

“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	3
2	Introduction	Error ! Bookmark not defined.
3	Procedure	3
3.1	Requirements.....	3
3.1.1	Platform Setup AWS EC2 instance	3
3.1.2	Java.....	5
3.1.3	Jenkins	6
3.1.4	Jenkins First Setup.....	8
3.2	Create a GitHub Repository	11
3.3	Slack.....	14
3.3.1	Create a Slack Channel	14
3.3.2	Integrate Jenkins into Slack.....	16
3.4	Jenkins Configuration for Notification Delivery to Slack.....	20
3.5	Create a Jenkins Job.....	26
3.6	Testing Slack Notification.....	33
3.7	Triggering builds directly from slack.....	37
3.7.1	Pipedream solution.....	37
3.7.2	Slash Command Integration Solution.....	52
4	Conclusion	64

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 screenshot shows the AWS Quick Start interface for selecting an Amazon Machine Image (AMI). The 'Quick Start' tab is selected. Under the 'Ubuntu' category, the 'Ubuntu Server 24.04 LTS (HVM), SSD Volume Type' AMI is highlighted. The details pane shows the AMI ID as ami-075449515af5df0d1, the architecture as 64-bit (x86), and the username as ubuntu. A green button labeled 'Verified provider' is visible.

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:

The screenshot shows the AWS EC2 Security Groups page. A success message at the top states: "Inbound security group rules successfully modified on security group (sg-05fd6df44996e7c93 | WebServerSG) > Details". The main view shows the security group sg-05fd6df44996e7c93 - WebServerSG with its details: Security group name (WebServerSG), Security group ID (sg-05fd6df44996e7c93), Description (Allows HTTP on port 8080), VPC ID (vpc-0a982a6709297d88a), Owner (605134439183), Inbound rules count (4 Permission entries), and Outbound rules count (1 Permission entry). The 'Inbound rules' tab is selected, showing four rules listed in a table. The last three rules, which define an incoming connection on port 8080, are highlighted with a red border.

Name	Security group rule ID	IP version	Type	Protocol
-	sgr-0c629af82d0a73115	IPv4	SSH	TCP
-	sgr-0db0b3229304ed540	IPv4	Custom TCP	TCP
-	sgr-0ba4ebf529e7eafbd	IPv4	All TCP	TCP
-	sgr-0ccfd2fe36faa1805	IPv4	HTTP	TCP

This is required when connecting via SSH from the local laptop or when Slack needs to invoke the Jenkins'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-keyring
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:~$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
> https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
> /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:

The screenshot shows the AWS Management Console interface for the EC2 service. On the left, there's a navigation sidebar with options like Dashboard, EC2 Global View, Events, Instances, Images, and Elastic Block Store. The main content area is titled 'Instances (1/1)' and shows a single instance named 'Ubuntu Server...'. Below the instance name, it displays the Instance ID (i-053dfbaabd3e861d7), Public IPv4 address (13.61.26.71), Private IPv4 address (172.31.42.71), and Public IPv4 DNS (ec2-13-61-26-71.eu-north-1.compute.amazonaws.com). A red box highlights the Public IPv4 DNS field.

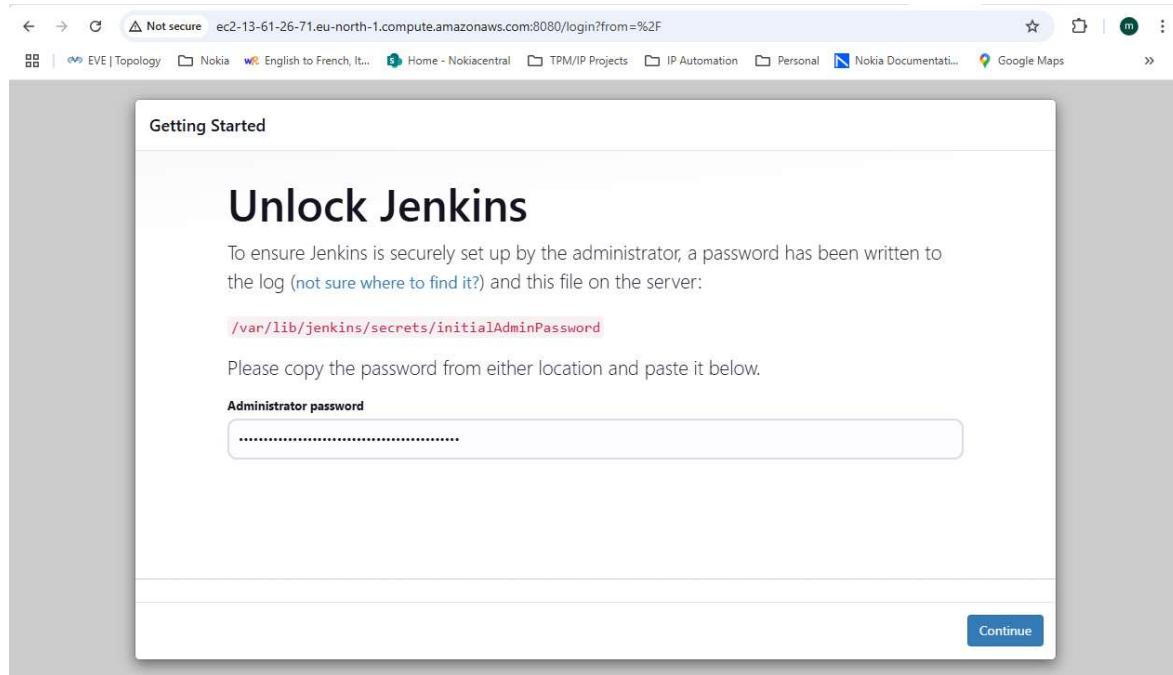
Open a web browser and enter the following url

<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

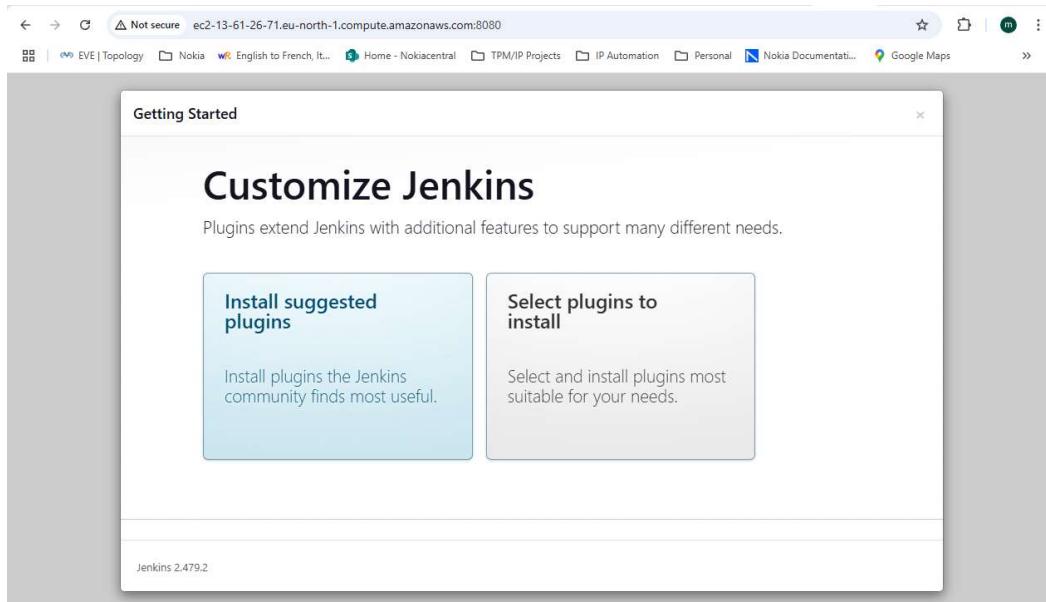
The URL above lands to the following page:



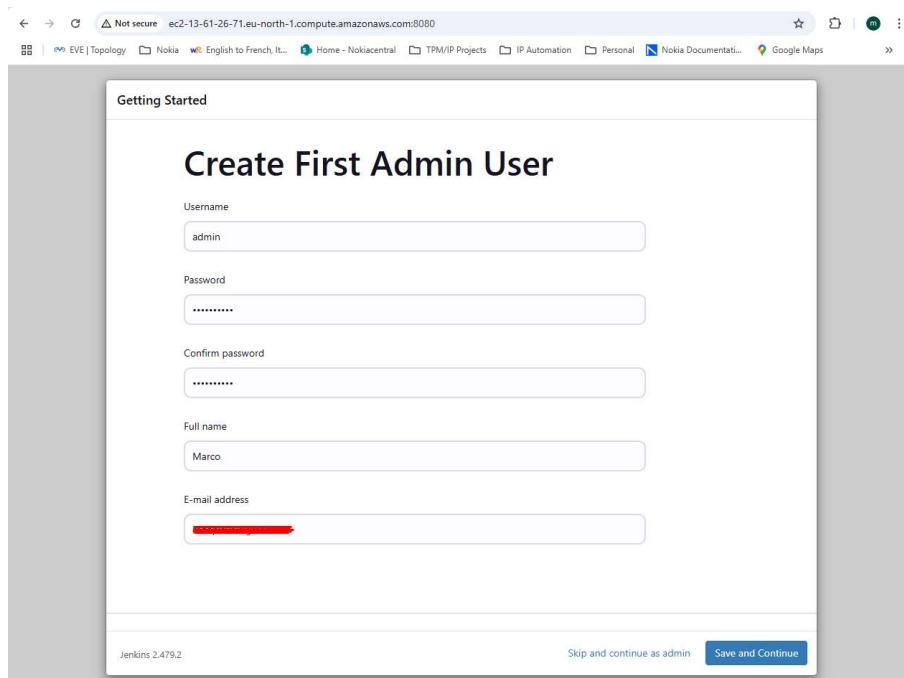
1. Please verify the password in the location specified in the page above and click on “Continue”:

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

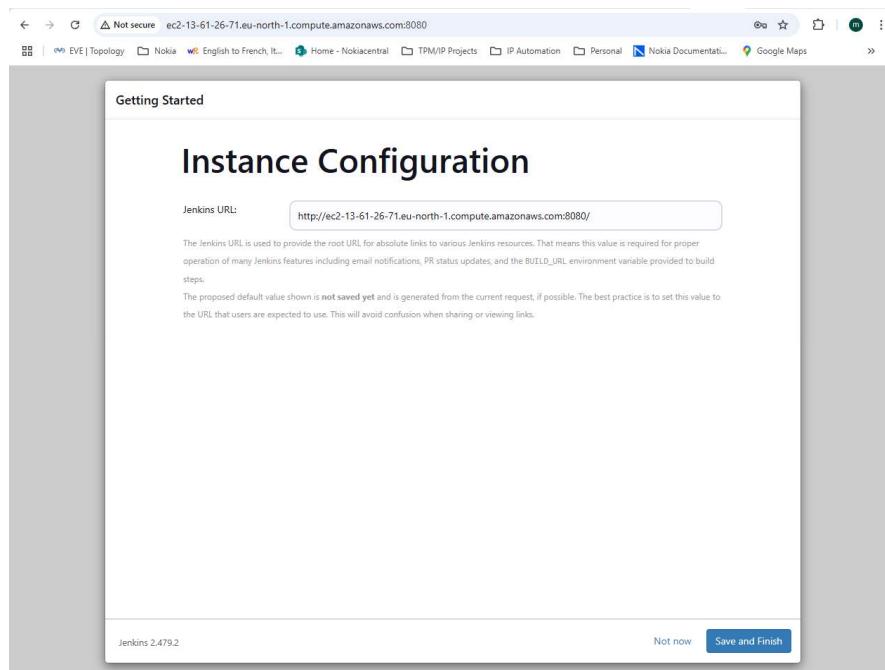
2. Proceed by installing the suggested plugins:



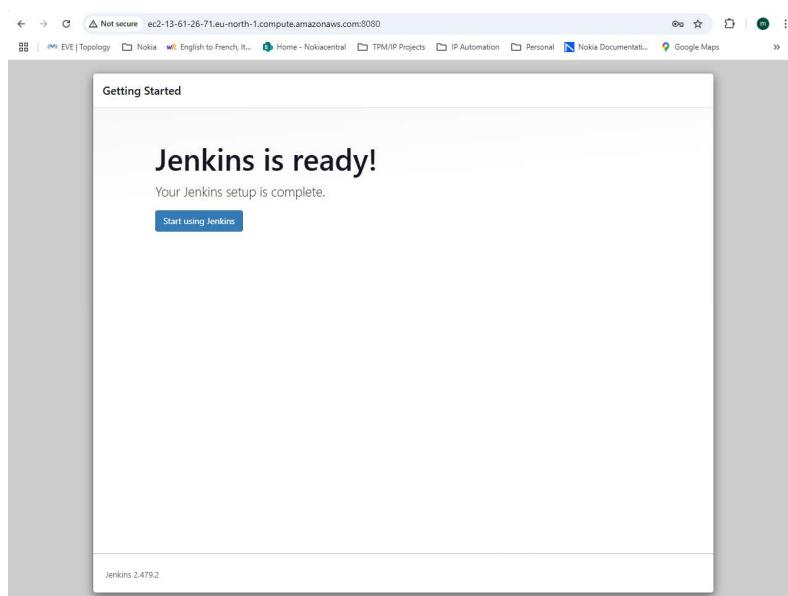
3. After the installation of the suggested plugins, create the first admin user:



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”:

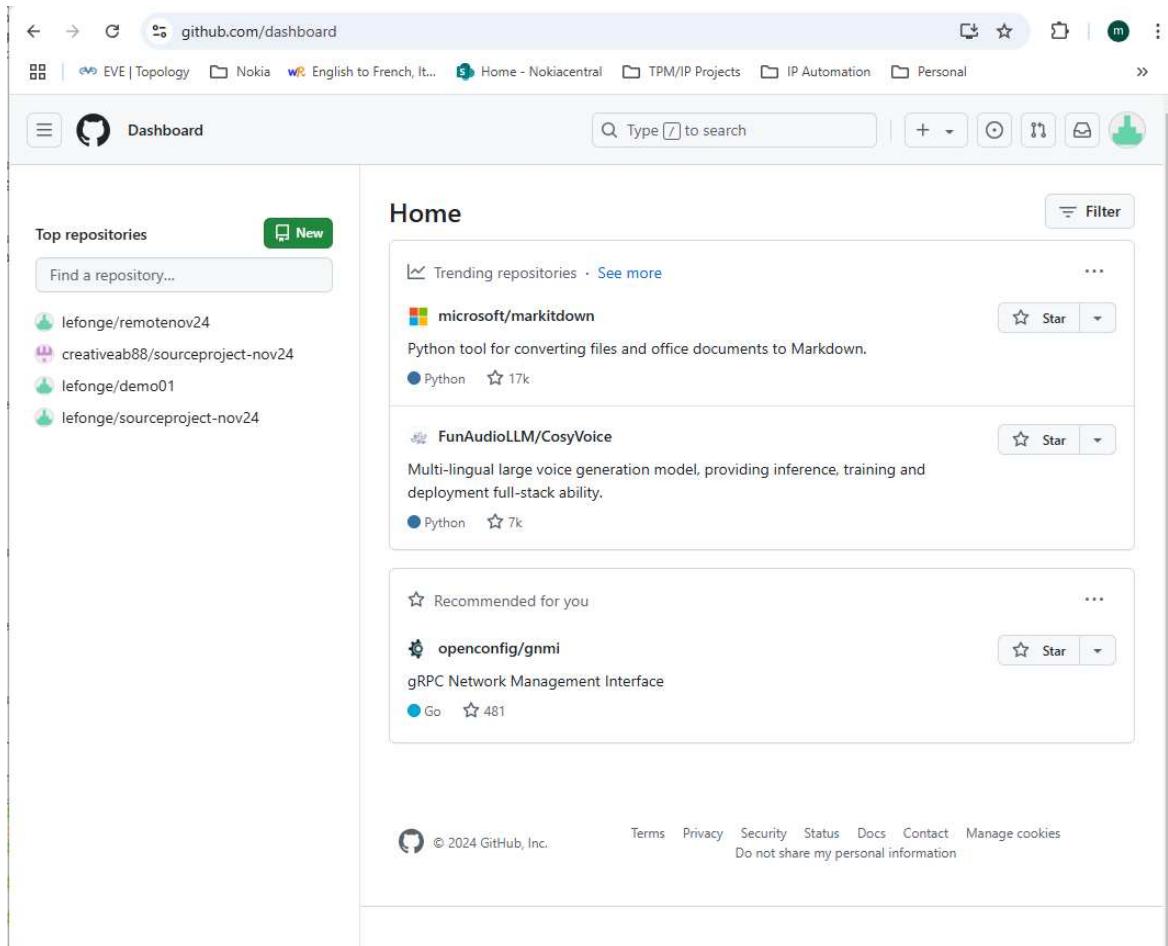


5. Finally, click on “Start using Jenkins”:



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:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner * lefonge / **Repository name *** la_market_repository
 la_market_repository is available.

Great repository names are short and memorable. Need inspiration? How about [jubilant-spork](#) ?

Description (optional)
 Repository for La Market Project implementation

Public
 Anyone on the internet can see this repository. You choose who can commit.
 Private
 You choose who can see and commit to this repository.

Initialize this repository with:
 Add a README file
 This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore
 .gitignore template: None

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license
 License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set **main** as the default branch. Change the default name in your [settings](#).

