

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

What is Continuous Integration (C.I.)?

Continuous integration is a **DevOps** software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

The key goals of continuous integration are to find and address bugs quicker, improve software quality, and reduce the time it takes to validate and release new software updates.

Benefits:

Continuous Integration Benefits



Improve Developer Productivity

Continuous integration helps your team be more productive by freeing developers from manual tasks and encouraging behaviors that help reduce the number of errors and bugs released to customers.



Find and Address Bugs Quicker

With more frequent testing, your team can discover and address bugs earlier before they grow into larger problems later.



Deliver Updates Faster

Continuous integration helps your team deliver updates to their customers faster and more frequently.

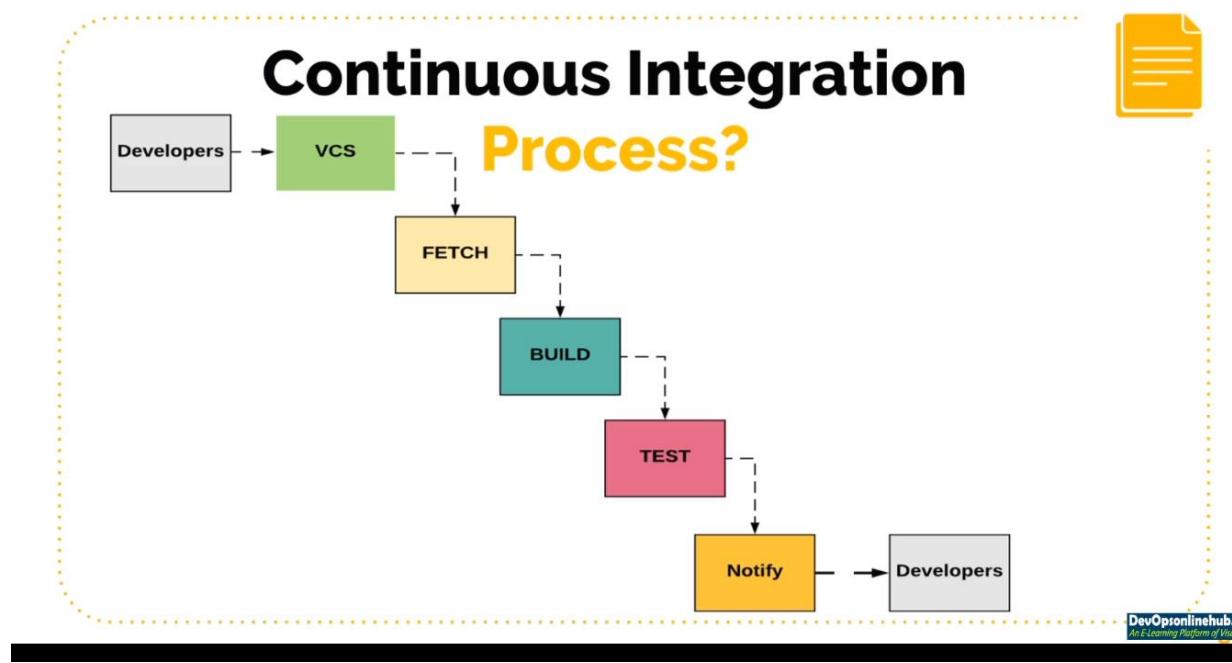
Merged But **Not Integrated**



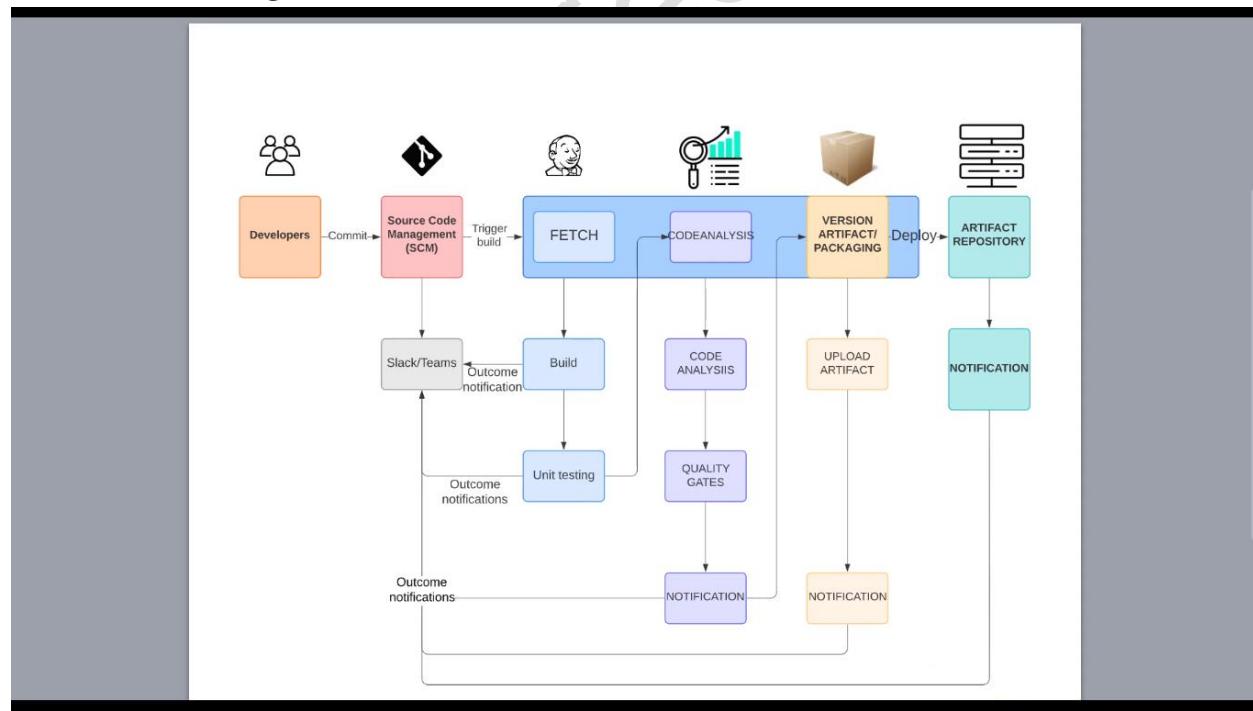
Developers keep merging code to VCS several times in a day. All the code collected from different developers would have generated conflicts and bugs.

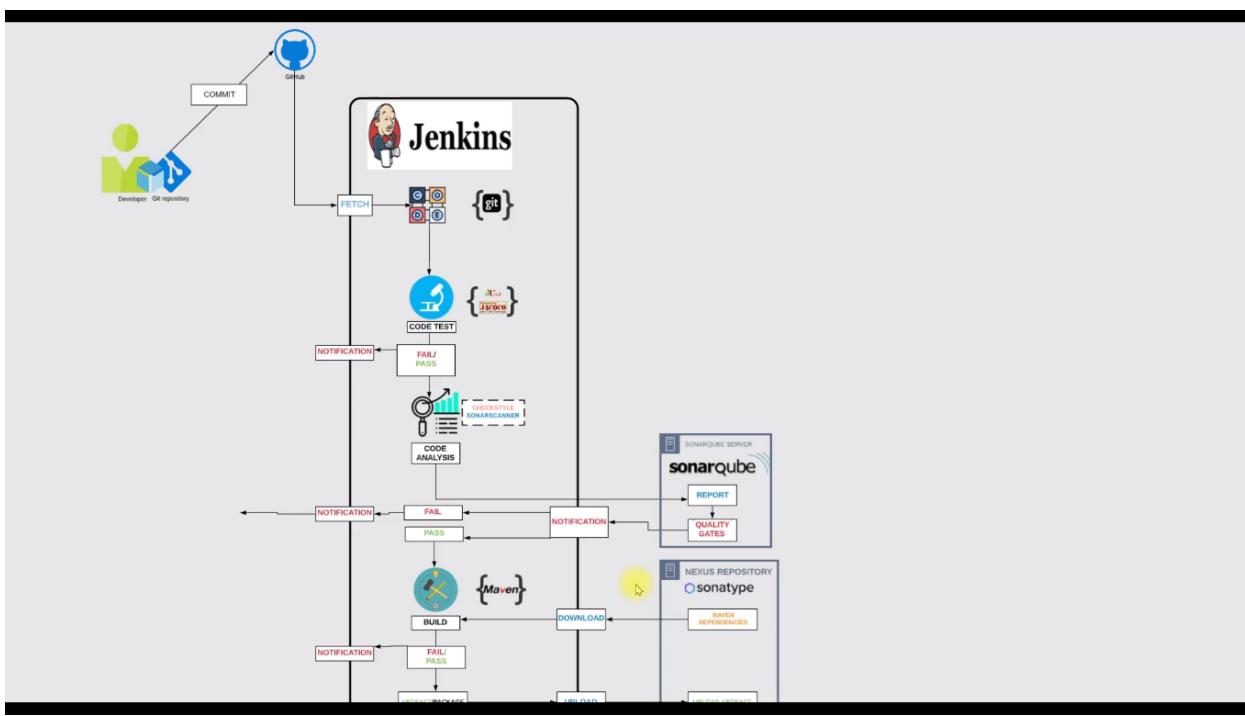
Code Merged from a long time when built throws Conflict, Bugs and errors. All these conflict, bugs & errors need to be resolved, which takes a very long time and halts development.

Process of CI:-



Continuous Integration WorkFlow:-





Pre-requisites:

1. Basic understanding of Jenkins
2. Basic knowledge of EC2 instances and linux
3. Awareness of Tools such as Nexus and SonarQube

Tools used:

Jenkins	Git	Maven	SonarQube	
Checkstyle	AWS EC2	Nexus3		

Stage -1

Key pair setup

Login to the AWS management console → EC2 service → Keypair and Create a new keypair for the purpose of creating EC2 instances

The screenshot shows two consecutive screenshots of the AWS Management Console interface.

Screenshot 1: Key Pairs List

- The left sidebar shows the navigation menu with "Key Pairs" selected under "Network & Security".
- The main content area displays a table titled "Key pairs (3) Info" with columns: Name, Fingerprint, and ID.
- The table lists three existing key pairs:
 - bhskrdevops_keypair (Fingerprint: 6d:f1:e6:57:05:ed:f1:07:a9:5cf6:0b:a..., ID: key-0c13b14b9f33a9726)
 - logic_labs_demo_keypair (Fingerprint: 7d:4b:61:15:04:f6:84:3a:97:d9:c7:4d..., ID: key-0fc76f434c31069b9)
 - ubuntu_docker (Fingerprint: 51:88:19:3c:92:0e:db:e4:ed:21:0d:88..., ID: key-091edaf6e0c1e0752)

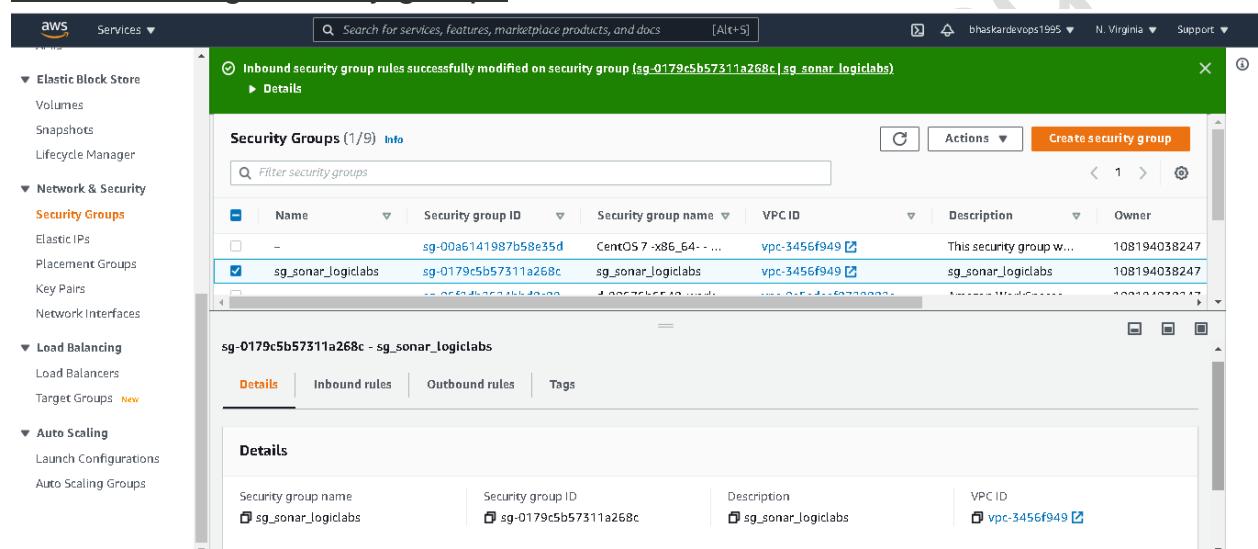
Screenshot 2: Create Key Pair Wizard

- The left sidebar shows the navigation menu with "Create key pair" selected under "Key Pairs".
- The main content area displays the "Create key pair" wizard with the following steps completed:
 - Key pair:** A key pair, consisting of a private key and a public key, is a set of security credentials that you use to prove your identity when connecting to an instance.
 - Name:** An input field labeled "Enter key pair name" with placeholder text "Name" and a note: "The name can include up to 255 ASCII characters. It can't include leading or trailing spaces."
 - Private key file format:** A radio button selection for ".ppk" (selected) and ".pem" (for use with OpenSSH).
 - Tags (Optional):** A note stating "No tags associated with the resource." and a "Add tag" button.
- At the bottom right of the wizard are "Cancel" and "Create key pair" buttons.

