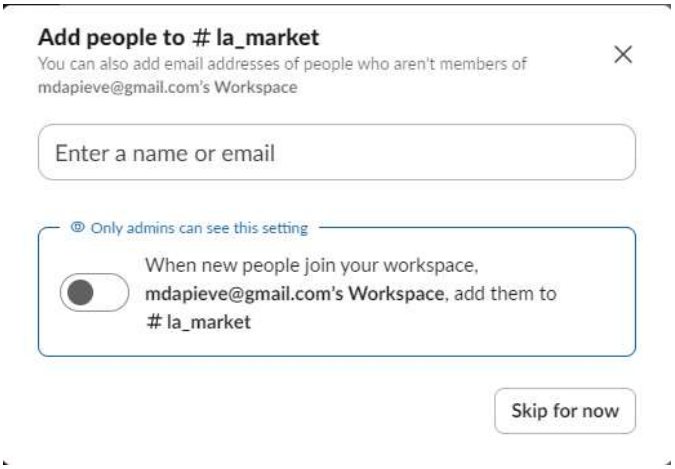
2. Click on Next:



3. Specify a name and then click create

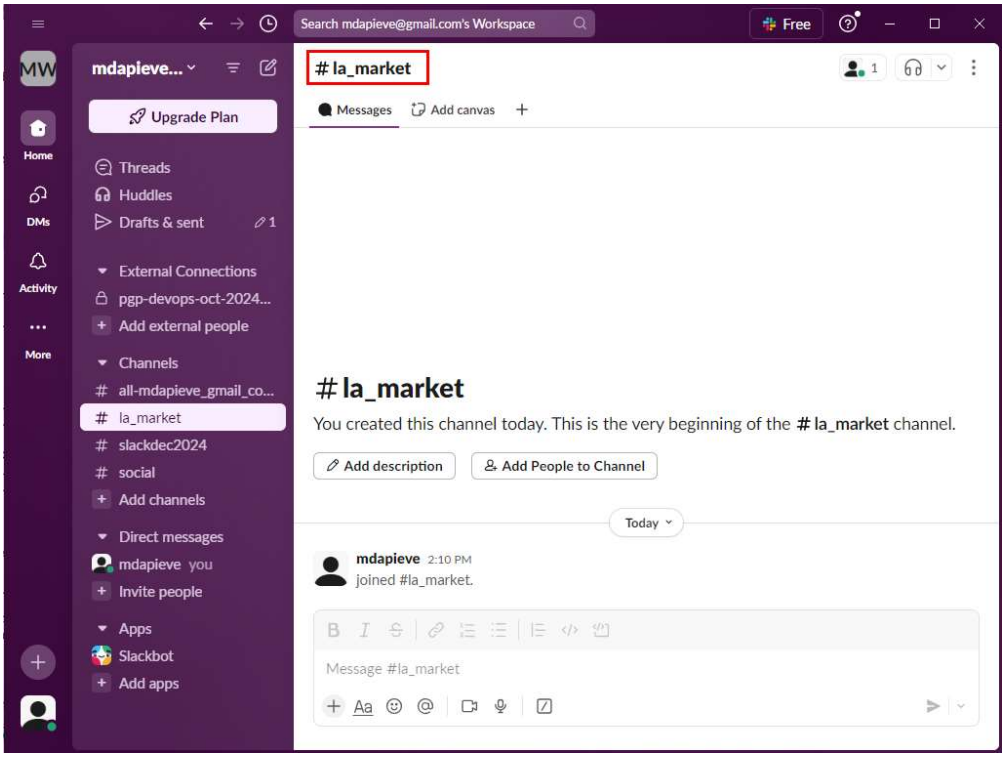


4. For the scope of this project, no additional members will be added and this step can be skipped

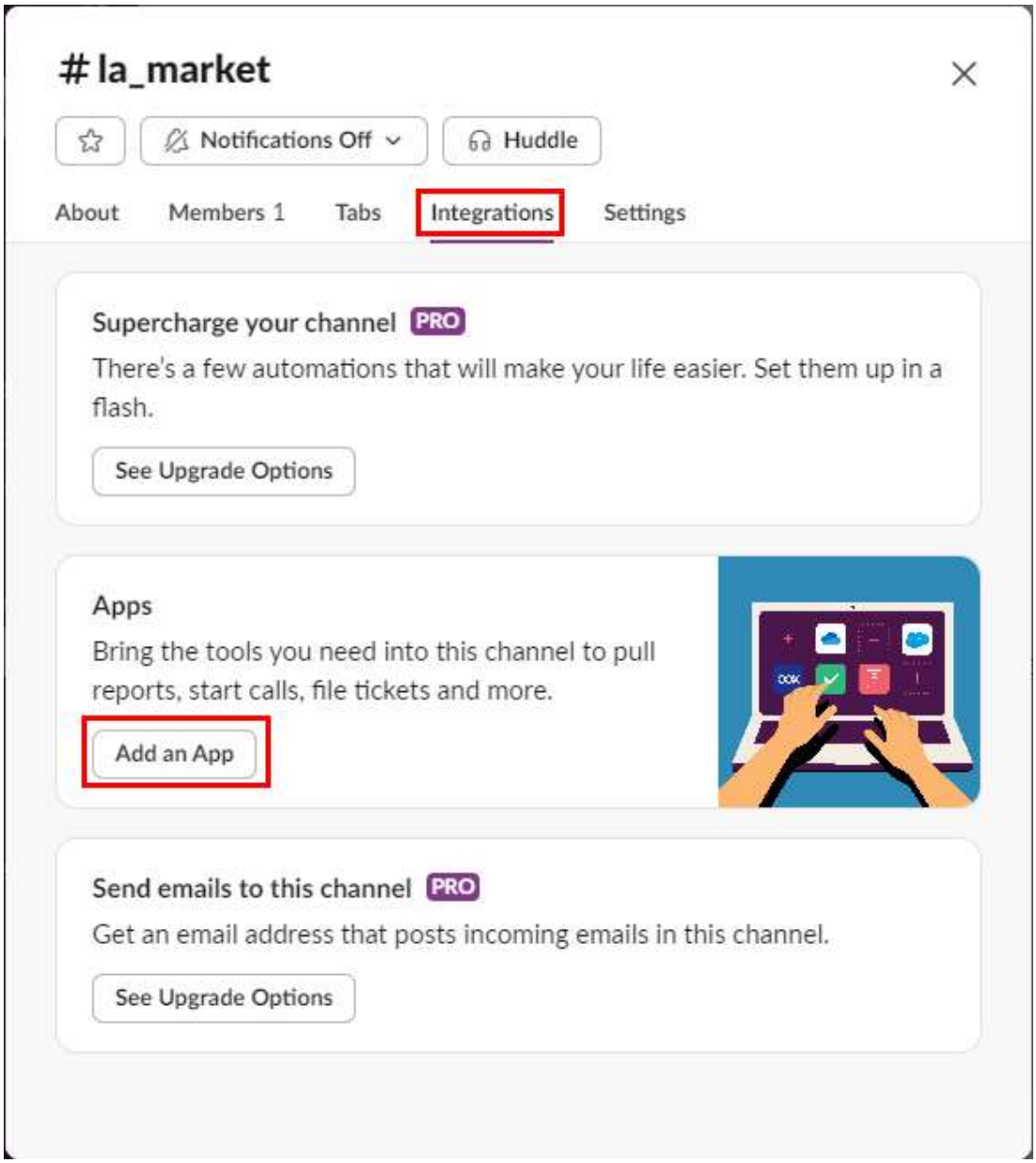


2.3.2 Integrate Jenkins into Slack

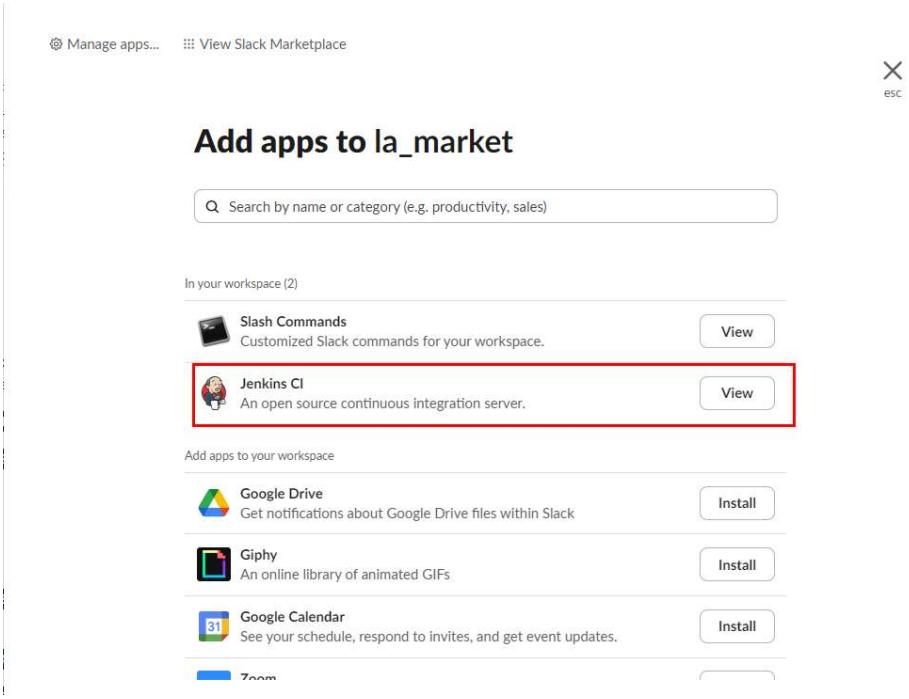
1. Once the channel is created, click on the channel name



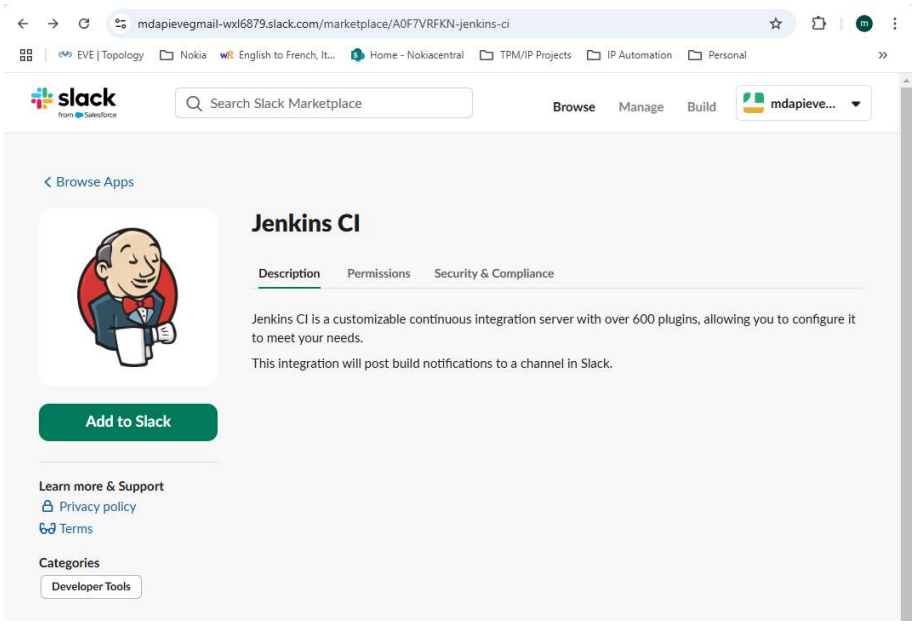
2. Switch to the “Integrations” tab and click on “Add an App”



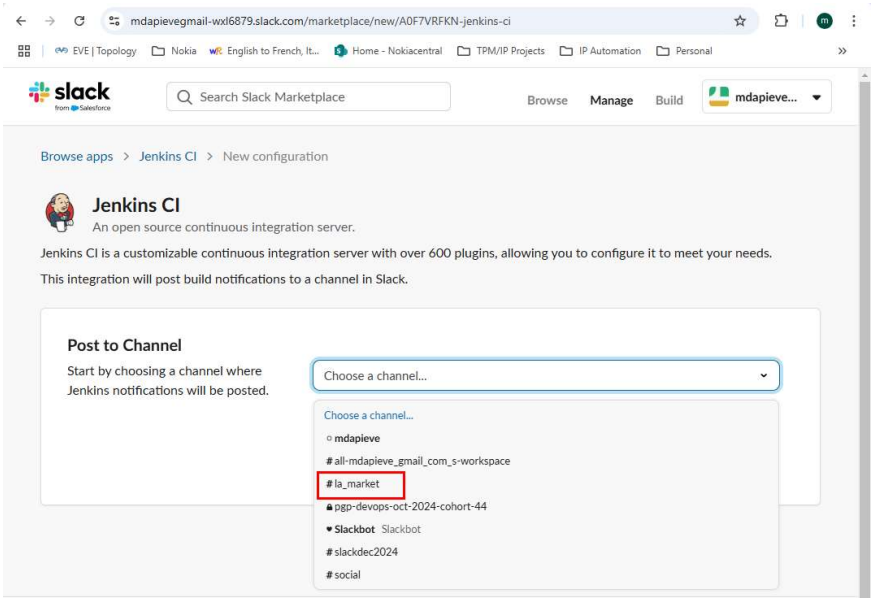
3. If not already installed, please search and install “Jenkins CI” in Slack and then click on it



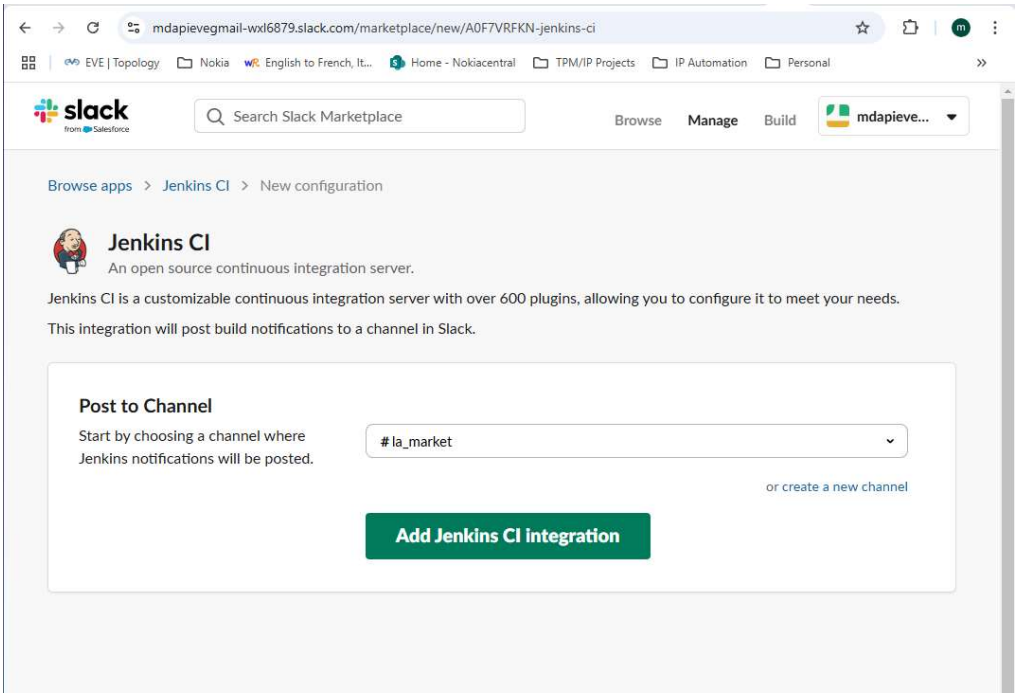
4. The following page opens, click on “Add to Slack”



5. Choose the correct channel



6. Click on “Add Jenkins CI Integration”



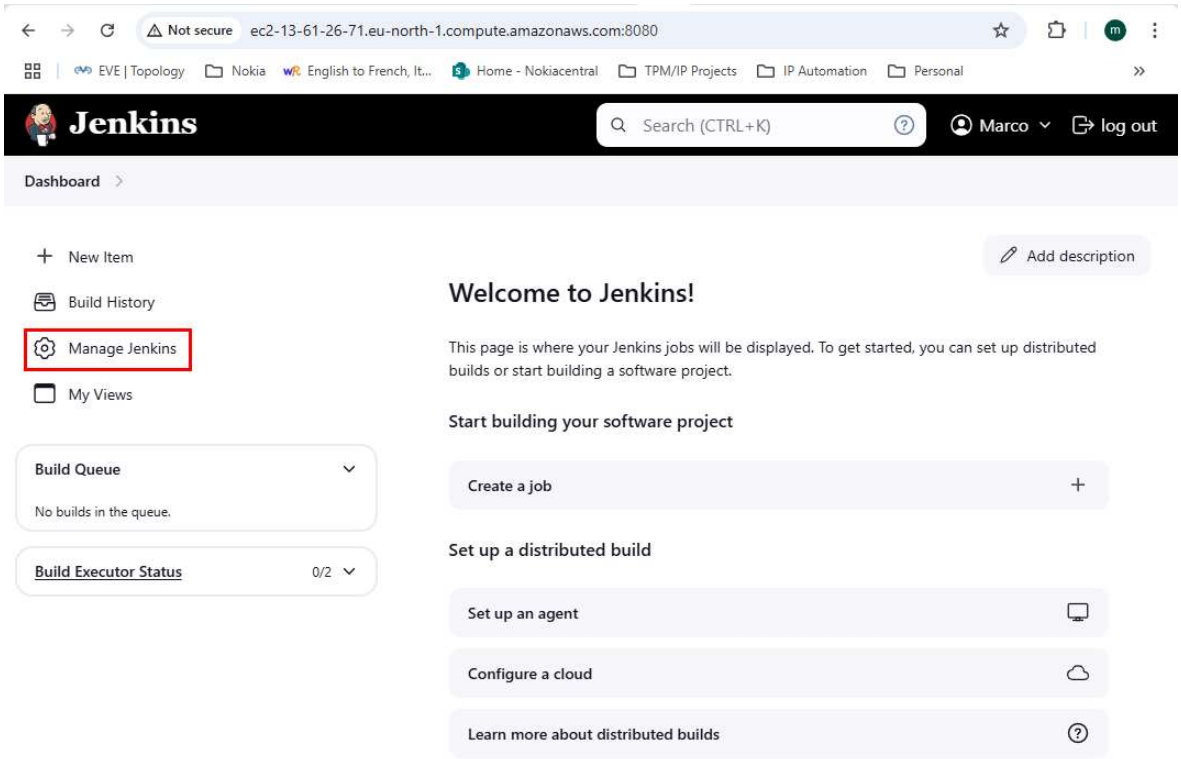
7. After clicking the button in the page above, a list of steps is presented to perform what is described in the next section. Step 3 of the list of steps includes some important information that needs to be noted somewhere to be used later on. Such information is:

    - **Team Subdomain:** mdapievegmail-wxl6879
    - **Integration Token Credential ID:** Create a secret text credential using g1oUexBgZt3l7HxykBr9BbEJ as the value

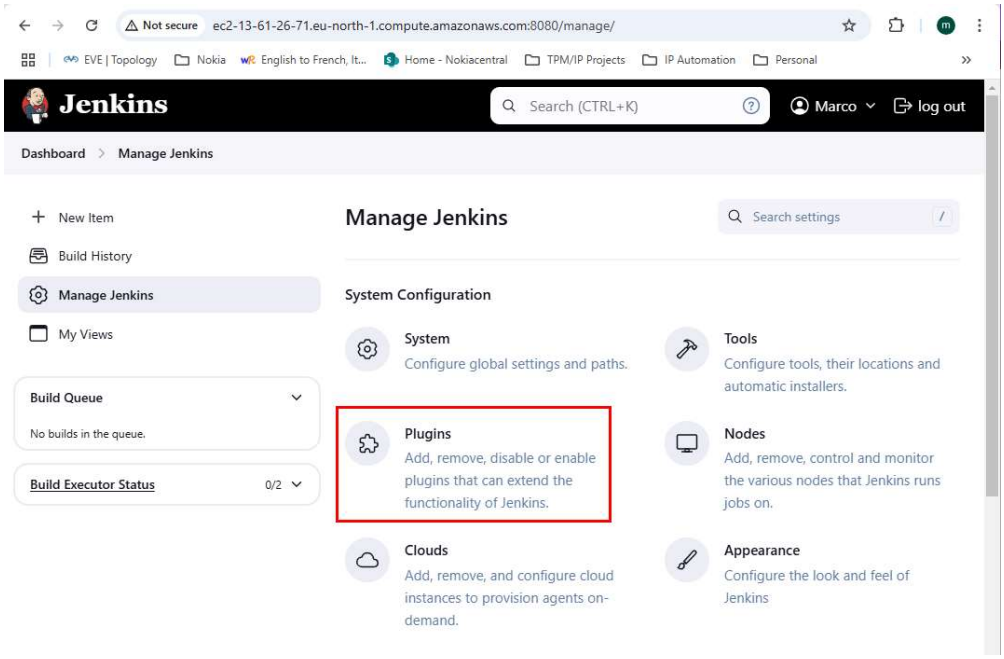
2.4 Jenkins Configuration for Notification Delivery to Slack

Please follow the steps that are presented in the next page and that are reported here below for commodity.