ⓘ You are creating a public repository in your personal account.

Create repository

- 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.

The screenshot shows a GitHub repository page for 'la_market_repository'. The repository is public and has one branch, 'main', which contains one commit from 'lefonge' labeled 'Initial commit' at 0af69be · now. There is also a file named 'README.md' with an 'Initial commit' at 'now'. The repository description is 'Repository for La Market Project Implementation'. On the right side, there are sections for 'About', 'Releases', and 'Packages'. The 'About' section includes a link to 'Create a new release'. The 'Packages' section indicates 'No packages published' and 'Publish your first package'. At the bottom, there is a footer with links to GitHub's Terms, Privacy, Security, Status, Docs, Contact, Manage cookies, and a 'Do not share my personal information' checkbox.

github.com/lefonge/la_market_repository

Code Issues Pull requests Actions Projects Wiki Security Insights Settings

la_market_repository Public

main 1 Branch 0 Tags

Go to file t + Code

lefonge Initial commit 0af69be · now 1 Commit

README.md Initial commit now

README

la_market_repository

Repository for La Market Project Implementation

About

Repository for La Market Project Implementation

Readme

Activity

0 stars

1 watching

0 forks

Releases

No releases published

Create a new release

Packages

No packages published

Publish your first package

© 2024 GitHub, Inc. Terms Privacy Security Status Docs Contact Manage cookies Do not share my personal information

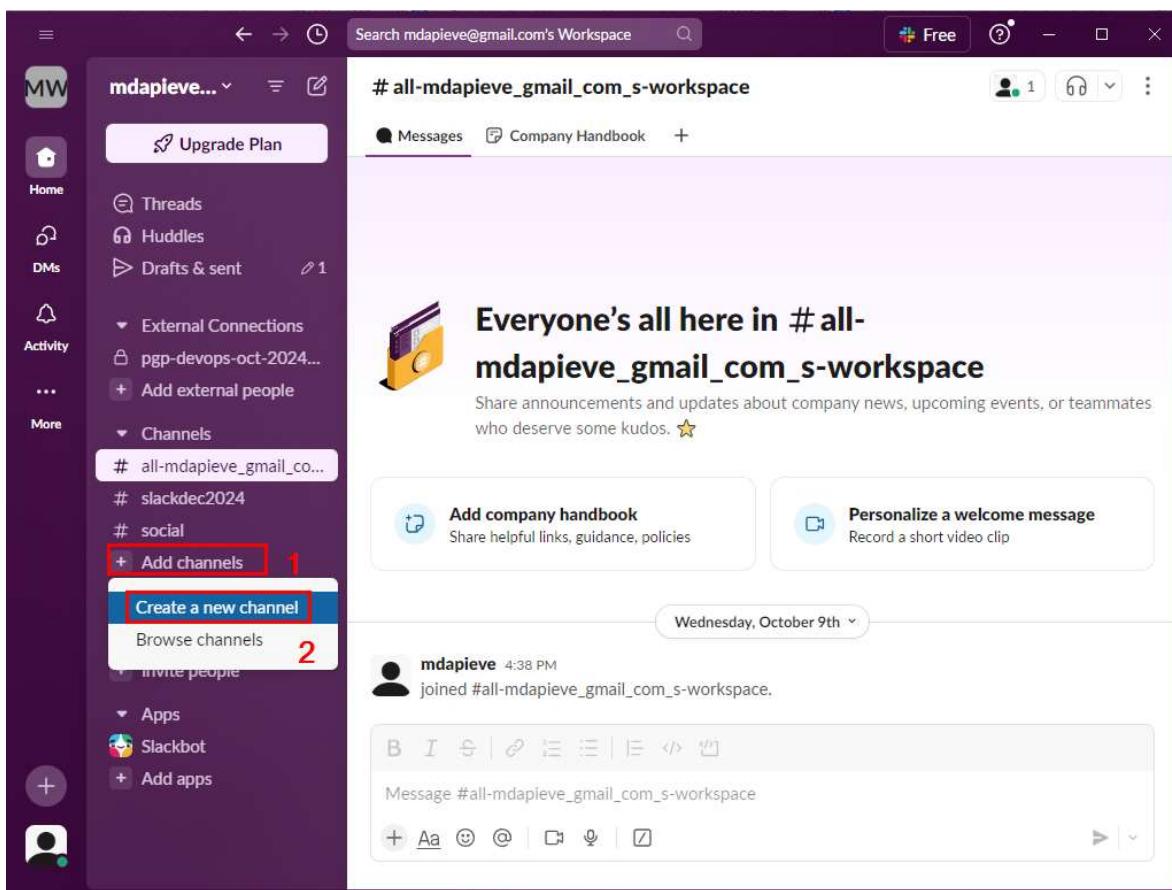
2.3 Slack

Please install Slack on your local machine

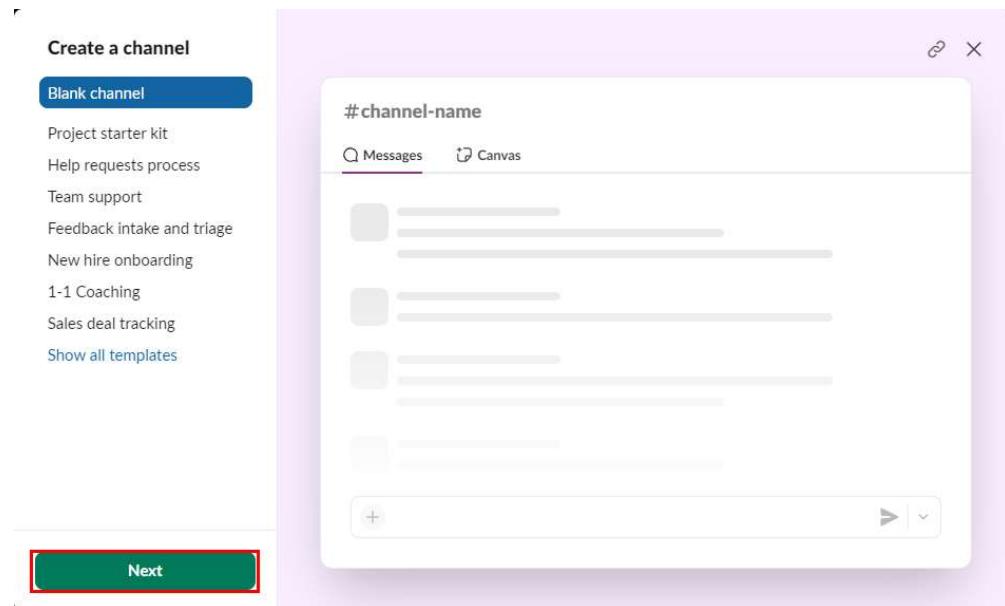
2.3.1 Create a Slack Channel

For this step, an account on SLACK is required.

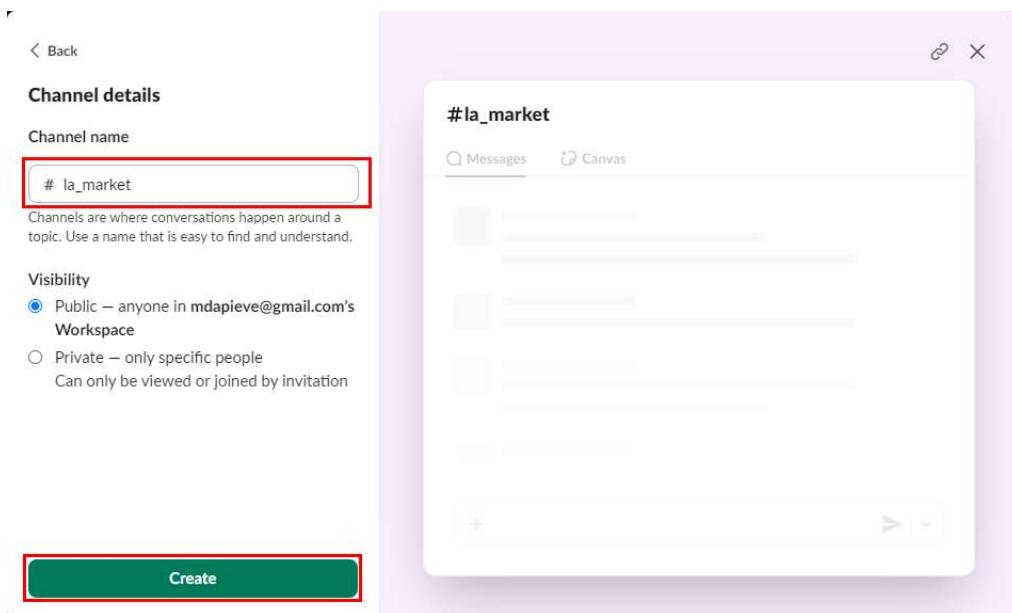
1. To create a new slack channel, click on add channel and create a new channel



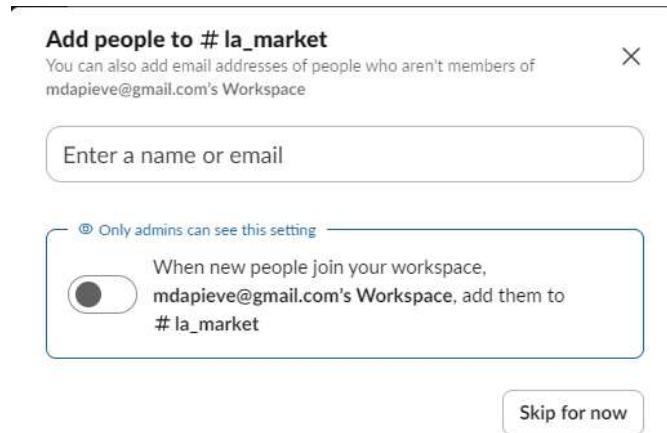
2. Click on Next:



3. Specify a name and then click create

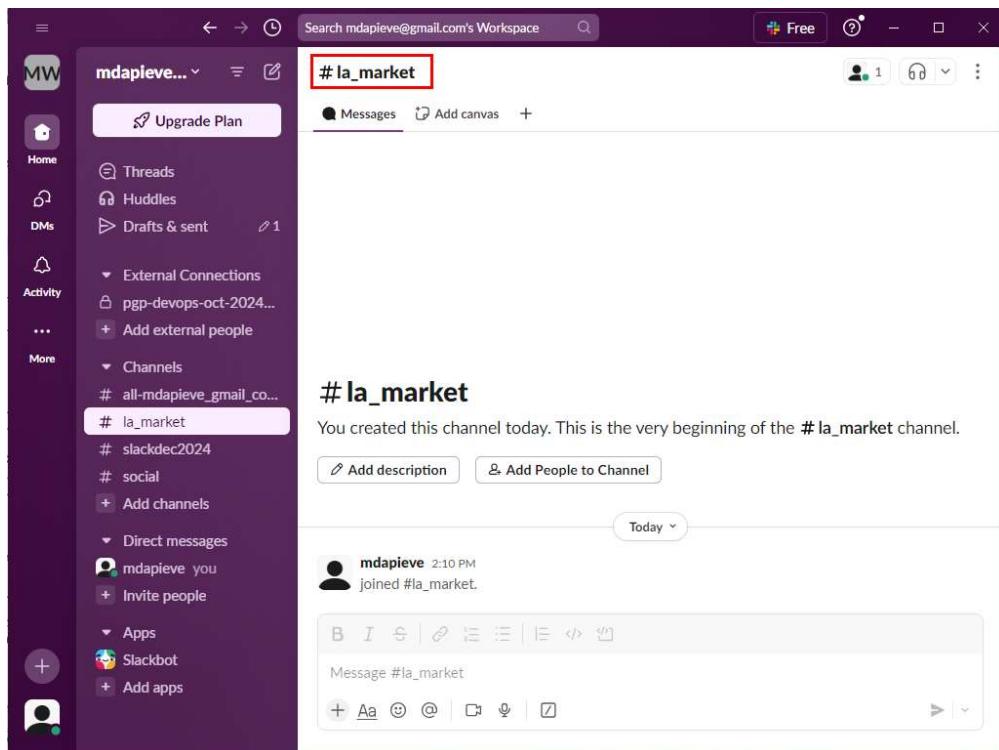


- For the scope of this project, no additional members will be added and this step can be skipped



2.3.2 Integrate Jenkins into Slack

- Once the channel is created, click on the channel name



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

la_market

☆ Notifications Off Huddle

[About](#) [Members 1](#) [Tabs](#) [Integrations](#) [Settings](#)

Supercharge your channel **PRO**

There's a few automations that will make your life easier. Set them up in a flash.

[See Upgrade Options](#)

Apps

Bring the tools you need into this channel to pull reports, start calls, file tickets and more.

[Add an App](#)



Send emails to this channel **PRO**

Get an email address that posts incoming emails in this channel.

[See Upgrade Options](#)

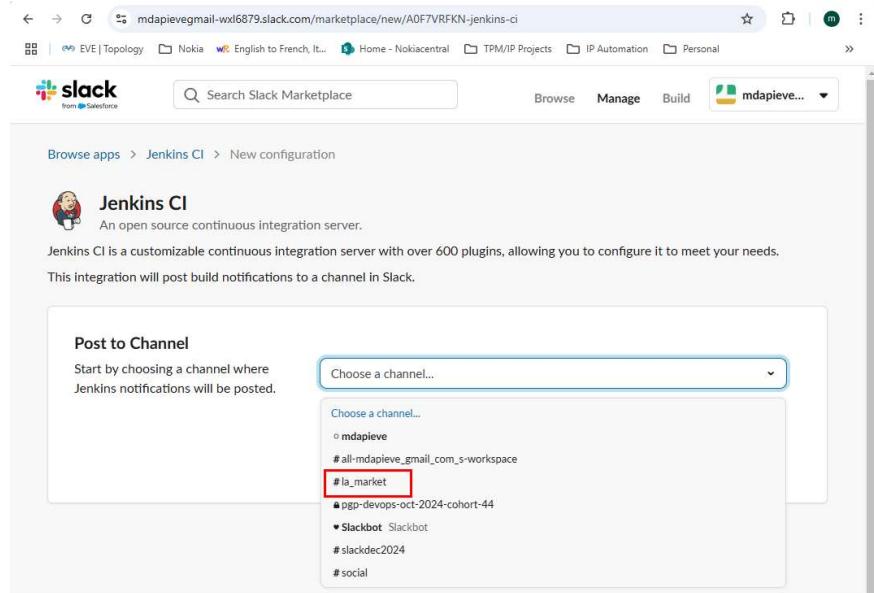
3. If not already installed, please search and install “Jenkins CI” in Slack and then click on it

The screenshot shows the Slack Marketplace interface. At the top, there are links to "Manage apps..." and "View Slack Marketplace". On the right, there is a close button (X) and an "esc" key indicator. The main title is "Add apps to la_market". Below it is a search bar with the placeholder "Search by name or category (e.g. productivity, sales)". A section titled "In your workspace (2)" lists two apps: "Slash Commands" and "Jenkins CI". The "Jenkins CI" card is highlighted with a red border. It features a cartoon character icon, the app name, and a brief description: "An open source continuous integration server." To the right of the card is a "View" button. Below this section, there is another heading "Add apps to your workspace" followed by three more app cards: "Google Drive", "Giphy", and "Google Calendar", each with an "Install" button.

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

The screenshot shows the Jenkins CI app details page in the Slack Marketplace. The URL in the browser is "mdapieve@gmail-wx16879.slack.com/marketplace/A0F7VRFKN-jenkins-ci". The page header includes the Slack logo and a search bar. The main content area shows the app's name "Jenkins CI" with a cartoon character icon. Below the name are tabs for "Description", "Permissions", and "Security & Compliance", with "Description" being the active tab. The description text reads: "Jenkins CI is a customizable continuous integration server with over 600 plugins, allowing you to configure it to meet your needs. This integration will post build notifications to a channel in Slack." A large green "Add to Slack" button is prominently displayed. At the bottom of the page, there are links for "Learn more & Support", "Privacy policy", "Terms", and "Categories" (with "Developer Tools" selected).