Security Groups setup

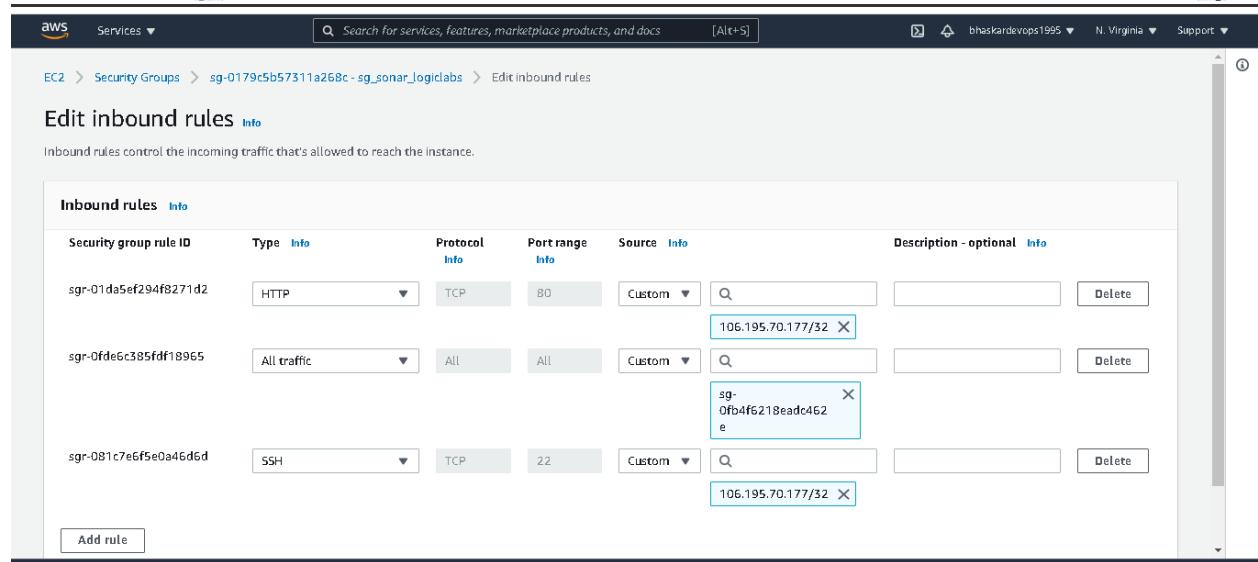
1. Login to AWS console-> Ec2-> Security Groups
2. Create three Security Groups (S.Gs) one for each- Jenkins, Sonar and Nexus
3. Edit inbound rules for each security group as per the screenshot below and leave outbound as it is.
4. If unclear about inbound rules, set “Allow all” for the practice purpose.

Creating security groups



The screenshot shows the AWS EC2 Security Groups page. A green success message at the top states: "Inbound security group rules successfully modified on security group (sg-0179c5b57311a268c|sg_sonar_logilabs)". Below this, the "Security Groups (1/9)" table lists one item: "sg_sonar_logilabs" (sg-0179c5b57311a268c). The table includes columns for Name, Security group ID, Security group name, VPC ID, Description, and Owner. The "sg_sonar_logilabs" row is selected. At the bottom of the table, there are "Details", "Inbound rules", "Outbound rules", and "Tags" tabs, with "Details" being the active tab.

Edit inbound rules



The screenshot shows the "Edit inbound rules" page for the "sg_sonar_logilabs" security group. It displays a table of three existing inbound rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-01da5ef294fb271d2	HTTP	TCP	80	Custom (106.195.70.177/32)	
sgr-0fde6c385fd18965	All traffic	All	All	Custom (sg-0fb4f6218eadc462e)	
sgr-081c7e6f5e0a46d6d	SSH	TCP	22	Custom (106.195.70.177/32)	

A "Delete" button is visible next to each rule entry. At the bottom left, there is a "Add rule" button.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

6

The screenshot shows two separate browser windows demonstrating AWS management console features.

Top Window: Shows the AWS Services navigation pane on the left with sections like Elastic Block Store, Network & Security (Security Groups highlighted), Load Balancing, and Auto Scaling. The main content area displays a table titled "Security Groups (1/9) Info" with one item selected: "sg_nexus_logilabs". Below this, a detailed view of the selected security group is shown, including its name, ID, description, and VPC ID.

Bottom Window: Shows the "Edit inbound rules" page for the selected security group. It lists four existing rules:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-064c8efeb97b464e	All traffic	All	All	Custom	sg-0f4f6218eadc462e
sgr-07054f0604a47d9f5	All traffic	All	All	My IP	106.195.70.177/32
sgr-0f23235717b5b48df	SSH	TCP	22	My IP	106.195.70.177/32
sgr-053eca1d75b9cd9fa	Custom TCP	TCP	8081	My IP	106.195.70.177/32

An "Add rule" button is visible at the bottom left of the page.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

7

The screenshot shows the AWS EC2 console interface. On the left, a navigation sidebar lists services: Elastic Block Store, Network & Security (Security Groups selected), Load Balancing, and Auto Scaling. The main content area displays the 'Security Groups' page with the title 'Security Groups (1/9) Info'. A table lists two security groups:

Name	Security group ID	Security group name	VPC ID	Description	Owner
sg_Jenkins_logilabs	sg-0fb4f6218eadc462e	sg_Jenkins_logilabs	vpc-3456f949	sg_Jenkins_logilabs	108194038247
default	sg-af1216a5	default	vpc-3456f949	default VPC security g...	108194038247

Below the table, a section titled 'sg-0fb4f6218eadc462e - sg_Jenkins_logilabs' contains tabs for Details, Inbound rules, Outbound rules, and Tags. The Details tab is selected, showing the following information:

Security group name sg_Jenkins_logilabs	Security group ID sg-0fb4f6218eadc462e	Description sg_Jenkins_logilabs	VPC ID vpc-3456f949
Owner 108194038247	Inbound rules count 8 Permission entries	Outbound rules count 2 Permission entries	

The browser address bar shows the URL: <https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#>. The bottom right corner of the screen shows the date and time: 09:03 09/07/2021.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

8

The screenshot shows the AWS EC2 Management Console with the 'Security Groups' section open. The table lists nine security groups, each with a unique ID, name, VPC ID, description, and owner. Below this, the 'Inbound rules' section is shown, listing five rules with their respective IDs, types, protocols, port ranges, sources, and descriptions.

Name	Security group ID	Security group name	VPC ID	Description	Owner
-	sg-00a141987b58e35d	CentOS 7-x86_64 - ...	vpc-3456f949	This security group w...	108194038247
sg_sonar_logilabs	sg-0179c5b57311a268c	sg_sonar_logilabs	vpc-3456f949	sg_sonar_logilabs	108194038247
-	sg-05f2db2624bb8c89	d-90676b548_work...	vpc-0a5adaef8729882a...	Amazon WorkSpaces ...	108194038247
-	sg-0719809445891239	default	vpc-0a5adaef8729882a...	default VPC security g...	108194038247
sg_nexus_logilabs	sg-09c995f450660a722	sg_nexus_logilabs	vpc-3456f949	sg_nexus_logilabs	108194038247
-	sg-035e080d604581b	launch-wizard-1	vpc-3456f949	launch-wizard-1 creat...	108194038247
-	sg-0f49d2b4a1c09275	d-90676b548_contr...	vpc-0a5adaef8729882a...	AWS created security ...	108194038247
sg_Jenkins_logilabs	sg-0fb4f6218eadc462e	sg_Jenkins_logilabs	vpc-3456f949	sg_Jenkins_logilabs	108194038247
-	sg-af1216a5	default	vpc-3456f949	default VPC security g...	108194038247

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0c083b3021f66a8ac	All traffic	All	All	Custom	sg-09c995f450660a722
sgr-016277bd10bde1f1d	All traffic	All	All	Custom	sg-0179c5b57311a268c
sgr-004a349cf1afac17a	Custom TCP	TCP	8080	My IP	106.195.70.177/32
sgr-0eb832d699ef5ebd1	HTTP	TCP	80	My IP	106.195.70.177/32
sgr-05bd6f12434980f3c	SSH	TCP	22	My IP	106.195.70.177/32

Server setup

We need to create 3 servers for Jenkins, Sonar and Nexus3.

Step1: clone source code and user data from repository

<https://github.com/gnsharma530/Logilabs-cidemo.git>

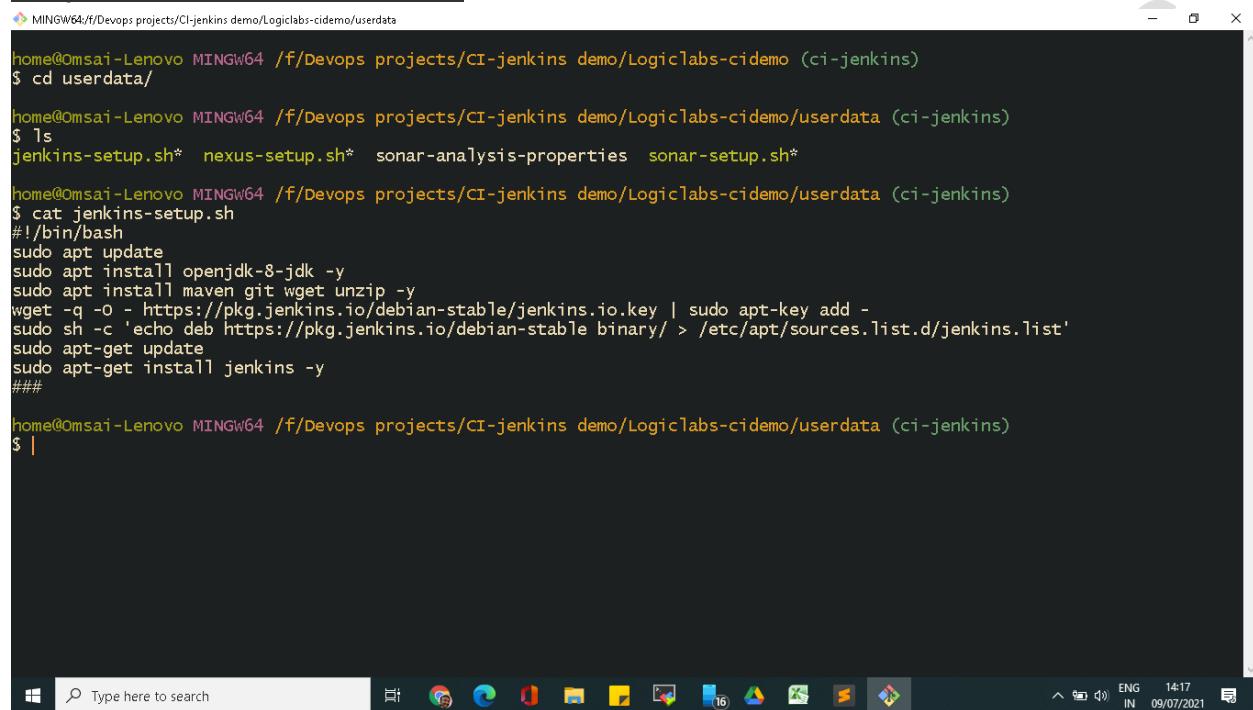
Step2: checkout to ci-jenkins branch and open userdata folder:

Commands to checkout branch:

git checkout ci-jenkins

cd userdata

Step3 : observe the files in it



```
MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata
$ cd userdata

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo (ci-jenkins)
$ ls
jenkins-setup.sh* nexus-setup.sh* sonar-analysis-properties sonar-setup.sh*
home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata (ci-jenkins)
$ cat jenkins-setup.sh
#!/bin/bash
sudo apt update
sudo apt install openjdk-8-jdk -y
sudo apt install maven git wget unzip -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
sudo apt-get update
sudo apt-get install jenkins -y
###

home@Omsai-Lenovo MINGW64 /f/Devops projects/CI-jenkins demo/Logiclabs-cidemo/userdata (ci-jenkins)
$ |
```

step4 : open the AWS console to create the server ec2 instances

AWS console→ ec2→ create instance

Choice of Instance:

Jenkins:t2.small

SonarQube:t2.small

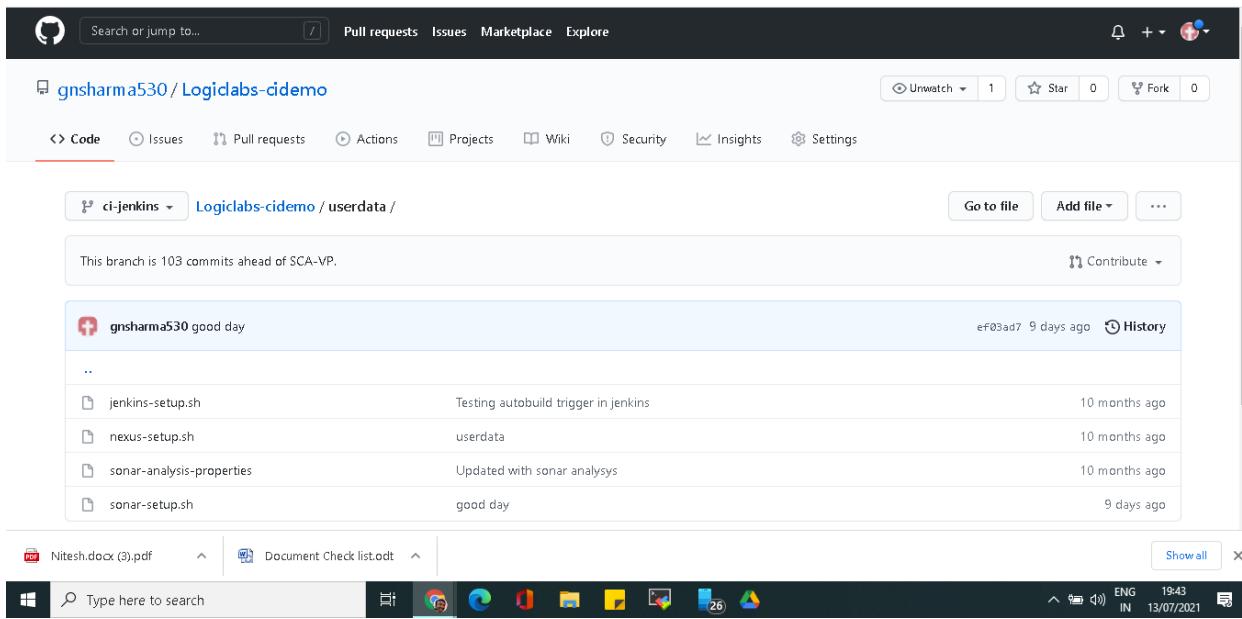
Nexus: t2.medium

-Select the relevant security groups

-paste the proper user data from the user data folder of the repository.