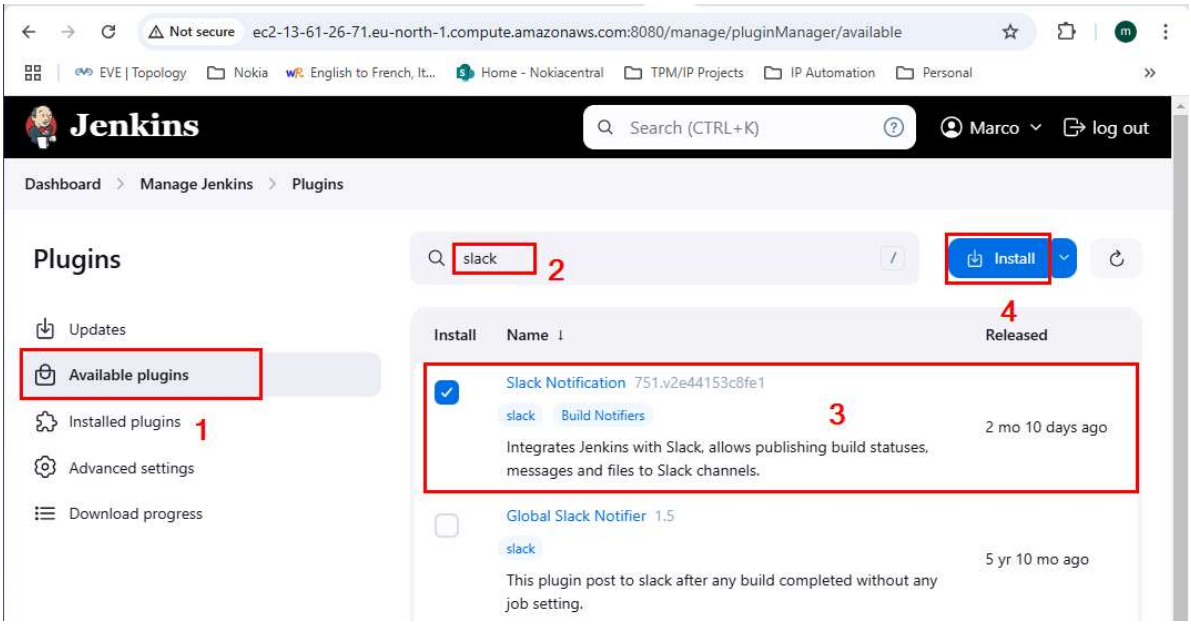
1. Access Jenkins' homepage and click on "Manage Jenkins"



2. Click on “Plugins”

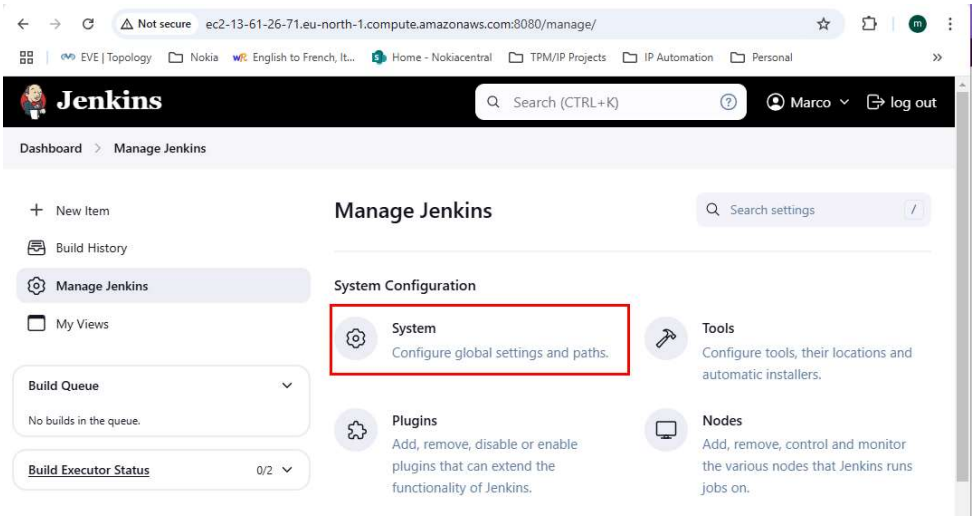


3. Switch to the “Available Plugins”, search for “slack”, select the “Slack Notification” plugin” and then install

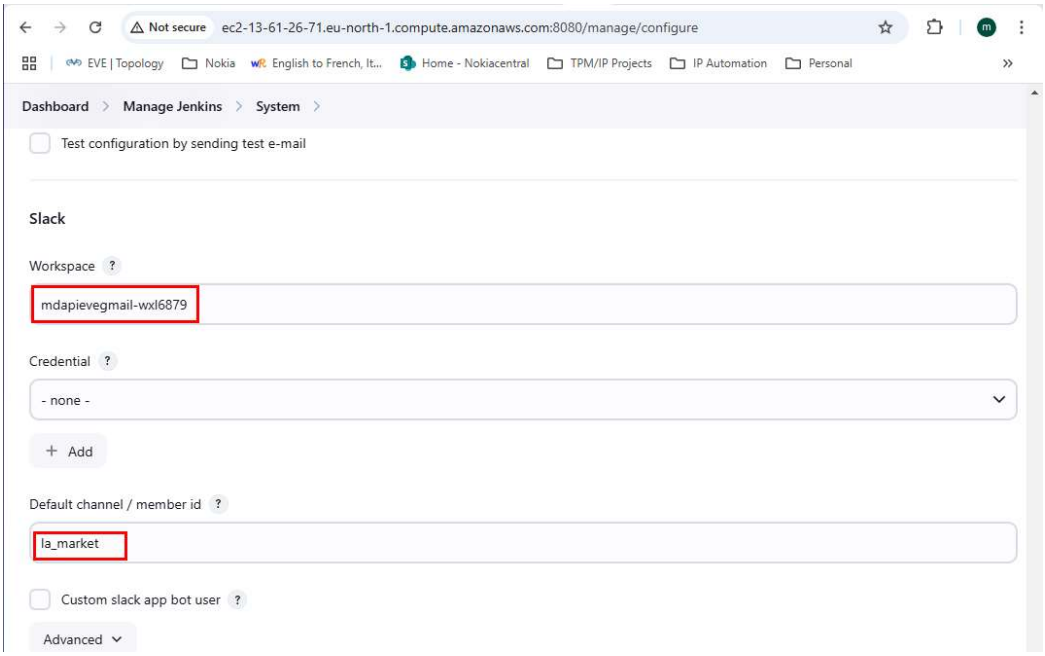


The installation of this plugin does not require Jenkins to be restarted

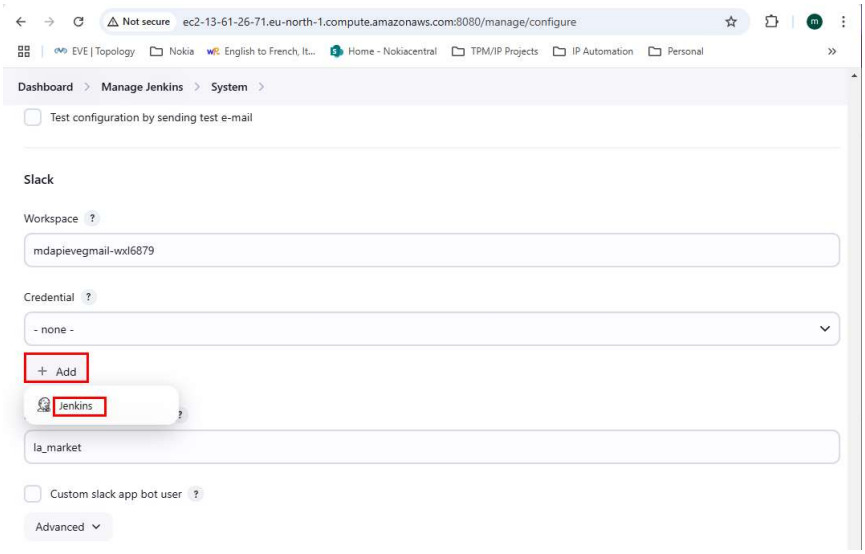
4. Go to “Manage Jenkins” again and click on “System”



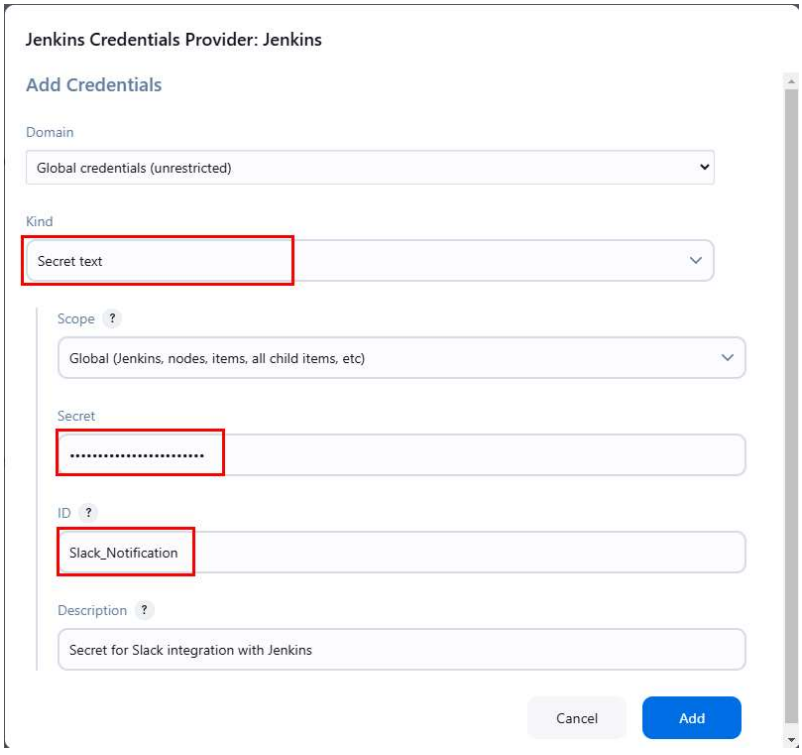
5. Add the “Workspace” and the “Default channel/member id” (The workspace name and the Secret are provided in step 7 from section 3.3.2)



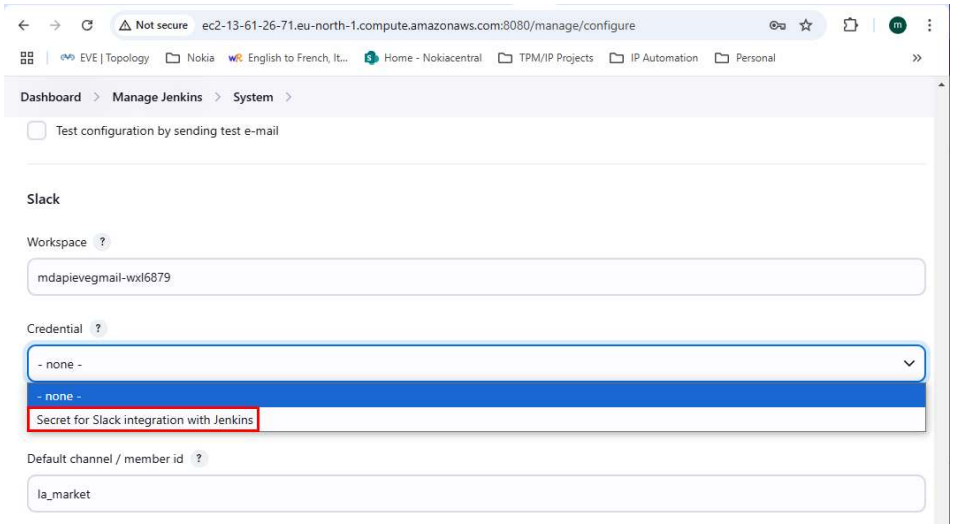
6. Now click on “+ Add” button in the “Credential” section



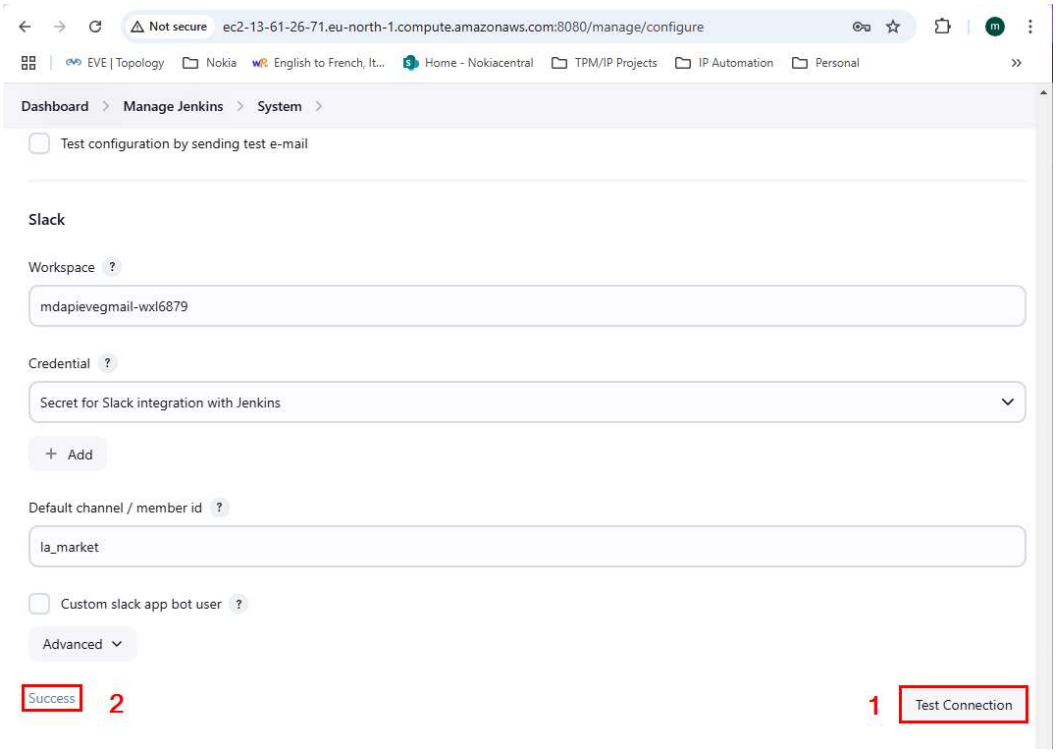
7. Populate the highlighted field correctly as described below and then click on “Add” (The workspace name and the Secret are provided in step 7 from section 3.3.2).



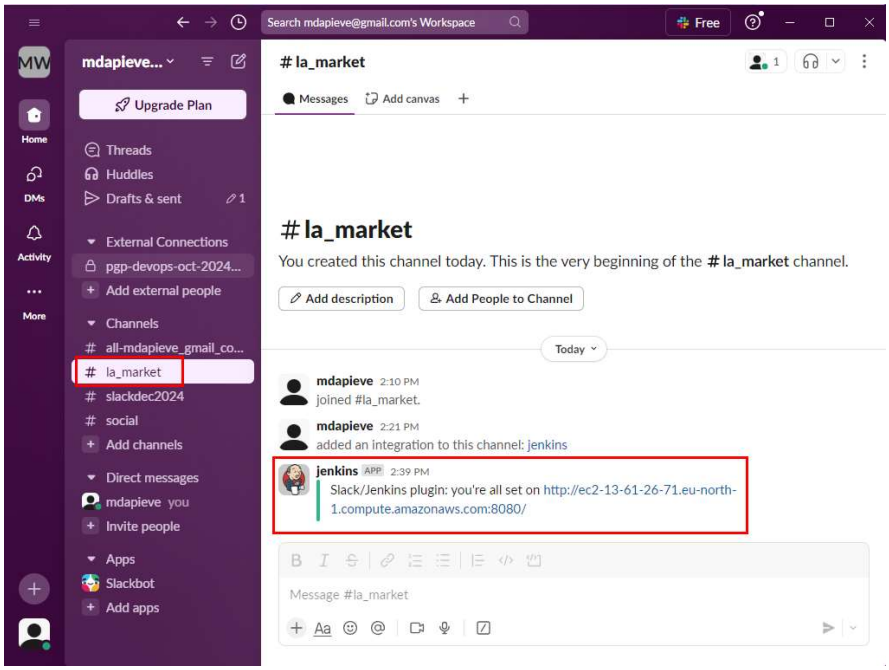
8. On the underlying page, it is now possible to specify the Credential



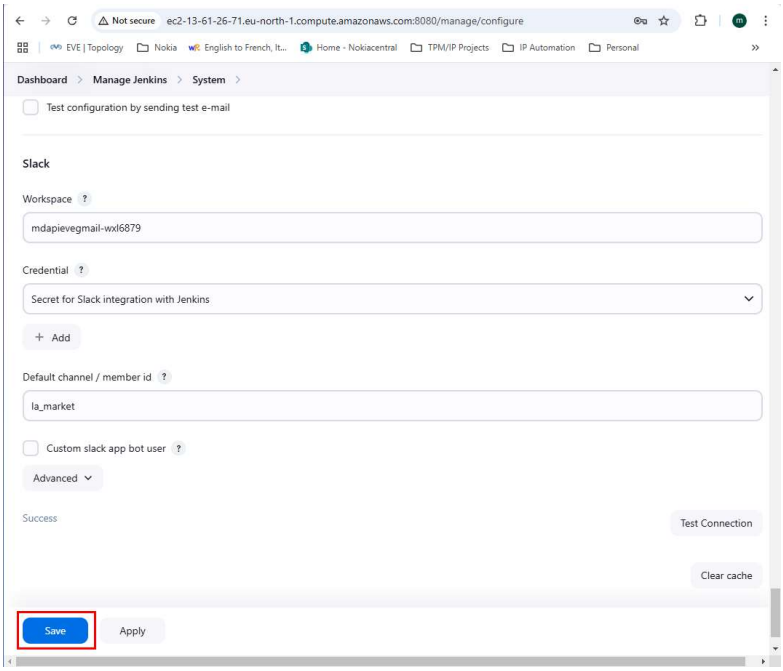
9. It is possible to verify the successful integration by click on “Test Connection” and check if the “Success” message appears:



10. This test also results in a message on the slack channel



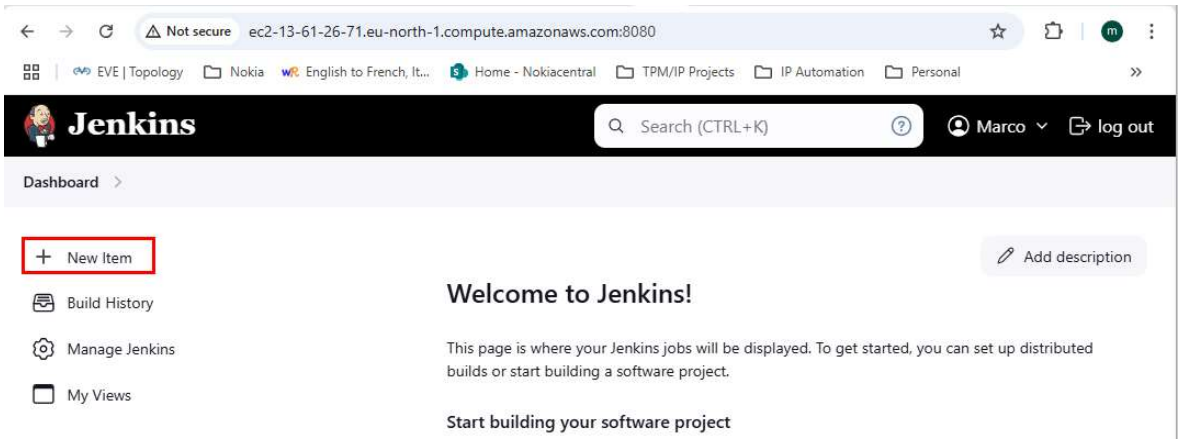
11. It is now possible to save the page on Jenkins



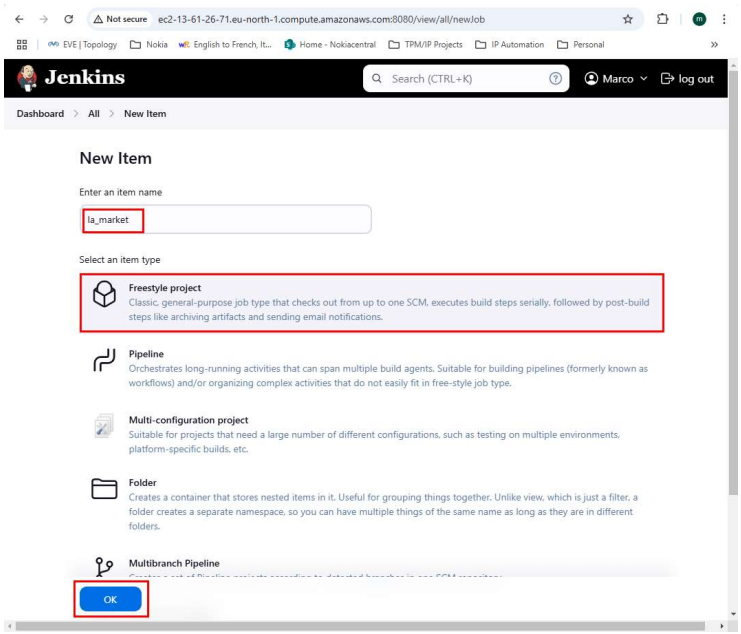
2.5 Create a Jenkins Job

In order to receive notification on the “la_market” slack channel, a Jenkins job is created and the the github “la_market_repository” is references as SCM.