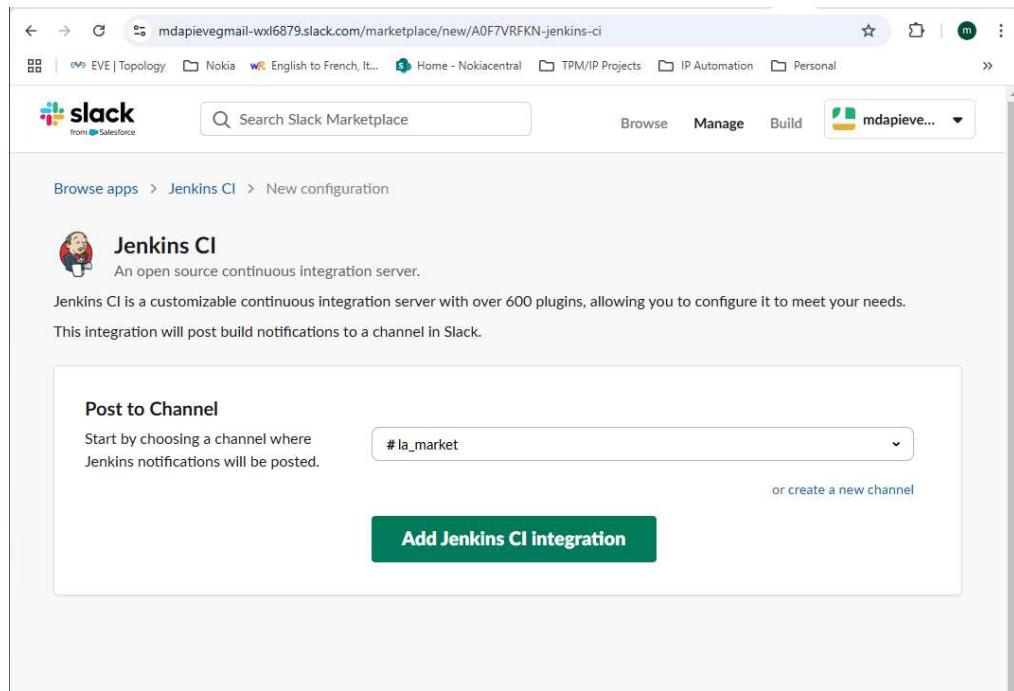
5. Choose the correct channel



The screenshot shows the Slack Marketplace interface for adding a Jenkins CI integration. In the center, there's a section titled "Post to Channel" with a dropdown menu labeled "Choose a channel...". Below the dropdown is a list of available channels:

- mdapieve
- # all-mdapieve_gmail_com_s-workspace
- # la_market** (This channel is highlighted with a red box.)
- ▲ pgp-devops-oct-2024-cohort-44
- ▼ Slackbot Slackbot
- # slackdec2024
- # social

6. Click on “Add Jenkins CI Integration”



The screenshot shows the continuation of the Jenkins CI integration setup in the Slack Marketplace. The "Post to Channel" dropdown now shows "# la_market" selected. At the bottom of the form, there is a prominent green button labeled "Add Jenkins CI integration".

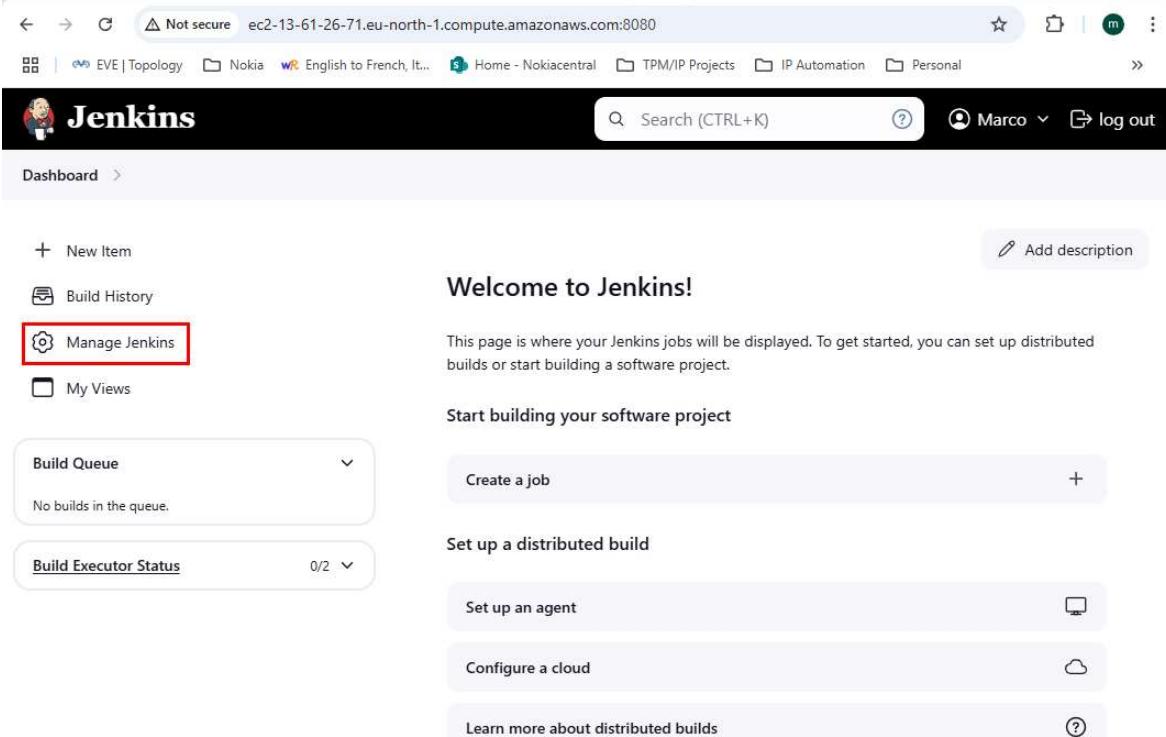
7. After clicking the button in the page above, a list of steps is presented to perform what is described in the next section. Step 3 of the list of steps includes some important information that needs to be noted somewhere to be used later on. Such information is:

- **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"



The screenshot shows the Jenkins homepage. At the top, there is a navigation bar with links like 'Not secure', 'ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080', and user profile icons. Below the navigation bar is the Jenkins logo and the title 'Welcome to Jenkins!'. On the left side, there is a sidebar with links: '+ New Item', 'Build History', 'Manage Jenkins' (which is highlighted with a red box), and 'My Views'. The main content area has a heading 'Start building your software project' and several buttons: 'Create a job' (with a '+' icon), 'Set up a distributed build', 'Set up an agent' (with a monitor icon), 'Configure a cloud' (with a cloud icon), and 'Learn more about distributed builds' (with a question mark icon). There are also sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '0/2').

2. Click on “Plugins”

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with options like 'New Item', 'Build History', 'Manage Jenkins' (which is selected and highlighted in purple), and 'My Views'. Below this are sections for 'Build Queue' (no builds in the queue) and 'Build Executor Status' (0/2). The main area is titled 'Manage Jenkins' and contains several configuration sections: 'System Configuration' (with 'System' and 'Plugins' items), 'Tools' (with 'Nodes' and 'Tools' items), and 'Appearance'. The 'Plugins' section is specifically highlighted with a red box. It describes the function of adding, removing, disabling, or enabling plugins to extend Jenkins' functionality.

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

The screenshot shows the Jenkins Plugin Manager. On the left, there are tabs for 'Updates', 'Available plugins' (which is selected and highlighted in red), 'Installed plugins' (with a count of 1), and 'Advanced settings'. The main area has a search bar with 'slack' typed in (highlighted with a red box) and a blue 'Install' button. A detailed view of the 'Slack Notification' plugin is shown: it's version 751.v2e44153c0fe1, released 4 days ago, and has 3 reviews. The description states it integrates Jenkins with Slack to allow publishing build statuses, messages, and files to Slack channels. Another plugin, 'Global Slack Notifier', is also listed below it.

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

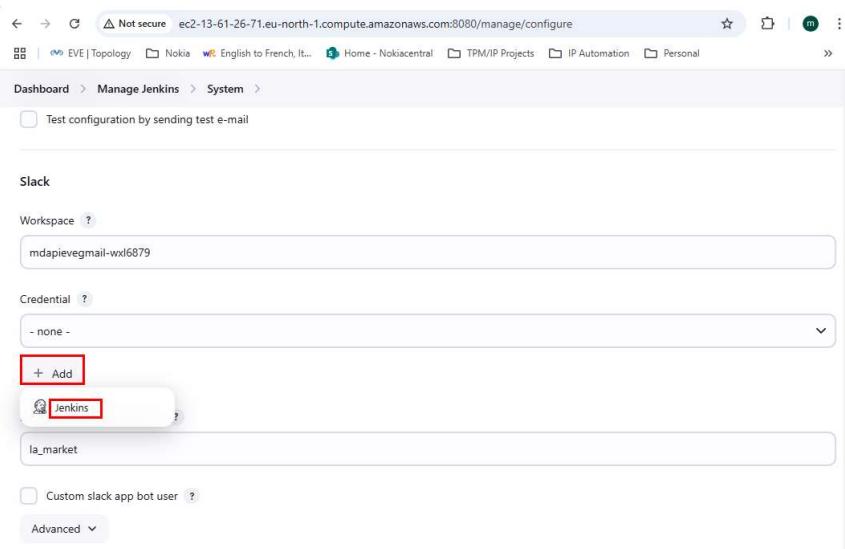
4. Go to “Manage Jenkins” again and click on “System”

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with options like 'New Item', 'Build History', 'Manage Jenkins' (which is selected and highlighted in grey), 'My Views', 'Build Queue' (with a note 'No builds in the queue.'), and 'Build Executor Status' (with a note '0/2'). The main content area is titled 'Manage Jenkins' and contains a 'System Configuration' section. This section includes a 'System' card (with a gear icon) which is highlighted with a red box, labeled 'Configure global settings and paths.', and a 'Tools' card (with a wrench icon) labeled 'Configure tools, their locations and automatic installers.' Below these are 'Plugins' and 'Nodes' cards. A search bar at the top right says 'Search settings'.

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)

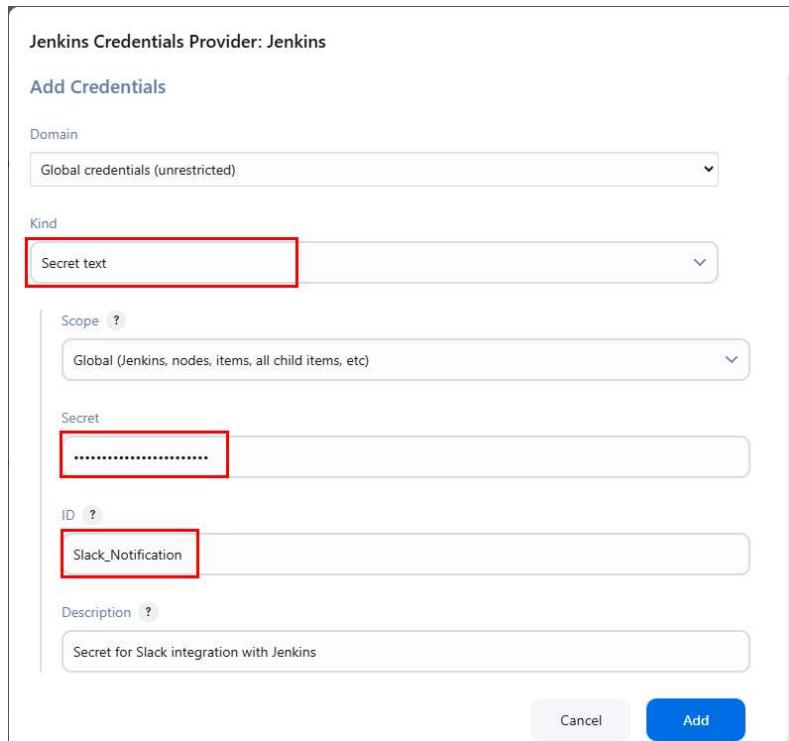
The screenshot shows the Jenkins System configuration page under the 'Manage Jenkins' section. It has a 'Test configuration by sending test e-mail' checkbox. The 'Slack' section is expanded, showing a 'Workspace' field containing 'mdapieve@gmail-wx16879' (which is highlighted with a red box). Below it is a 'Credential' dropdown set to '- none -' and an 'Add' button. The 'Default channel / member id' field contains 'la_market' (also highlighted with a red box). At the bottom, there's a 'Custom slack app bot user' checkbox and an 'Advanced' dropdown.

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



The screenshot shows the Jenkins System configuration page. In the 'Slack' section, there is a 'Workspace' field containing 'mdapieve@gmail-wx16879'. Below it, a 'Credential' dropdown is set to '- none -'. A red box highlights the '+ Add' button. To its right, a dropdown menu is open, showing 'Jenkins' with a red box around it. Other options in the menu include 'Custom slack app bot user' and 'Advanced'. At the bottom of the workspace input field, the text 'la_market' is visible.

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).



The dialog box is titled 'Jenkins Credentials Provider: Jenkins' and contains the following fields:

- Add Credentials**
- Domain**: Global credentials (unrestricted)
- Kind**: Secret text (highlighted with a red box)
- Scope**: Global (Jenkins, nodes, items, all child items, etc)
- Secret**: A field containing a redacted secret value (highlighted with a red box)
- ID**: Slack_Notification (highlighted with a red box)
- Description**: Secret for Slack integration with Jenkins

At the bottom right are 'Cancel' and 'Add' buttons, with 'Add' being highlighted in blue.

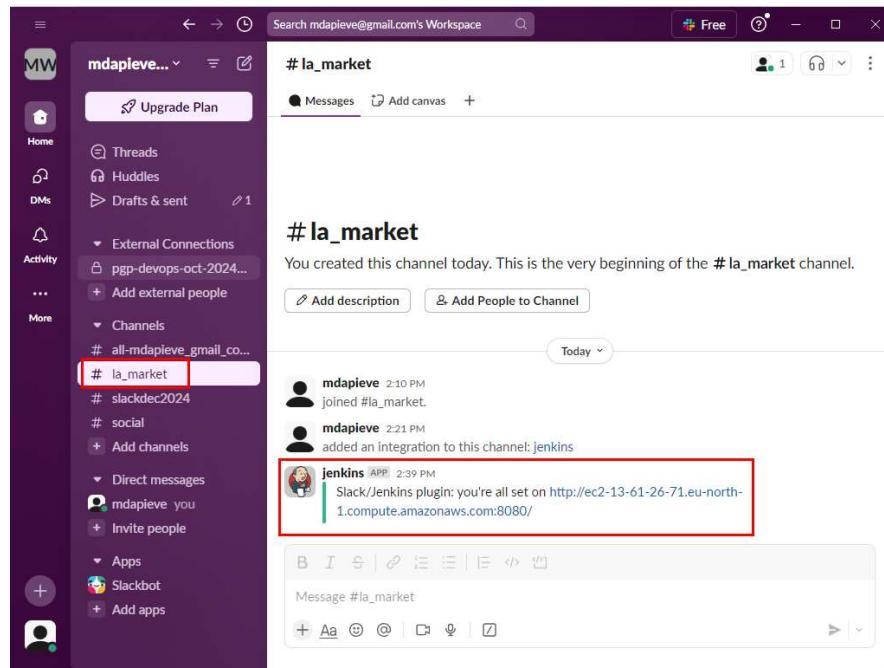
8. On the underlying page, it is now possible to specify the Credential

The screenshot shows the Jenkins System configuration page under the Manage Jenkins section. In the Slack section, the 'Workspace' field contains 'mdapievegmai-wxl6879'. The 'Credential' dropdown menu is open, showing two options: '- none -' and 'Secret for Slack integration with Jenkins', with the second option highlighted by a red box. The 'Default channel / member id' field contains 'la_market'.

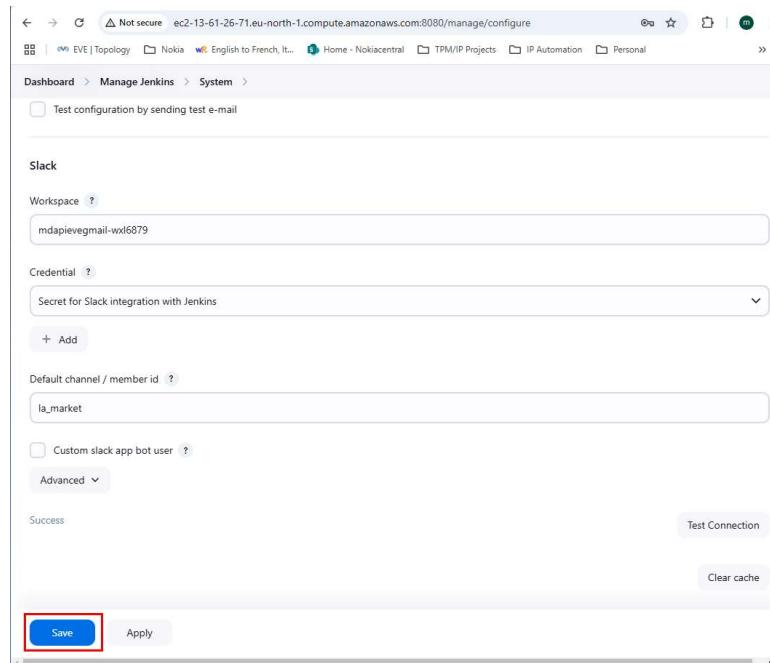
9. It is possible to verify the successful integration by click on “Test Connection” and check if the “Success” message appears:

The screenshot shows the Jenkins System configuration page under the Manage Jenkins section. The 'Success' message is displayed in a red box at the bottom left. To its right, the number '2' is enclosed in a red box. At the bottom right, the 'Test Connection' button is enclosed in a red box, with the number '1' to its left.