Clone the repository to the user data:

<https://github.com/gnsharma530/Logiclabs-cidemo.git>



Step 4.1:

Jenkins server creation : OS→ UBUNTU

Instance type → t2 small/medium

** Avoid t2 micro as it might not handle the load smoothly

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

11

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▾ Current generation ▾ Show/Hide Columns

Currently selected: t2.small (~ ECUs, 1 vCPUs, 2.5 GHz, ~ 2 GiB memory, EBS only)

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
t2	t2.micro Free tier eligible	1	1	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/> t2	<input checked="" type="checkbox"/> t2.small	1	2	EBS only	-	Low to Moderate	Yes
t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

Copy user data from the cloned repo and paste in the user data field. Select the relevant security group

<https://github.com/gnsharma530/Logiclabs-cidemo/tree/ci-jenkins/userdata>

Step 3: Configure Instance Details

Additional charges apply.

Credit specification ▾ Unlimited Additional charges may apply

File systems ▾ Add file system Create new file system

Advanced Details

Enclave ▾ Enable

Metadata accessible ▾ Enabled

Metadata version ▾ V1 and V2 (token optional)

Metadata token response hop limit ▾ 1

User data ▾ As text As file Input is already base64 encoded

```
#!/bin/bash
sudo apt update
sudo apt install openjdk-8-jdk -y
sudo apt install maven git wget unzip -y
wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
```

Cancel Previous Review and Launch Next: Add Storage

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

12

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group:

- Create a new security group
- Select an existing security group

Security Group ID	Name	Description
sg-00a6141987b58e35d	CentOS 7 -x86_64 - with Updates HVM-2002_01-AutogenByAWSMP	This security group was generated by AWS Marketplace and is based on recommended settings for CentOS 7 (x86_64) - with Updat...
sg-a1f1b65	default	default VPC security group
sg-035e080df04581b	launch-wizard-1	launch-wizard-1 created 2021-06-02T22:04:00.760+05:30
sg-0fb4f6218eadc462e	sg_Jenkins_logiclabs	sg_Jenkins_logiclabs
sg-09c995f450660a722	sg_nexus_logiclabs	sg_nexus_logiclabs

Inbound rules for sg-0fb4f6218eadc462e (Selected security groups: sg-0fb4f6218eadc462e)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
HTTP	TCP	80	::/0	
Custom TCP Rule	TCP	8080	0.0.0.0/0	

[Cancel](#) [Previous](#) [Review and Launch](#)

Step 7: Review Instance Launch

Please review your instance launch details. You can go back at any time.

AMI Details

Ubuntu Server 18.04 LTS (HVM), SSD Volume Type

Instance Type

Instance Type	ECUs	vCPUs	Memory (GiB)
t2.micro	0.05	1	1.75

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance. Amazon EC2 supports ED25519 and RSA key pair types. ED25519 keys are smaller and faster while offering the same level of security as RSA keys. Use ED25519 keys to improve the speed of authentication or if you have regulatory requirements that mandate the use of ED25519 keys.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair
 Select a key pair
logic_labs_demo_keypair

I acknowledge that I have access to the selected private key file (logic_labs_demo_keypair.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

[Edit AMI](#) [Edit instance type](#) [Work Performance](#)

[Cancel](#) [Previous](#) [Launch](#)

Step 5: Repeat the same steps for the below servers

Sonatype Nexus3 server:

OS: Centos7 & Instance type: t2 medium

NEXUS SERVER SETUP

The screenshot shows the 'Launch instance wizard | EC2 Management Console' on the AWS website. The user is at Step 1: Choose an Amazon Machine Image (AMI). A search bar at the top contains 'centos'. The results list shows two entries under 'AWS Marketplace (424)'. The first entry is 'CentOS 7 (x86_64) - with Updates HVM' by CentOS.org, which is marked as 'Free tier eligible'. The second entry is another 'CentOS 7 (x86_64) - with Updates HVM' by Amazon Web Services. Both entries have a 'Select' button to the right. The browser's address bar shows the URL: https://console.aws.amazon.com/ec2/v2/home?region=us-east-1#LaunchInstanceWizard:.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

14

Instance Type	Software	EC2	Total
t2.nano	\$0.00	\$0.006	\$0.006/hr
t2.micro	\$0.00	\$0.012	\$0.012/hr
t2.small	\$0.00	\$0.023	\$0.023/hr
t2.medium	\$0.00	\$0.046	\$0.046/hr
t2.large	\$0.00	\$0.093	\$0.093/hr
t2.xlarge	\$0.00	\$0.186	\$0.186/hr
t2.2xlarge	\$0.00	\$0.371	\$0.371/hr
t3.nano	\$0.00	\$0.005	\$0.005/hr
t3.micro	\$0.00	\$0.01	\$0.01/hr
t3.small	\$0.00	\$0.021	\$0.021/hr
t3.medium	\$0.00	\$0.042	\$0.042/hr
t3.large	\$0.00	\$0.083	\$0.083/hr
t3.xlarge	\$0.00	\$0.166	\$0.166/hr
t3.2xlarge	\$0.00	\$0.333	\$0.333/hr
t3a.nano	\$0.00	\$0.005	\$0.005/hr
t3a.micro	\$0.00	\$0.009	\$0.009/hr
t3a.small	\$0.00	\$0.019	\$0.019/hr
t3a.medium	\$0.00	\$0.038	\$0.038/hr
t3a.large	\$0.00	\$0.076	\$0.076/hr

Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

15

The screenshot shows a Windows desktop environment with several open windows:

- Sublime Text:** The main window displays a file named `nexus-setup.sh` containing bash script code for setting up Nexus. The code includes commands like `yum install java-1.8.0-openjdk.x86_64`, `mkdir -p /opt/nexus/`, and configuration for a systemd service named `nexus`.
- AWS Management Console:** An EC2 instance creation wizard is open at step 3: Configure Instance Details. In the `User data` section, a shell script is pasted:

```
echo 'run_as_user="nexus"' > /opt/nexus/$NEXUSDIR/bin/nexus.rc
systemctl daemon-reload
systemctl start nexus
systemctl enable nexus
```
- Taskbar:** The taskbar at the bottom shows various pinned icons including File Explorer, Edge, Google Chrome, FileZilla, and others.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

16

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-0cb4f601f0d3c	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Cancel Previous Review and Launch Next: Add Tags

Step 6: Configure Security Group

Assign a security group: Create a new security group Select an existing security group

Security Group ID	Name	Description
sg-00a6141987b58e35d	CentOS 7 -x86_64- - with Updates HVM-2002_01-AutogenByAWSMP	This security group was generated by AWS Marketplace and is based on recommended settings for CentOS 7 (x86_64) - with Updates HVM-2002_01-AutogenByAWSMP
sg-a1f216a5	default	default VPC security group
sg-0a35e080df604581b	launch-wizard-1	launch-wizard-1 created 2021-06-02T22:04:00.760+05:30
sg-0fb4f6218eadc462e	sg_Jenkins_logiclabs	sg_Jenkins_logiclabs
sg-09c995450660a722	sg_nexus_logiclabs	sg_nexus_logiclabs
sg-0179c5b57311a268c	sg_sonar_logiclabs	sg_sonar_logiclabs

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	
All traffic	All	All	sg-0fb4f6218eadc462e (sg_Jenkins_logicla	
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	::/0	

Cancel Previous Review and Launch

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

17

Type	Protocol	Port Range	Source	Description
All traffic	All	All	0.0.0.0/0	
All traffic	All	All	::/0	
All traffic	All	All	sg-0fb4f6218eadc462e (sg_Jenkins_logiclab)	
SSH	TCP	22	0.0.0.0/0	
SSH	TCP	22	::/0	
Custom TCP Rule	TCP	8081	0.0.0.0/0	
Custom TCP Rule	TCP	8081	::/0	

Step 6: Configure Security Group
Inbound rules for sg-09c995f450660a722 (Selected security groups: sg-09c995f450660a722)

Cancel Previous Review and Launch

Step 7: Review Instance Launch
sg-09c995f450660a722

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair
Select a key pair
logic_labs_demo_keypair

I acknowledge that I have access to the corresponding private key file, and that without this file, I won't be able to log into my instance.

Cancel Launch Instances

sg_nexus_logiclabs

Instance Details Storage Tags

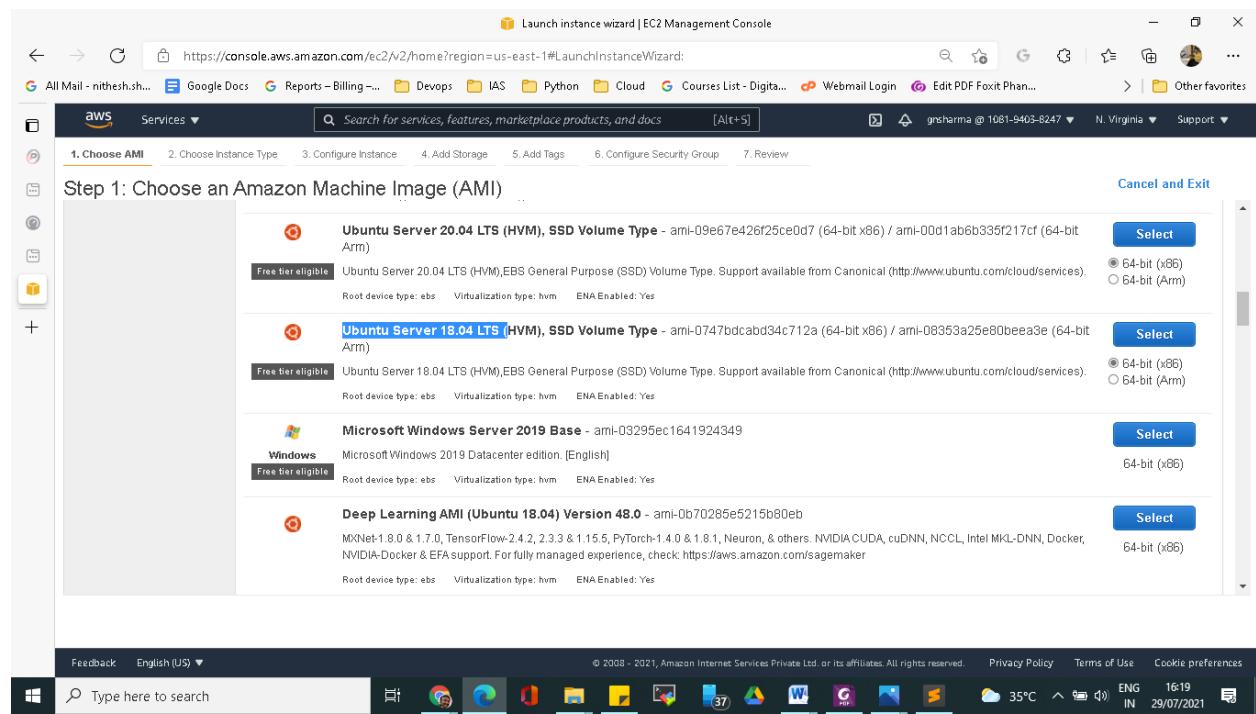
Edit instance details Edit storage Edit tags

Cancel Previous Launch

LogicLab Technologies

SONARQUBE SERVER SETUP:

OS: Ubuntu Instance type: t2 medium



A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

20

The screenshot shows the 'Launch instance wizard | EC2 Management Console' on the AWS website. The user is at Step 2: Choose an Instance Type. A table lists various t2 instance types with their details:

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes

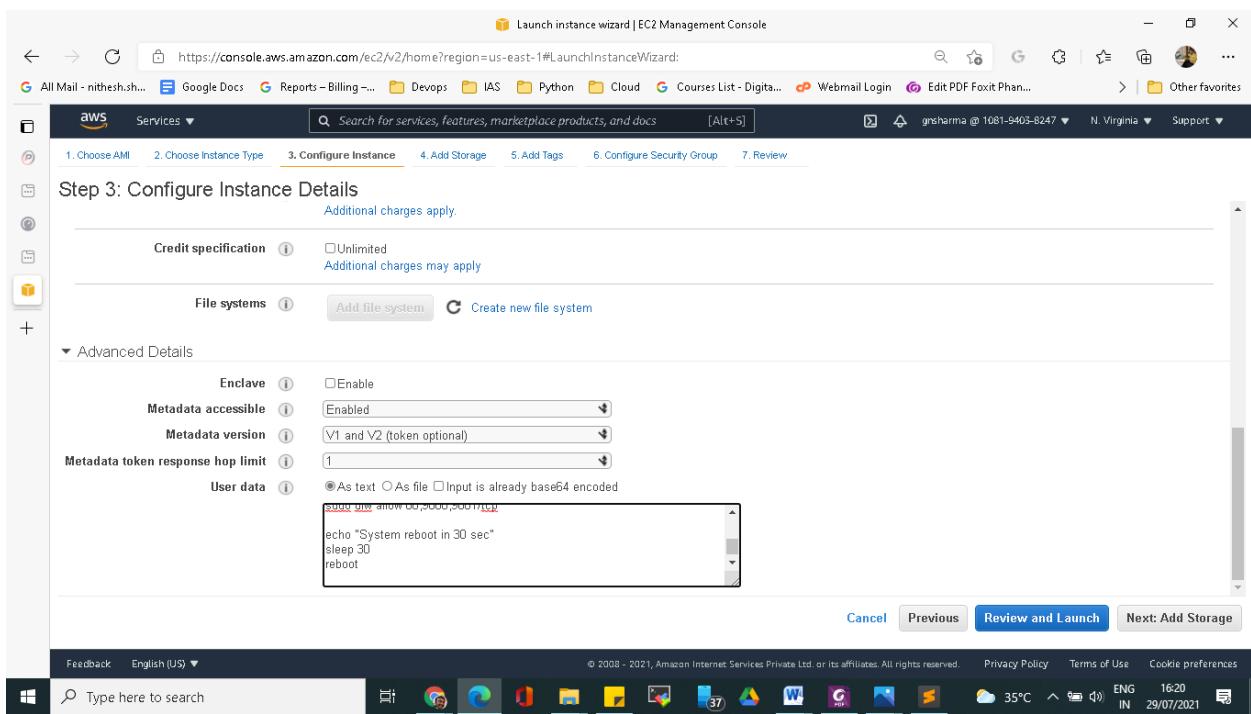
Buttons at the bottom include 'Cancel', 'Previous', 'Review and Launch' (highlighted in blue), and 'Next: Configure Instance Details'.