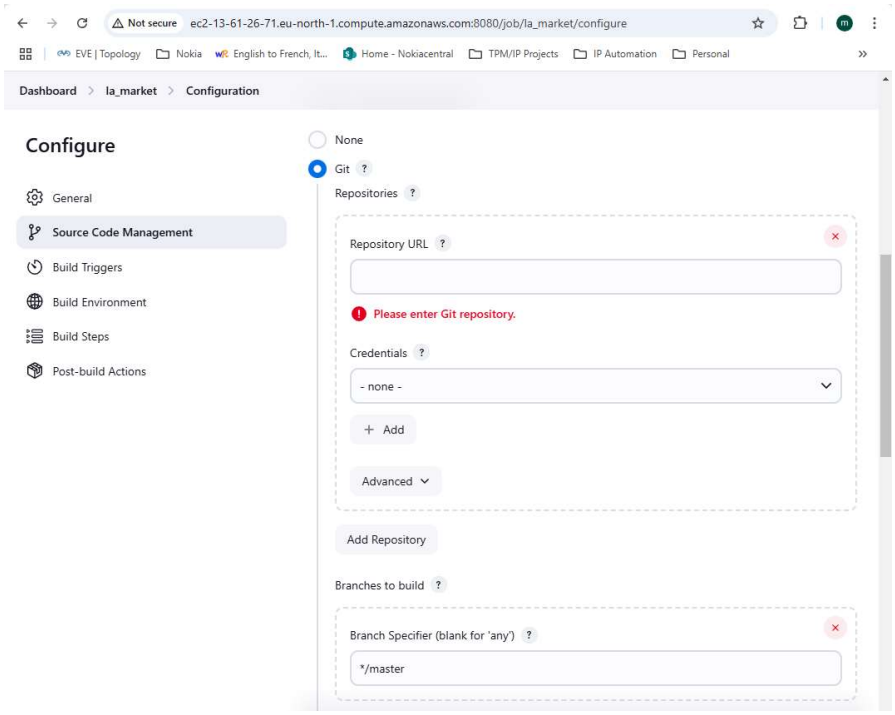
1. To do this, from the Jenkins dashboard, click on “New Item”



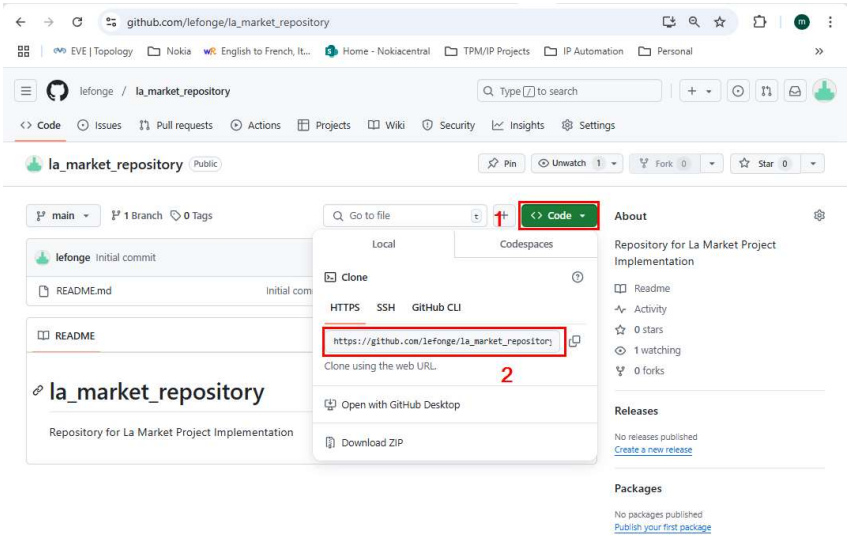
2. Provide a name for the job; specify a Freestyle project and click on "OK”



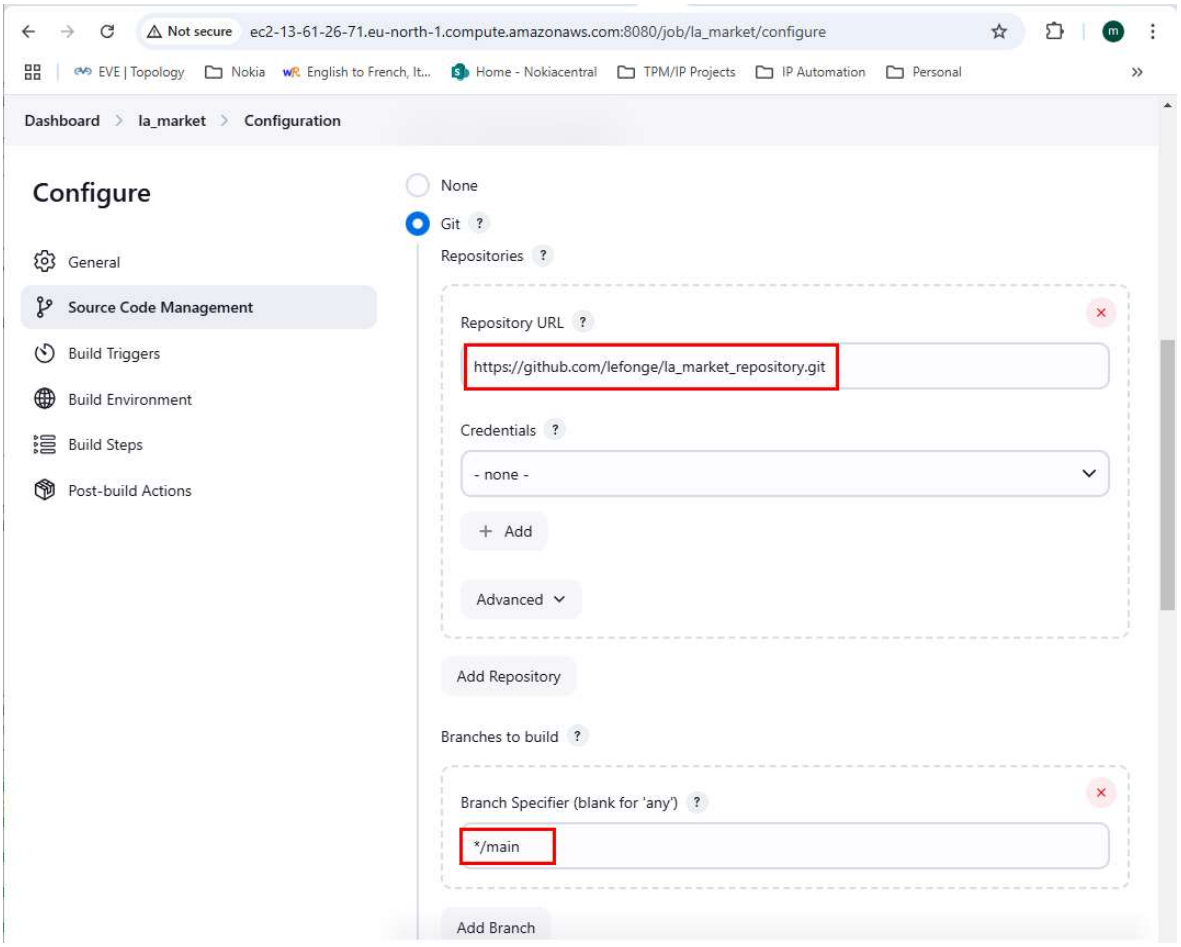
3. When the page loads, go to the “Source Code Management” section and select the “Git” checkbox:



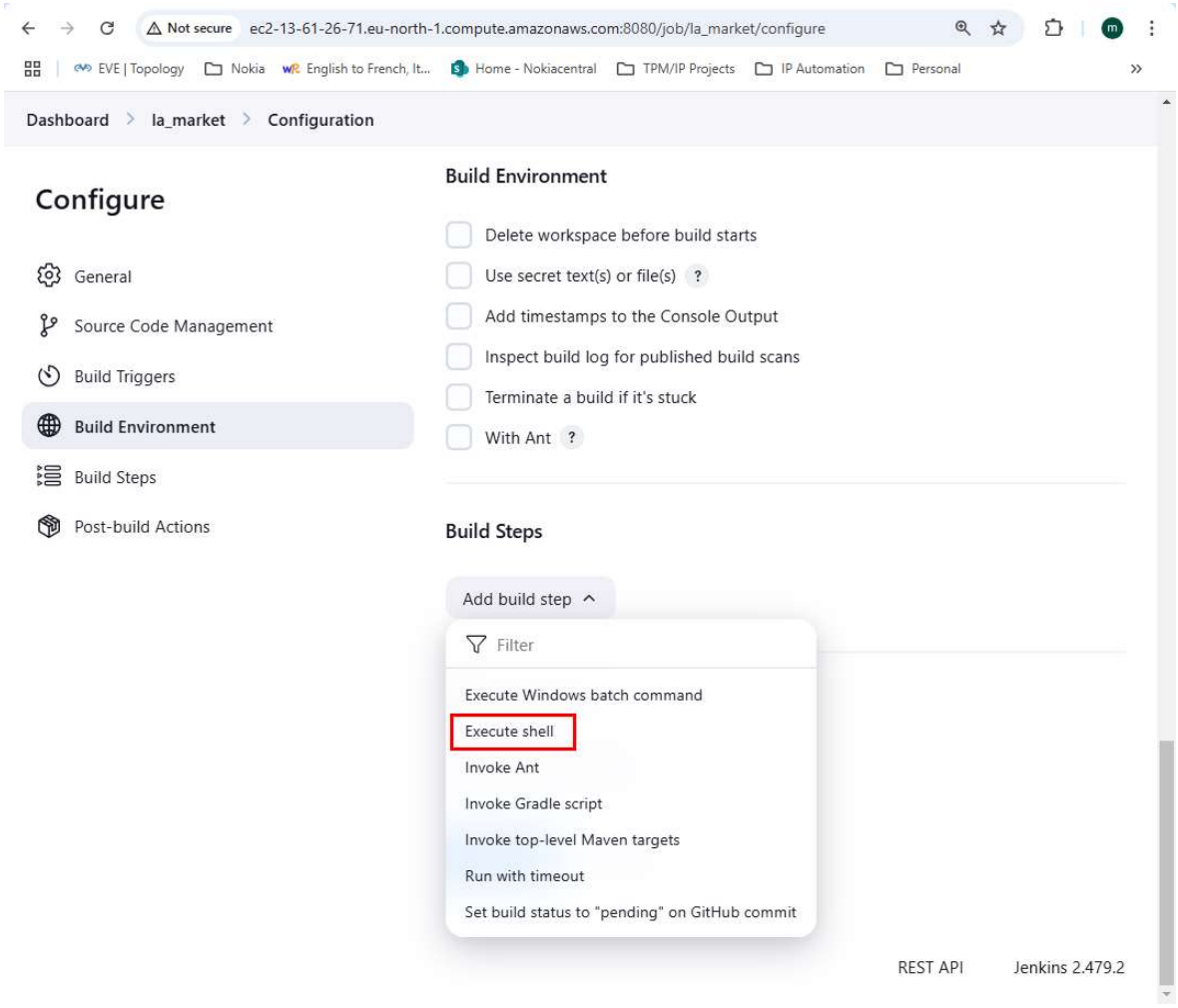
4. The repository URL can be retrieved from the Github’s repository page and is https://github.com/lefonge/la_market_repository.git



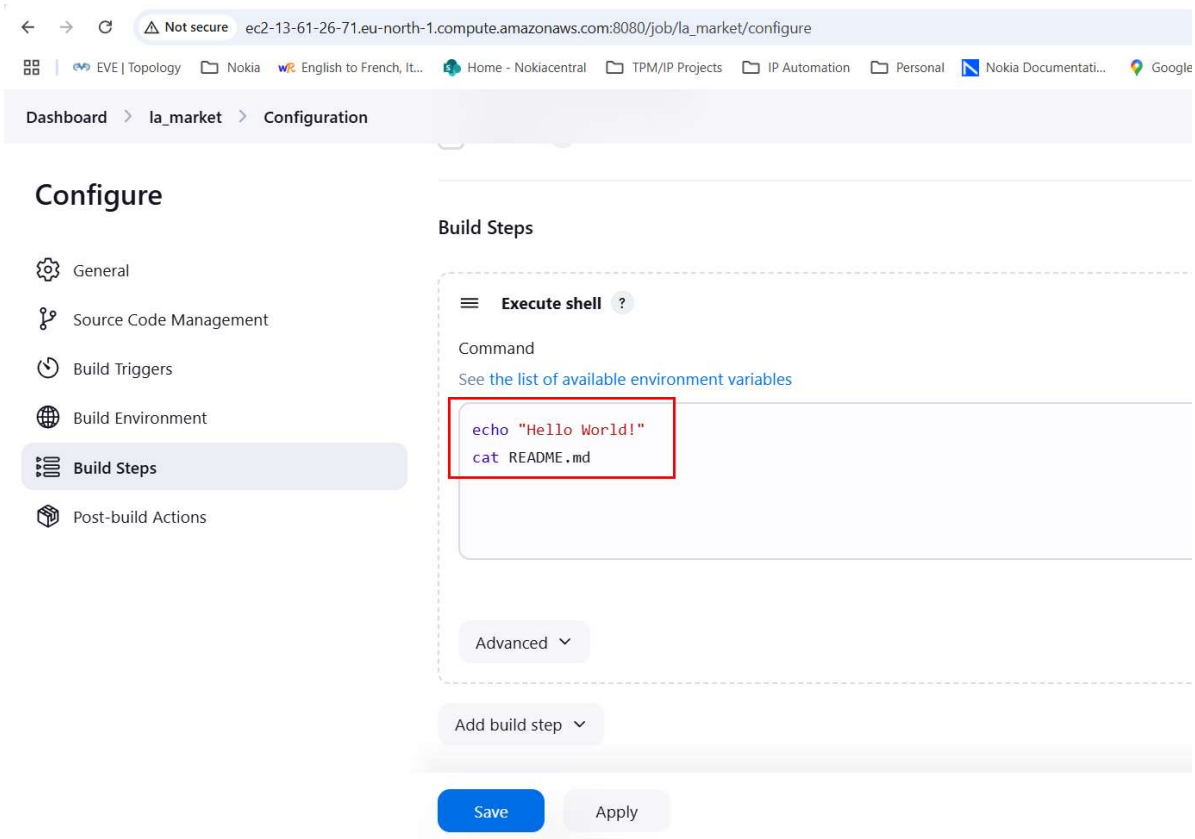
5. Go back to the Jenkins' job page and paste the repository URL in the "Repository URL" field and then change the Branch name from "master" to "main". Credentials in this case are not necessary as the repository is public.



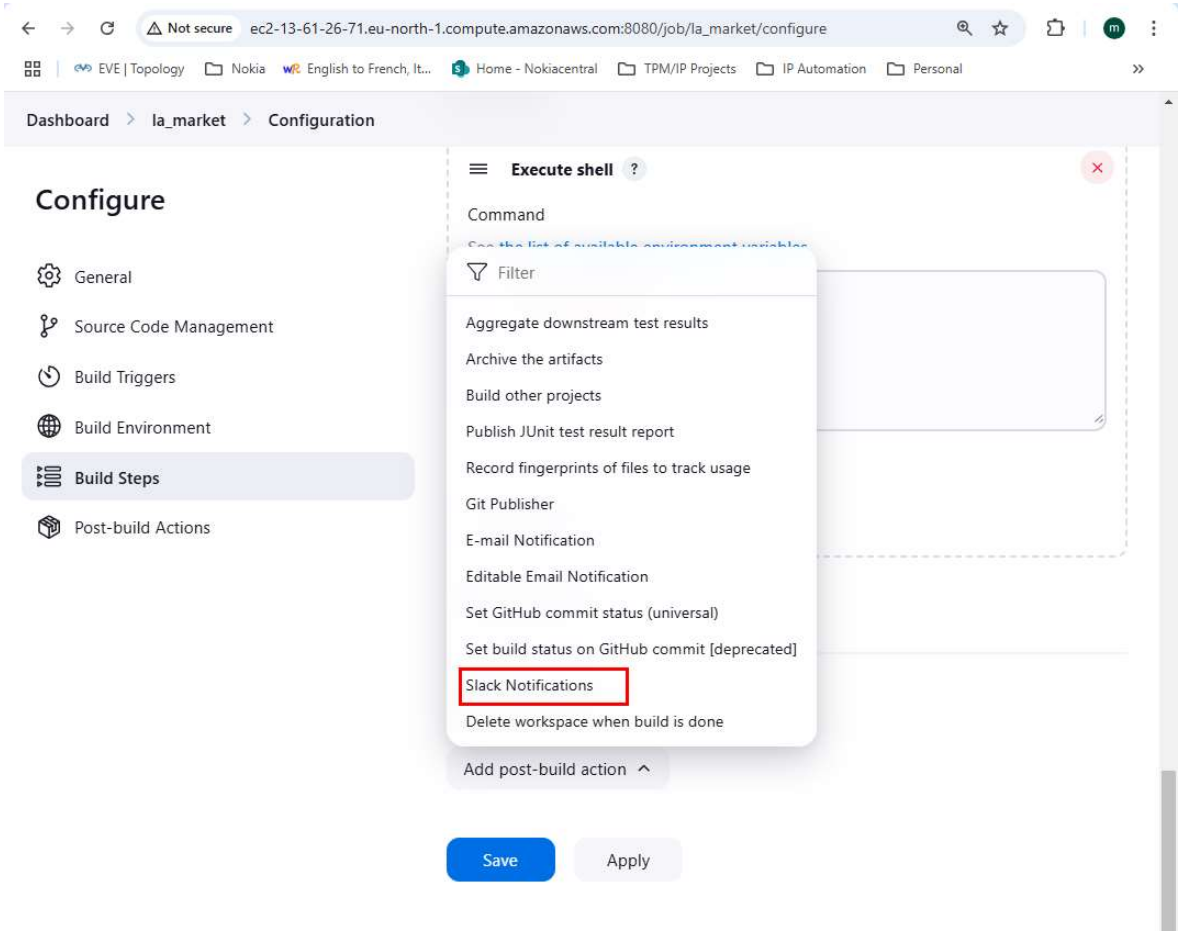
6. Go to the “Build Steps” section, click on the drop-down menu and select “Execute shell”:



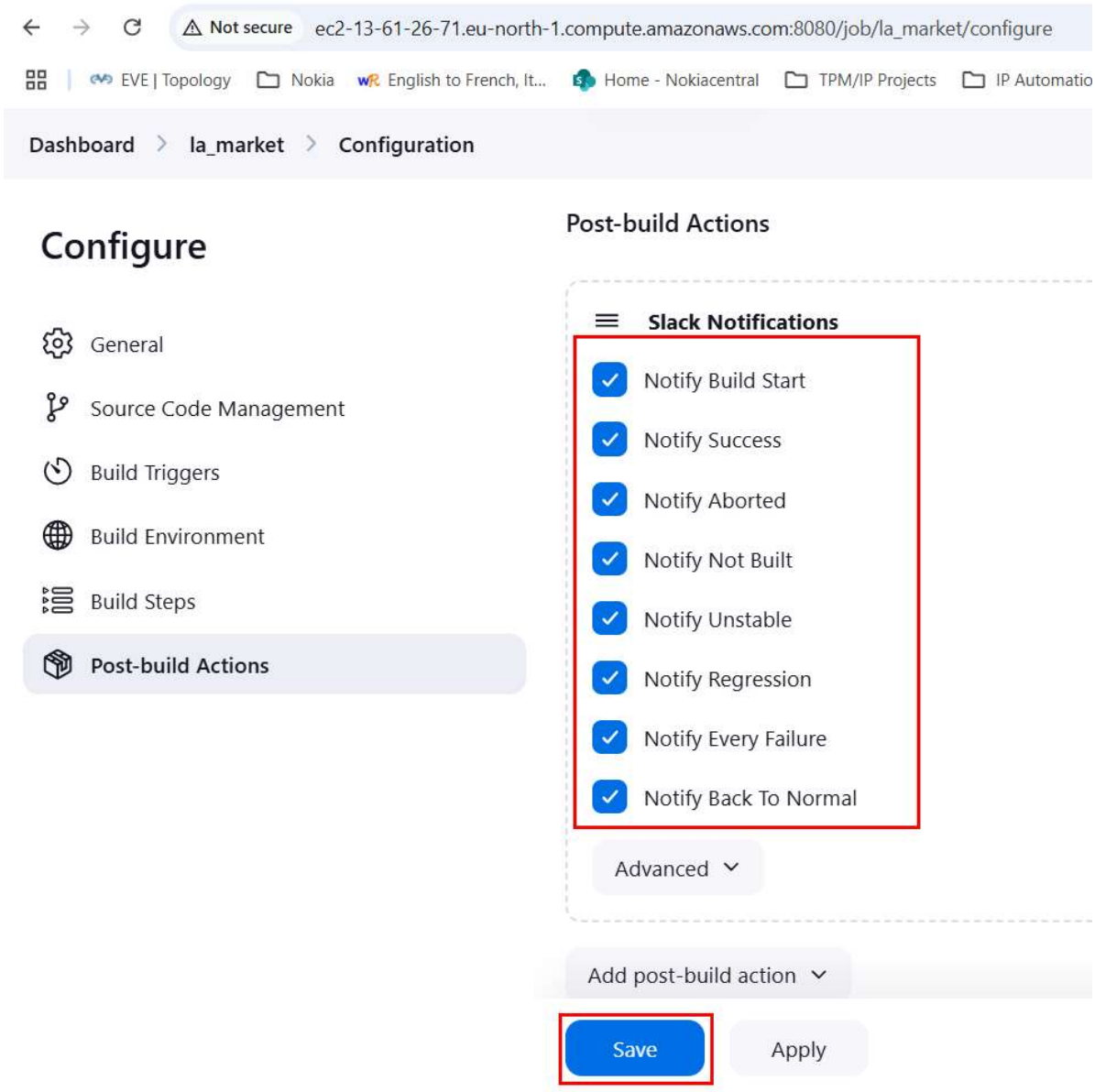
7. Just add an “echo” and a “cat” of the README.md file available from the Github repository, for testing purpose:



8. Go to “Post-build Actions”, click on the drop-down menu and click on “Slack Notification”:

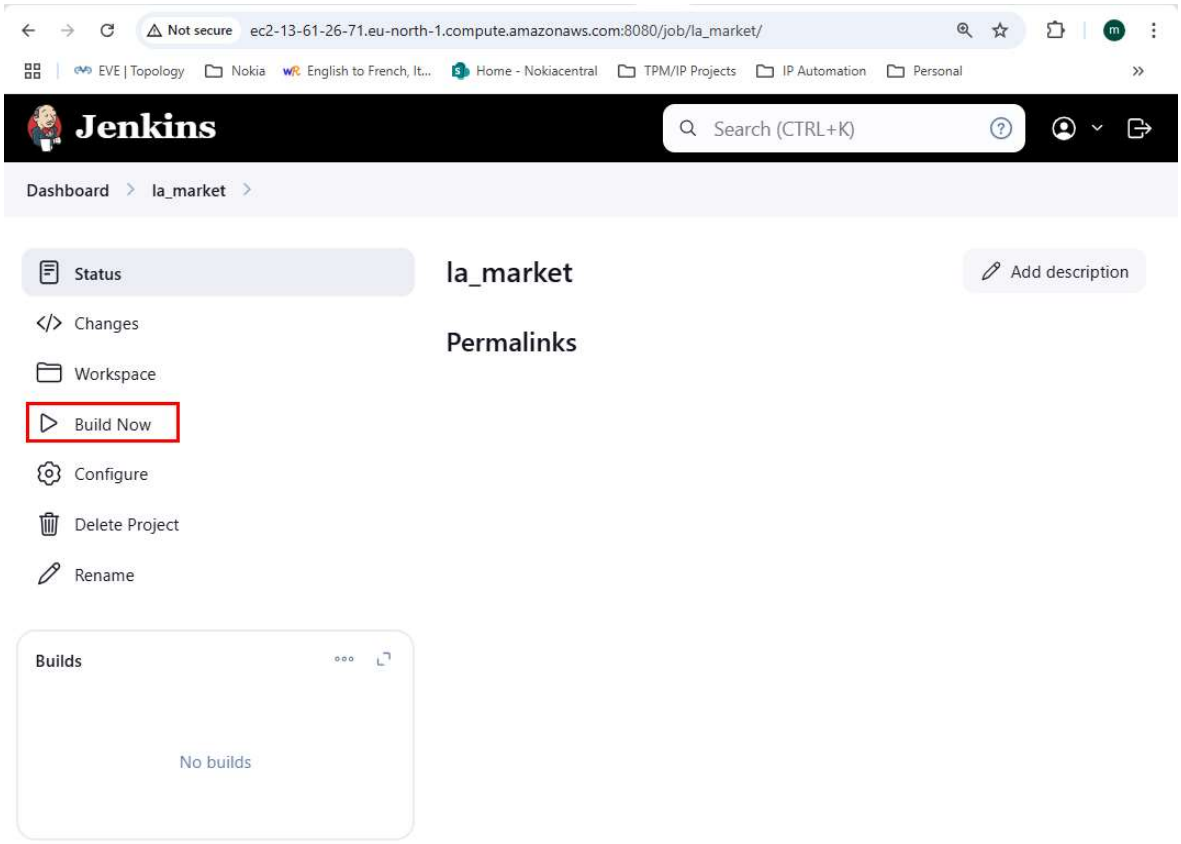


9. Select the following check-boxes and then click on “Save”:

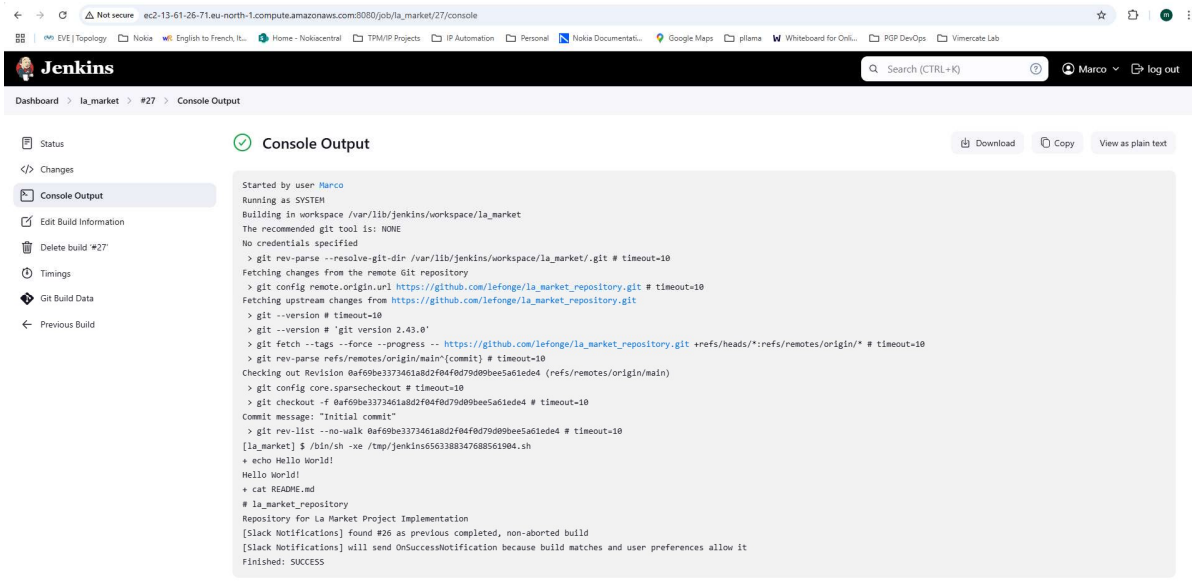


2.6 Testing Slack Notification

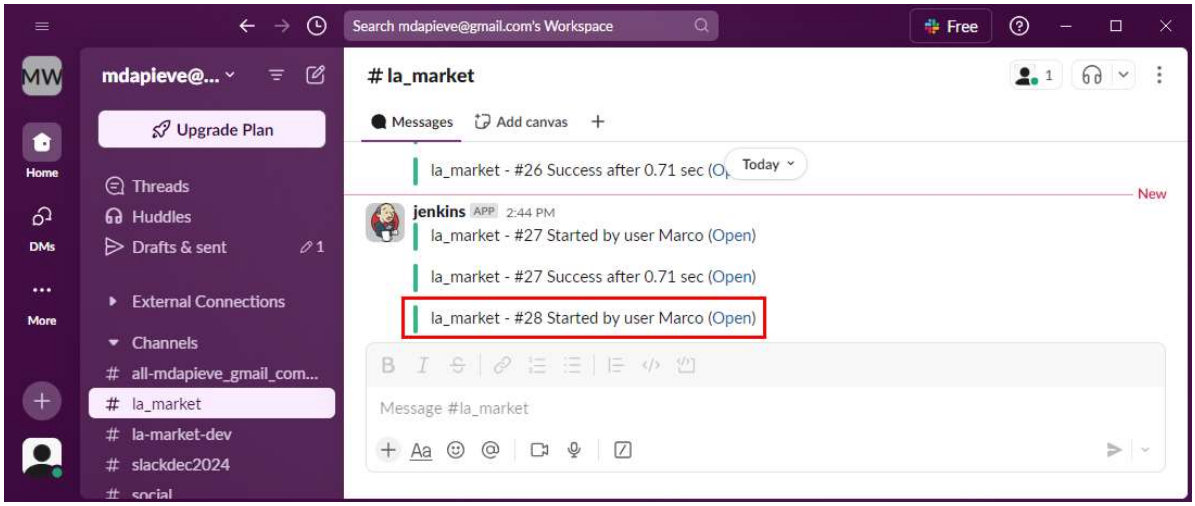
To test the correct functionality of receiving build notification on the slack channel, click on "Build" from the Jenkins' job page:



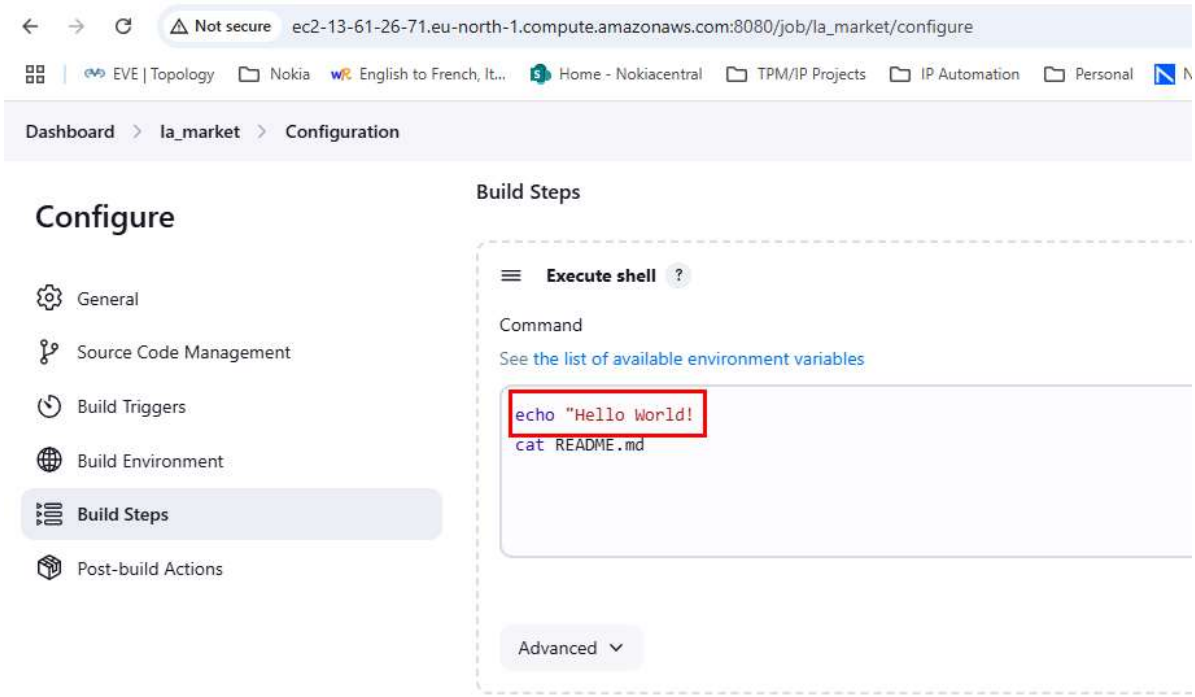
This results in a successful build and the Console output can be checked for analysis



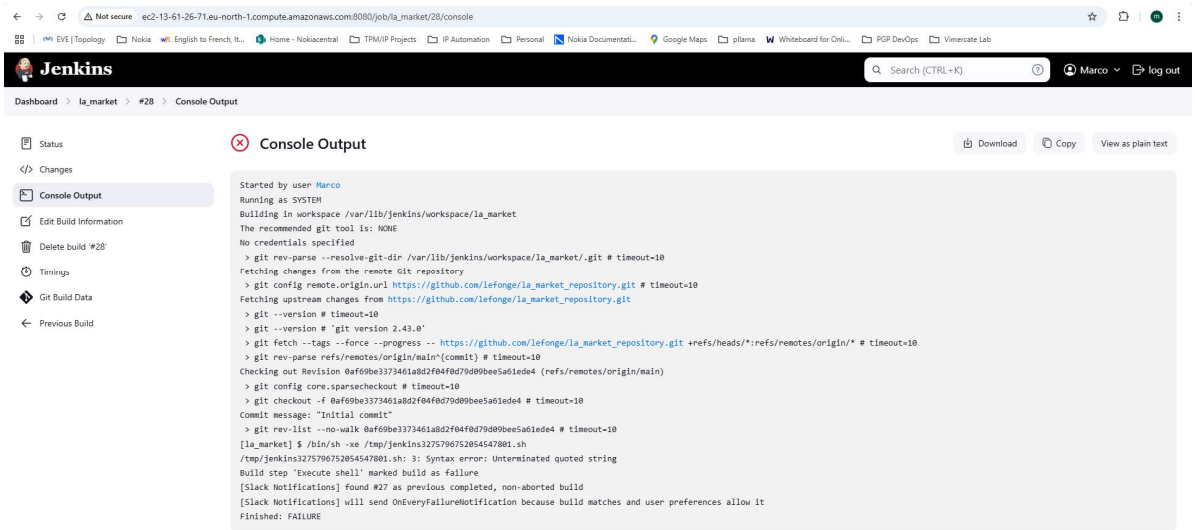
This also resulted in a successful notification on the Slack's channel



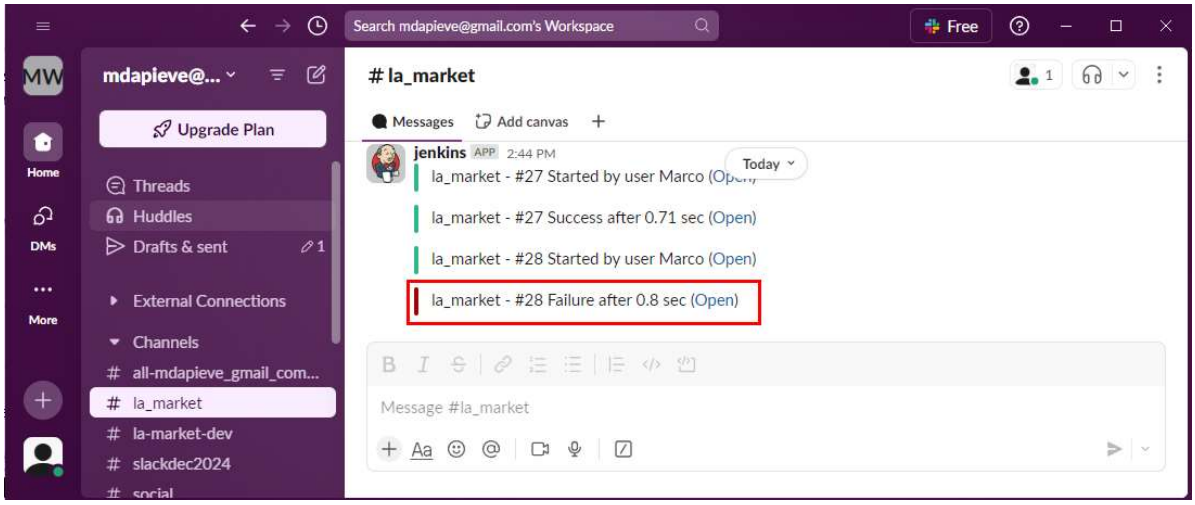
In order to test also the failure notification of the build the “echo” command will be modified to trigger a syntax error in the following way (the closing quotation is missing):



Now the build is failing



And this resulted in a failure notification sent to the Slack's channel



2.7 Triggering builds directly from slack

In this procedure, two ways for triggering a build directly from Slack will be presented:

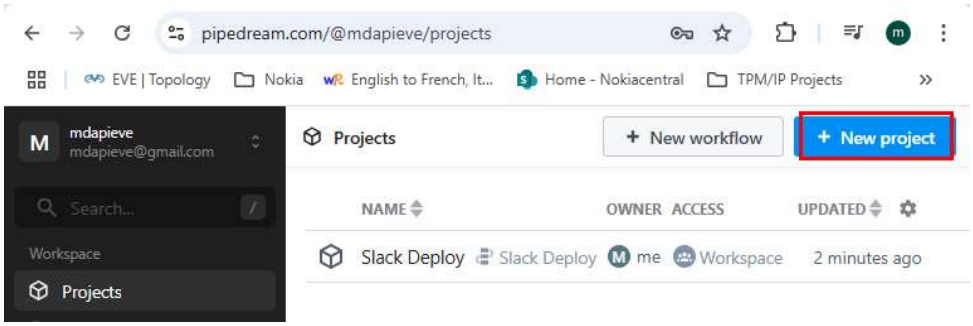
- One makes use of Pipedream and a Slack App that is created for this specific use case (recommended as the second method is a legacy integration method)
- The other makes use of the Slash Command App integration available in Slack

2.7.1 Pipedream solution

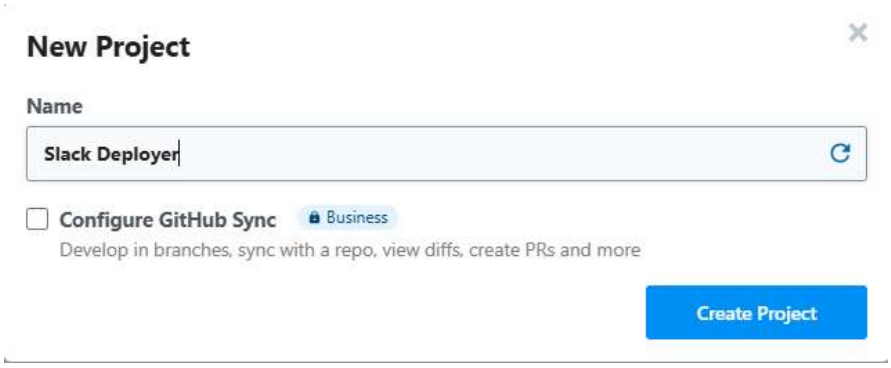
2.7.1.1 Pipedream workflow creation

Before configuring Slack to be able to send a slash command to trigger a build on Jenkins, it is necessary to create a workflow on pipedream.com. Please create an account.

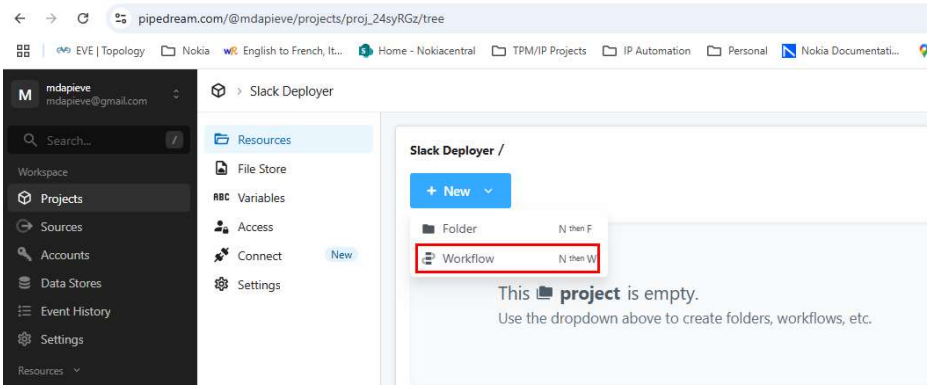
1. For the “Request URL” field required in the Slack App creation, it is necessary to create a URL from pipedream.com. On pipedream a new project needs to be created:



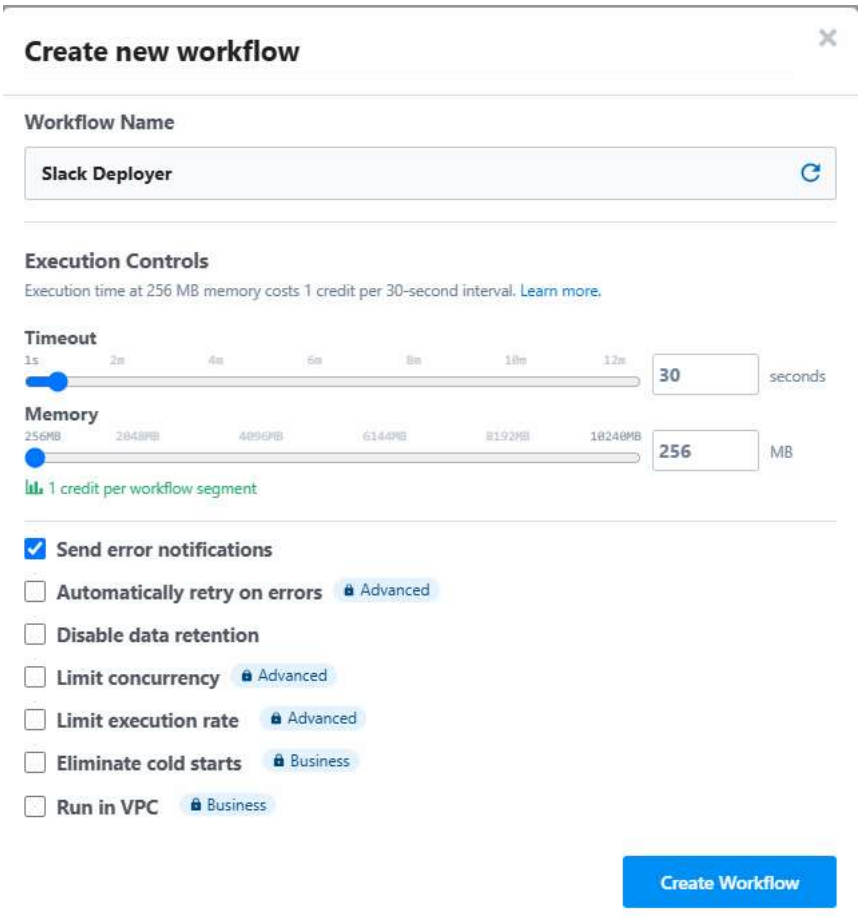
2. Specify a name for the project and click on Create Project