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



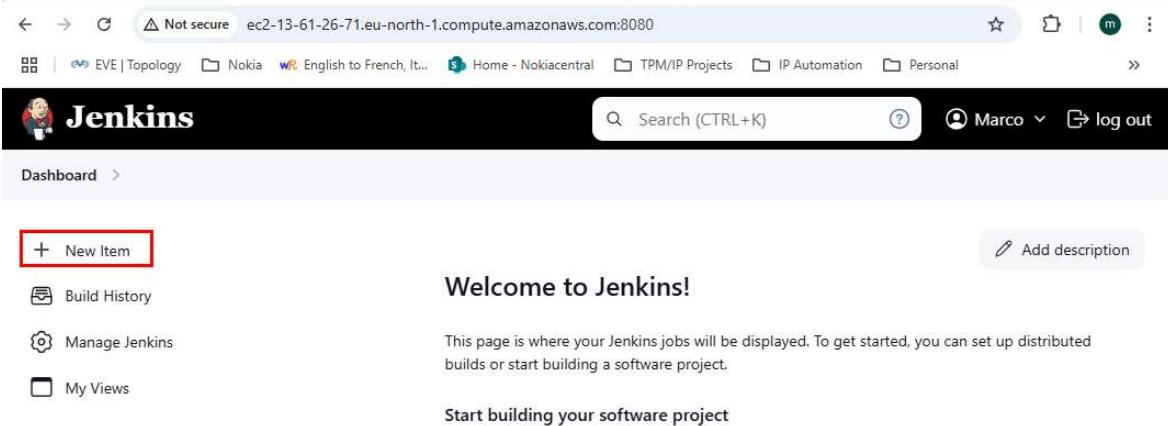
11. It is now possible to save the page on Jenkins



2.5 Create a Jenkins Job

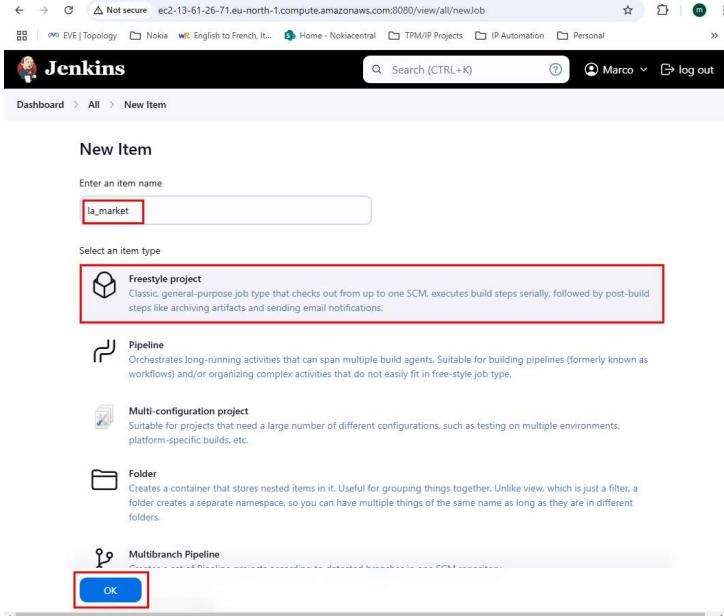
In order to receive notification on the “la_market” slack channel, a Jenkins job is created and the the github “la_market_repository” is references as SCM.

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



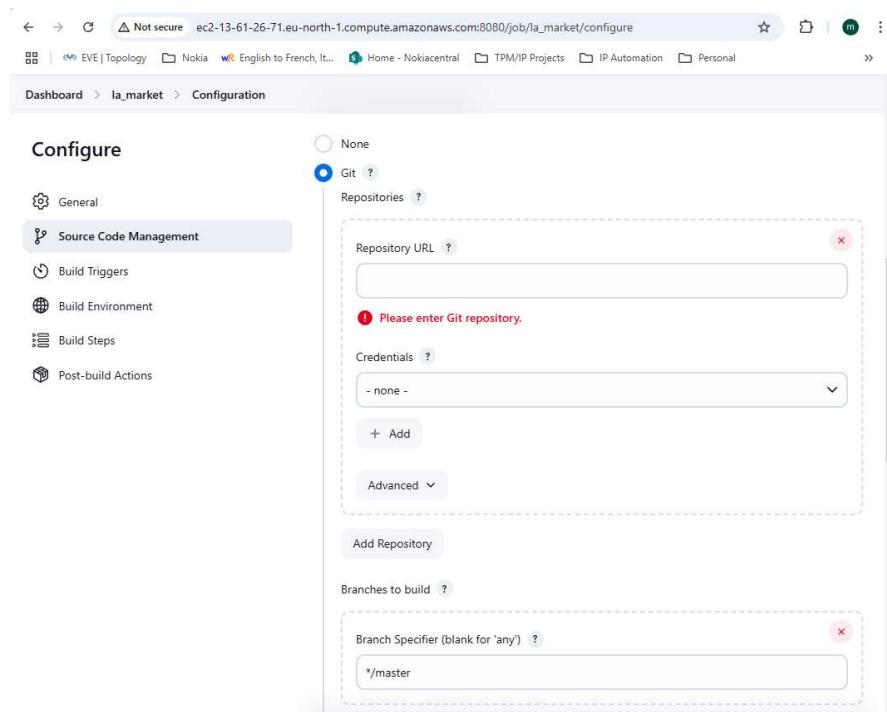
The screenshot shows the Jenkins dashboard at <http://ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080>. The top navigation bar includes links for EVE | Topology, Nokia, English to French, It..., Home - Nokiacentral, TPM/IP Projects, IP Automation, Personal, and a user profile for Marco. The main content area features a "Welcome to Jenkins!" message and a "Start building your software project" button. On the left, there are links for Build History, Manage Jenkins, and My Views. A red box highlights the "+ New Item" button in the top-left corner of the dashboard area.

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

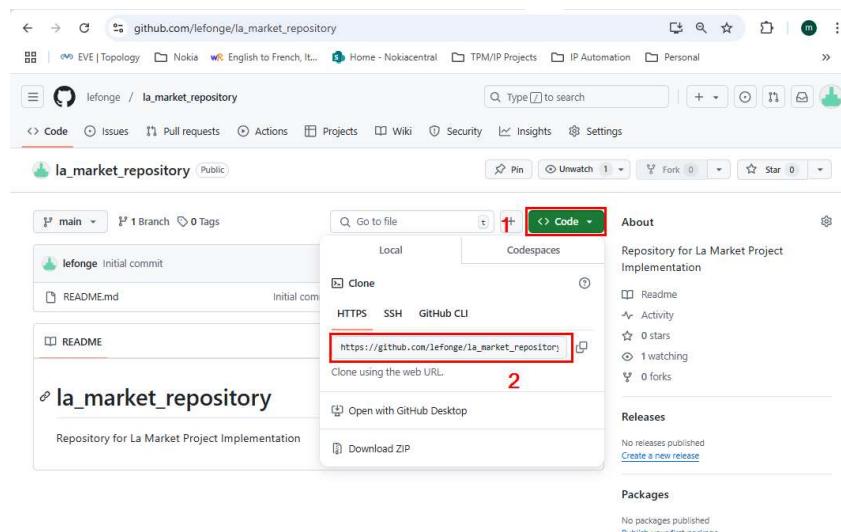


The screenshot shows the "New Item" creation dialog. At the top, it says "Enter an item name" with "la_market" typed in. Below that, it says "Select an item type" with "Freestyle project" selected. A red box highlights the "Freestyle project" option. Other options shown are Pipeline, Multi-configuration project, Folder, and Multibranch Pipeline. At the bottom, a red box highlights the "OK" button.

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.

The screenshot shows the Jenkins configuration page for the 'la_market' job. The 'Source Code Management' section is selected. It is configured to use a Git repository. The 'Repository URL' is set to `https://github.com/lefonge/la_market_repository.git`. The 'Branch Specifier' is set to `*/main`.

Configure

Source Code Management (selected)

Repository URL: `https://github.com/lefonge/la_market_repository.git`

Branch Specifier: `*/main`

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

The screenshot shows the Jenkins configuration interface for a job named 'la_market'. The left sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment (which is selected and highlighted in grey), Build Steps (selected and highlighted in blue), and Post-build Actions. The main content area is titled 'Build Environment' and contains several checkboxes for build-related options. Below this is the 'Build Steps' section, which includes a 'Add build step ^' button and a dropdown menu titled 'Filter'. The dropdown menu lists several build step options: 'Execute Windows batch command', 'Execute shell' (which is highlighted with a red box), 'Invoke Ant', 'Invoke Gradle script', 'Invoke top-level Maven targets', 'Run with timeout', and 'Set build status to "pending" on GitHub commit'.

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

The screenshot shows a web-based build configuration interface. At the top, the URL is `ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080/job/la_market/configure`. Below the URL, the page title is "Configure". The left sidebar lists several sections: General, Source Code Management, Build Triggers, Build Environment, **Build Steps**, and Post-build Actions. The "Build Steps" section is currently selected. On the right, under the heading "Build Steps", there is a single step titled "Execute shell". The "Command" field contains the following two lines of code, which are highlighted with a red box:

```
echo "Hello World!"  
cat README.md
```

Below the command field is an "Advanced" dropdown menu. At the bottom of the screen are two buttons: "Save" and "Apply".

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

The screenshot shows the Jenkins configuration interface for a job named 'la_market'. The 'Build Steps' section is currently selected. A dropdown menu titled 'Execute shell' is open, listing various post-build actions. The 'Slack Notifications' option is highlighted with a red box.

- Aggregate downstream test results
- Archive the artifacts
- Build other projects
- Publish JUnit test result report
- Record fingerprints of files to track usage
- Git Publisher
- E-mail Notification
- Editable Email Notification
- Set GitHub commit status (universal)
- Set build status on GitHub commit [deprecated]
- Slack Notifications** (highlighted with a red box)
- Delete workspace when build is done

At the bottom of the dropdown, there is a link 'Add post-build action ^'.

Buttons at the bottom of the configuration screen include 'Save' and 'Apply'.

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

The screenshot shows the Jenkins configuration interface for a job named "la_market". The left sidebar lists various configuration sections: General, Source Code Management, Build Triggers, Build Environment, Build Steps, and Post-build Actions. The "Post-build Actions" section is currently selected and highlighted with a blue background. On the right, under the heading "Post-build Actions", there is a "Slack Notifications" section. This section contains eight checkboxes, all of which are checked (indicated by a blue square with a white checkmark). The checkboxes are: Notify Build Start, Notify Success, Notify Aborted, Notify Not Built, Notify Unstable, Notify Regression, Notify Every Failure, and Notify Back To Normal. A red rectangular box highlights this entire list of checkboxes. Below this section is an "Advanced" dropdown menu. At the bottom of the page, there are two buttons: a large blue "Save" button with a red border, and a smaller "Apply" button.

Configure

Post-build Actions

General

Source Code Management

Build Triggers

Build Environment

Build Steps

Post-build Actions

Slack Notifications

- Notify Build Start
- Notify Success
- Notify Aborted
- Notify Not Built
- Notify Unstable
- Notify Regression
- Notify Every Failure
- Notify Back To Normal

Advanced ▾

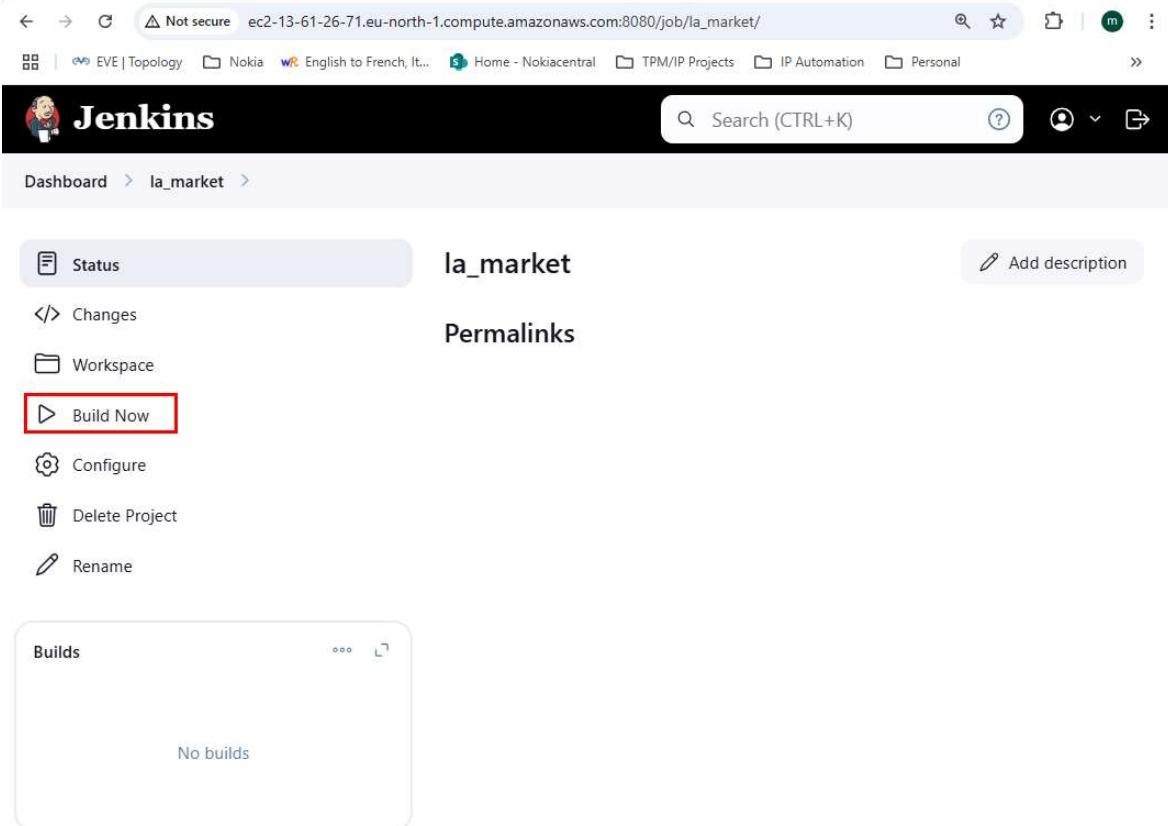
Add post-build action ▾

Save

Apply

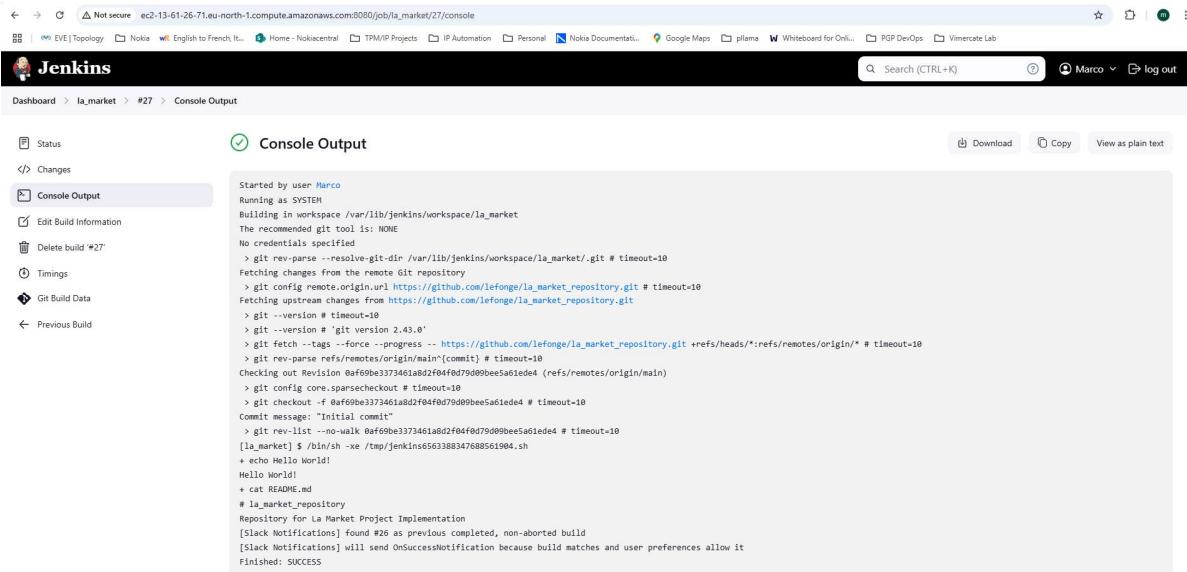
2.6 Testing Slack Notification

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



The screenshot shows the Jenkins interface for the 'la_market' job. The top navigation bar includes links for EVE | Topology, Nokia, English to French, It..., Home - Nokiacentral, TPM/IP Projects, IP Automation, Personal, and a search bar. Below the header, the job name 'la_market' is displayed. On the left, a sidebar lists options: Status (highlighted), Changes, Workspace, Build Now (highlighted with a red box), Configure, Delete Project, and Rename. A 'Permalinks' section is also present. The main content area is titled 'Builds' and displays the message 'No builds'.