The screenshot shows a Windows desktop environment. In the foreground, a Sublime Text window is open, displaying a bash script named 'sonar-setup.sh'. The script contains commands for setting up SonarQube, including file backups, ulimits, and PostgreSQL configuration. The Sublime Text interface includes a sidebar showing project files like 'vprofile-project', 'sonar-analysis-properties', and 'sonar-setup.sh'. The taskbar at the bottom shows icons for various applications including a browser, file explorer, and system tools. The system tray indicates the date as 29/07/2021 and the time as 16:19.

```
#!/bin/bash
cp /etc/sysctl.conf /root/sysctl.conf_backup
cat <<EOT> /etc/sysctl.conf
4 vm.max_map_count=262144
5 fs.file-max=65536
6 ulimit -n 65536
7 ulimit -u 4096
8 EOT
9 cp /etc/security/limits.conf /root/sec_limit.conf_backup
10 cat <<EOT> /etc/security/limits.conf
11 sonarqube - nofile 65536
12 sonarqube - nproc 409
13 EOT
14
15 sudo apt-get update -y
16 sudo apt-get install openjdk-11-jdk -y
17 sudo update-alternatives --config java
18 java -version
19
20 sudo apt update
21 wget -q https://www.postgresql.org/media/keys/ACCC4CF8.asc -O - | sudo apt-key add -
22
23 sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt/ `lsb_release -cs`-pgdg main" >> /etc/a
24 sudo apt install postgresql postgresql-contrib -y
25 #sudo -u postgres psql -c "SELECT version();"
26 sudo systemctl enable postgresql.service
27 sudo systemctl start postgresql.service
28 sudo echo "postgres:admin123" | chpasswd
29 runuser -l postgres -c "createuser sonar"
30 sudo -i -u postgres psql -c "ALTER USER sonar WITH ENCRYPTED PASSWORD 'admin123';"
31 sudo -i -u postgres psql -c "CREATE DATABASE sonarqube OWNER sonar;"
```

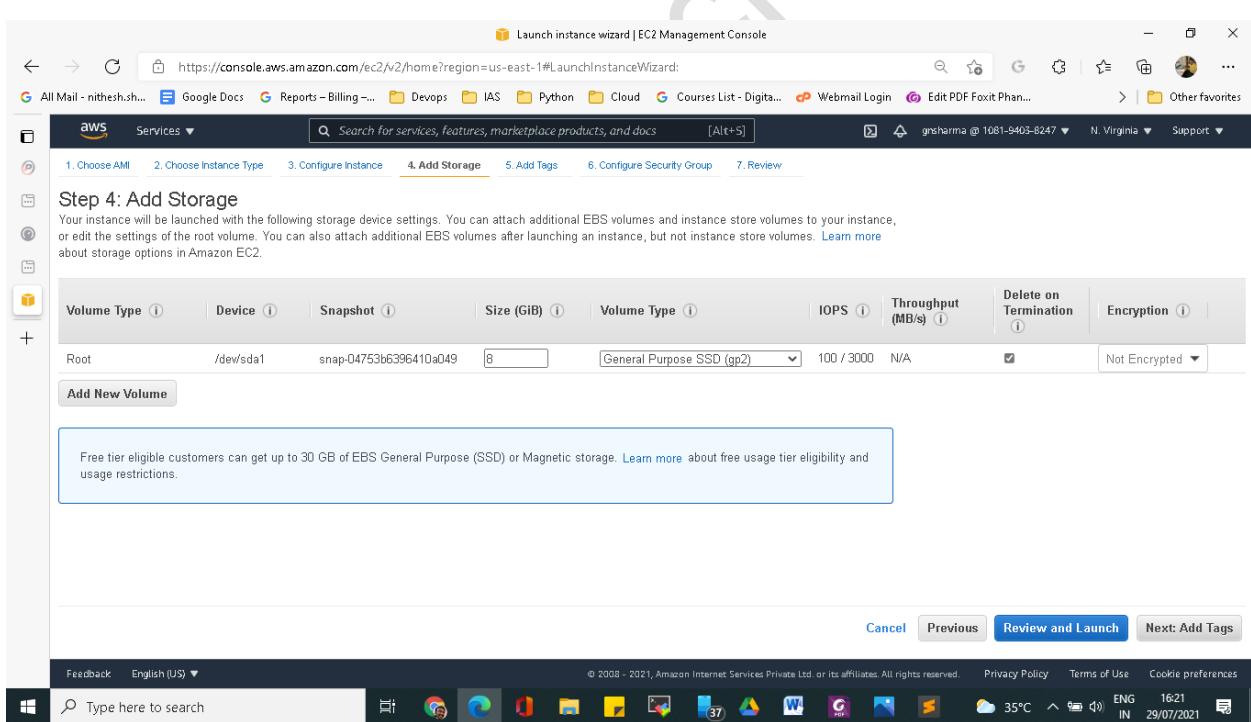
A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

21



The screenshot shows the AWS EC2 Management Console Launch instance wizard at Step 3: Configure Instance Details. The User data field contains the following shell script:

```
#!/bin/bash
echo "System reboot in 30 sec"
sleep 30
reboot
```



The screenshot shows the AWS EC2 Management Console Launch instance wizard at Step 4: Add Storage. A new volume is being added with the following settings:

Volume Type	Device	Snapshot	Size (GiB)	Volume Type	IOPS	Throughput (MB/s)	Delete on Termination	Encryption
Root	/dev/sda1	snap-i04753b6396410a049	8	(General Purpose SSD (gp2))	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

22

The screenshot shows the AWS EC2 Management Console Launch Instance Wizard at Step 6: Configure Security Group. The user has selected an existing security group named 'sg_0179c5b57311a268c'. The list of security groups includes:

Security Group ID	Name	Description
sg-00a6141987b58e35d	CentOS 7 - x86_64 - with Updates HVM-2002_01-AutogenByAWSMP	This security group was generated by AWS Marketplace and is based on recommended settings for CentOS 7 (x86_64) - with Updates HVM-2002_01-AutogenByAWSMP
sg-af1216a5	default	default VPC security group
sg-0a35e080df6045b1b	launch-wizard-1	launch-wizard-1 created 2021-06-02T22:04:00Z+05:30
sg-0fb4f6218eadc462e	sg_jenkins_logiclabs	sg_jenkins_logiclabs
sg-09c9954f450660a722	sg_nexus_logiclabs	sg_nexus_logiclabs
sg-0179c5b57311a268c	sg_sonar_logiclabs	sg_sonar_logiclabs

Inbound rules for sg-0179c5b57311a268c (Selected security groups: sg-0179c5b57311a268c)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
All traffic	All	All	sg-0fb4f6218eadc462e (sg_jenkins_logiclab)	
SSH	TCP	22	0.0.0.0/0	

Buttons: Cancel, Previous, Review and Launch.

The screenshot shows the AWS EC2 Management Console Launch Instance Wizard at Step 6: Configure Security Group. The user has selected an existing security group named 'sg_0179c5b57311a268c'. The list of security groups is identical to the previous screenshot.

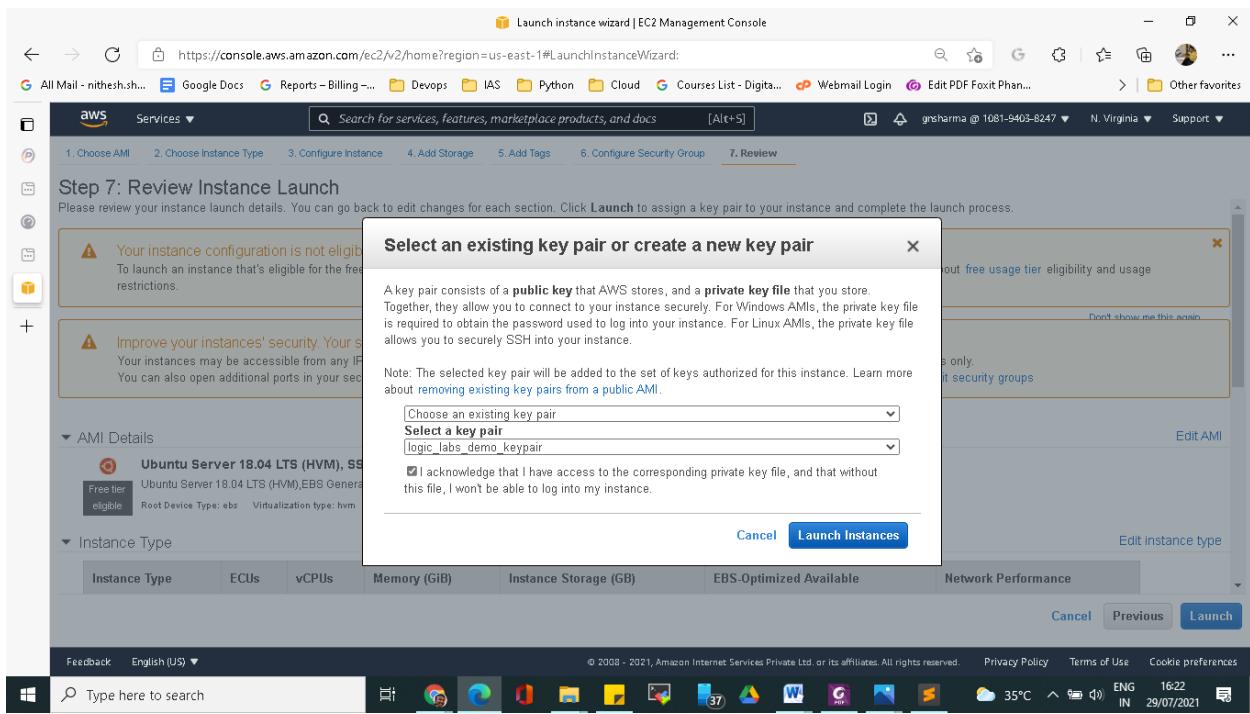
Inbound rules for sg-0179c5b57311a268c (Selected security groups: sg-0179c5b57311a268c)

Type	Protocol	Port Range	Source	Description
HTTP	TCP	80	0.0.0.0/0	
All traffic	All	All	sg-0fb4f6218eadc462e (sg_jenkins_logiclab)	
SSH	TCP	22	0.0.0.0/0	

Buttons: Cancel, Previous, Review and Launch.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

23



Verifying the installation:

1. Jenkins setup verification:

- Login using public ip of jenkins server followed by port number
Eg: 55.52.32.1:8080
- When the initial screen asks for Initial admin password, look into the server via ssh and copy the initial admin password

2. Sonar Setup verification

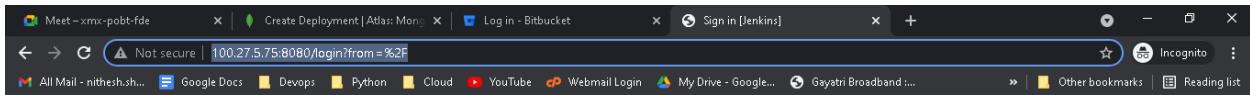
- Login using public ip of sonar server
-

3. Nexus setup Verification:

- Login using public ip of Nexus server followed by port number
Eg: 55.52.32.1:8081

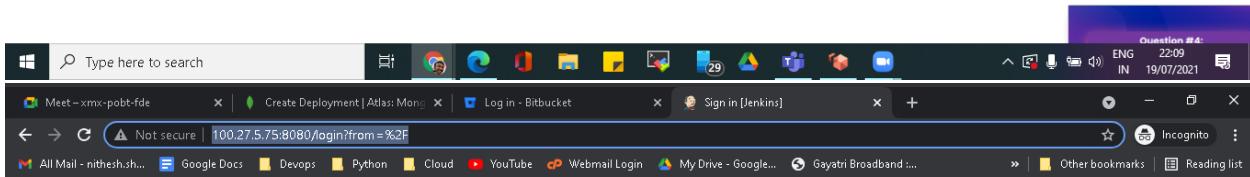
A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

24



Please wait while Jenkins is getting ready to work ...

Your browser will reload automatically when Jenkins is ready.