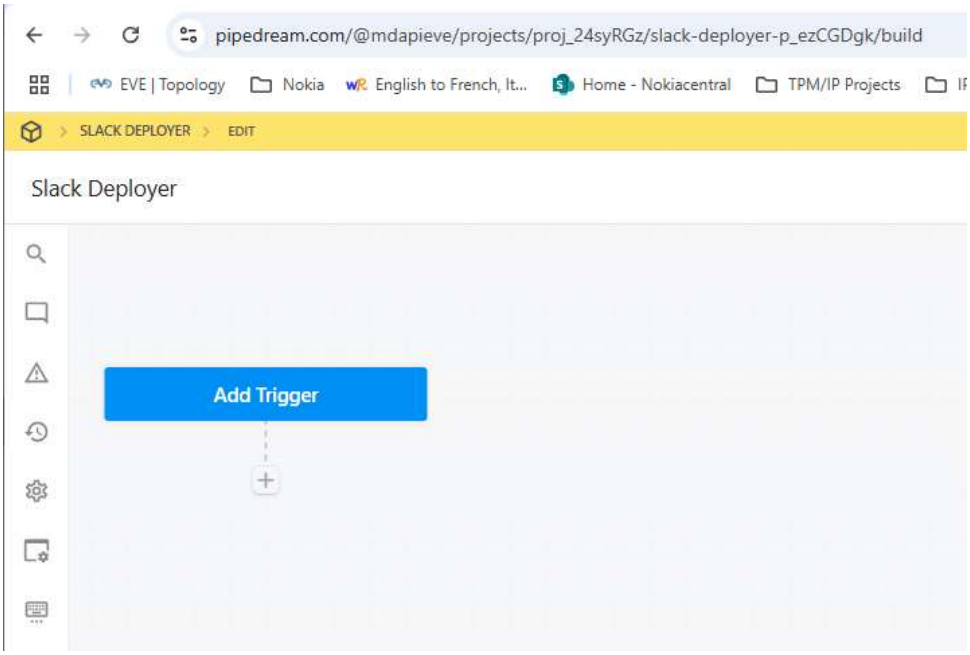
3. Then click on New and Workflow



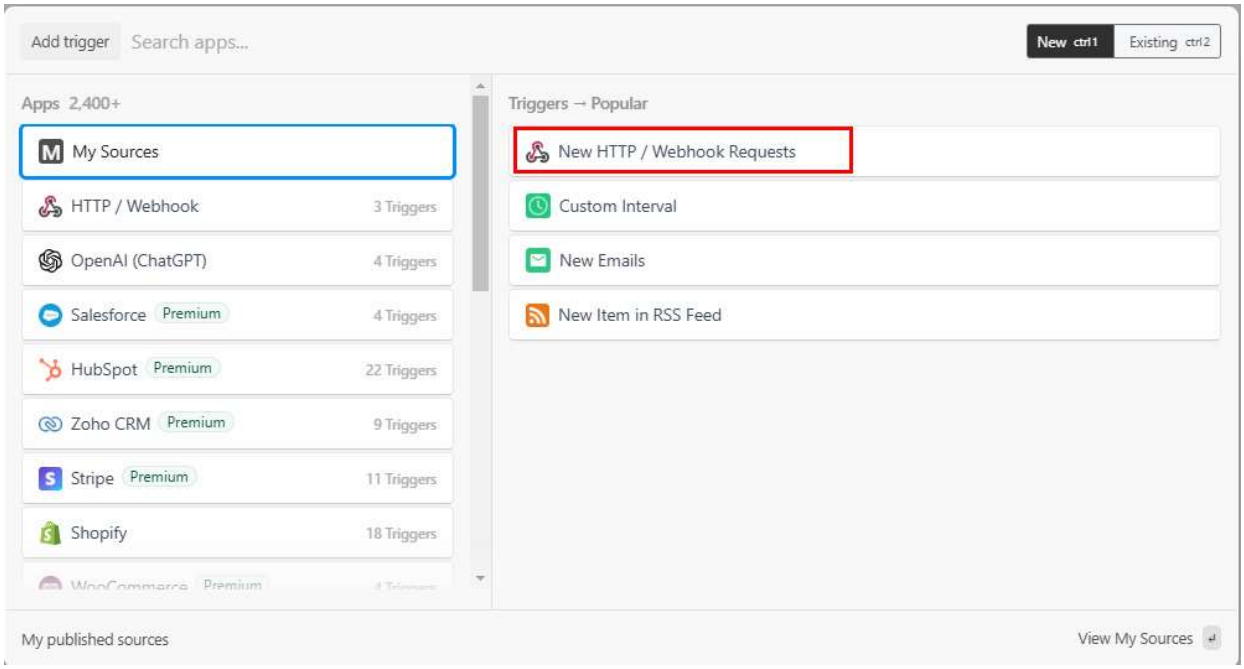
4. Specify a name for the workflow and click on Create Workflow



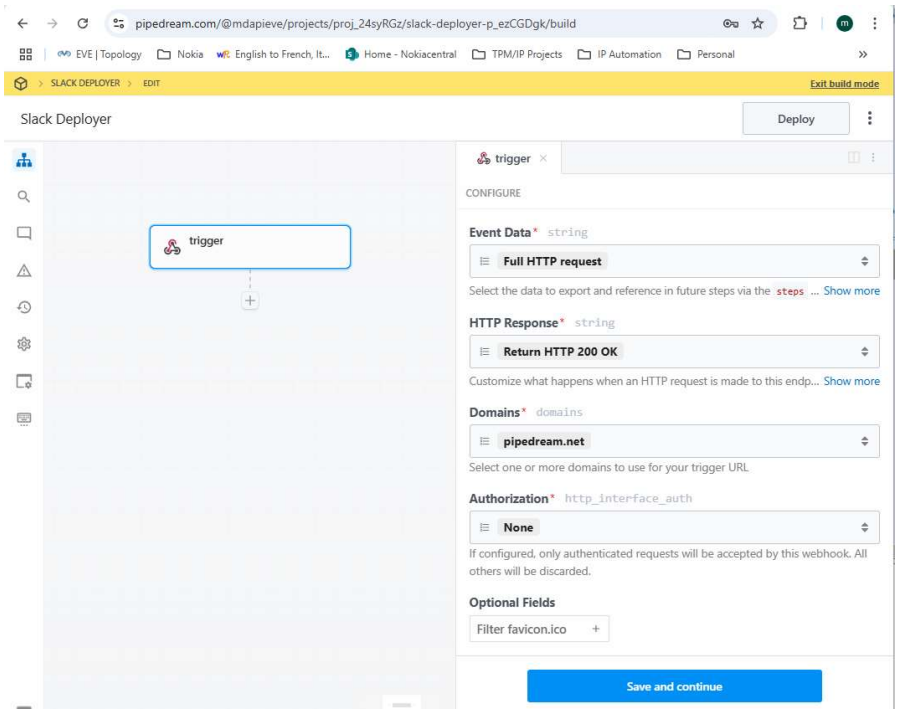
5. Add a trigger



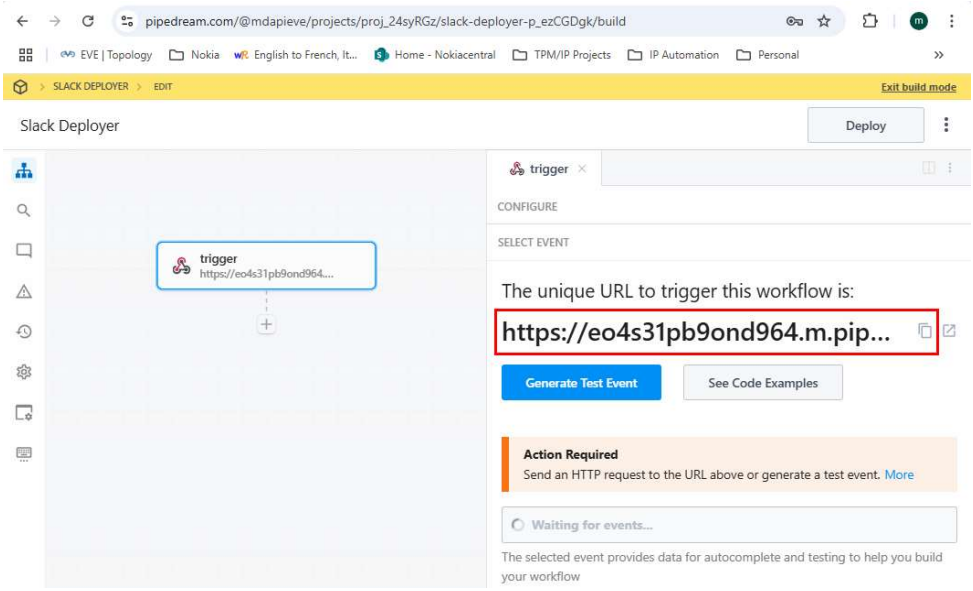
6. Select New HTTP/Webhook Requests



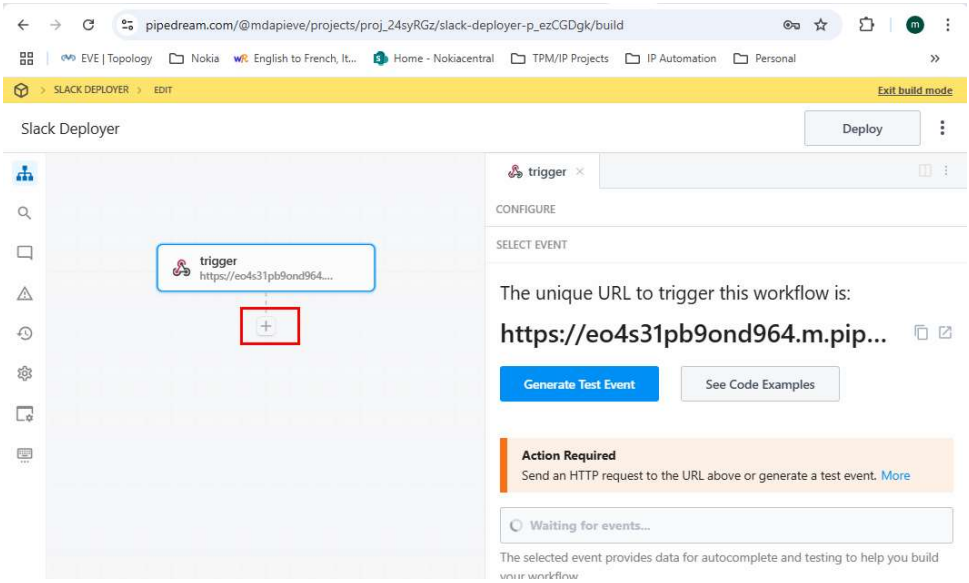
7. Then click on Save and Continue



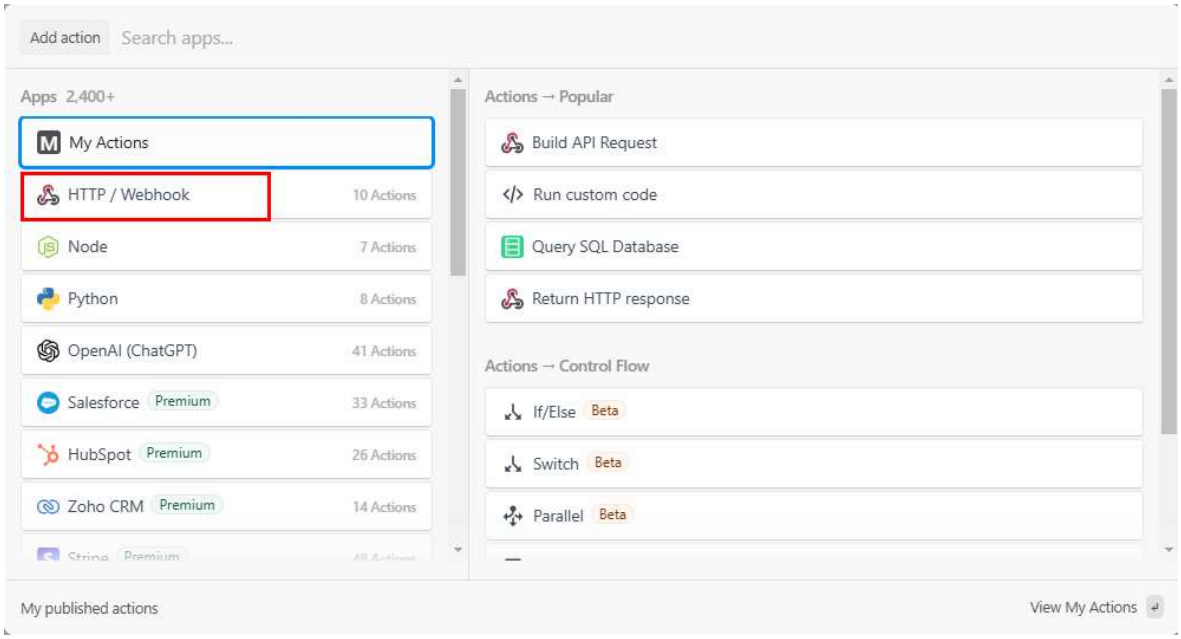
8. Copy and paste the URL somewhere as it will be needed later on during the Slack App creation for the slash command.



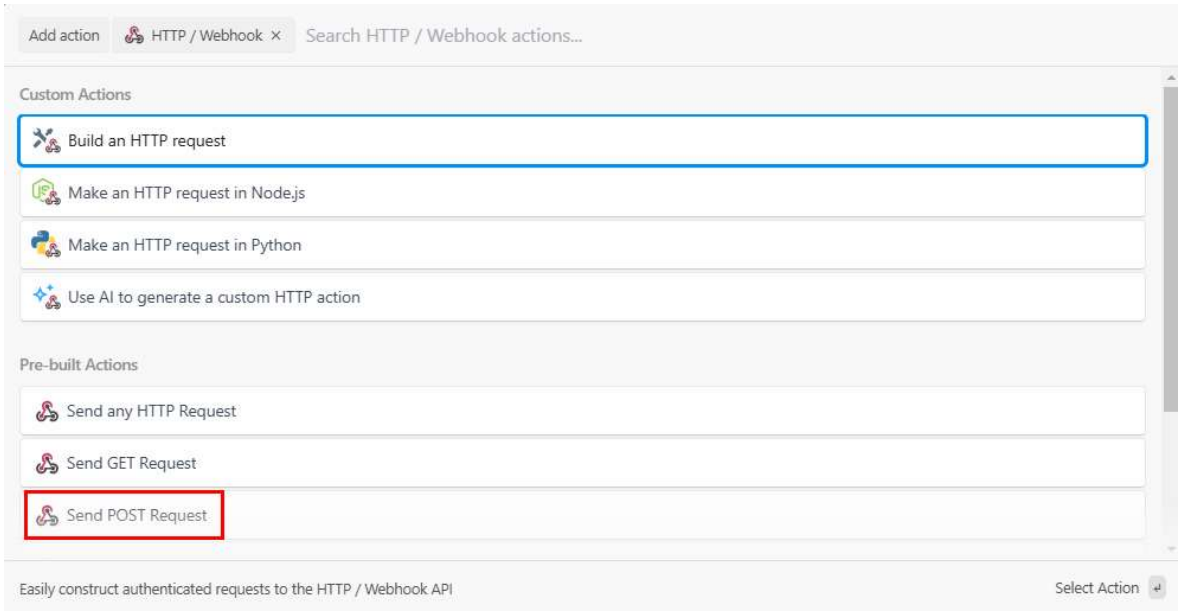
9. Another action needs to be added to this workflow which will actually send the HTTP POST Request to the Jenkins server to trigger the build. To do this, click on the "+" sign button



10. Select "HTTP/Webhook"

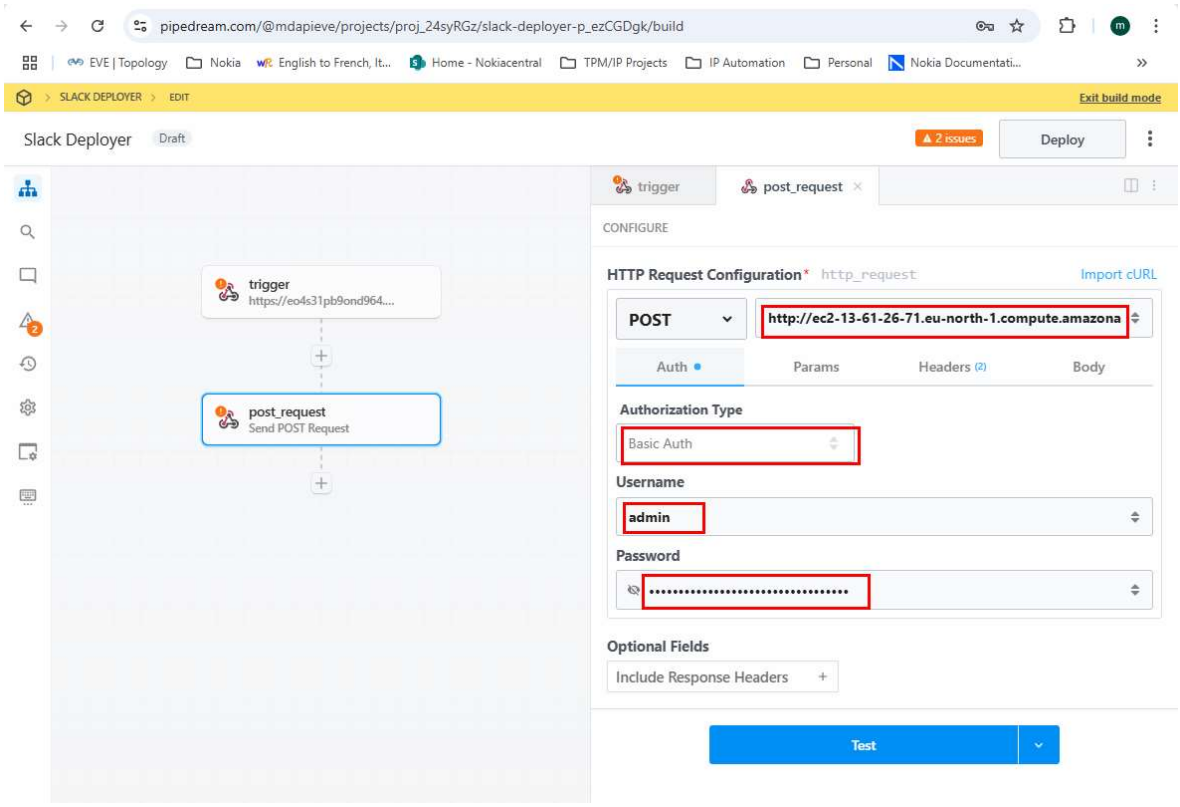


11. And then select “Send POST Request”:



12. Fill the required fields as below:

- The URL is the publicly reachable URL of the Jenkins server
- The Authorization Type is “Basic Auth” which provide the possibility to specific a username and password
- Username is the Jenkin’s username
- Password is the token created on Jenkins to allow API integration

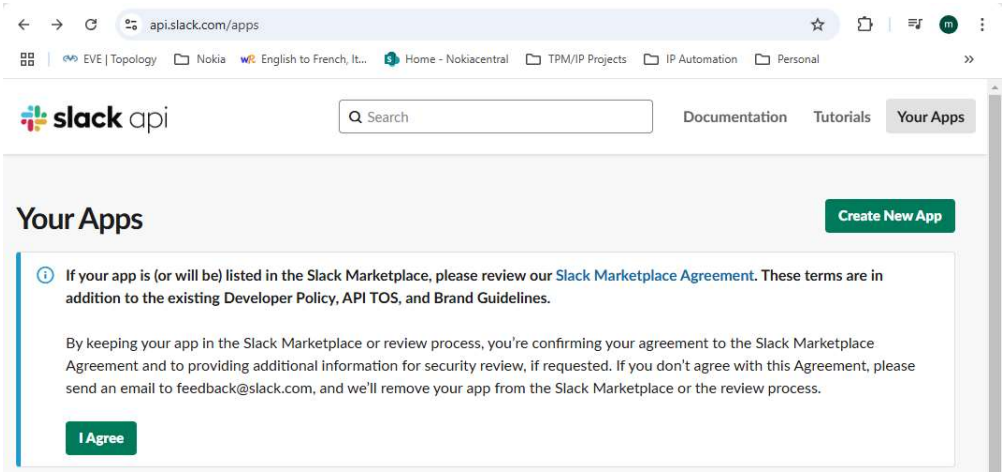


13. At the end of this process, click on the “Deploy” button.

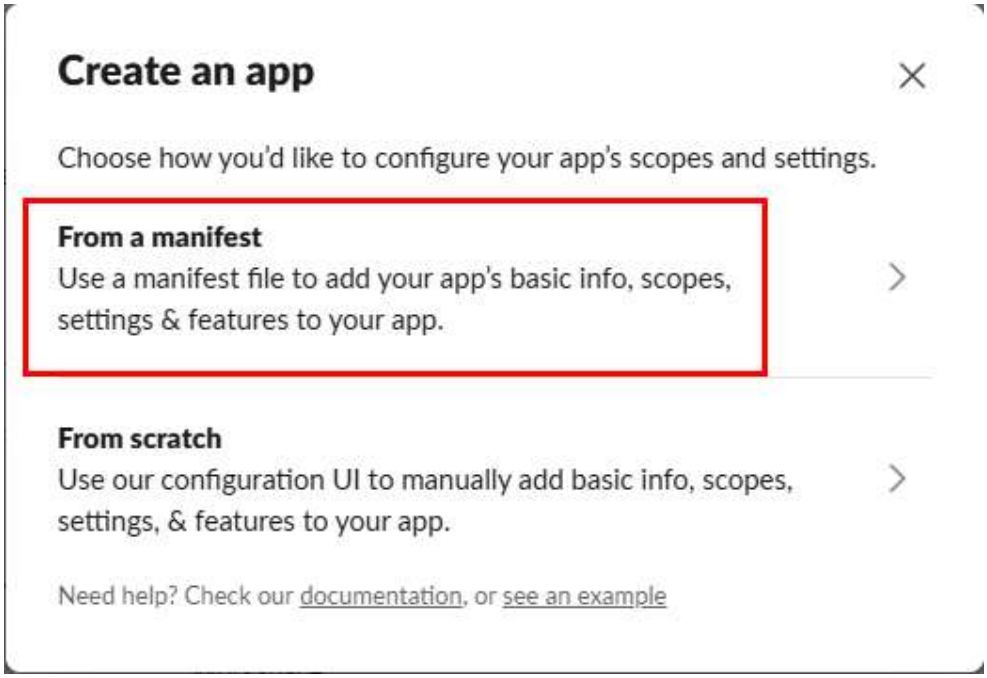
2.7.1.2 Slack app creation

Now a Slack App can be created.