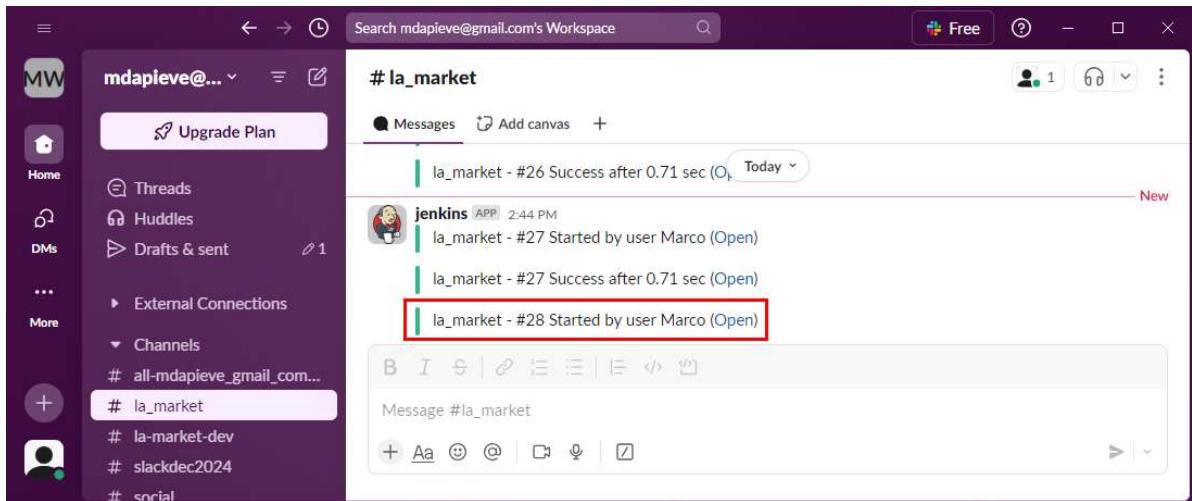
This results in a successful build and the Console output can be checked for analysis



The screenshot shows the Jenkins interface with the 'Console Output' tab selected. The build log displays a successful execution of a Jenkinsfile, including git fetch, checkout, and a 'Hello World!' print statement.

```
Started by user Marco
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/la_market
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/la_market/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lefonge/la_market_repository.git # timeout=10
Fetching upstream changes from https://github.com/lefonge/la_market_repository.git
> git -version # timeout=10
> git -version # git version 2.43.0'
> git fetch --tags --force --progress - <https://github.com/lefonge/la_market_repository.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0af69bdc3373461a8d2f0a0f079d0999ee5a1de4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0af69bdc3373461a8d2f0a0f079d0999ee5a1de4 # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 0af69bdc3373461a8d2f0a0f079d0999ee5a1de4 # timeout=10
[la_market]$ /bin/sh -xe /tmp/jenkins6563388347688561984.sh
+ echo Hello World!
Hello World!
+ cat README.md
# la_market_repository
Repository for La Market Project Implementation
[Slack Notifications] found #26 as previous completed, non-aborted build
[Slack Notifications] will send OnSuccessNotification because build matches and user preferences allow it
Finished: SUCCESS
```

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):

The screenshot shows the Jenkins configuration interface for a job named 'la_market'. The 'Build Steps' section is selected. Inside, there is a step titled 'Execute shell' containing the command 'echo "Hello World!'. A red box highlights this command. Below it, the command 'cat README.md' is visible. At the bottom right of the 'Execute shell' box, there is a button labeled 'Advanced'.

Now the build is failing

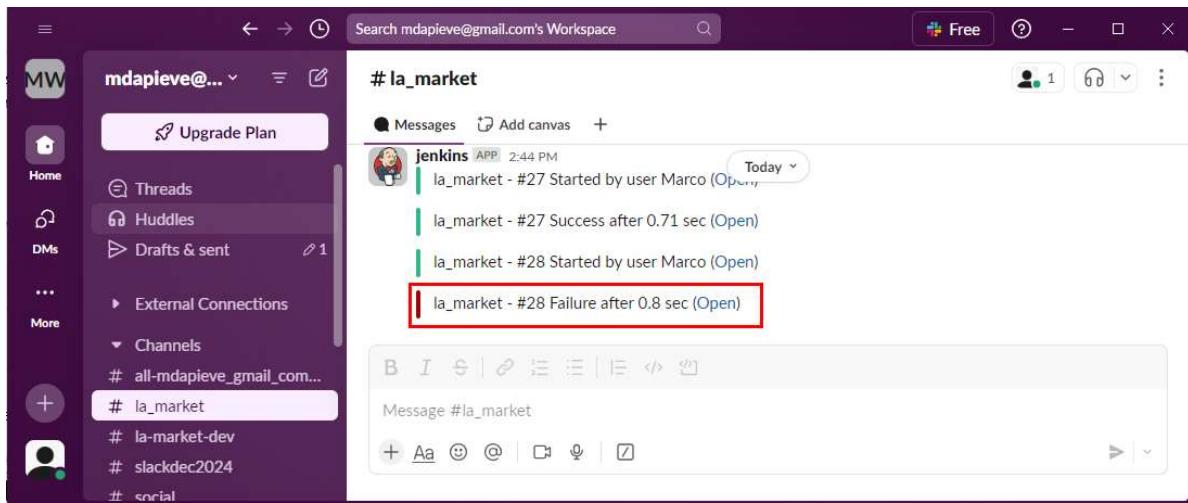
The screenshot shows the Jenkins console output for build #28. The 'Console Output' tab is selected. The output shows the build process, including cloning the repository and executing the 'Execute shell' step. The error message 'Syntax error: Unterminated quoted string' is visible, indicating the failure of the build step.

```

Started by user Marco
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/la_market
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/la_market/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lefonge/la_market_repository.git # timeout=10
Fetching upstream changes from https://github.com/lefonge/la_market_repository.git
> git --version # timeout=10
> git fetch --tags --force --progress -- https://github.com/lefonge/la_market_repository.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0af0be3373461a1a0d2f0404bd79d09bee5a61ede4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0af0be3373461a1a0d2f0404bd79d09bee5a61ede4 # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 0af0be3373461a1a0d2f0404bd79d09bee5a61ede4 # timeout=10
[la_market] $ /bin/sh -xe /tmp/jenkins3275796752054747801.sh
/tmp/jenkins3275796752054747801.sh: 3: Syntax error: Unterminated quoted string
Build step 'Execute shell' marked build as failure
[Slack Notifications] found #27 as previous completed, non-aborted build
[Slack Notifications] will send OnEveryFailureNotification because build matches and user preferences allow it
Finished: FAILURE

```

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:

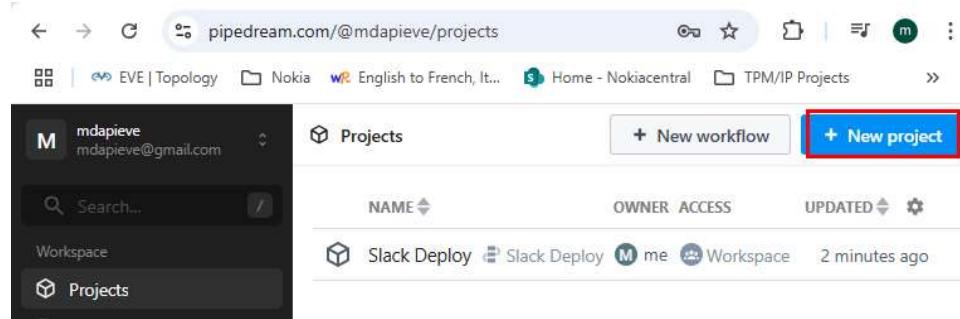
- One makes use of Pipedream and a Slack App that is created for this specific use case (recommended as the second method is a legacy integration method)
- The other makes use of the Slash Command App integration available in Slack

2.7.1 Pipedream solution

2.7.1.1 Pipedream workflow creation

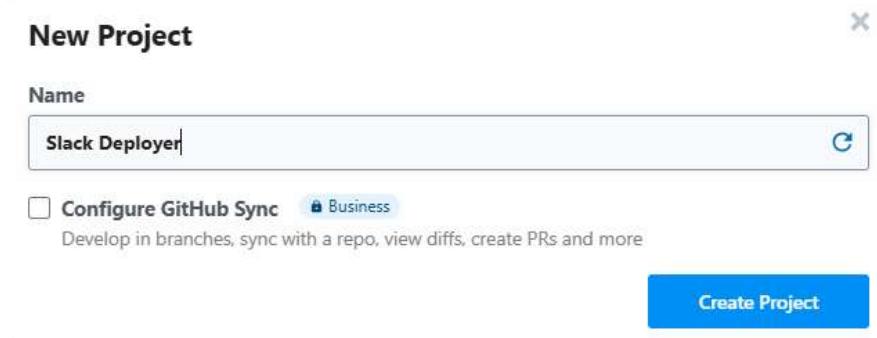
Before configuring Slack to be able to send a slash command to trigger a build on Jenkins, it is necessary to create a workflow on pipedream.com. Please create an account.

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:



The screenshot shows the Pipedream web interface. At the top, there's a navigation bar with links like 'EVE | Topology', 'Nokia', 'English to French, It...', 'Home - Nokiacentral', and 'TPM/IP Projects'. Below the navigation is a sidebar with 'Workspace' and 'Projects' tabs, where 'Projects' is currently selected. The main area is titled 'Projects' and shows a table with one row: 'Slack Deploy' by 'Slack Deploy' (owner), 'me' (access), 'Workspace' (team), and '2 minutes ago' (updated). To the right of the table are sorting and filtering options ('NAME', 'OWNER', 'UPDATED'). At the top right of the main area, there are buttons for '+ New workflow' and '+ New project'. A red box highlights the '+ New project' button.

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



The screenshot shows the 'New Project' dialog. It has a title 'New Project' and a close button 'X'. Below the title is a 'Name' input field containing 'Slack Deployer'. Underneath the input field is a checkbox labeled 'Configure GitHub Sync' with a 'Business' tag next to it. A note below the checkbox says 'Develop in branches, sync with a repo, view diffs, create PRs and more'. At the bottom right of the dialog is a large blue 'Create Project' button.

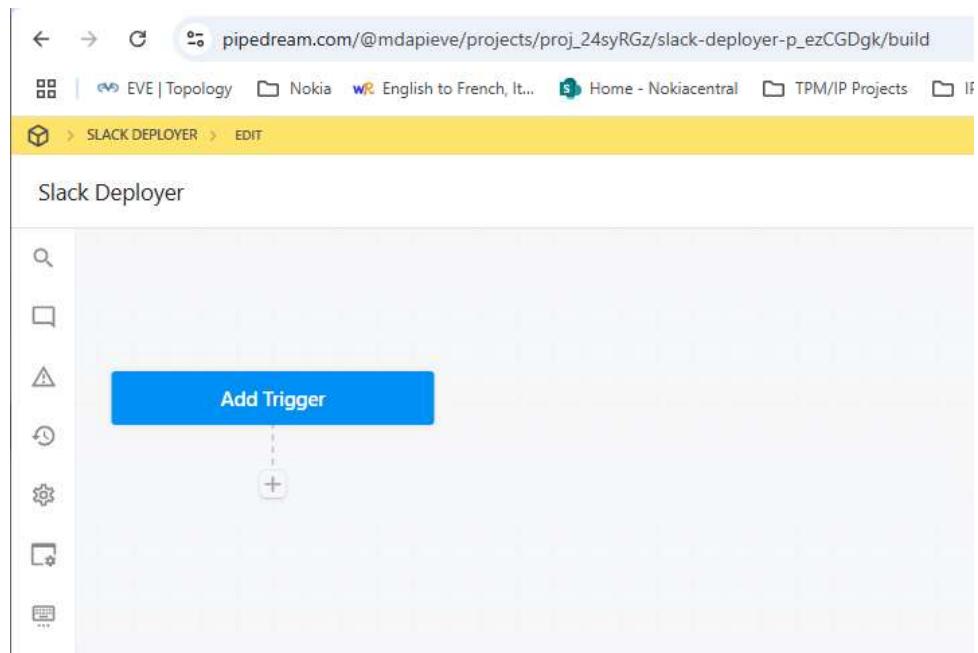
3. Then click on New and Workflow

The screenshot shows the Pipedream platform interface. On the left, there's a sidebar with various options like Workspace, Projects, Sources, Accounts, Data Stores, Event History, Settings, and Resources. The 'Projects' option is currently selected. In the main content area, there's a header 'Slack Deployer'. Below it, there's a '+ New' dropdown menu with two options: 'Folder' and 'Workflow'. The 'Workflow' option is highlighted with a red box. A message below the dropdown says 'This project is empty. Use the dropdown above to create folders, workflows, etc.'.

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

The screenshot shows the 'Create new workflow' dialog. At the top, it says 'Create new workflow'. Below that is a 'Workflow Name' field containing 'Slack Deployer'. Under 'Execution Controls', there are sections for 'Timeout' (set to 30 seconds) and 'Memory' (set to 256 MB). A note says '1 credit per workflow segment'. Below these are several configuration options with checkboxes: 'Send error notifications' (checked), 'Automatically retry on errors' (unchecked), 'Disable data retention' (unchecked), 'Limit concurrency' (unchecked), 'Limit execution rate' (unchecked), 'Eliminate cold starts' (unchecked), and 'Run in VPC' (unchecked). At the bottom right is a large blue 'Create Workflow' button.

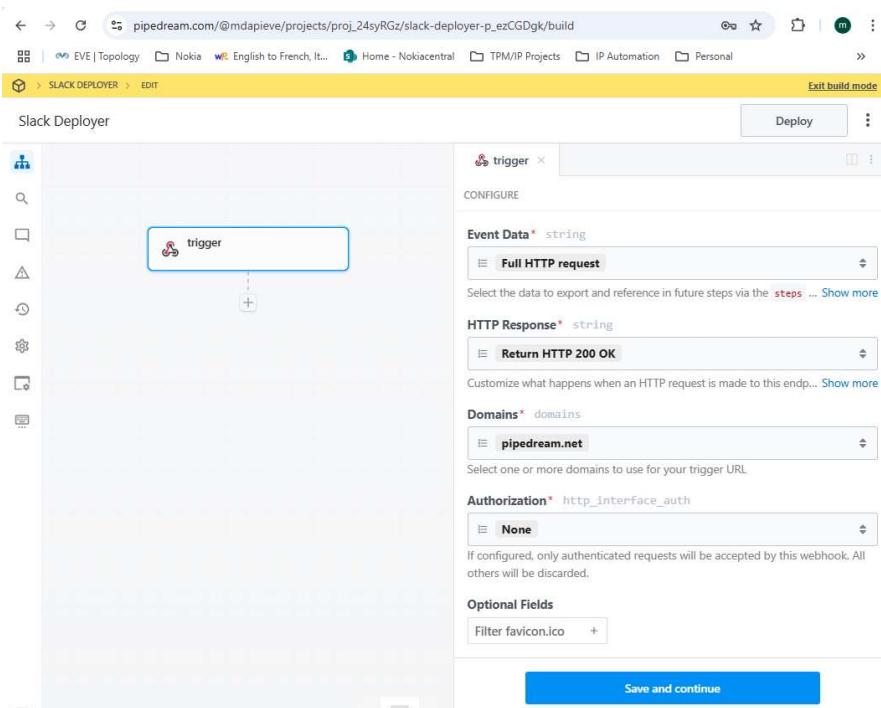
5. Add a trigger



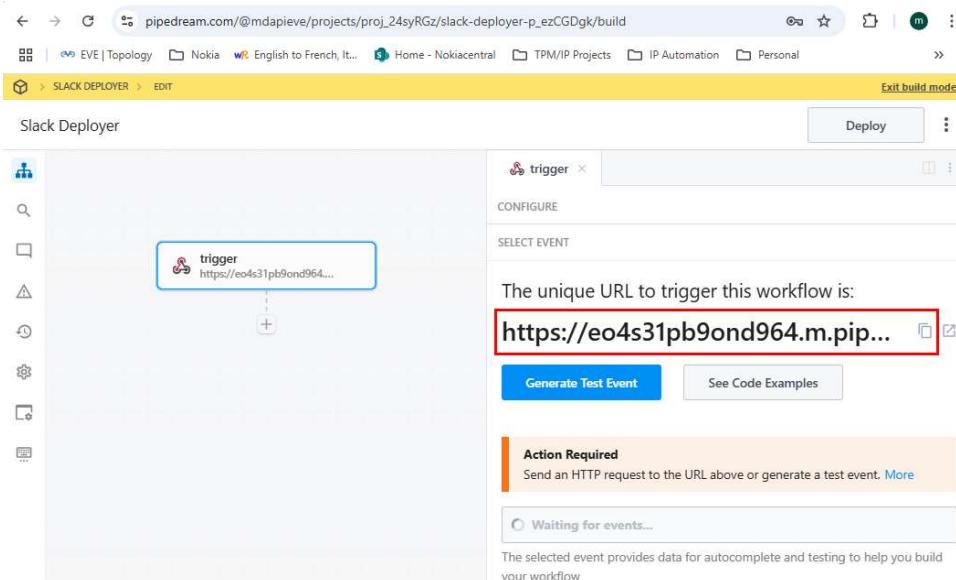
6. Select New HTTP/Webhook Requests

A screenshot of the Pipedream trigger selection interface. On the left, there's a sidebar with "Add trigger" and a search bar. Below it is a list of "Apps 2,400+": "My Sources" (selected), "HTTP / Webhook" (3 triggers), "OpenAI (ChatGPT)" (4 triggers), "Salesforce" (Premium, 4 triggers), "HubSpot" (Premium, 22 triggers), "Zoho CRM" (Premium, 9 triggers), "Stripe" (Premium, 11 triggers), "Shopify" (18 triggers), and "WooCommerce" (Premium, 4 triggers). On the right, under "Triggers → Popular", there is a list: "New HTTP / Webhook Requests" (highlighted with a red box), "Custom Interval", "New Emails", and "New Item in RSS Feed". At the bottom, there are buttons for "View My Sources" and a "View" button.