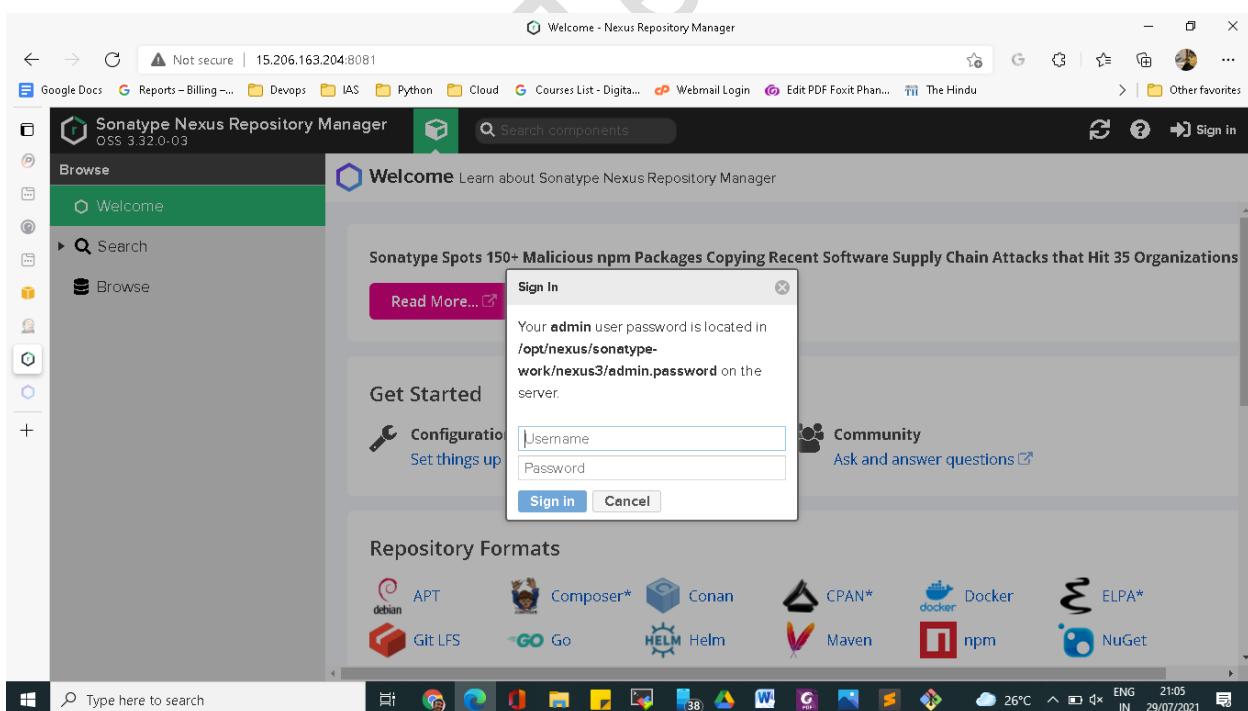
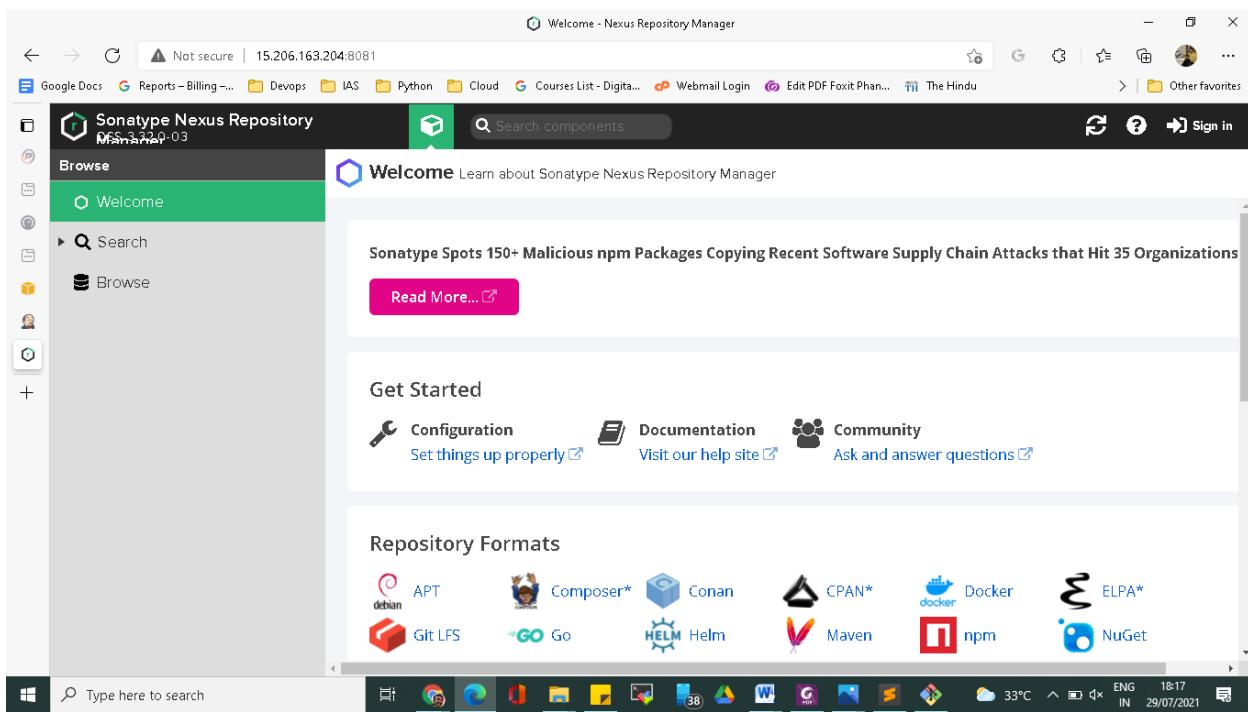
Welcome to Jenkins!

Sign in

Keep me signed in



Validating Nexus configuration:



SSH into nexus server to get admin password

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

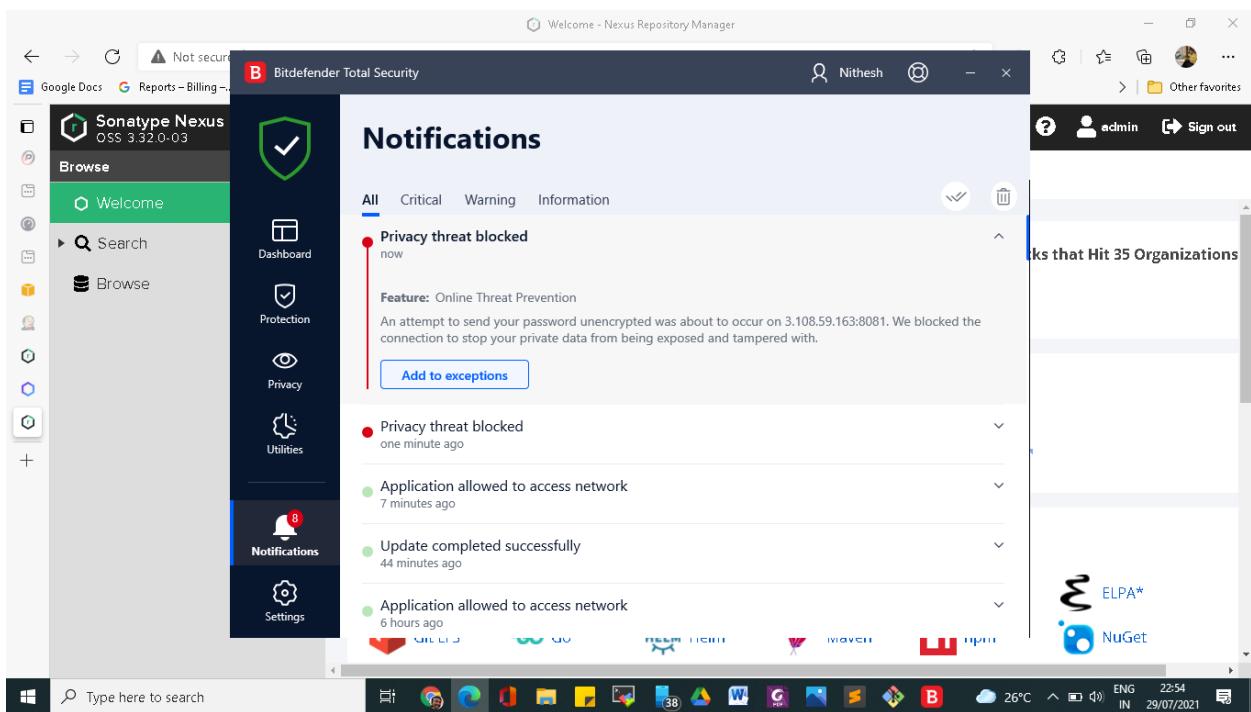
26

The screenshot shows a Windows desktop environment. At the top, there is a terminal window titled 'centos@ip-172-31-0-186:~\$' displaying a command-line session. Below the terminal is a web browser window titled 'Welcome - Nexus Repository Manager'. The browser shows the Sonatype Nexus Repository Manager interface, version OSS 3.32.0-03. A 'Sign In' dialog box is open in the center of the page, prompting for an 'admin' password. The desktop taskbar at the bottom shows various pinned icons and the system tray.

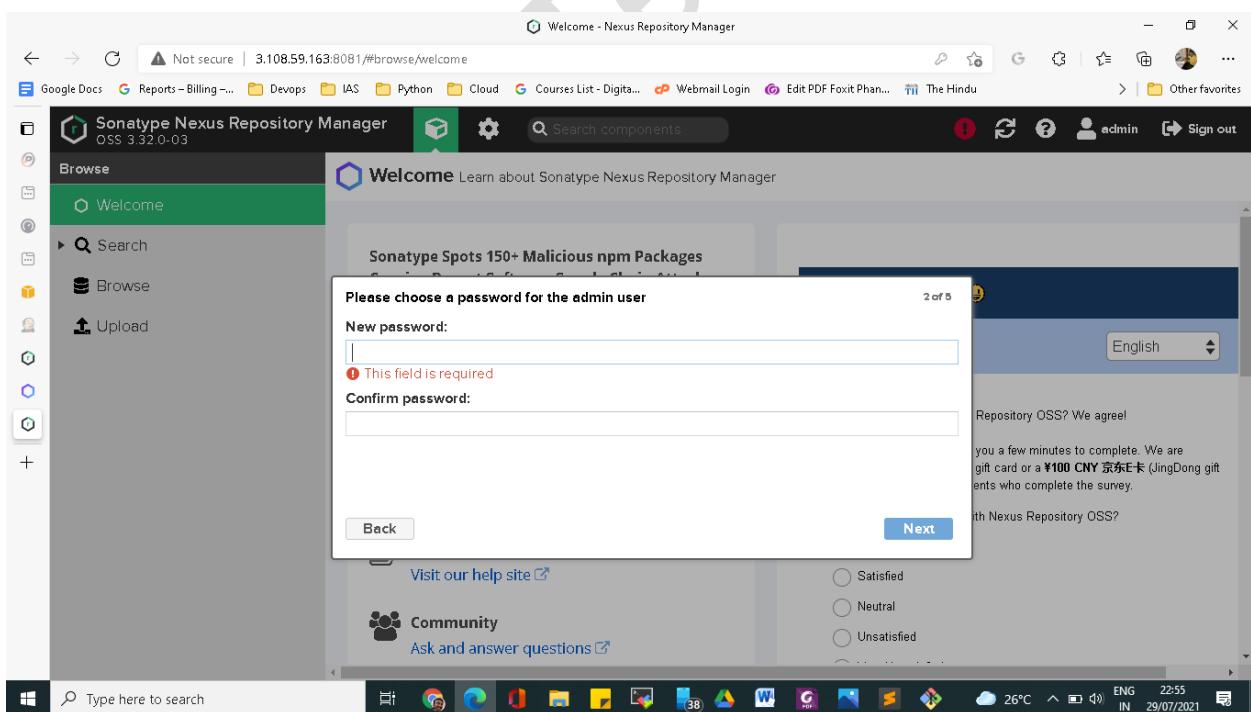
If your antivirus throws an error, please make necessary changes accordingly...

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

27



There will be option to change the default password.



Select the below given settings for anonymous access..

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

28

The screenshot shows the Sonatype Nexus Repository Manager interface. A modal window titled "Configure Anonymous Access" is open, asking if users can search, browse, and download components without credentials. It includes a note about security implications and two radio button options: "Enable anonymous access" (selected) and "Disable anonymous access". Below the modal, there's a "Community" section with a "Ask and answer questions" link. The background shows a banner about malicious npm packages and a survey for Nexus Repository OSS.

This screenshot shows the same Nexus Repository Manager interface after the survey has been completed. The survey results are displayed on the right side, showing satisfaction levels: Very Satisfied (radio button selected), Satisfied, Neutral, and Unsatisfied. A "Help Us Improve!" section with a smiley face emoji is also present. The background banners remain the same as in the previous screenshot.

Go to browser option and create new repositories:

1. Maven release
2. Maven central

3. Maven group

Repositories → create new repository → Maven2 (hosted)

Sonatype Nexus Repository Manager OSS 3.32.0-03

Administration

- Repository
 - Repositories** (selected)
 - Blob Stores
 - Cleanup Policies
 - Content Selectors
 - Proprietary Repositories
 - Routing Rules
- Security
 - Privileges
 - Roles
 - Users
 - Anonymous Access
- LDAP

Repositories / Select Recipe

Recipe ↑
apt (hosted)
apt (proxy)
bower (group)
bower (hosted)
bower (proxy)
cocoapods (proxy)
conan (proxy)
conda (proxy)
docker (group)
docker (hosted)
docker (proxy)
gitlfs (hosted)
go (group)
go (proxy)
helm (hosted)
helm (proxy)
maven2 (group)
maven2 (hosted)
maven2 (proxy)
npm (group)

Sonatype Nexus Repository Manager OSS 3.32.0-03

Administration

- Repository
 - Repositories** (selected)
 - Blob Stores
 - Cleanup Policies
 - Content Selectors
 - Proprietary Repositories
 - Routing Rules
- Security
 - Privileges
 - Roles
 - Users
 - Anonymous Access
- LDAP

Repositories Manage repositories

Create repository

Name ↑	Type	Format	Status	URL	Health check	IQ Policy Vi...
maven-central	proxy	maven2	Online - Ready to Co...	[copy]	Analyze	[edit]
maven-public	group	maven2	Online	[copy]	[edit]	[edit]
maven-releases	hosted	maven2	Online	[copy]	[edit]	[edit]
maven-snapshots	hosted	maven2	Online	[copy]	[edit]	[edit]
nuget-group	group	nuget	Online	[copy]	[edit]	[edit]
nuget-hosted	hosted	nuget	Online	[copy]	[edit]	[edit]
nuget.org-proxy	proxy	nuget	Online - Ready to Co...	[copy]	Analyze	[edit]

Fill in the details as given in the screenshot below.

Start with the release repository, then proceed to the central repo and finally create a group repository.