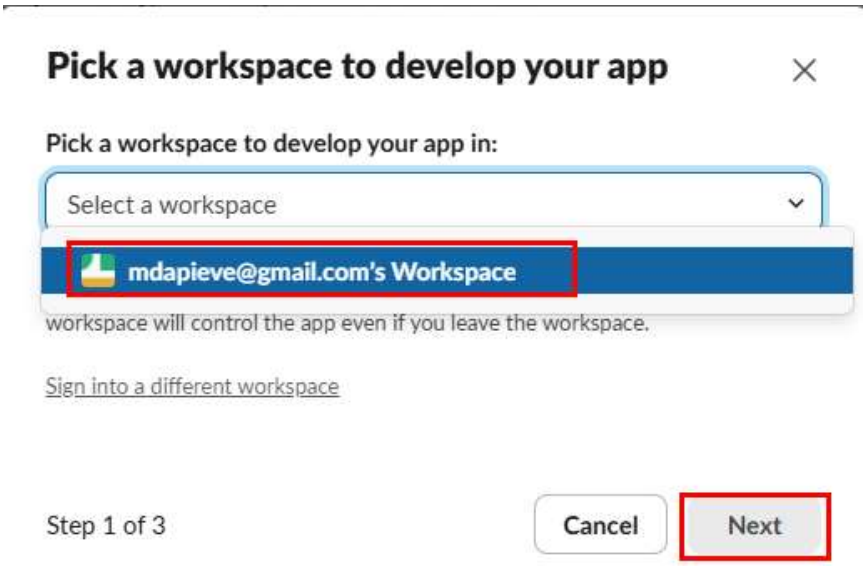
1. Access the api.slack.com/apps and click on “Create New App”



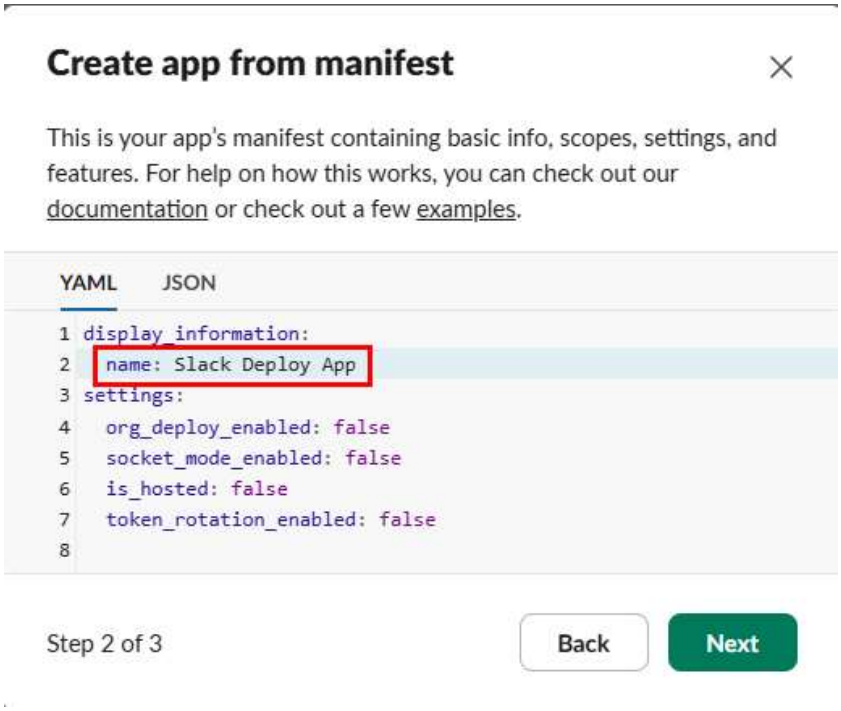
2. Click on “From a Manifest”:



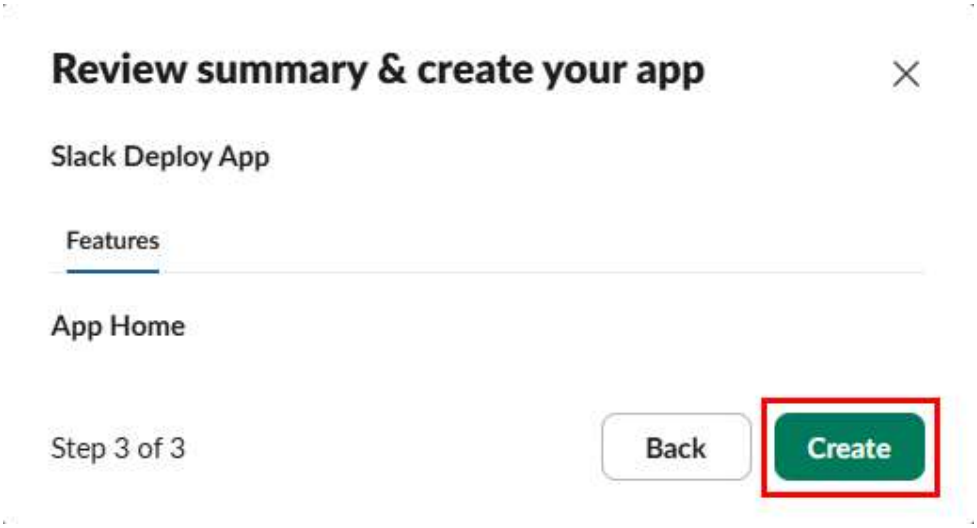
3. Select the workspace



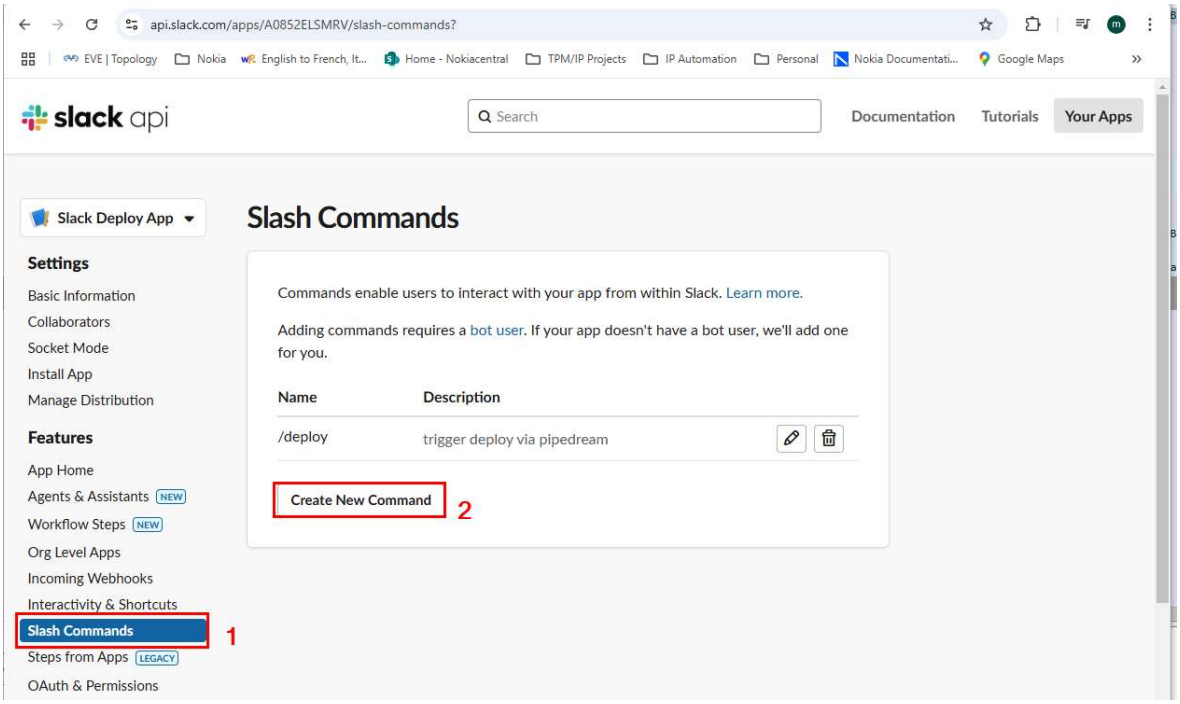
4. Edit the name of the App as you like:



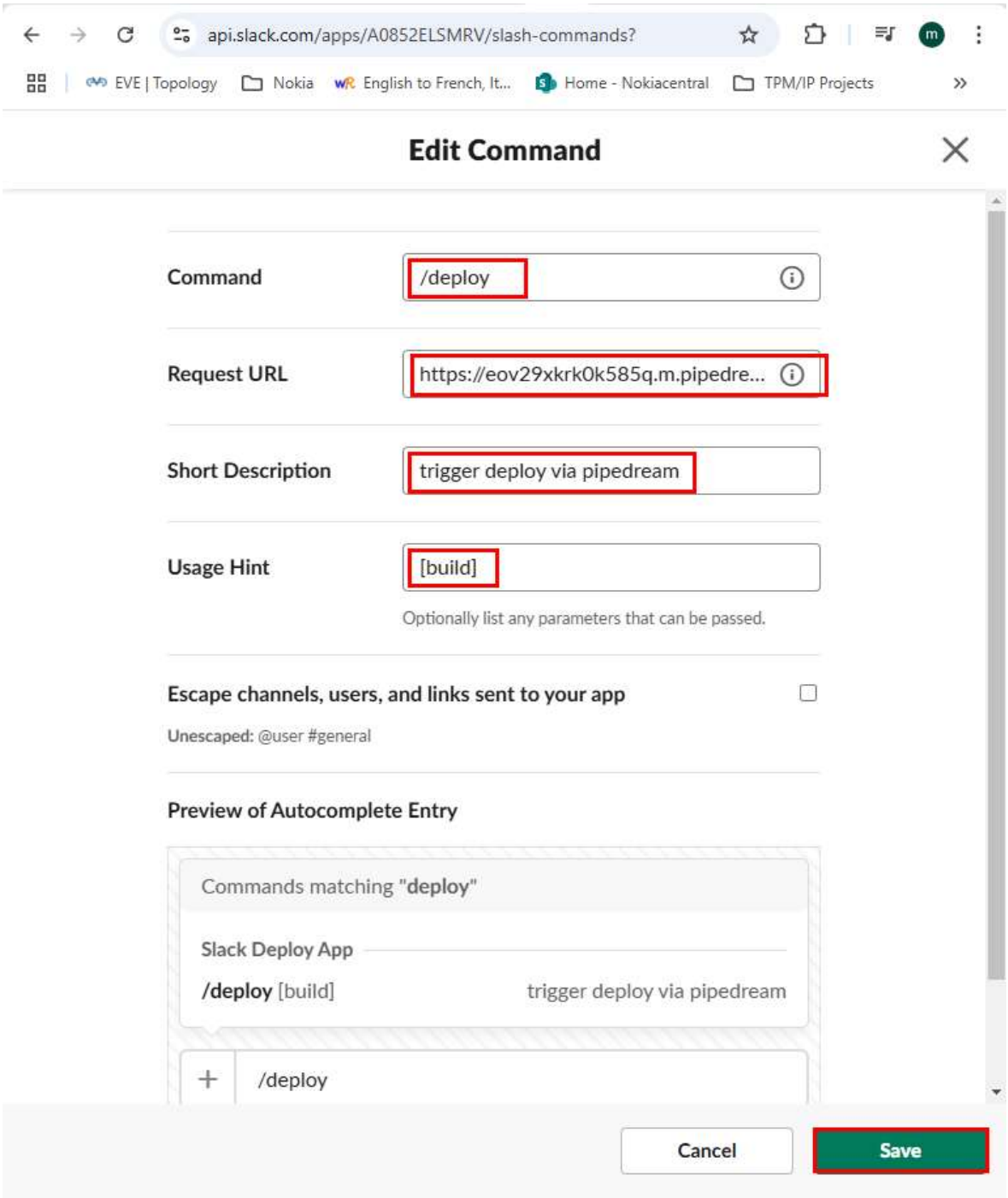
5. And then Create:



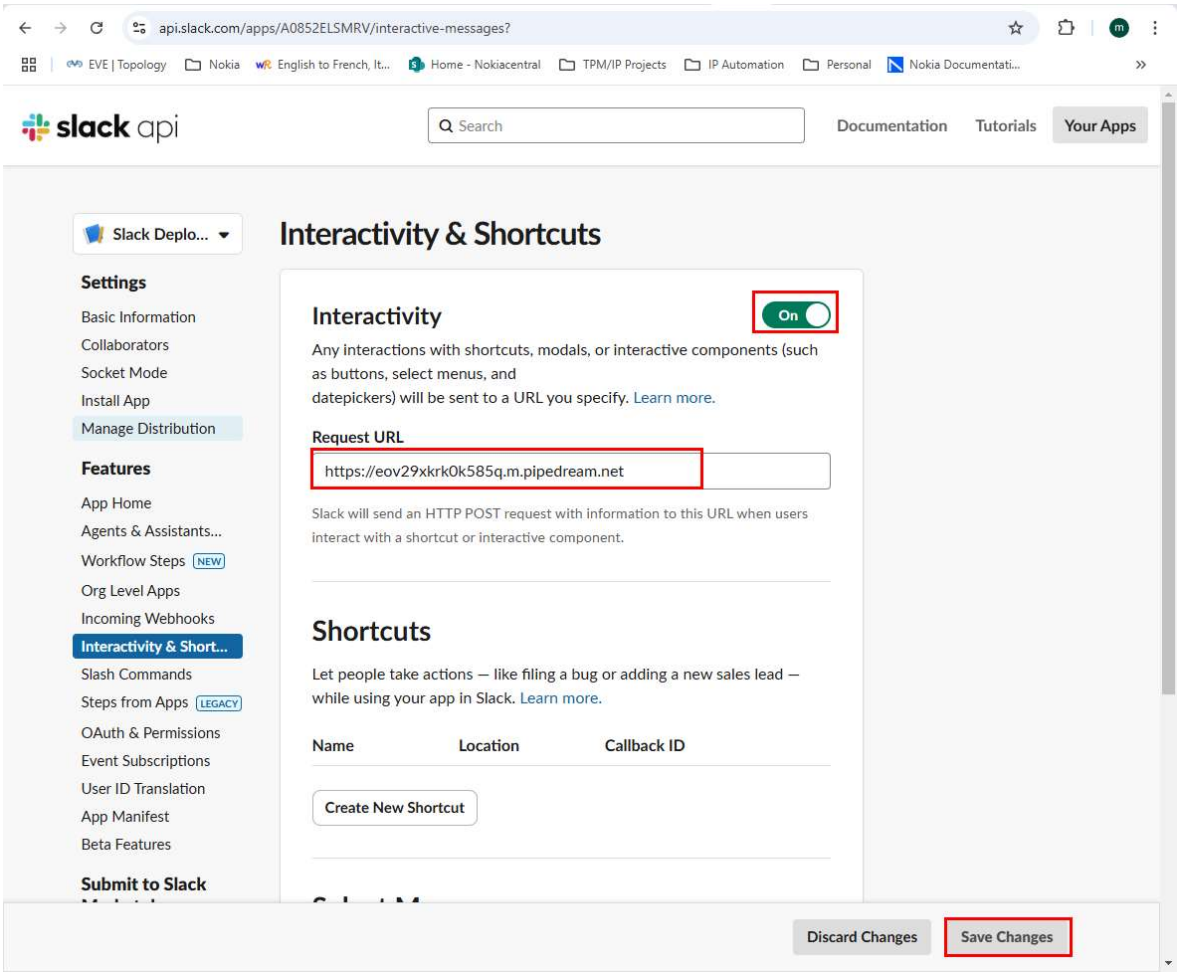
6. Then switch to the “Slash Commands” section and click on “Create New Command”:



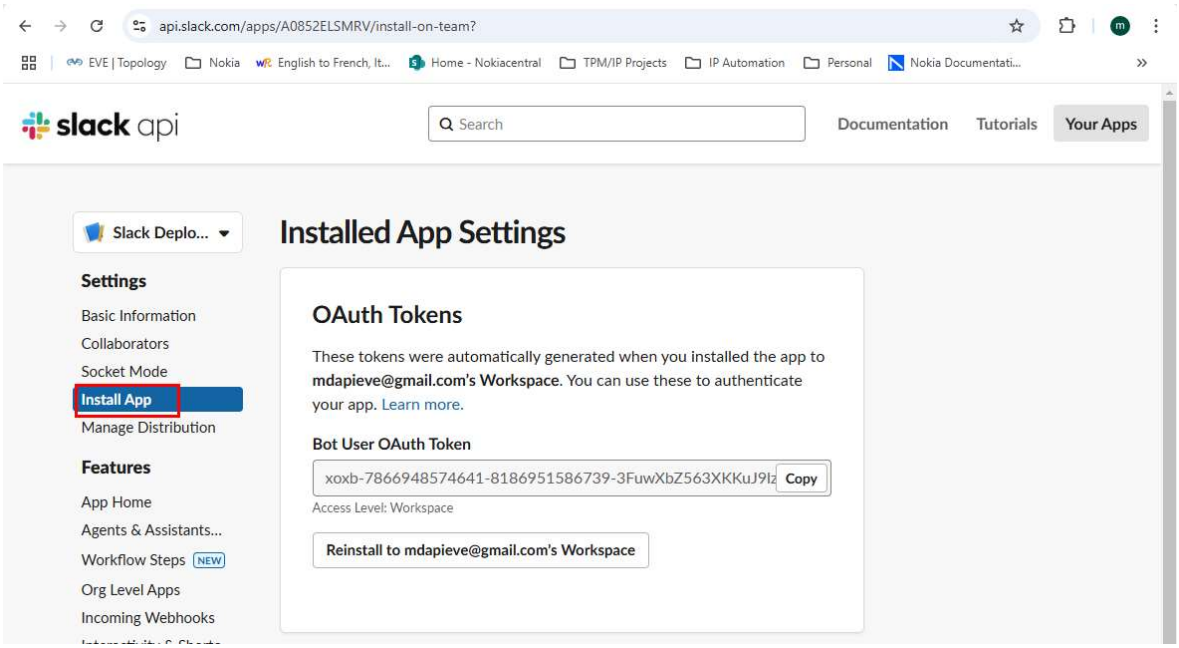
7. Fill the fields below as needed (For the Request URL, please refer to step 8 of the previous section) and click on Save:



8. Then switch to “Interactivity and Shortcuts” and follow the steps below. Please note that the URL is the same as the screenshot above (i.e. step 8 of the previous section):

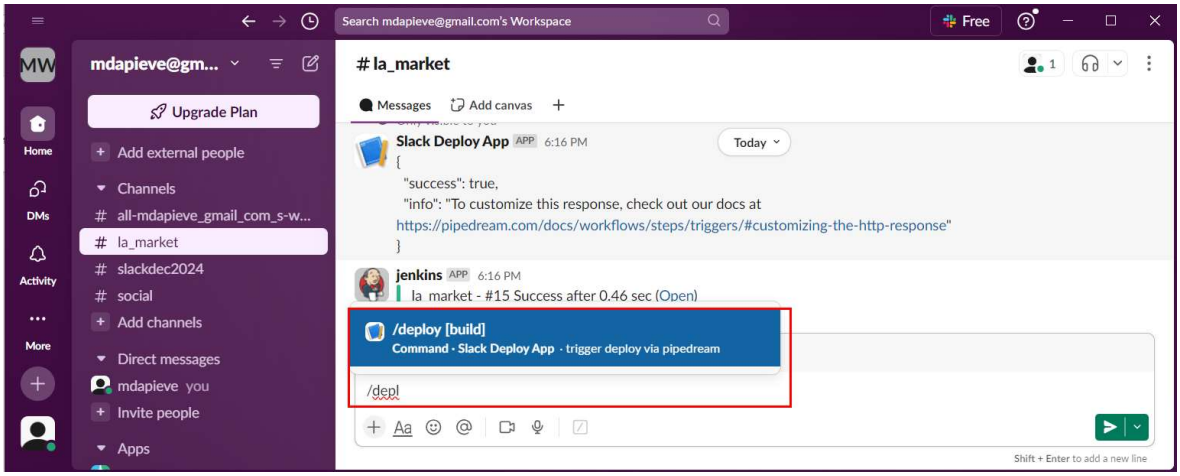


9. Then switch to the “Install App” to install the app (in the screenshot below, the App is already installed):

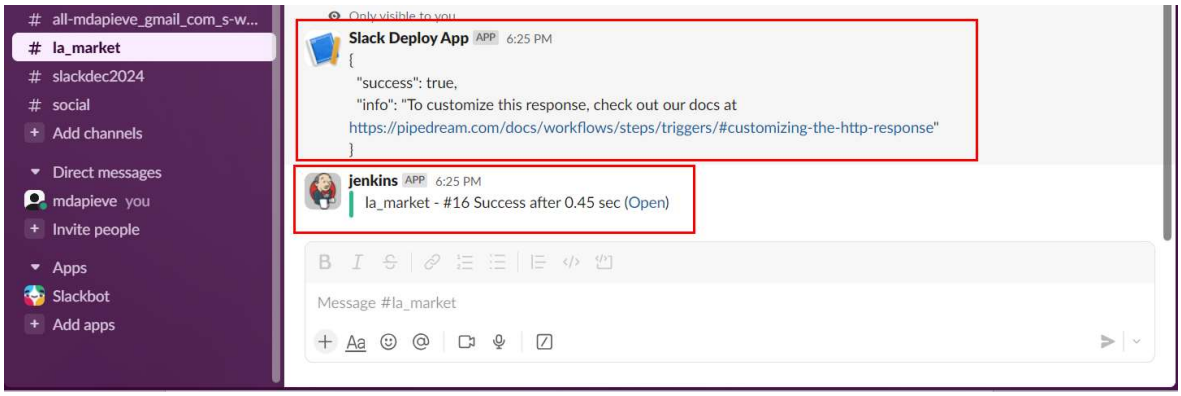


2.7.1.3 Testing

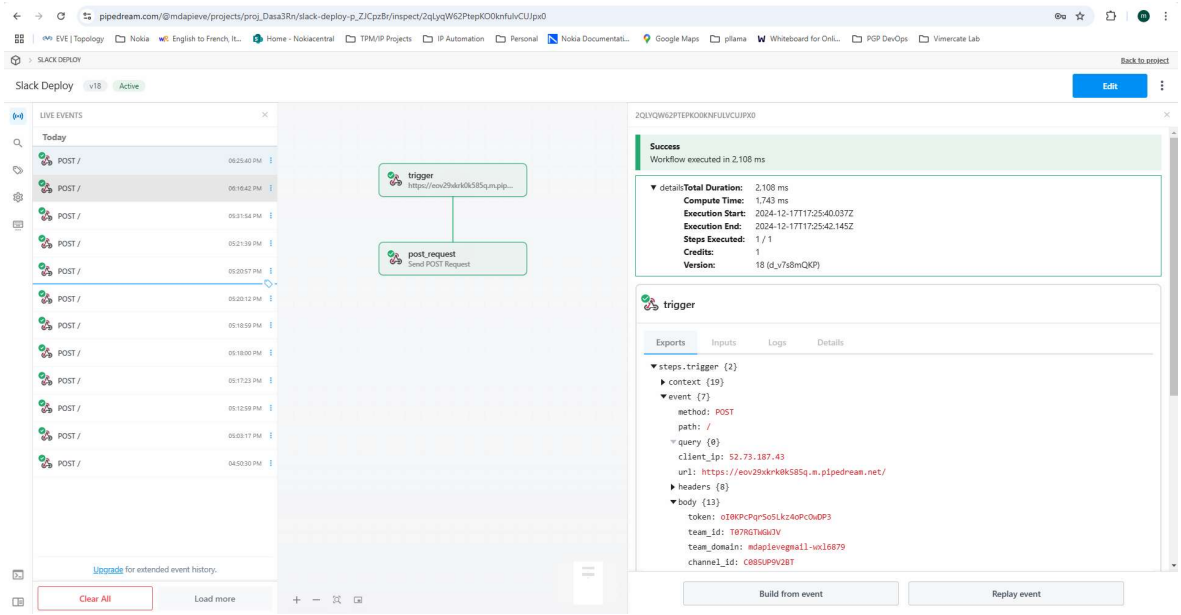
Now the /deploy command is available on Slack to be entered:



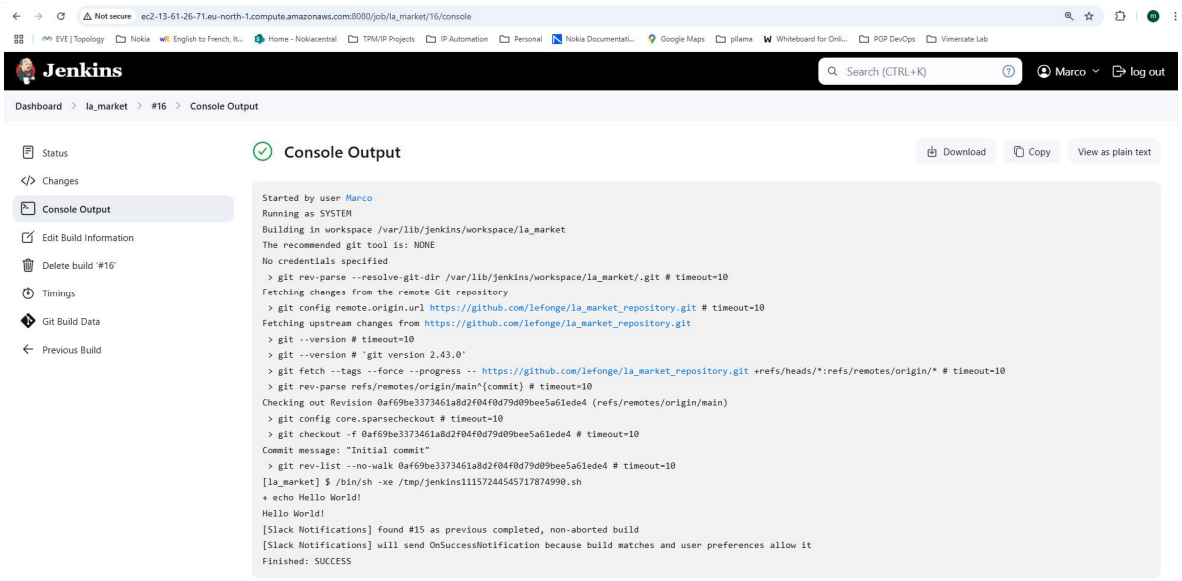
By issuing the command, Slack will trigger the pipedream workflow to send the HTTP POST REQUEST to the Jenkins Server:



This can be further analyzed on the pipedream dashboard:



And finally, this can be viewed also on Jenkins where the Build has been successfully executed:



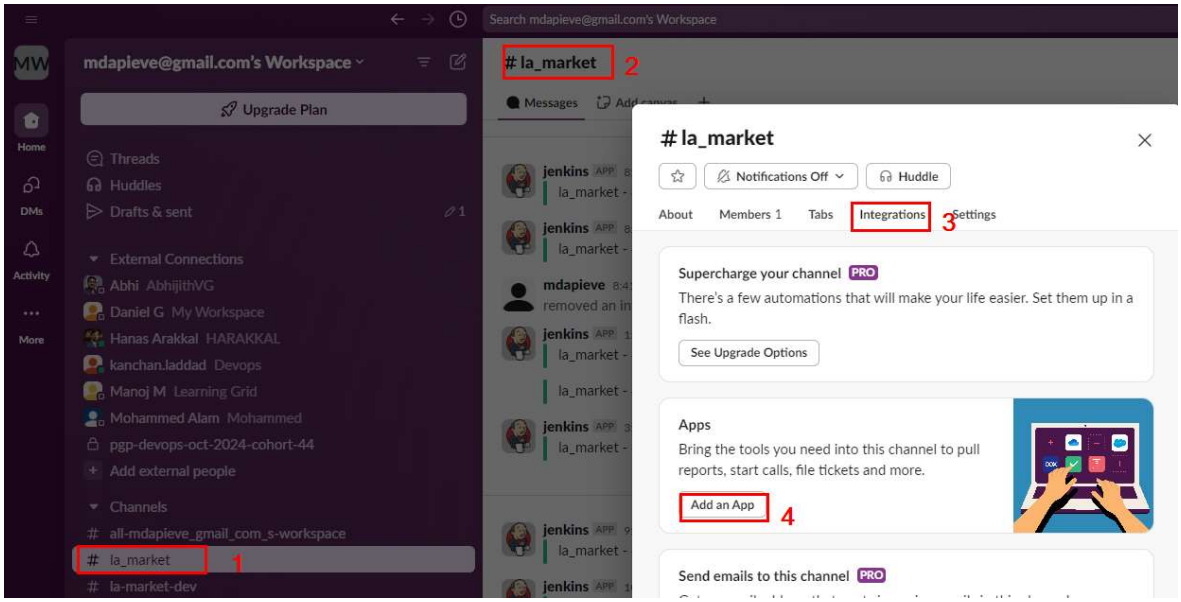
2.7.2 Slash Command Integration Solution

There are two alternatives to implement the solution by using the Slash Command Integration. The first shown here does not make use of an additional plugin and a GET request is sent from Slack to Jenkins to build the job. The second way is to use a plugin named “Generic Webhook” plugin to execute the job when Jenkins receive a POST Request from Slack.

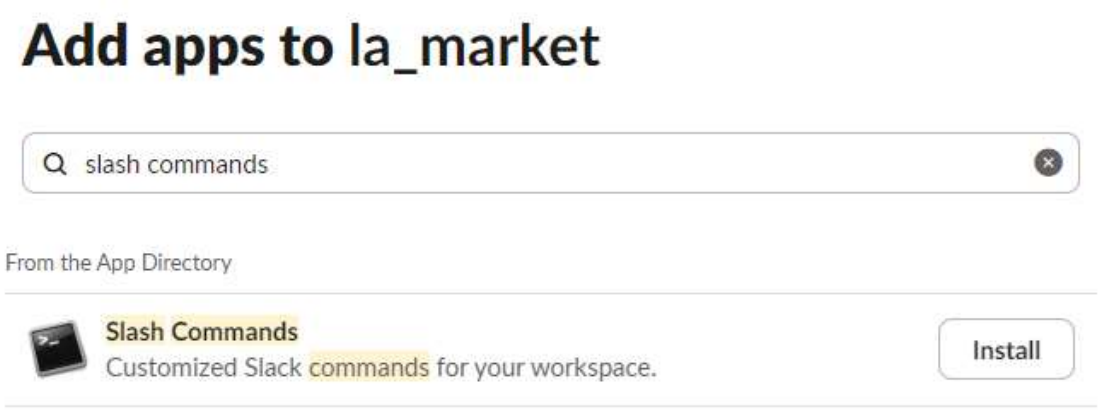
2.7.2.1 Legacy Integration configuration – GET Method

It is also possible to achieve the task of triggering a Jenkins job from slack by using a legacy integration of the slash command in Slack.

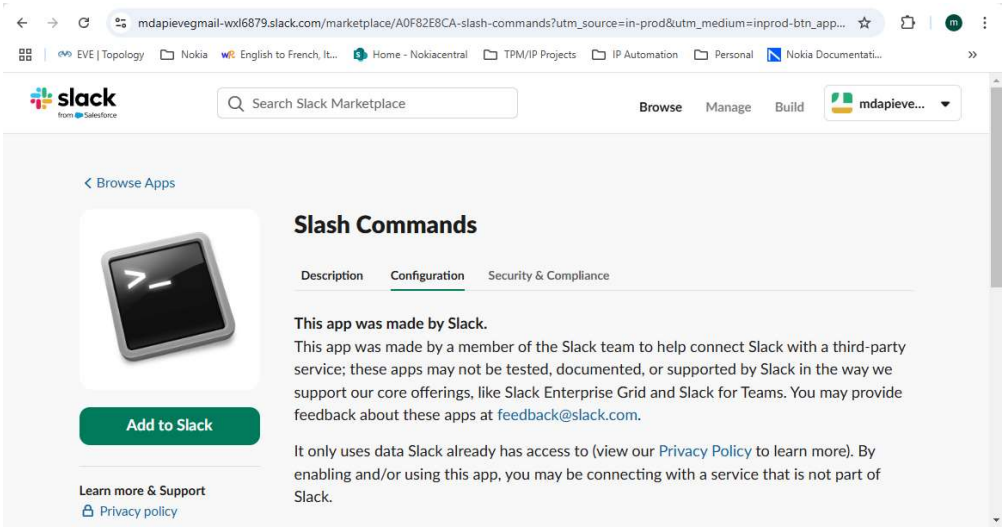
1. To do this, it is required to add Slash Command integration into slack and configure it. From the Slack channel, perform the steps below:



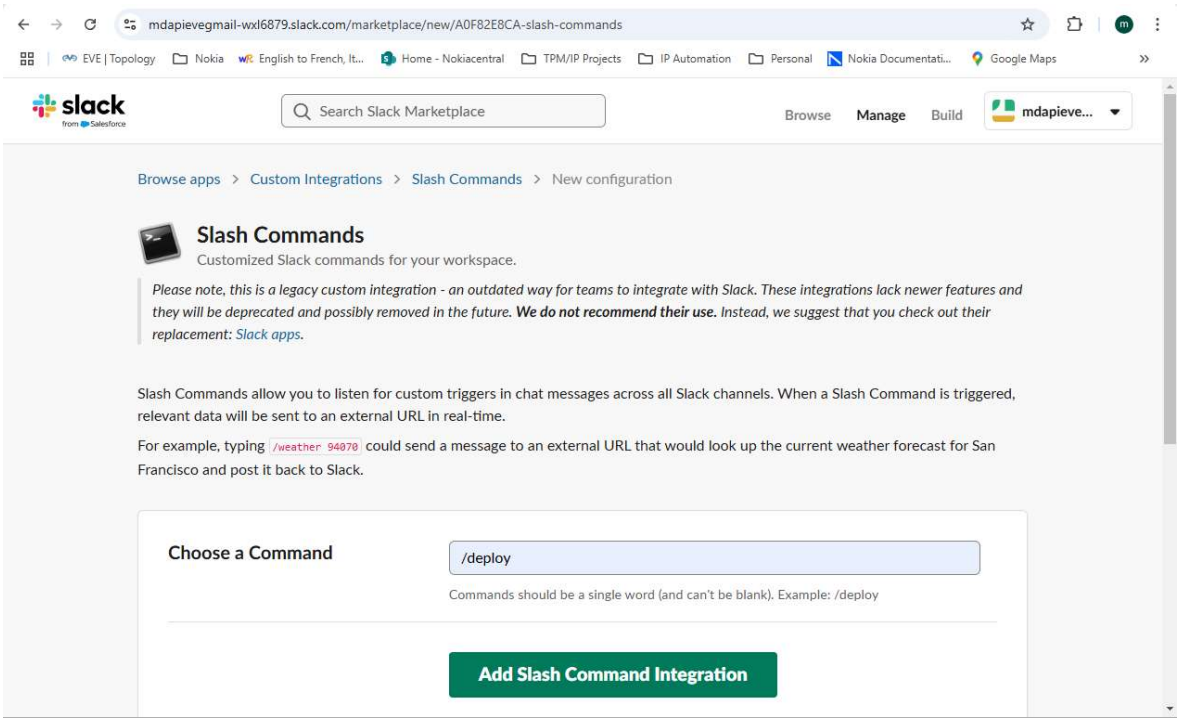
2. Look for “Slash Commands” and click on “Install”



3. A web page opens, click on “Add to Slack”



4. Specify the command that will be issued on Slack channel and that will trigger the job on Jenkins:

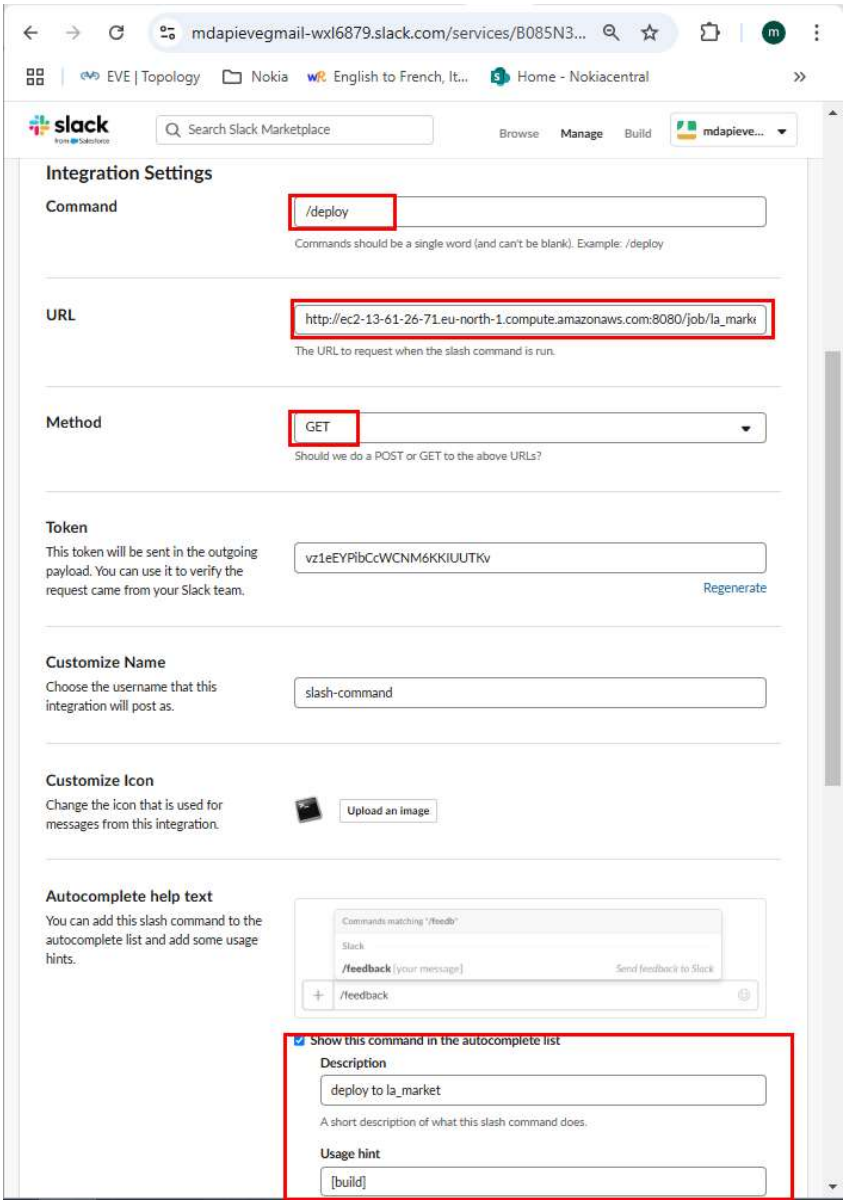


5. Configure the following field as specified below. Take into account that the URL needs to be in this form:

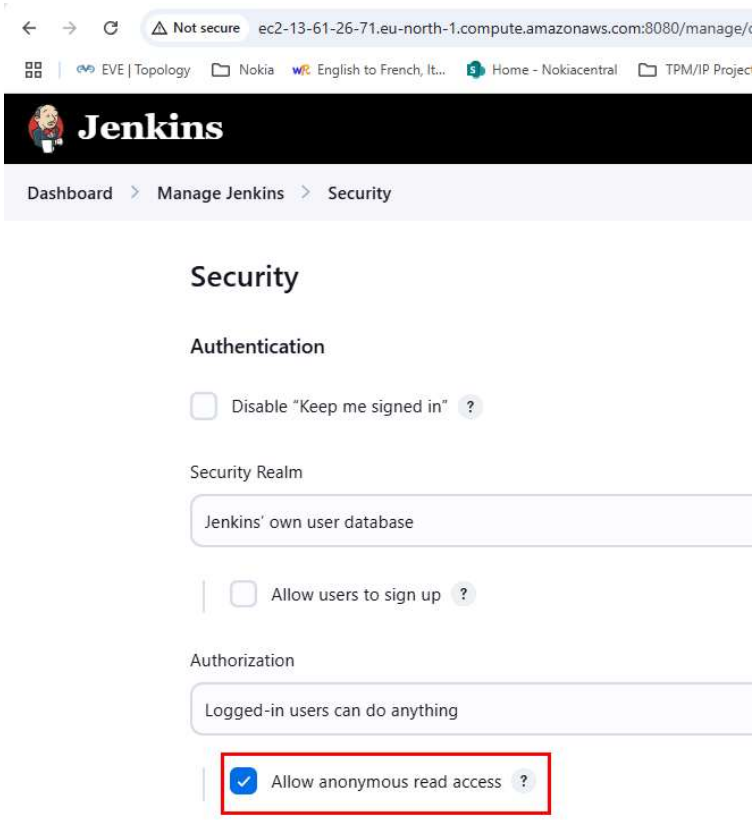
   http://ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080/job/la_market/build?token=token123

   a general form of the URL is:

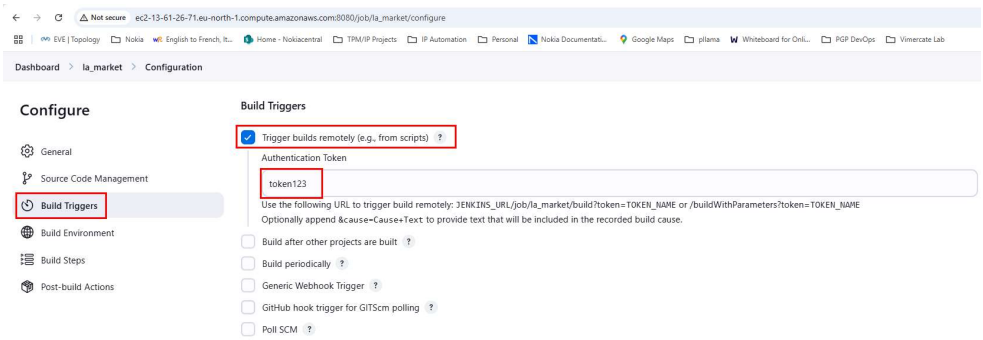
   http://*ipaddress|DNS*:*JenkinsPort*/job/*job_name* /build?token=*token_name*



6. Finally click on “Save Integration”. Then, go to Jenkins and check the box below:

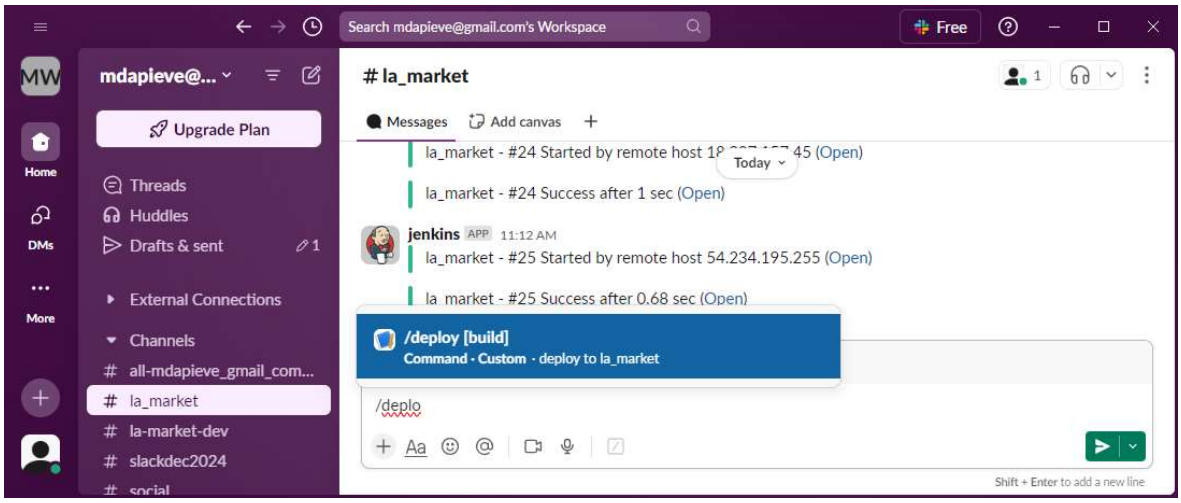


7. Then, go to the job and click on “Configure”, enable the “Trigger Build remotely” and specify the token name that must correspond to the one specified in the URL entered during the slack integration configuration above