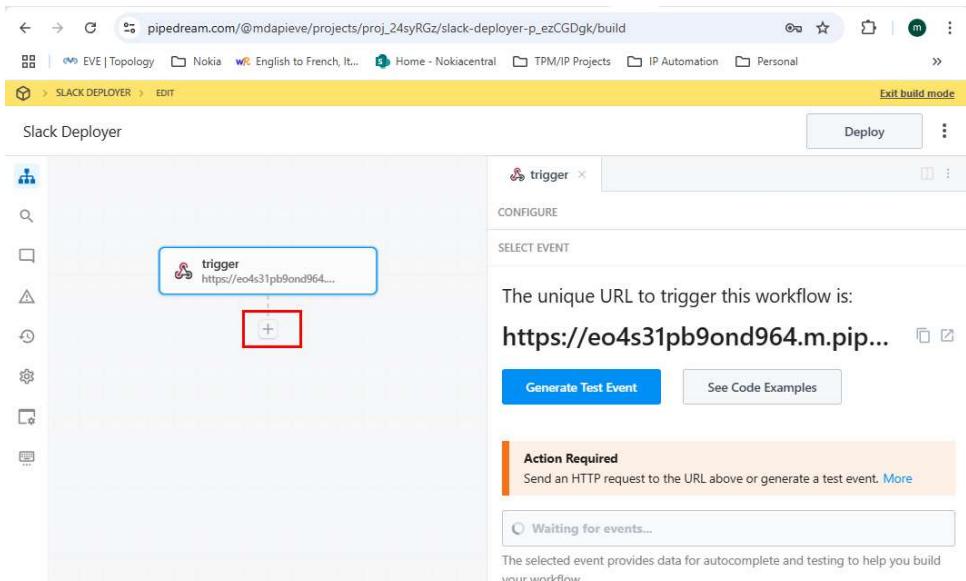
7. Then click on Save and Continue



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”

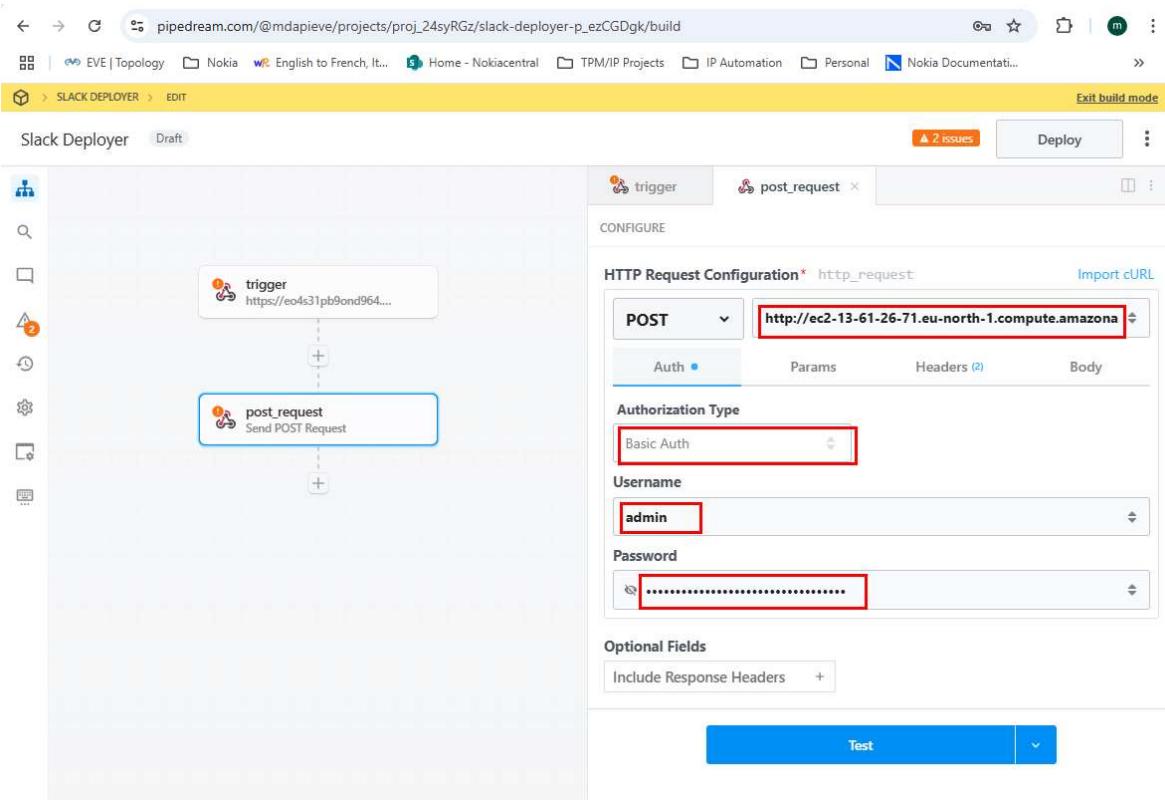
The screenshot shows the "My Actions" library in Pipedream. On the left, there's a sidebar with "Add action" and "Search apps...". The main area lists various actions under "Actions → Popular": "Build API Request", "Run custom code", "Query SQL Database", and "Return HTTP response". Under "Actions → Control Flow", there are "If/Else" (Beta), "Switch" (Beta), and "Parallel" (Beta). On the left, there's a list of published actions: "My Actions" (selected), "HTTP / Webhook" (highlighted with a red box), "Node", "Python", "OpenAI (ChatGPT)", "Salesforce Premium", "HubSpot Premium", "Zoho CRM Premium", and "Stripe Premium". At the bottom, there's a "View My Actions" button.

11. And then select “Send POST Request”:

The screenshot shows the Zapier interface for selecting an action under the 'HTTP / Webhook' category. The 'Custom Actions' section at the top contains three items: 'Build an HTTP request' (selected), 'Make an HTTP request in Node.js', and 'Make an HTTP request in Python'. Below this is the 'Pre-built Actions' section, which includes 'Send any HTTP Request', 'Send GET Request', and 'Send POST Request' (highlighted with a red box). A note at the bottom states 'Easily construct authenticated requests to the HTTP / Webhook API'. A 'Select Action' button is located in the bottom right corner.

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

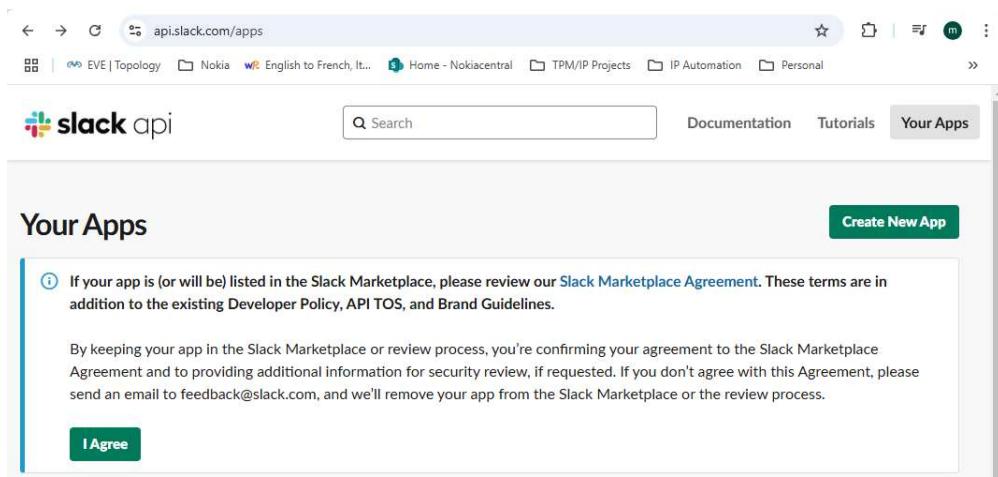


13. At the end of this process, click on the “Deploy” button.

2.7.1.2 Slack app creation

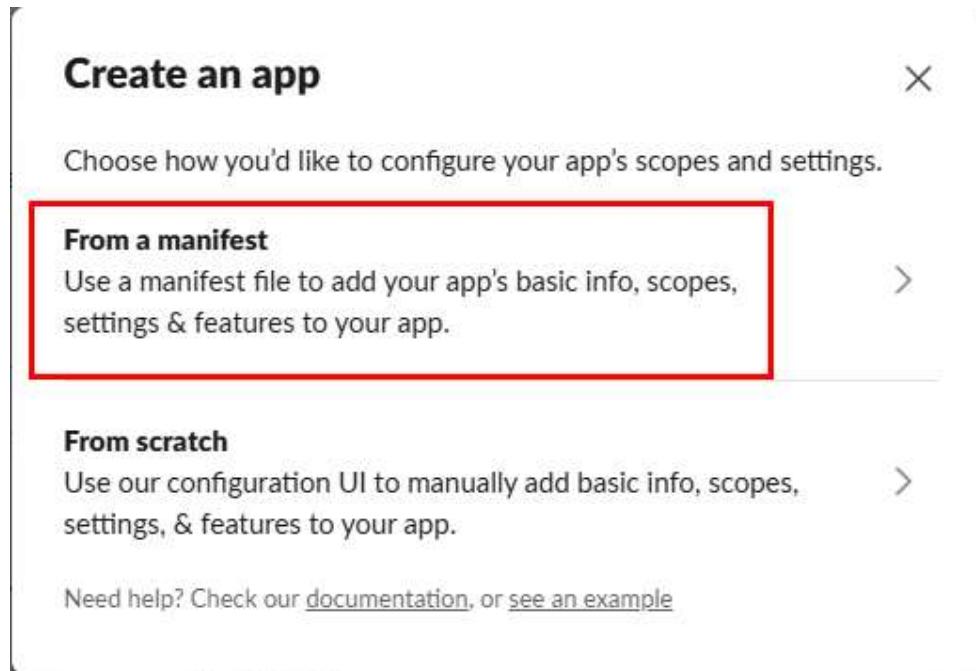
Now a Slack App can be created.

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



The screenshot shows a web browser window with the URL api.slack.com/apps in the address bar. The page header includes the Slack logo and navigation links for Documentation, Tutorials, and Your Apps. A prominent green button labeled "Create New App" is located in the top right corner. Below the button, there is a callout box with the following text:
If your app is (or will be) listed in the Slack Marketplace, please review our [Slack Marketplace Agreement](#). These terms are in addition to the existing Developer Policy, API TOS, and Brand Guidelines.
By keeping your app in the Slack Marketplace or review process, you're confirming your agreement to the Slack Marketplace Agreement and to providing additional information for security review, if requested. If you don't agree with this Agreement, please send an email to feedback@slack.com, and we'll remove your app from the Slack Marketplace or the review process.
A green "I Agree" button is at the bottom of the callout box.

2. Click on “From a Manifest”:



The screenshot shows a "Create an app" configuration page. At the top, it says "Choose how you'd like to configure your app's scopes and settings." Two options are listed: "From a manifest" and "From scratch". The "From a manifest" section is highlighted with a red box and contains the following text:
From a manifest
Use a manifest file to add your app's basic info, scopes, settings & features to your app.
The "From scratch" section contains:
From scratch
Use our configuration UI to manually add basic info, scopes, settings, & features to your app.
At the bottom, there is a link: "Need help? Check our [documentation](#), or [see an example](#)".

3. Select the workspace

Pick a workspace to develop your app

Pick a workspace to develop your app in:

Select a workspace

mdapieve@gmail.com's Workspace

workspace will control the app even if you leave the workspace.

[Sign into a different workspace](#)

Step 1 of 3

Cancel

Next

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

Create app from manifest

This is your app's manifest containing basic info, scopes, settings, and features. For help on how this works, you can check out our [documentation](#) or check out a few [examples](#).

[YAML](#) [JSON](#)

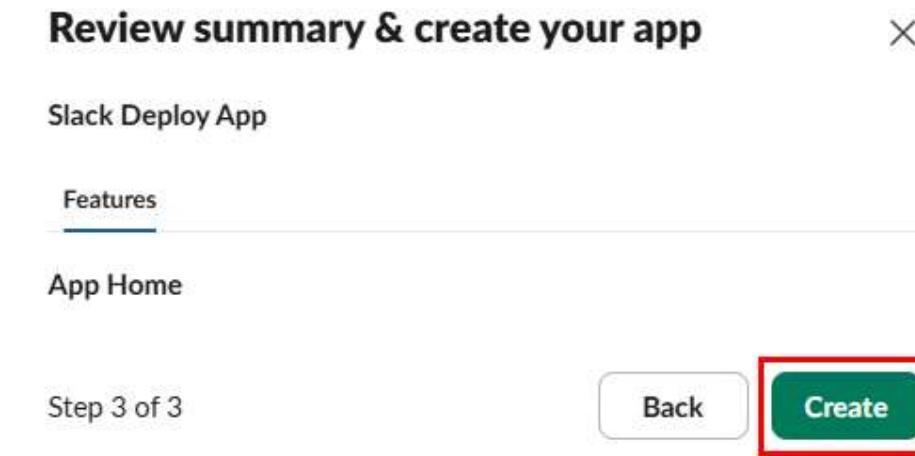
```
1 display_information:  
2   name: Slack Deploy App  
3 settings:  
4   org_deploy_enabled: false  
5   socket_mode_enabled: false  
6   is_hosted: false  
7   token_rotation_enabled: false  
8
```

Step 2 of 3

Back

Next

5. And then Create:



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

The screenshot shows the Slack API interface at the URL api.slack.com/apps/A0852ELSMRV/slash-commands. The page title is 'Slack Commands'. On the left, a sidebar lists sections like 'Settings', 'Features' (with 'Slash Commands' highlighted by a red box labeled '1'), and 'OAuth & Permissions'. The main content area shows a table for slash commands. A new command is being created with the name '/deploy' and the description 'trigger deploy via pipedream'. A 'Create New Command' button is highlighted with a red box labeled '2'.

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

← → ⌂ api.slack.com/apps/A0852ELSMRV/slash-commands? ⭐ 🗑️ 🔍 m

EVE | Topology Nokia English to French, It... Home - Nokiacentral TPM/IP Projects

Edit Command

Command	/deploy	i
Request URL	https://eov29xkrk0k585q.m.pipedre...	i
Short Description	trigger deploy via pipedream	
Usage Hint	[build]	
Optionally list any parameters that can be passed.		
Escape channels, users, and links sent to your app		<input type="checkbox"/>
Unescaped: @user #general		
Preview of Autocomplete Entry		
<div style="border: 1px solid #ccc; padding: 10px;"> <p>Commands matching "deploy"</p> <p>Slack Deploy App</p> <p>/deploy [build] trigger deploy via pipedream</p> </div>		
<input style="width: 40px; height: 40px; margin-right: 10px;" type="button" value="+"/> /deploy		

- 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):

The screenshot shows the configuration interface for a Slack app. The URL in the browser is `api.slack.com/apps/A0852ELSMRV/interactive-messages?`. The main header includes the Slack API logo, a search bar, and navigation links for Documentation, Tutorials, and Your Apps.

The left sidebar contains a dropdown menu for "Slack Deployments" and several sections: Settings (Basic Information, Collaborators, Socket Mode, Install App, Manage Distribution), Features (App Home, Agents & Assistants..., Workflow Steps [NEW], Org Level Apps, Incoming Webhooks, Interactivity & Shortcuts...), and Submit to Slack.

The "Interactivity & Shortcuts" section is currently selected. It has two main sections: "Interactivity" and "Shortcuts".

- Interactivity:** A toggle switch labeled "On" is highlighted with a red box. Below it, a description states: "Any interactions with shortcuts, modals, or interactive components (such as buttons, select menus, and datepickers) will be sent to a URL you specify. [Learn more.](#)"
- Request URL:** A text input field containing the URL `https://eov29xkrk0k585q.m.pipedream.net` is highlighted with a red box. Below the field, a note says: "Slack will send an HTTP POST request with information to this URL when users interact with a shortcut or interactive component."
- Shortcuts:** A table header with columns Name, Location, and Callback ID. A "Create New Shortcut" button is visible.
- Buttons at the bottom:** "Discard Changes" and "Save Changes" (which is also highlighted with a red box).

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

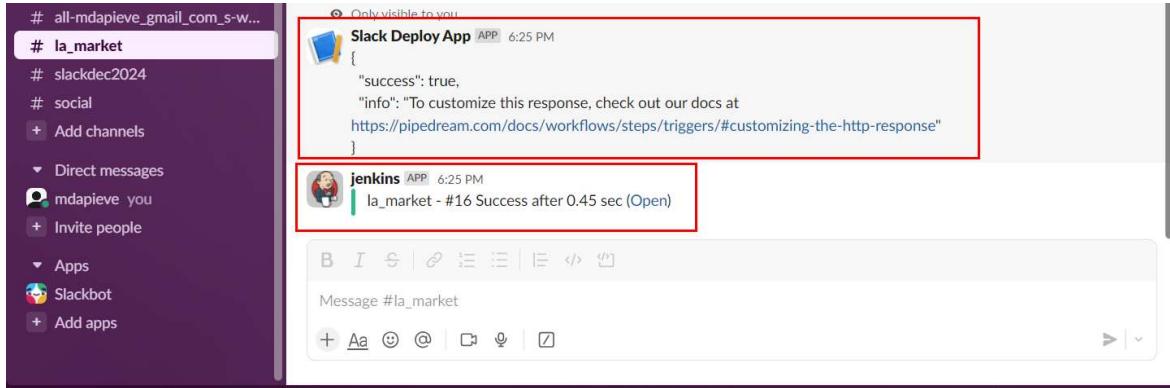
The screenshot shows the 'Installed App Settings' page for the 'Slack Deploy' app. On the left, there's a sidebar with sections like 'Settings' (Basic Information, Collaborators, Socket Mode), 'Features' (App Home, Agents & Assistants..., Workflow Steps, Org Level Apps, Incoming Webhooks), and an 'Install App' button which is highlighted with a red box. The main content area is titled 'OAuth Tokens'. It explains that tokens were generated for the workspace and provides a 'Bot User OAuth Token' field containing 'xoxb-7866948574641-8186951586739-3FuwXbZ563XKKuJ9lz'. A 'Copy' button is next to it, and below is an 'Access Level: Workspace' note. A 'Reinstall to mdapieve@gmail.com's Workspace' button is also present.