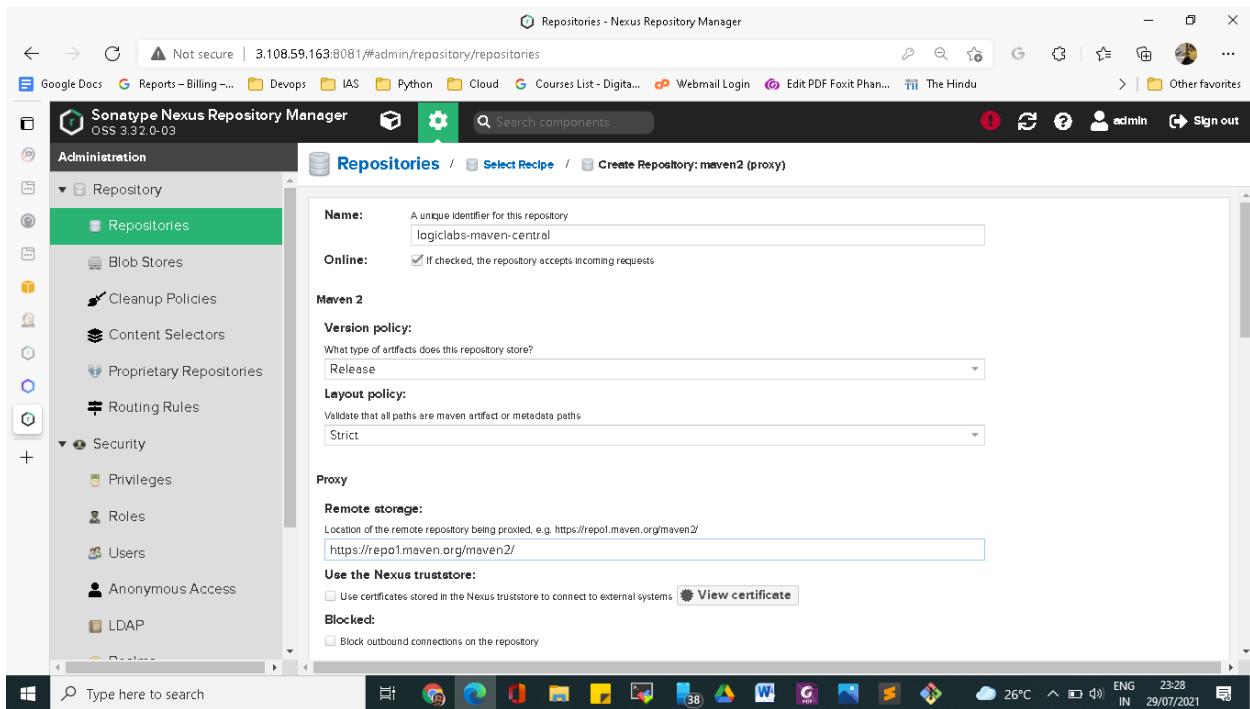
The screenshot shows the 'Repositories' section of the Nexus Repository Manager. A new repository is being created with the following details:

- Name:** logiclabs-maven-release
- Online:** Checked
- Maven 2**
 - Version policy:** Release
 - Layout policy:** Strict
- Storage**
 - Blob store:** default
 - Strict Content Type Validation:** Checked
- Hosted**

The screenshot shows the continuation of the repository creation process. The configuration includes:

- Controls if deployments of and updates to artifacts are allowed:** Disable redeploy
- Proprietary Components:** Components in this repository count as proprietary for namespace conflict attacks (requires Sonatype Nexus Firewall)
- Cleanup**
 - Cleanup Policies:** Components that mismatch of the Applied policies will be deleted
 - Available:** Filter
 - Applied:** Empty

At the bottom, there are 'Create repository' and 'Cancel' buttons.



The above step is to save dependencies from maven repo to nexus repo.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

32

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is titled "Administration" and includes sections for Repository (Repositories, Blob Stores, Cleanup Policies, Content Selectors, Proprietary Repositories, Routing Rules), Security (Privileges, Roles, Users, Anonymous Access, LDAP), and other management tools. The main content area is titled "Repositories" and shows a table of existing repositories. A green banner at the top right indicates "Repository updated: logilabs-maven-central". The table columns are Name, Type, Format, Status, URL, Health check, and IQ Policy Vi... (with a "Filter" button). The repositories listed are:

Name	Type	Format	Status	URL	Health check	IQ Policy Vi...
logilabs-maven-central	proxy	maven2	Online - Ready to Connect	[copy]	Analyze	
logilabs-release	hosted	maven2	Online	[copy]		
maven-central	proxy	maven2	Online - Ready to Connect	[copy]	Analyze	
maven-public	group	maven2	Online	[copy]		
maven-releases	hosted	maven2	Online	[copy]		
maven-snapshots	hosted	maven2	Online	[copy]		
nuget-group	group	nuget	Online	[copy]		
nuget-hosted	hosted	nuget	Online	[copy]		
nuget.org-proxy	proxy	nuget	Online - Ready to Connect	[copy]	Analyze	

Create a group repository

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is identical to the previous screenshot. The main content area is titled "Repositories / Select Recipe". A sidebar on the right lists various "Recipe" options, with "maven2 (group)" currently selected and highlighted in grey. Other recipes listed include bower (group), bower (hosted), bower (proxy), cocoapods (proxy), conan (proxy), conda (proxy), docker (group), docker (hosted), docker (proxy), gitlfs (hosted), go (group), go (proxy), helm (hosted), helm (proxy), maven2 (hosted), maven2 (proxy), npm (group), npm (hosted), and npm (proxy).

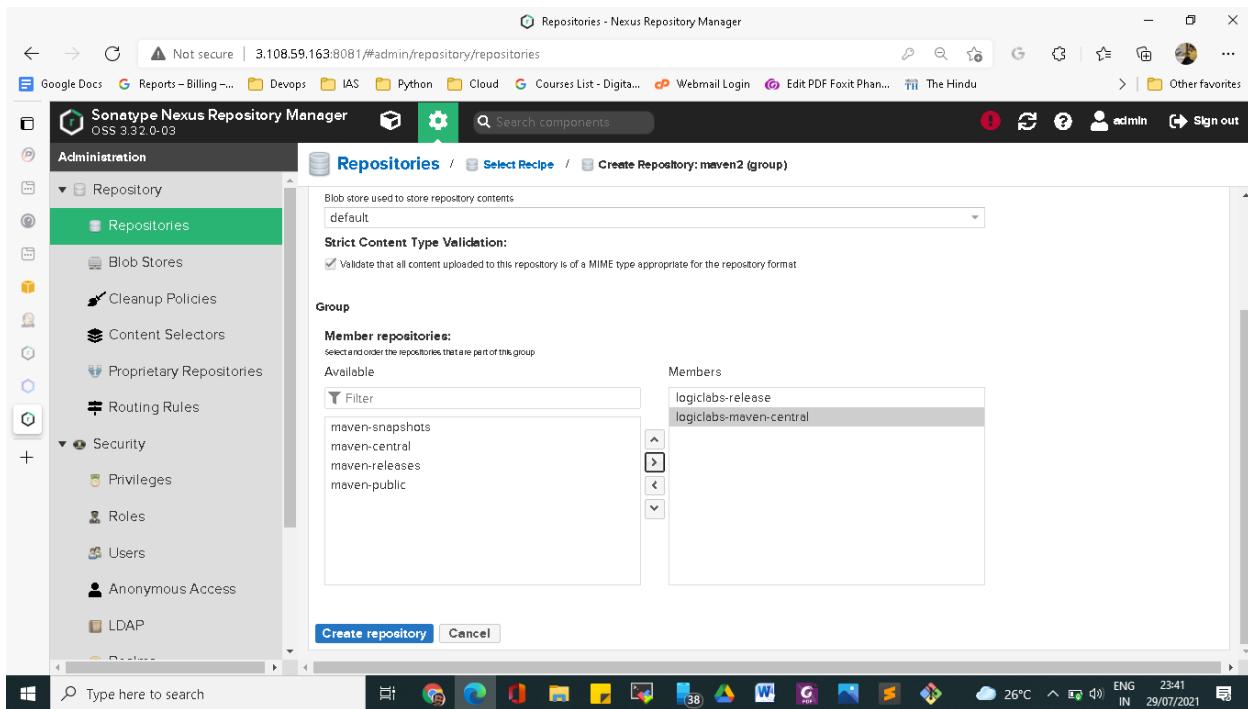
A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

33

The screenshot shows the Sonatype Nexus Repository Manager interface. The left sidebar is titled 'Administration' and includes sections for Repository, Blob Stores, Cleanup Policies, Content Selectors, Proprietary Repositories, Routing Rules, Security, Privileges, Roles, Users, Anonymous Access, and LDAP. The 'Repositories' section is currently selected. The main panel is titled 'Repositories / Select Recipe / Create Repository: maven2 (group)'. It has fields for 'Name' (set to 'logiclabs-maven-group') and 'Online' (checked). Under 'Storage', 'Blob store' is set to 'default'. Under 'Strict Content Type Validation', there is a checked checkbox. In the 'Group' section, 'Member repositories' is expanded, showing a list of available repositories: 'maven-snapshots', 'maven-central', 'maven-releases', 'maven-public', and 'logiclabs-release'. The 'logiclabs-release' repository is selected and highlighted in blue. To its right, under 'Members', there is a red rectangular box highlighting the area where member repositories are listed.

This screenshot shows the same interface as the previous one, but with a different configuration. The 'logiclabs-release' repository is no longer selected in the 'Available' list; instead, 'logiclabs-maven-central' is selected and highlighted in blue. The 'Members' list on the right contains the single entry 'logiclabs-maven-central'. The 'Create repository' button at the bottom is visible.

Add the relevant repositories already created to the group repository as given below.



If needed, a snapshot repository can be created from maven2(host) type and change the policy from release to snapshot...however for this demo, we aren't using a snapshot repo.

Inorder for maven to pick up dependencies from nexus repo instead of public repo, we provide necessary details in settings.xml file in our users directory.

Note: the text editor used in this demonstration is sublime text and any other alternative can be used.

```

<?xml version="1.0" encoding="UTF-8"?>
<settings xmlns="http://maven.apache.org/SETTINGS/1.1.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.1.0 http://maven.apache.org/xsd/settings-1.1.0.xsd">

  <servers>
    <server>
      <id>${SNAP-REPO}</id>
      <username>${NEXUS-USER}</username>
      <password>${NEXUS-PASS}</password>
    </server>
    <server>
      <id>${RELEASE-REPO}</id>
      <username>${NEXUS-USER}</username>
      <password>${NEXUS-PASS}</password>
    </server>
    <server>
      <id>${CENTRAL-REPO}</id>
      <username>${NEXUS-USER}</username>
      <password>${NEXUS-PASS}</password>
    </server>
    <server>
      <id>${NEXUS-GRP-REPO}</id>
      <username>${NEXUS-USER}</username>
      <password>${NEXUS-PASS}</password>
    </server>
  </servers>
  <mirrors>
    <mirror>
      <id>${CENTRAL-REPO}</id>
      <name>Central Repository</name>
      <url>${CENTRAL-REPO}</url>
      <mirrorOf>*</mirrorOf>
    </mirror>
  </mirrors>

```

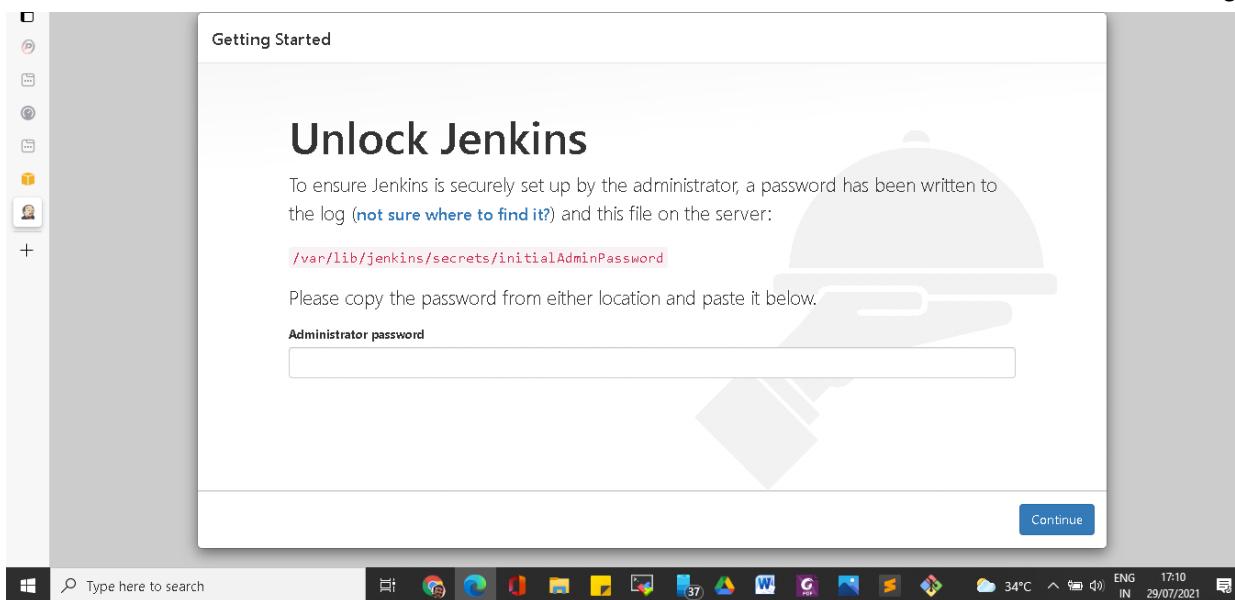
We will mention in jenkins to use this settings.xml file instead of the default one in the .m2 folder of source code.

In the settings.xml and pom.xml files, we mention the variables where the repository values and server URLs will be passed.

POST INSTALLATION CONFIGURATION

JENKINS:

1. Initial admin password can be obtained by logging in to the jenkins ubuntu server and going to the path:
/var/lib/jenkins/secrets/initialadminpassword
2. Create a new admin password when prompted
3. Select install suggested plugins
- 4.
- 5.