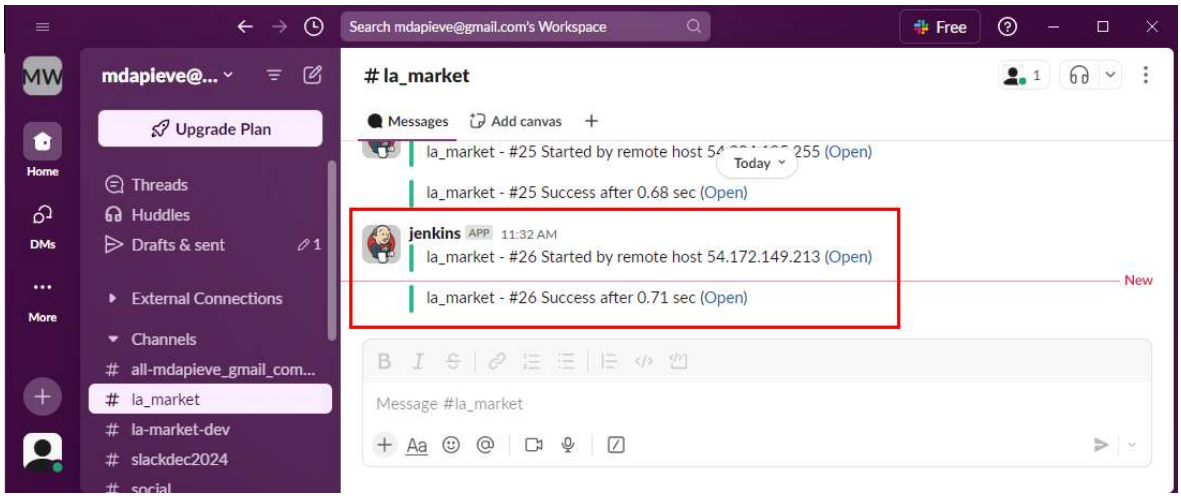


2.7.2.1.1 Testing

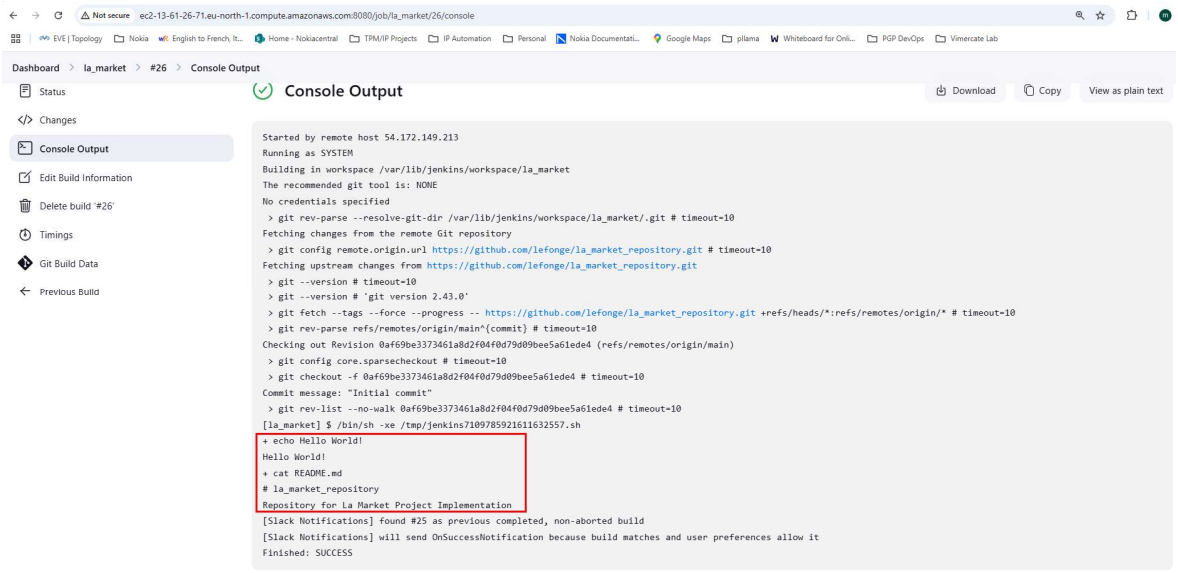
Now go to the Slack Channel and issue the"/deploy" command:



The job is executed successfully:

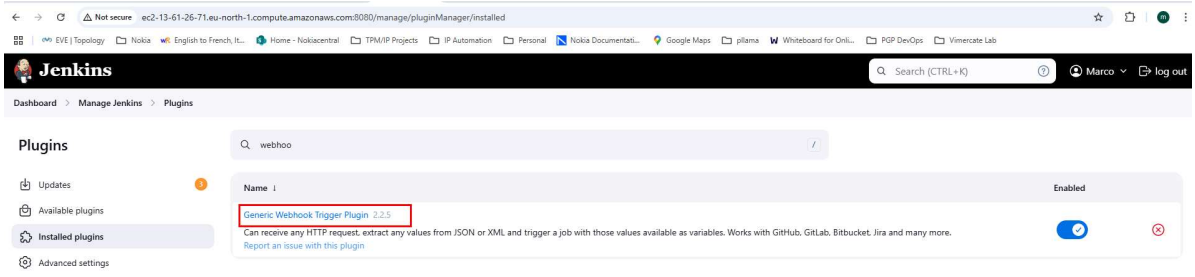


We can see the console output of the Jenkins Job's corresponding build triggered above where an echo command prints "Hello World" and a cat of the README.md file available on the provided github repository

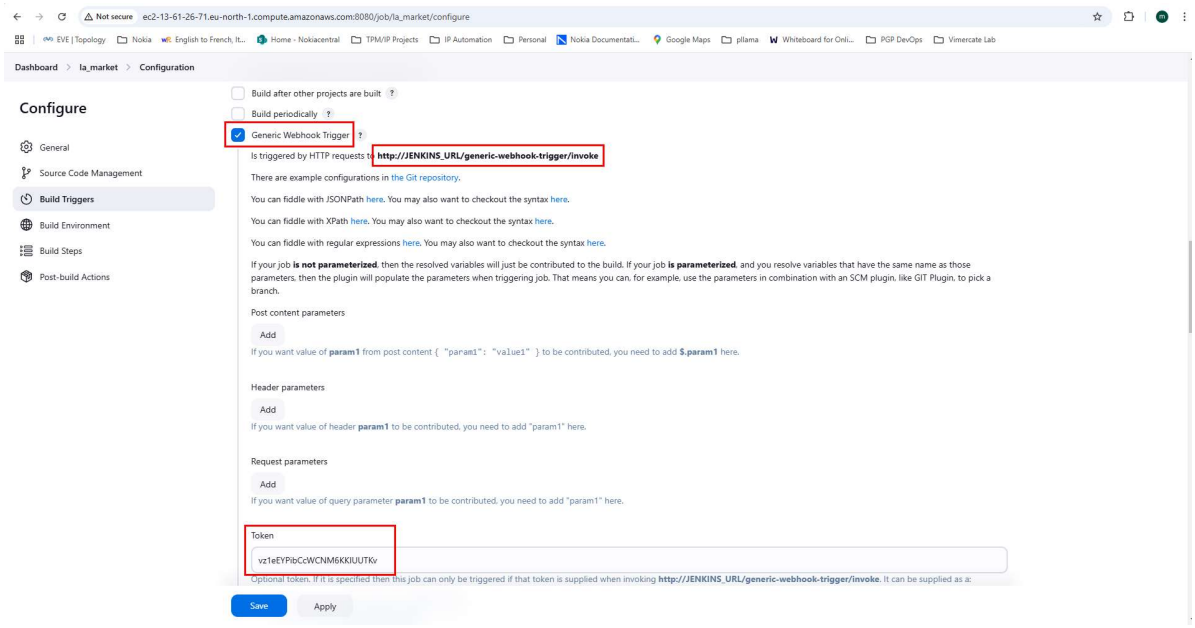


2.7.2.2 Legacy Integration configuration – POST Method

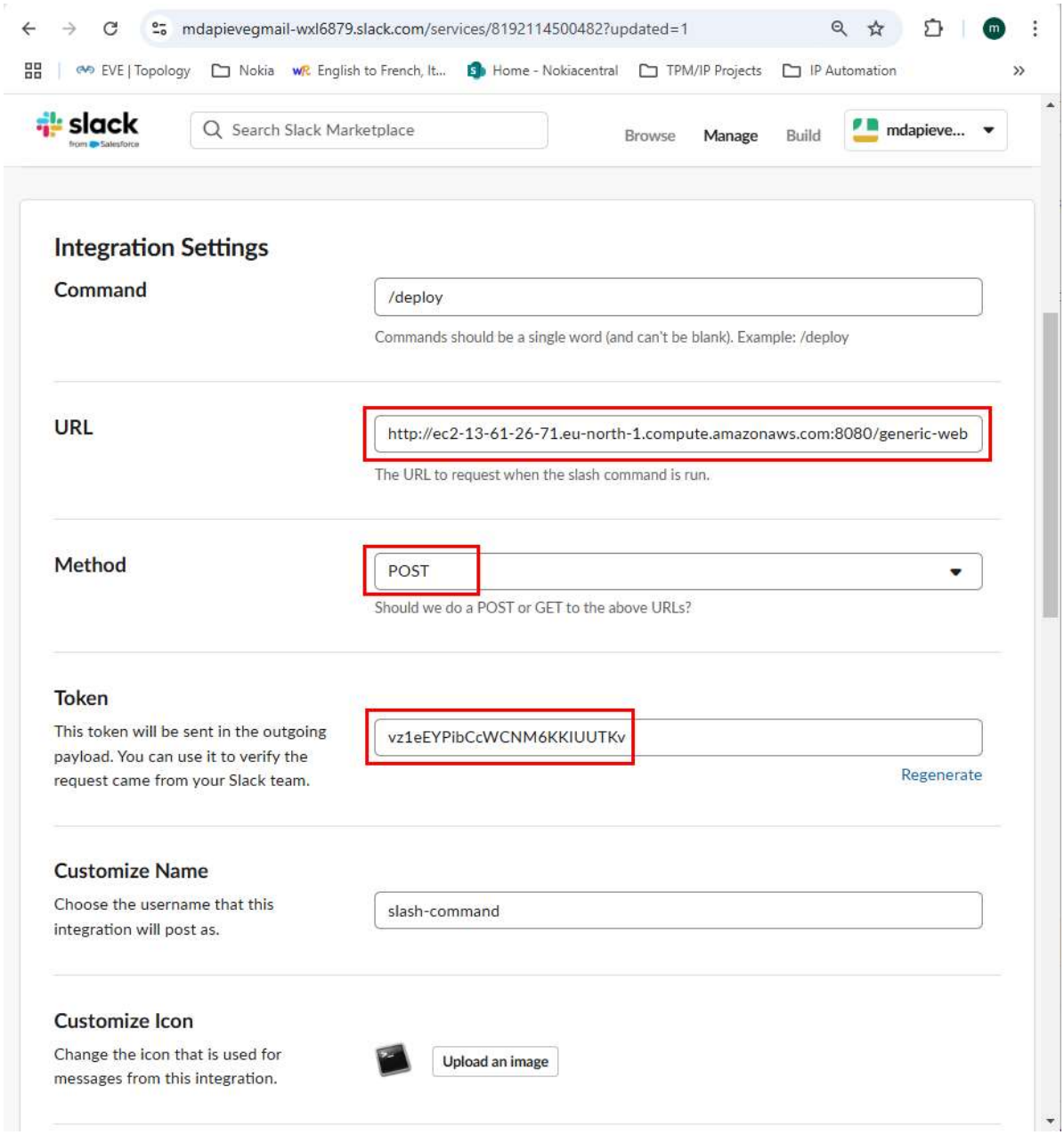
In this case, the following plugin needs to be installed



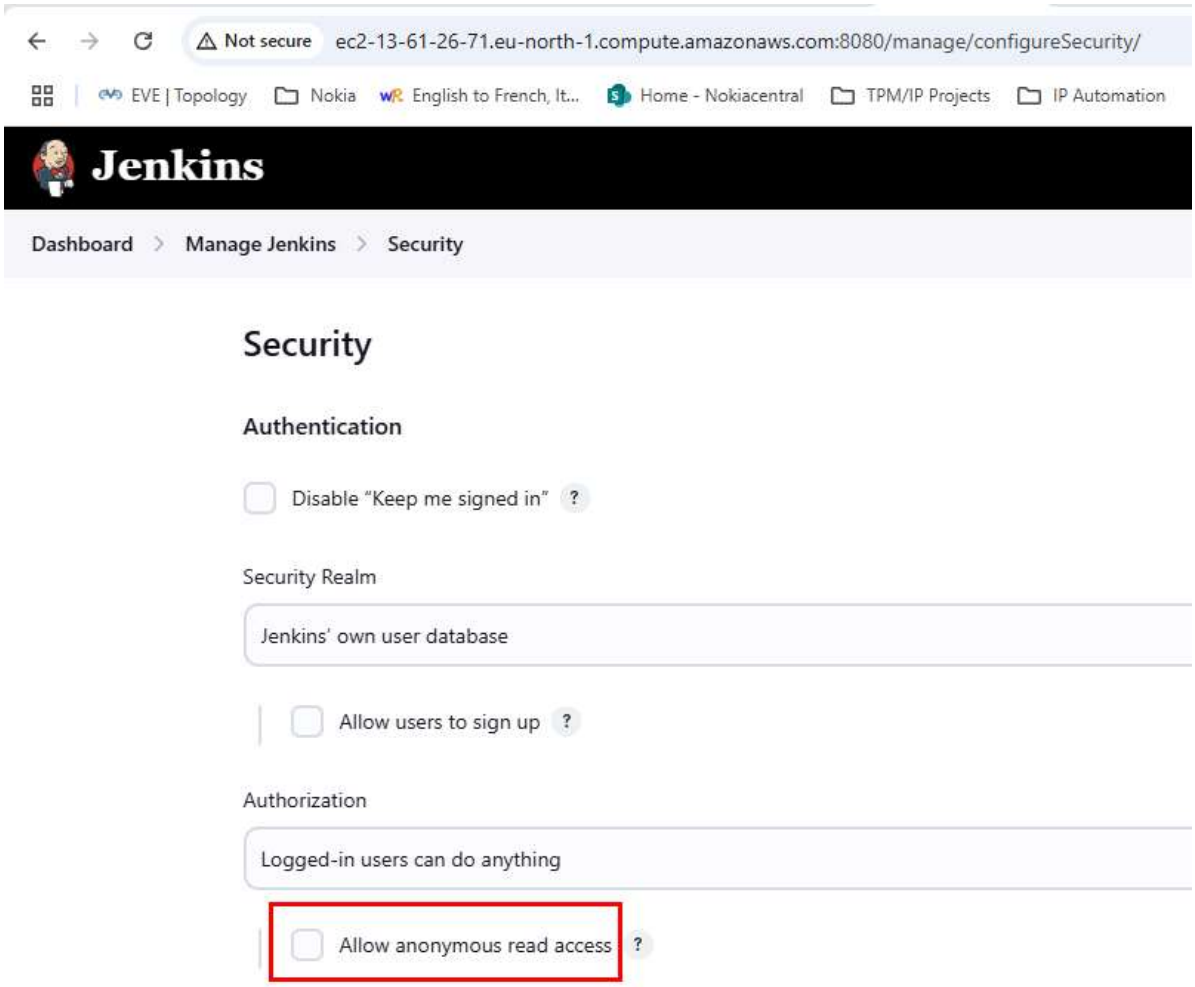
1. To make use of this plugin, go to the Jenkins Job and Build triggers. The Token added in the screenshot below is taken from the Slash Command integration Configuration from Slack. Check next step



2. The Webhook URL needs to be used in the Slash Command Integration window below

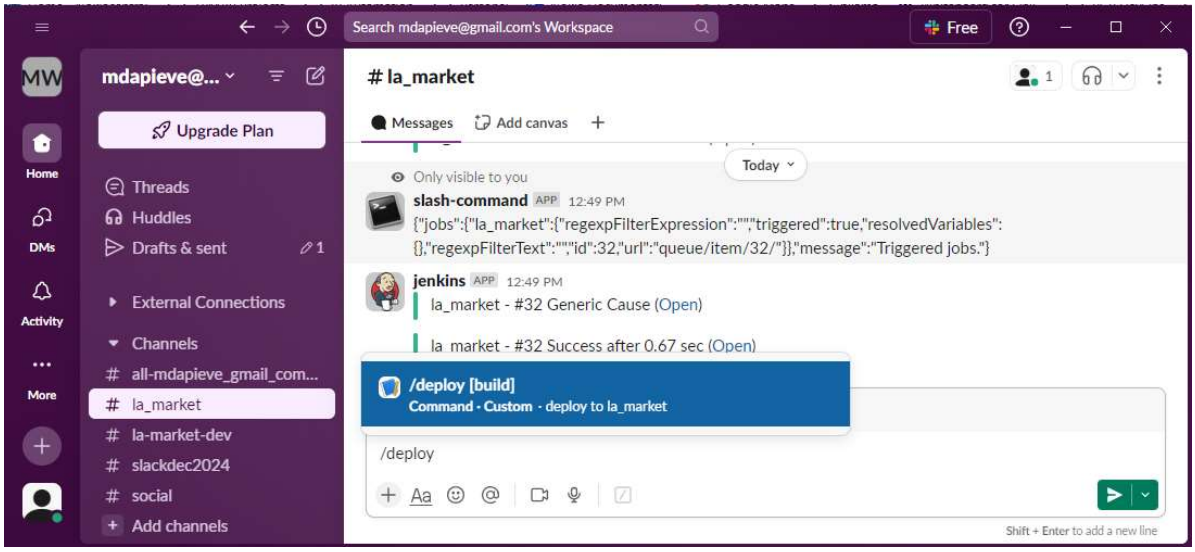


3. The following checkbox in the Jenkin's security settings can be disabled:

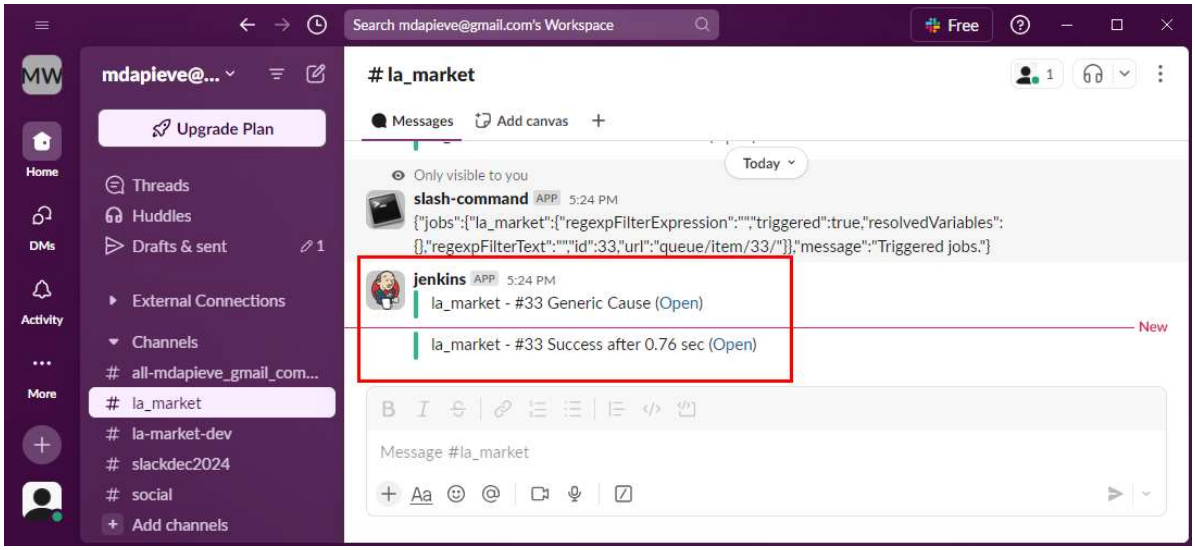


2.7.2.2.1 Testing

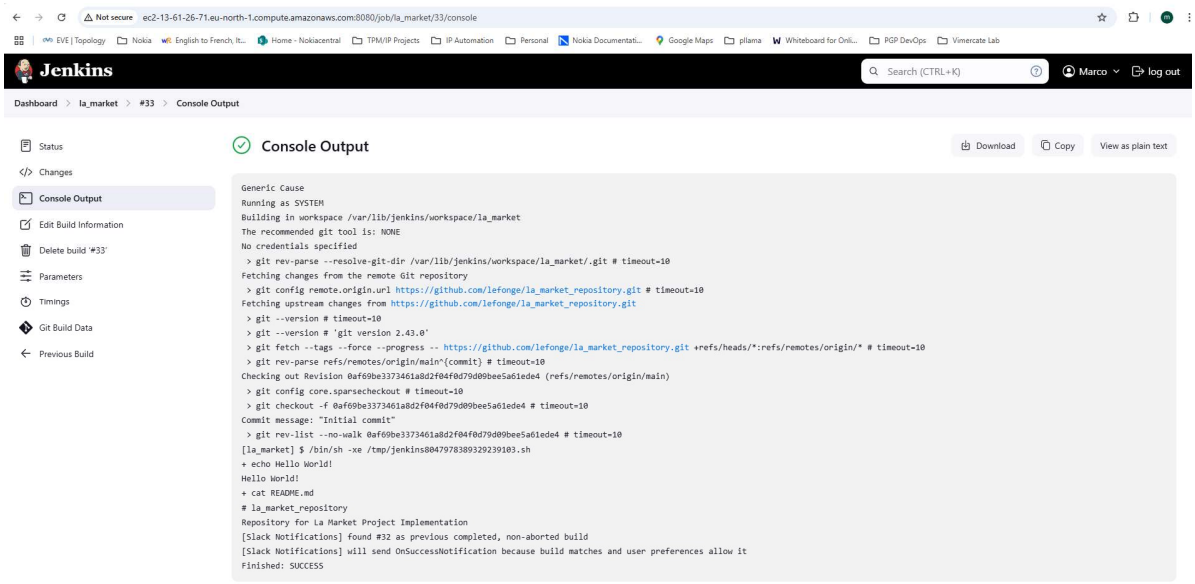
Go to slack and enter “/deploy” in the channel:



The Job is executed successfully and Slack is notified back:



The following is the successful job execution on Jenkins:



3 Conclusion

This procedure shows how to achieve the same objective by implementing different solutions. The first solution is the one that is recommended as it is in line with Jenkins policies of creating an app and provide a slash command within that app. This solution also showcases how Pipedream can be used in order to serve as a midpoint between Slack and Jenkins.

Regardless of the solution adopted, the integration of a slash command that trigger a job execution from Slack will enhance the deployment process and will enable the developers to trigger and monitor Jenkins builds from Slack.