2.7.1.3 Testing

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

The screenshot shows a Slack channel named '# la_market'. A message from the 'Slack Deploy App' is visible, followed by a message from 'jenkins APP'. In the message input field at the bottom, the command '/deploy [build]' is typed, with a red box highlighting the input field.

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:

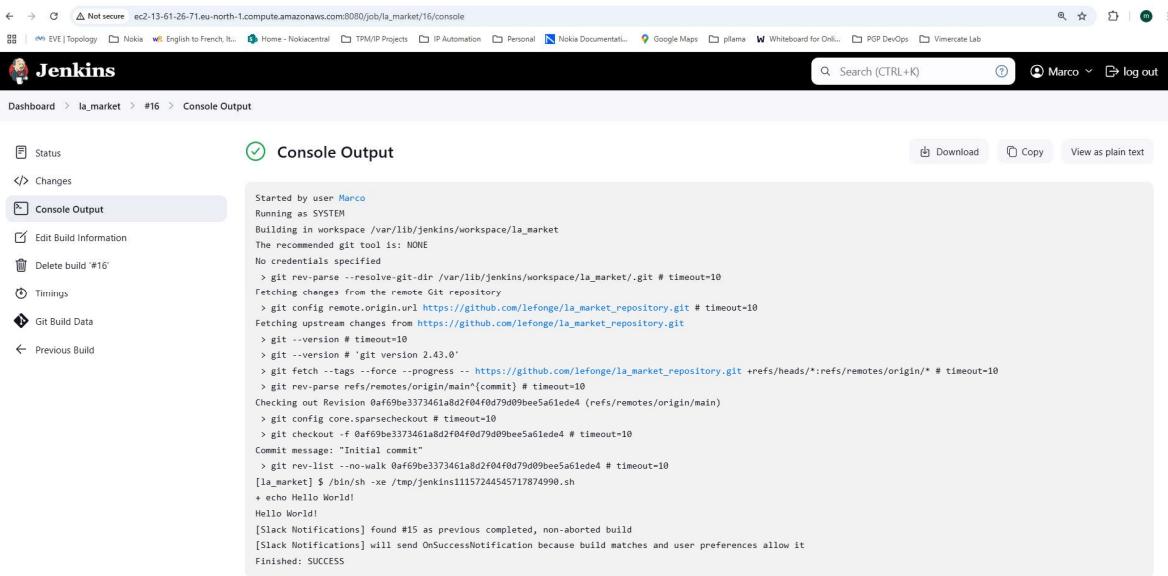
The screenshot shows the Pipedream dashboard for the 'Slack Deploy' workflow. On the left, there's a timeline of 'LIVE EVENTS' showing multiple POST requests triggered by Slack. On the right, the 'Success' section shows the execution details:

- Total Duration: 2.108 ms
- Compute Time: 1.743 ms
- Execution Start: 2024-12-17T17:25:40.037Z
- Execution End: 2024-12-17T17:25:42.145Z
- Steps Executed: 1
- Credits: 1
- Version: 18 (d_v7s8mQK0)

The 'trigger' step is expanded, showing the event details:

- method: POST
- path: /
- query: {}
- client_ip: 52.73.187.43
- url: https://ev29kkrk0K5B5q.m.pipedream.net/
- headers: {}
- body: {}
- token: o18KPCPgr5o5Lkz4oPc0wD9
- team_id: T07RG7NG0J0V
- team_domain: mdapiemail-wx16879
- channel_id: C08UPU9V2BT

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



The screenshot shows a Jenkins job named "la_market" with build number #16. The "Console Output" tab is selected. The output window displays the following log entries:

```
Started by user Marco
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/la_market
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/la_market/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lefonge/la_market_repository.git # timeout=10
Fetching upstream changes from https://github.com/lefonge/la_market_repository.git
> git --version # timeout=10
> git -v
> git fetch --tags --force --progress -- https://github.com/lefonge/la_market_repository.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0af69be3373461a8d2f04fd79d09bee5a61de4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0af69be3373461a8d2f04fd79d09bee5a61de4 # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 0af69be3373461a8d2f04fd79d09bee5a61de4 # timeout=10
[la_market] $ /bin/sh -xe /tmp/jenkins11157244545717874990.sh
+ echo Hello World!
Hello World!
[Slack Notifications] found #15 as previous completed, non-aborted build
[Slack Notifications] will send OnSuccessNotification because build matches and user preferences allow it
Finished: SUCCESS
```

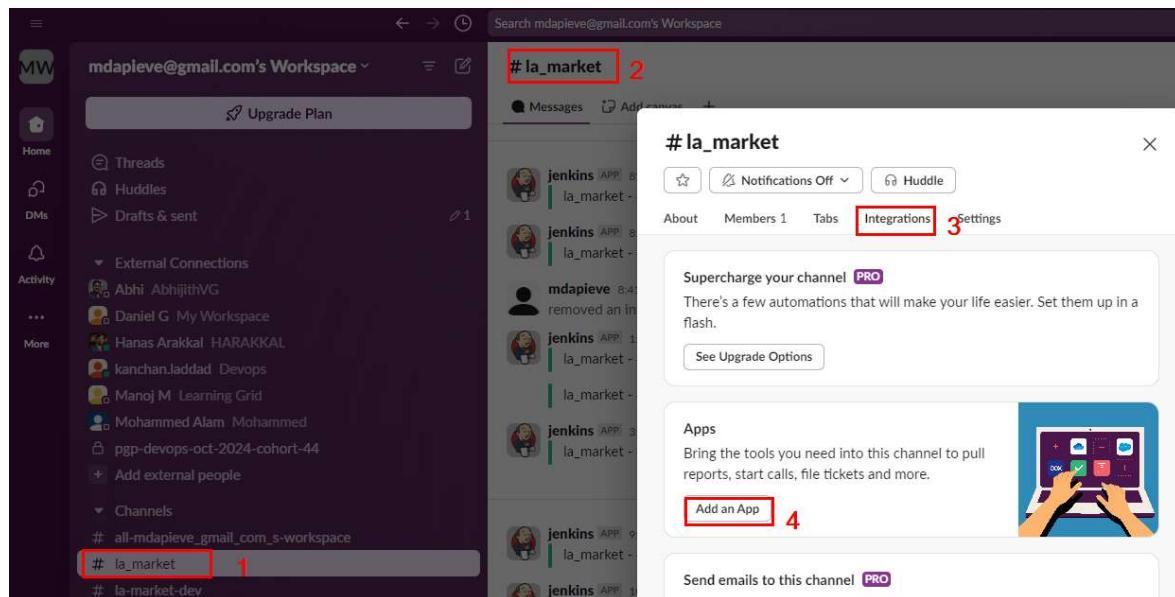
2.7.2 Slash Command Integration Solution

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

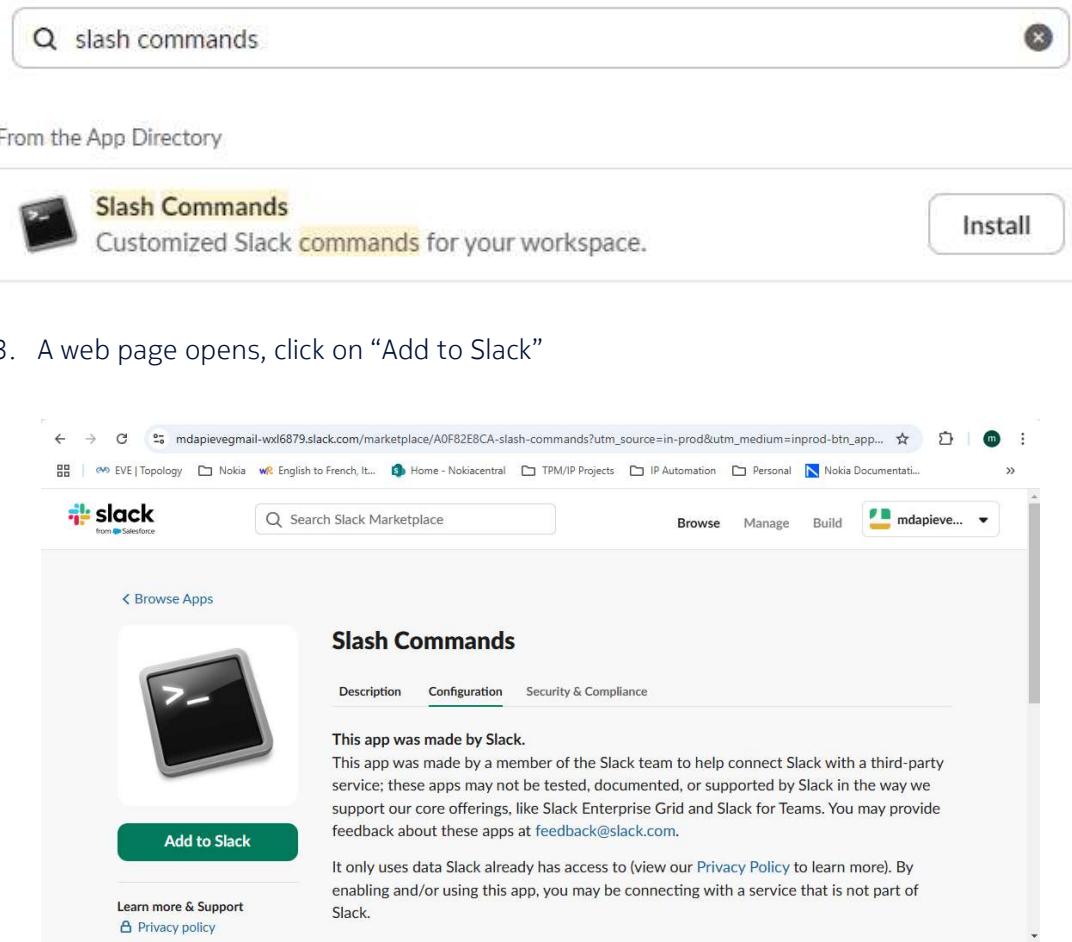
It is also possible to achieve the task of triggering a Jenkins job from slack by using a legacy integration of the slash command in Slack.

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”

Add apps to la_market



The screenshot shows the Slack Marketplace search results for "slash commands". A search bar at the top contains the query "slash commands". Below the search bar, a section titled "From the App Directory" lists the "Slash Commands" app. The app card includes a thumbnail of a keyboard key with a slash symbol, the app name "Slash Commands", a description "Customized Slack commands for your workspace.", and a prominent "Install" button.

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

The screenshot shows a web browser window displaying the Slack Marketplace page for the "Slash Commands" app. The URL in the address bar is https://mdapieve@gmail-wxld6879.slack.com/marketplace/A0F82E8CA-slash-commands?utm_source=in-prod&utm_medium=inprod-btn_app.... The page features the Slack logo and a search bar labeled "Search Slack Marketplace". The main content area displays the "Slash Commands" app card, which includes a thumbnail of a keyboard key with a slash symbol, the app name, a brief description, and a large green "Add to Slack" button. Navigation links like "Browse", "Manage", and "Build" are visible at the top right, along with a user profile icon.

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

The screenshot shows a web browser window with the URL mdapievegmai-wxl6879.slack.com/marketplace/new/A0F82E8CA-slash-commands. The page is titled "Slash Commands" under "Custom Integrations". A note at the top states: "Please note, this is a legacy custom integration - an outdated way for teams to integrate with Slack. These integrations lack newer features and they will be deprecated and possibly removed in the future. We do not recommend their use. Instead, we suggest that you check out their replacement: [Slack apps](#)." Below this, it says "Slash Commands allow you to listen for custom triggers in chat messages across all Slack channels. When a Slash Command is triggered, relevant data will be sent to an external URL in real-time." An example is given: "For example, typing `/weather 94070` could send a message to an external URL that would look up the current weather forecast for San Francisco and post it back to Slack." A form on the right allows users to "Choose a Command" (with the input field containing "/deploy") and has a large green "Add Slash Command Integration" button.

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`

The screenshot shows the 'Integration Settings' page for a Slack integration. The 'Command' field is set to '/deploy'. The 'URL' field contains the Jenkins URL: `http://ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080/job/la_mark`. The 'Method' is set to 'GET'. In the 'Token' section, a token is entered: `vz1eEYPbCcWCNM6KKIUUTKv`. Under 'Customize Name', the username is set to 'slash-command'. In the 'Customize Icon' section, there is a placeholder icon and a 'Upload an image' button. The 'Autocomplete help text' section includes a screenshot of the Slack autocomplete interface for the '/feedback' command. The 'Show this command in the autocomplete list' checkbox is checked, and the 'Description' field contains 'deploy to la_market'. The 'Usage hint' field contains '[build]'. Other sections like 'Feedback' and 'Metrics' are partially visible at the bottom.

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

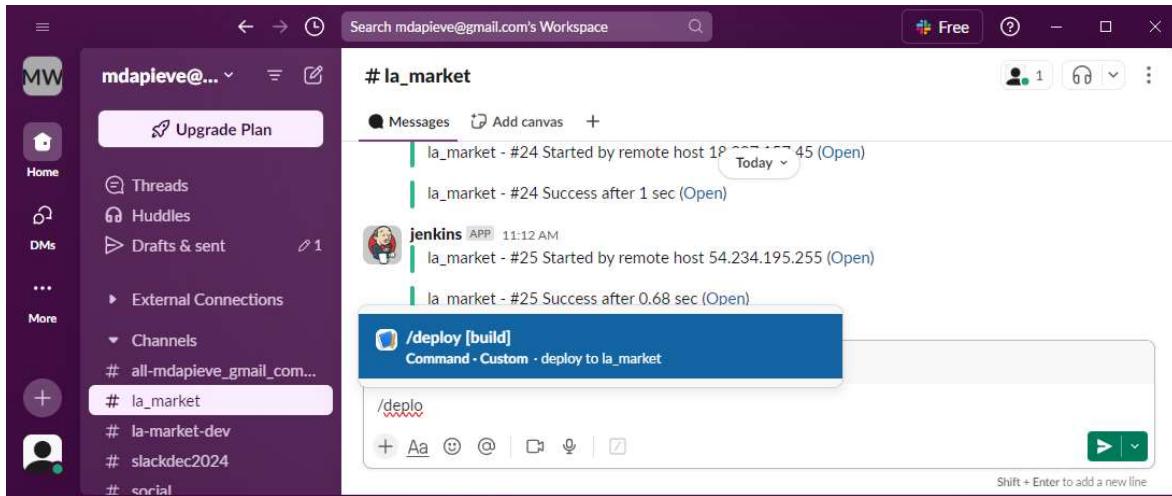
The screenshot shows the Jenkins Security configuration page. At the top, there's a navigation bar with links like 'Dashboard', 'Manage Jenkins', and 'Security'. Below that, the main title is 'Security'. Under 'Authentication', there's a checkbox for 'Disable "Keep me signed in"'. In the 'Security Realm' section, it says 'Jenkins' own user database'. Under 'Authorization', it says 'Logged-in users can do anything'. The 'Allow anonymous read access' checkbox is checked and has a red border around it, indicating it's the step being highlighted.

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

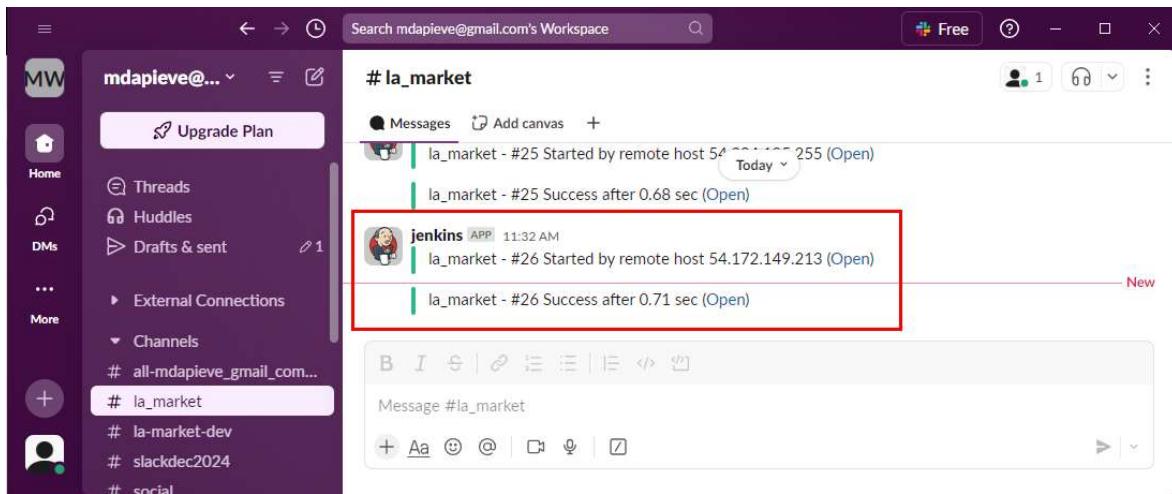
The screenshot shows the Jenkins job configuration page for a job named 'la_market'. Under the 'Build Triggers' section, the 'Trigger builds remotely (e.g., from scripts)' checkbox is checked and highlighted with a red box. Below it, there's a text input field containing 'token123'. To the right of the input field, there's a note: 'Use the following URL to trigger build remotely: JENKINS_URL/job/la_market/build?token=TOKEN_NAME or /buildWithParameters?token=TOKEN_NAME'. There are also other trigger options like 'Build after other projects are built', 'Build periodically', etc., which are not highlighted.