Either use putty or use git bash to ssh into the instance

```
ubuntu@ip-172-31-0-186:~  
home@Omsai-Lenovo MINGW64 ~  
$ ssh -i Downloads/ci-jenkins_mumbai.pem ubuntu@13.127.55.225  
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1045-aws x86_64)  
  
 * Documentation: https://help.ubuntu.com  
 * Management: https://landscape.canonical.com  
 * Support: https://ubuntu.com/advantage  
  
 System information as of Thu Jul 29 11:42:48 UTC 2021  
  
 System load: 0.08 Processes: 96  
 Usage of /: 24.7% of 7.69GB Users logged in: 0  
 Memory usage: 26% IP address for eth0: 172.31.0.186  
 Swap usage: 0%  
  
 68 packages can be updated.  
 45 of these updates are security updates.  
 To see these additional updates run: apt list --upgradable  
  
 New release '20.04.2 LTS' available.  
 Run 'do-release-upgrade' to upgrade to it.  
  
 Last login: Thu Jul 29 11:38:40 2021 from 103.160.236.10  
ubuntu@ip-172-31-0-186:~$ |
```

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

37

```
root@ip-172-31-0-186:~$ sudo su
ubuntu@ip-172-31-0-186:/home/ubuntu# cd /var/lib/jenkins/
root@ip-172-31-0-186:/var/lib/jenkins# ls
config.xml          jenkins.telemetry.correlator.xml  nodeMonitors.xml  secret.key      updates
hudson.model.UpdateCenter.xml  jobs                  nodes           secret.key.not-so-secret userContent
identity.key.enc    logs                  plugins        secrets
root@ip-172-31-0-186:/var/lib/jenkins# cd secrets/
root@ip-172-31-0-186:/var/lib/jenkins/secrets# ls
filepath-filters.d          org.jenkinsci.main.modules.instance_identity.InstanceIdentity.KEY
initialAdminPassword        slave-to-master-security-kill-switch
jenkins.model.Jenkins.crumbSalt  whitelisted-callables.d
master.key
root@ip-172-31-0-186:/var/lib/jenkins/secrets# cat initialAdminPassword
1497ee66e20547a29c089338110da040
root@ip-172-31-0-186:/var/lib/jenkins/secrets# |
```

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

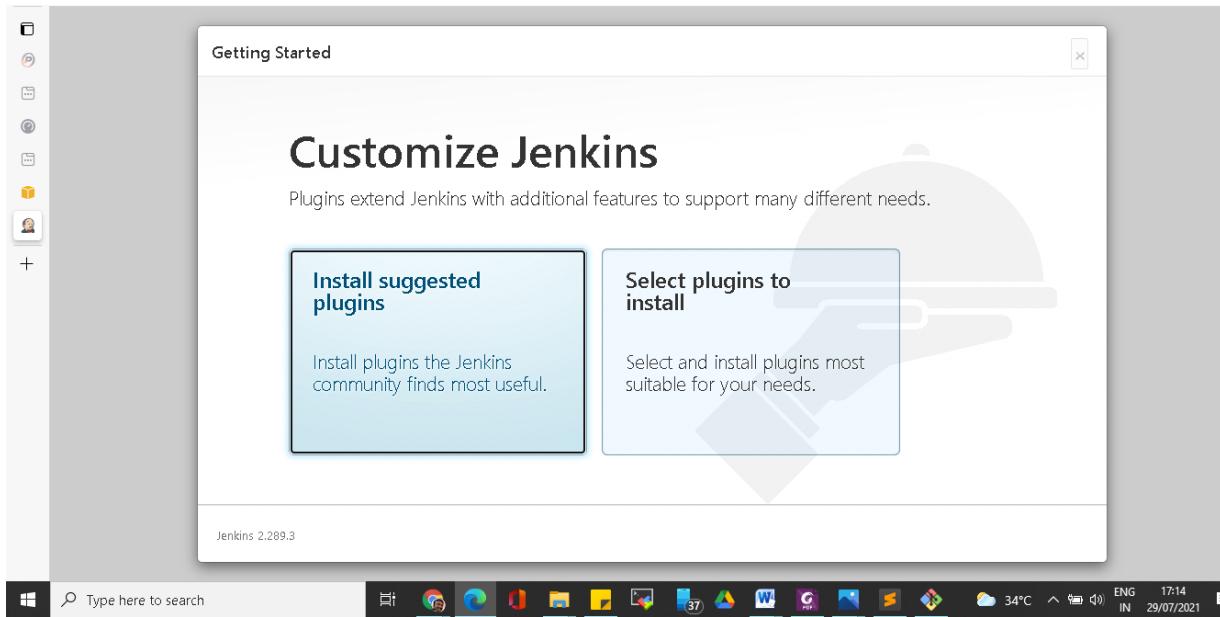
`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

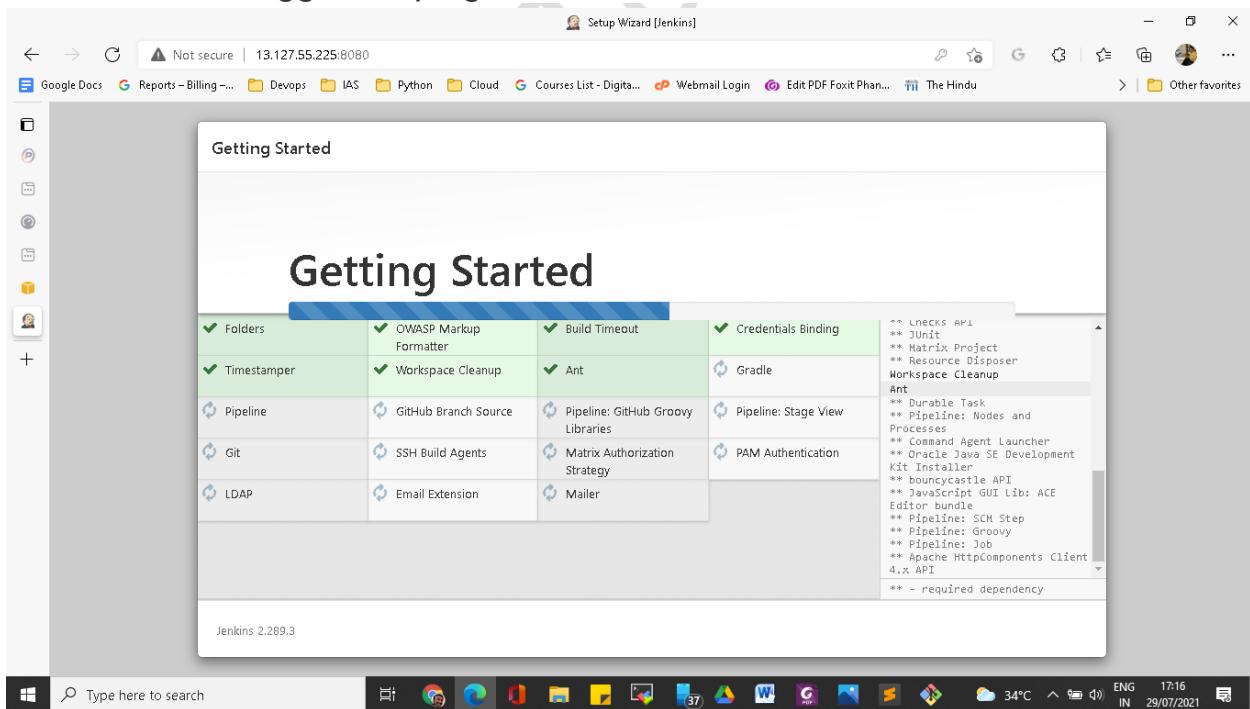
Administrator password

.....

Continue



Wait till all the suggested plugins are installed



Getting Started

Create First Admin User

Username:

Password:

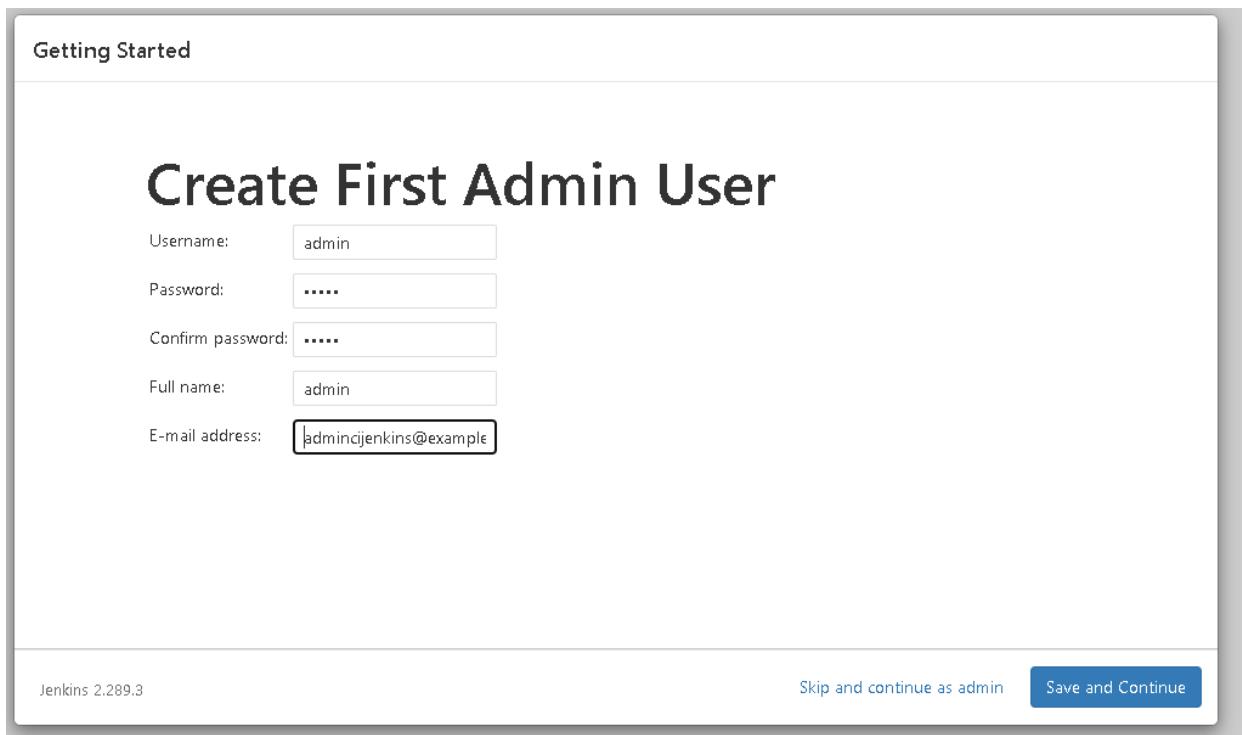
Confirm password:

Full name:

E-mail address:

Jenkins 2.289.3

[Skip and continue as admin](#) [Save and Continue](#)



Give your desired values for username and password, select continue to configure further steps.

Getting Started

Create First Admin User

Username:

Password:

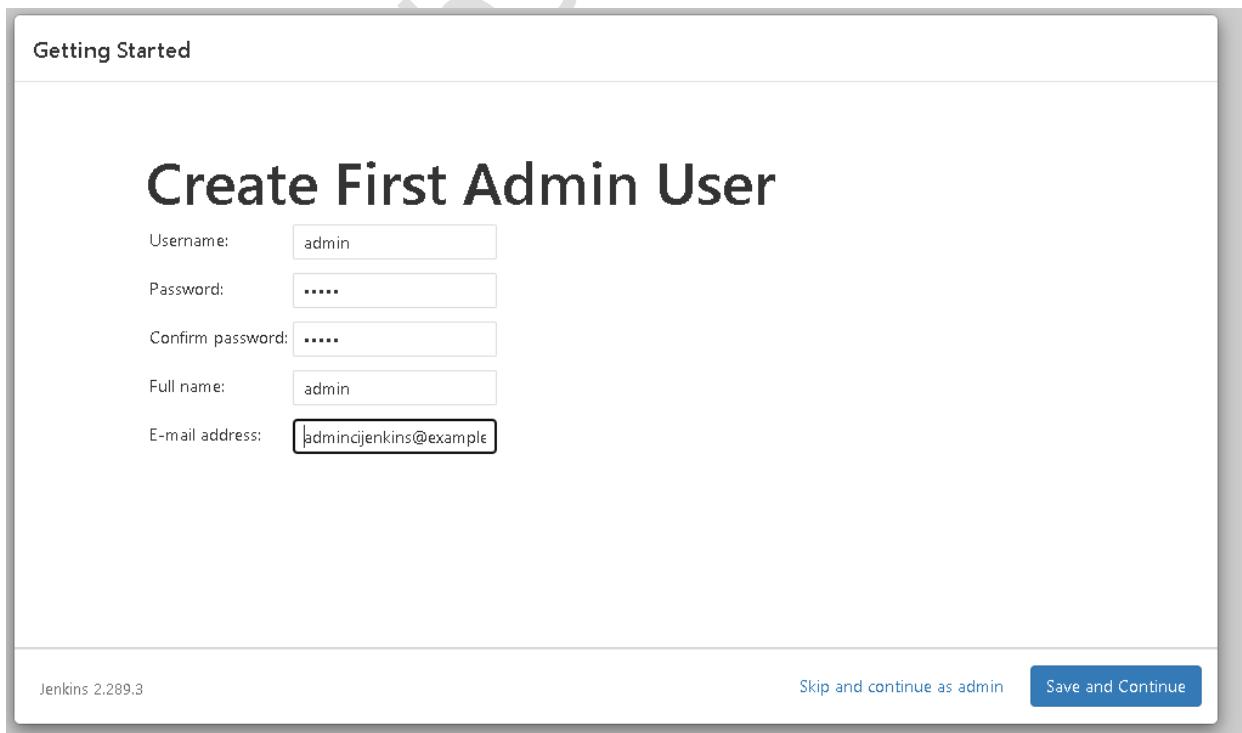
Confirm password:

Full name:

E-mail address:

Jenkins 2.289.3

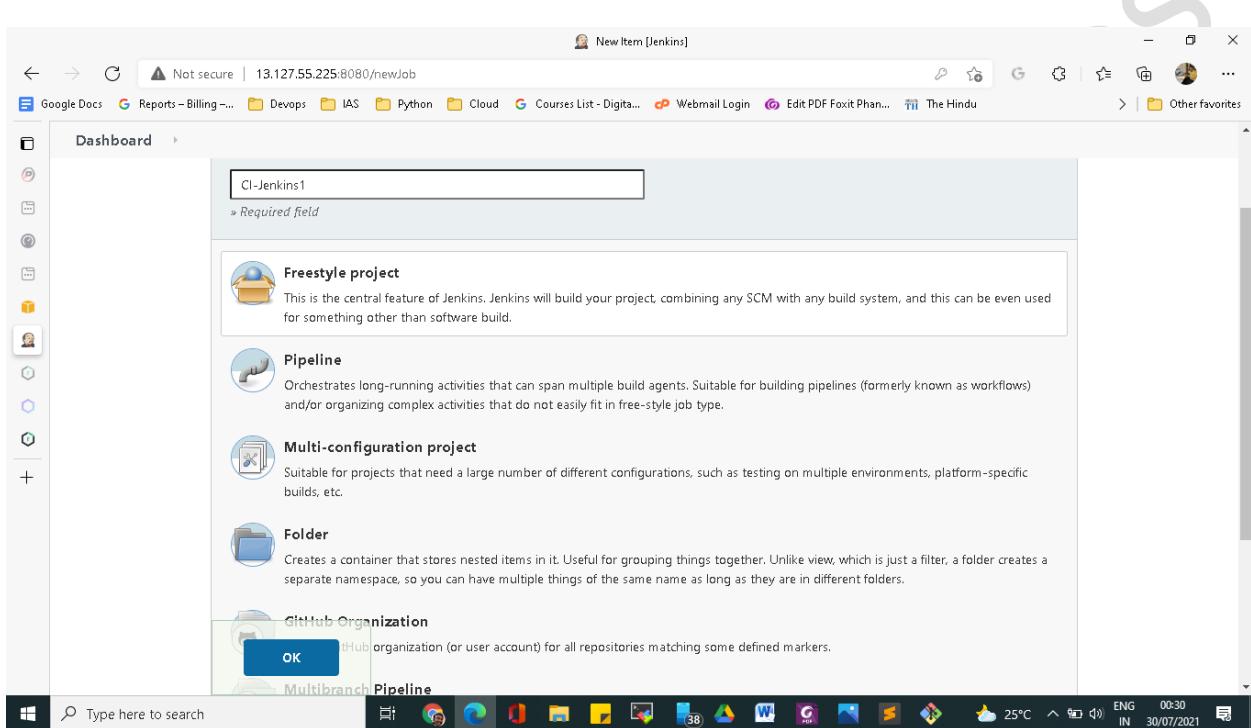
[Skip and continue as admin](#) [Save and Continue](#)



Creating a JENKINS Job:

Give a proper name (eg: CI-Jenkins1) and select freestyle project and click on ok to create a new jenkins job.