2.7.2.1.1 Testing

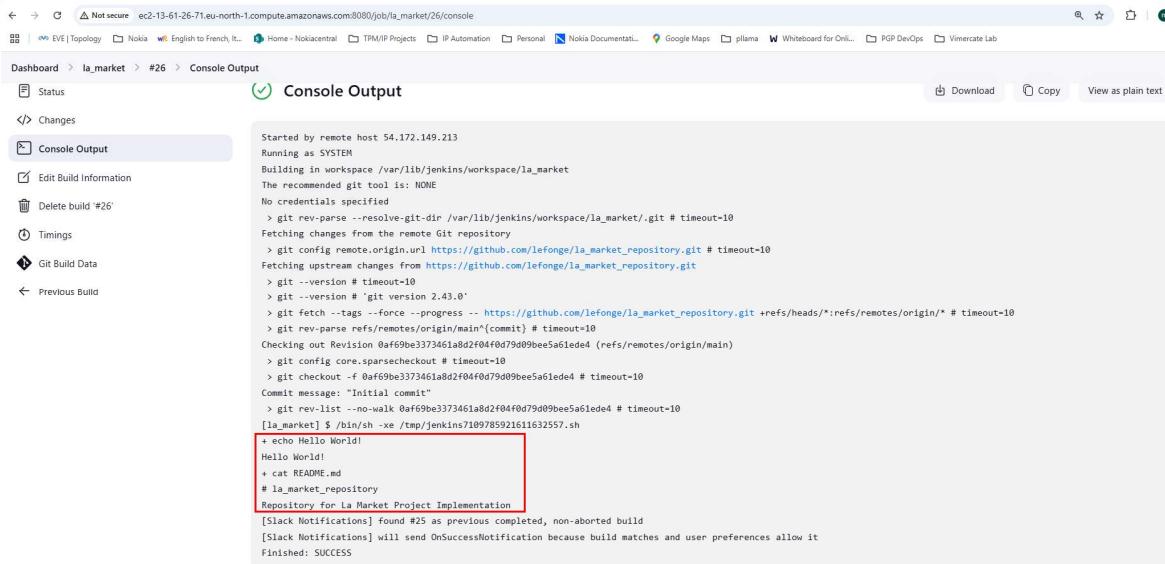
Now go to the Slack Channel and issue the "/deploy" command:



The job is executed successfully:



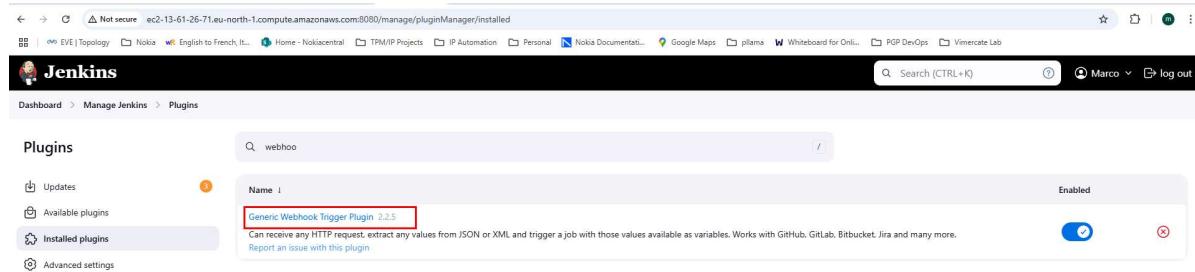
We can see the console output of the Jenkins Job's corresponding build triggered above where an echo command prints "Hello World" and a cat of the README.md file available on the provided github repository



```
Started by remote host 54.172.149.213
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/la_market
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/la_market/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lefonge/la_market_repository.git # timeout=10
Fetching upstream changes from https://github.com/lefonge/la_market_repository.git
> git -version # timeout=10
> git fetch --tags --force --progress -- https://github.com/lefonge/la_market_repository.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0af69be3373461a8d2f04f0d79d09bee5a61ede4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0af69be3373461a8d2f04f0d79d09bee5a61ede4 # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 0af69be3373461a8d2f04f0d79d09bee5a61ede4 # timeout=10
[la_market] $ /bin/sh -xe /tmp/jenkins7109785921611632597.sh
+ echo Hello World!
Hello World!
+ cat README.md
# la market repository
Repository for La Market Project Implementation
[Slack Notifications] found #25 as previous completed, non-aborted build
[Slack Notifications] will send OnSuccessNotification because build matches and user preferences allow it
Finished: SUCCESS
```

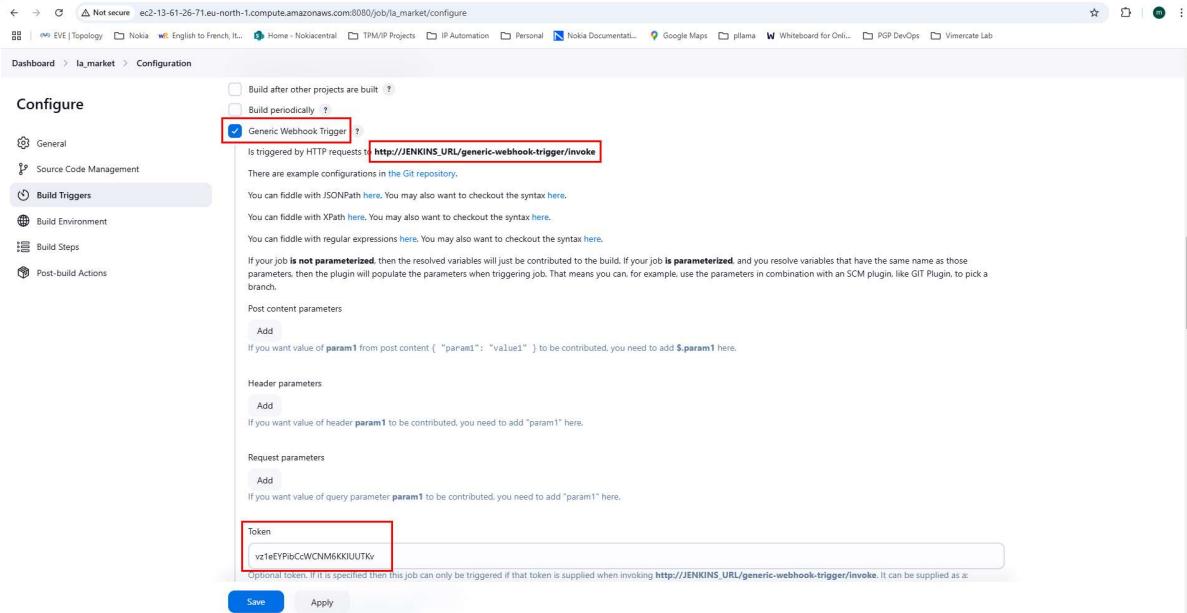
2.7.2.2 Legacy Integration configuration – POST Method

In this case, the following plugin needs to be installed



The screenshot shows the Jenkins Plugin Manager interface. The search bar at the top contains the text "webhook". In the "Installed plugins" section, the "Generic Webhook Trigger Plugin 2.2.5" is listed. The plugin's description is visible: "Can receive any HTTP request, extract any values from JSON or XML, and trigger a job with those values available as variables. Works with GitHub, GitLab, Bitbucket, Jira and many more." A link to report an issue is provided. The plugin is currently enabled, as indicated by the blue toggle switch.

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



The screenshot shows the configuration page for a Jenkins job named "la_market". Under the "Build Triggers" section, the "Generic Webhook Trigger" checkbox is selected, highlighted with a red box. Below this, the URL "http://JENKINS_URL/generic-webhook-trigger/invoke" is displayed. A token "v2TeEYRbCcWCNM6KKIUTKv" is entered into the "Token" field, also highlighted with a red box. At the bottom of the page, there are "Save" and "Apply" buttons.

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

The screenshot shows the 'Integration Settings' page for a Slack integration. The URL field and the token field are highlighted with red boxes.

Integration Settings

Command: /deploy

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

Method: POST

Token: vz1eEYPibCcWCNM6KKIUUTKv

Customize Name: slash-command

Customize Icon: (Icon placeholder) Upload an image

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

The screenshot shows the Jenkins 'Security' configuration page. At the top, there is a navigation bar with links to 'Dashboard', 'Manage Jenkins', and 'Security'. Below this, the main title is 'Security'. Under the 'Authentication' section, there is a checkbox labeled 'Disable "Keep me signed in"'. In the 'Security Realm' section, 'Jenkins' own user database' is selected. Under the 'Authorization' section, 'Logged-in users can do anything' is chosen. A red box highlights the checkbox 'Allow anonymous read access' under the authorization section.

← → ⚡ Not secure ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080/manage/configureSecurity/

EVE | Topology Nokia English to French, It... Home - Nokiacentral TPM/IP Projects IP Automation

Jenkins

Dashboard > Manage Jenkins > Security

Security

Authentication

Disable "Keep me signed in" ?

Security Realm

Jenkins' own user database

Allow users to sign up ?

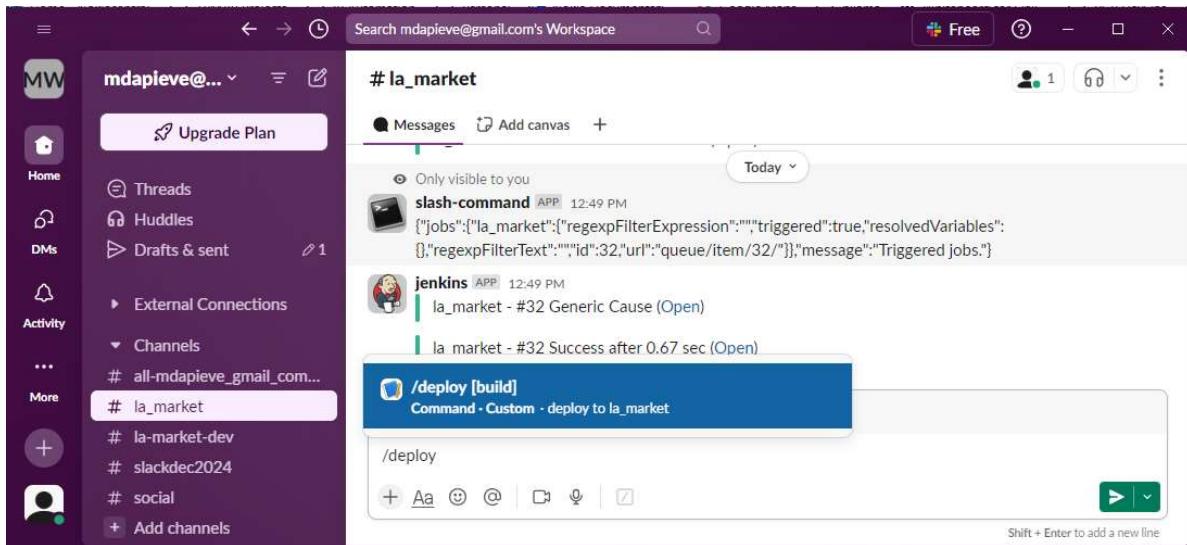
Authorization

Logged-in users can do anything

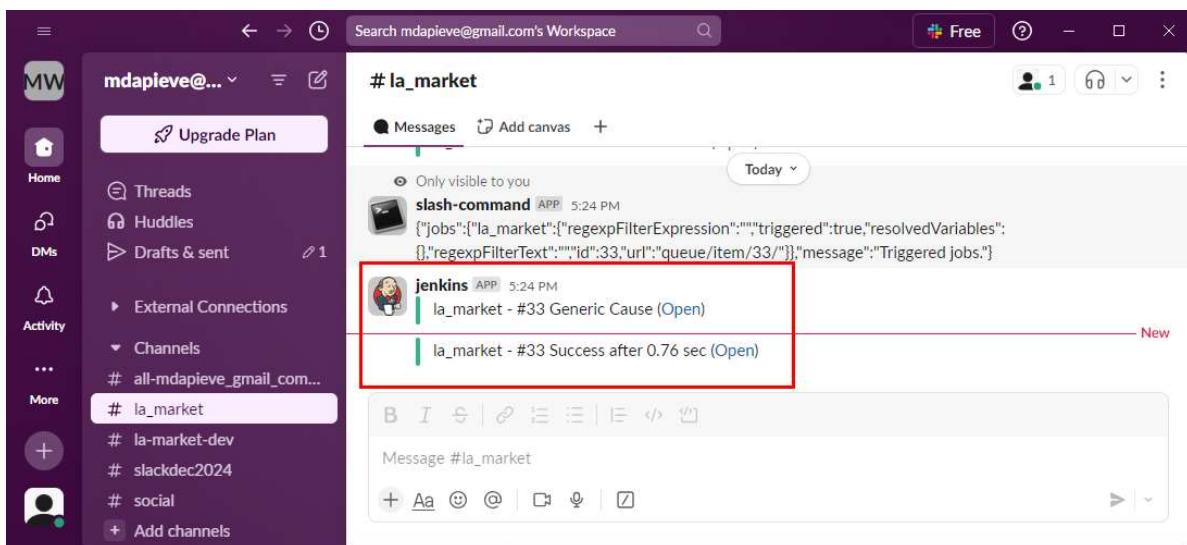
Allow anonymous read access ?

2.7.2.2.1 Testing

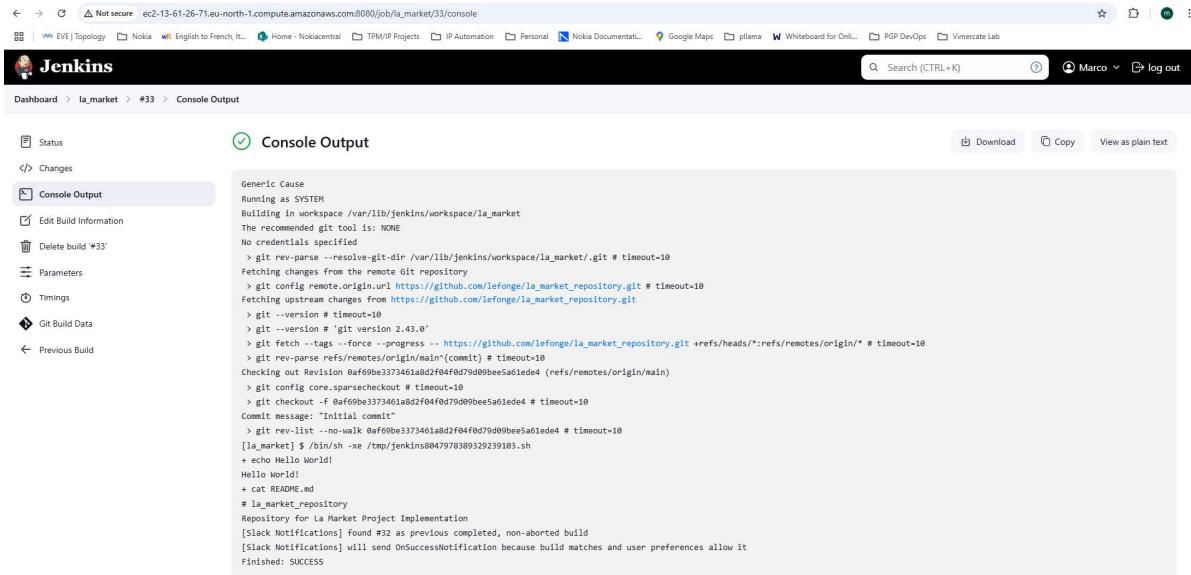
Go to slack and enter “/deploy” in the channel:



The Job is executed successfully and Slack is notified back:



The following is the successful job execution on Jenkins:



A screenshot of a Jenkins job's console output page. The URL in the address bar is `ec2-13-61-26-71.eu-north-1.compute.amazonaws.com:8080/job/la_market/33/console`. The page title is "Jenkins". The left sidebar shows navigation links: Status, Changes, Console Output (which is selected), Edit Build Information, Delete build #33, Parameters, Timings, Git Build Data, and Previous Build. The main content area is titled "Console Output" with a green checkmark icon. It displays the command-line output of the build process. The output shows the Jenkins environment, the git repository being cloned, the commit message "Initial commit", and the final "Finished: SUCCESS" message.

```
Generic Cause
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/la_market
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/la_market/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/lefonge/la_market_repository.git # timeout=10
Fetching upstream changes from https://github.com/lefonge/la_market_repository.git
> git -version # timeout=10
> git -version # git version 2.43.0"
> git fetch -t --progress - https://github.com/lefonge/la_market_repository.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/main^{commit} # timeout=10
Checking out Revision 0af60bce3373461a8d2f04f0d79080beef5a61de4 (refs/remotes/origin/main)
> git config core.sparsecheckout # timeout=10
> git checkout -f 0af60bce3373461a8d2f04f0d79080beef5a61de4 # timeout=10
Commit message: "Initial commit"
> git rev-list --no-walk 0af60bce3373461a8d2f04f0d79080beef5a61de4 # timeout=10
[la_market] $ /bin/sh -xe /tmp/jenkins864797838932939103.sh
+ echo Hello World!
Hello World!
+ cat README.md
# la_market_repository
Repository for La Market Project Implementation
[Slack Notifications] Found #32 as previous completed, non-aborted build
[Slack Notifications] will send OnSuccessNotification because build matches and user preferences allow it
Finished: SUCCESS
```

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.