1. Provide settings for the job as given in the screenshots below.



A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

41

CI-jenkins-Build1 Config [Jenkins]

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Repository URL
https://github.com/gnsharma530/Logilabs-cidemo.git

Credentials - none - Add

Name

Refspec

Add Repository

Branches to build

Branch Specifier (blank for 'any')
*/ci-jenkins

Save Apply Add Branch

CI-jenkins-Build1 Config [Jenkins]

General Source Code Management Build Triggers Build Environment Build Post-build Actions

GitHub hook trigger for GITScm polling

Poll SCM

Build Environment

Delete workspace before build starts

Use secret text(s) or file(s)

Abort the build if it's stuck

Add timestamps to the Console Output

Inspect build log for published Gradle build scans

With Ant

Build

Add build step ▾

- Execute Windows batch command
- Execute shell
- Invoke Ant
- Invoke Gradle script
- Invoke top-level Maven targets
- Run with timeout
- Set build status to "pending" on GitHub commit

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

42

The screenshot shows the Jenkins configuration page for the job 'CI-jenkins-Build1'. The 'Build Triggers' section is active, showing two options: 'GitHub hook trigger for GITScm polling' and 'Poll SCM'. The 'Build' section contains a step titled 'Invoke top-level Maven targets' with the goal 'Install -DskipTests' entered in the Goals field. The 'Post-build Actions' section is visible at the bottom.

We pass the goals as “*install - DskipTests*” to skip unit test as of now and only package our source code

SET VARIABLES:

Create a properties file and fill in the below given details. Change the below properties as per your server configurations

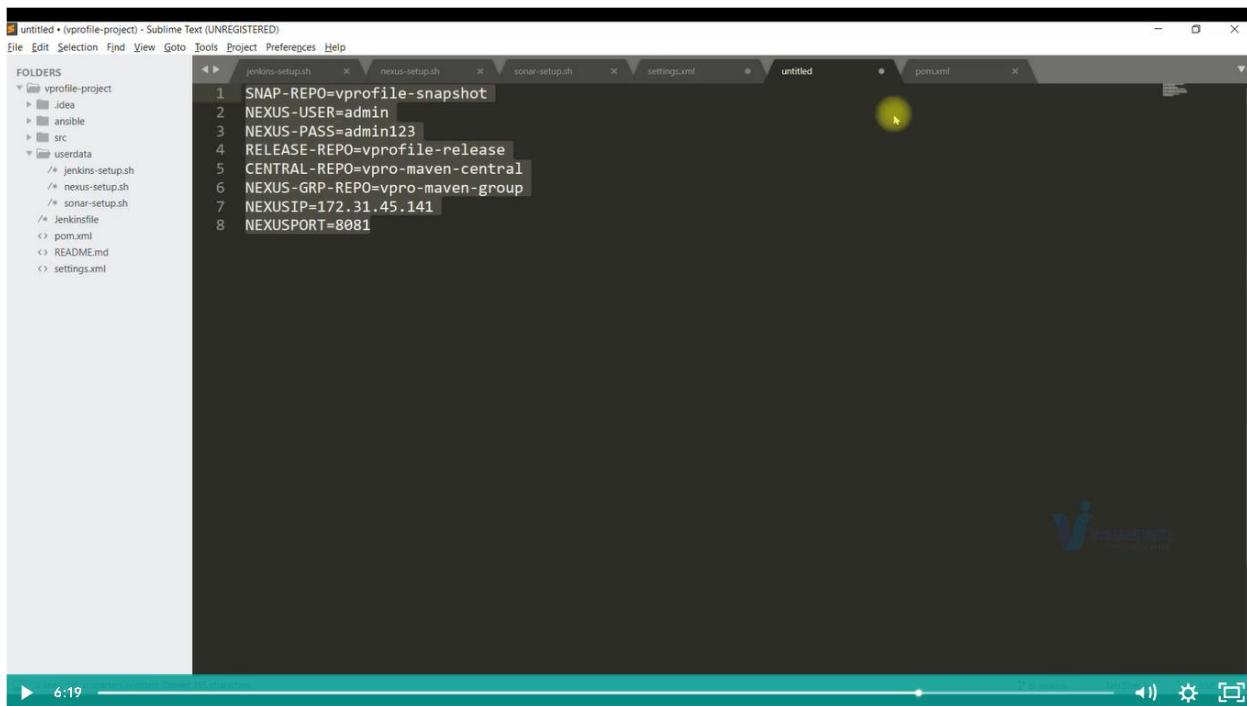
We obtain the necessary details from nexus repository configurations.

Paste the same values as you have given.

The values given in below screenshots may be different from what you might have given.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

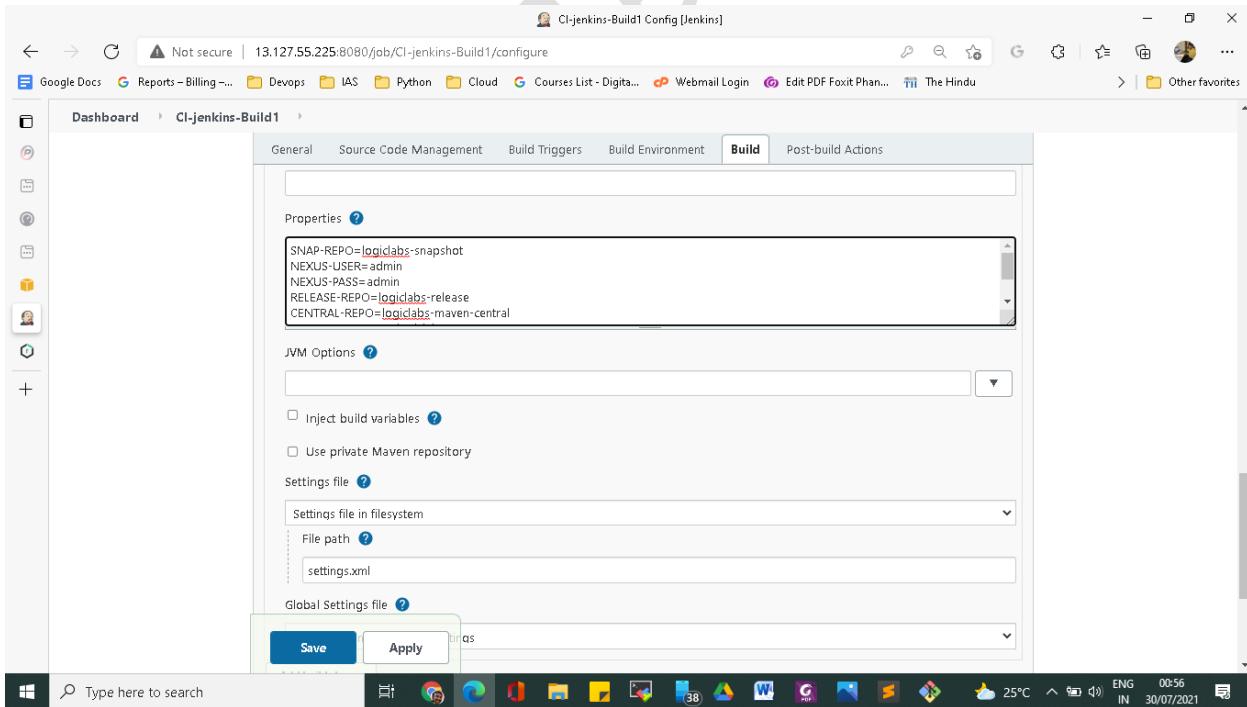
43



The screenshot shows a Sublime Text window with multiple tabs open. The tabs include 'jenkins-setup.sh', 'nexus-setup.sh', 'sonar-setup.sh', 'settings.xml', and 'pom.xml'. The left sidebar shows a project structure for 'vprofile-project' containing 'idea', 'ansible', 'src', and 'userdata' folders, along with 'jenkins-setup.sh', 'nexus-setup.sh', 'sonar-setup.sh', 'Jenkinsfile', 'pom.xml', 'README.md', and 'settings.xml' files.

```
1 SNAP-REPO=vprofile-snapshot
2 NEXUS-USER=admin
3 NEXUS-PASS=admin123
4 RELEASE-REPO=vprofile-release
5 CENTRAL-REPO=vpro-maven-central
6 NEXUS-GRP-REPO=vpro-maven-group
7 NEXUSIP=172.31.45.141
8 NEXUSPORT=8081
```

Copy these properties into the jenkins properties option;



The screenshot shows the Jenkins job configuration page for 'CI-jenkins-Build1'. The 'Build' tab is selected. In the 'Properties' section, there is a text input field containing the following environment variables:

```
SNAP-REPO=logilabs-snapshot
NEXUS-USER=admin
NEXUS-PASS=admin
RELEASE-REPO=logilabs-release
CENTRAL-REPO=logilabs-maven-central
```

Below the properties, there are sections for 'JVM Options' (checkboxes for 'Inject build variables' and 'Use private Maven repository'), 'Settings file' (dropdown for 'File path' set to 'settings.xml'), and 'Global Settings file' (dropdown for 'File path' set to 'settings.xml'). At the bottom of the configuration page are 'Save' and 'Apply' buttons.

Also provide 'File path' to settings.xml file as settings file in filesystem..

For that, go to settings file → file path → type “settings.xml”

Save the file.. And click on build now option to test our first build job.

The screenshot shows the Jenkins interface for the project "CI-jenkins-Build1". On the left sidebar, the "Build Now" button is selected and highlighted in blue. The main content area displays the project details: "Project CI-jenkins-Build1" and "CI-Jenkins project, first Build". It includes sections for "Workspace" and "Recent Changes". Below these are "Permalinks" and a "Build History" table with one entry: "#1 Jul 29, 2021 7:34 PM". At the bottom of the page, there are links for "Atom feed for all" and "Atom feed for failures". The status bar at the bottom right shows the date and time as 30/07/2021 01:04.

The screenshot shows the Jenkins console output for build #1. The log output is as follows:

```

Progress (2): 12 kB | 180/230 kB
Progress (2): 12 kB | 184/230 kB
Progress (2): 12 kB | 188/230 kB
Progress (2): 12 kB | 192/230 kB
Progress (2): 12 kB | 196/230 kB
Progress (2): 12 kB | 200/230 kB
Progress (2): 12 kB | 204/230 kB
Progress (2): 12 kB | 208/230 kB
Progress (2): 12 kB | 212/230 kB
Progress (2): 12 kB | 217/230 kB
Progress (2): 12 kB | 221/230 kB
Progress (2): 12 kB | 225/230 kB
Progress (2): 12 kB | 229/230 kB
Progress (2): 12 kB | 230 kB

Downloaded From logilabs-maven-central: http://172.31.0.227:8081/repository/logilabs-maven-group/org/codehaus/plexus/plexus-digest/1.0/plexus-digest-1.0.jar
(12 kB at 45 kB/s)
Downloaded From logilabs-maven-central: http://172.31.0.227:8081/repository/logilabs-maven-group/org/codehaus/plexus/plexus-utils/3.0.5/plexus-utils-3.0.5.jar
(230 kB at 682 kB/s)
[0@1:34mINFO@[]] Installing /var/lib/jenkins/workspace/CI-jenkins-Build1/target/vprofile-v2.war to
/var/lib/jenkins/.m2/repository/com/visualpathit/vprofile/v2/vprofile-v2.war
[0@1:34mINFO@[]] Installing /var/lib/jenkins/workspace/CI-jenkins-Build1/pom.xml to /var/lib/jenkins/.m2/repository/com/visualpathit/vprofile/v2/vprofile-e-2.pom
[0@1:34mINFO@[]] [1m.....[m
[0@1:34mINFO@[]] [1mBUILD SUCCESS[m
[0@1:34mINFO@[]] [1m.....[m
[0@1:34mINFO@[]] Total time: 01:57 min
[0@1:34mINFO@[]] Finished at: 2021-07-29T19:36:47Z
[0@1:34mINFO@[]] [1m.....[m
Finished: SUCCESS

```

The status bar at the bottom right shows the date and time as 30/07/2021 01:06.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

45

The screenshot shows a browser window with the title "CI-jenkins-Build1 #1 Console [Jenkins]". The URL is "13.127.55.225:8080/job/CI-jenkins-Build1/1/console". The page displays the Jenkins interface with a sidebar on the left and a main content area. The main content area is titled "Console Output" and shows the following text:

```
Skipping 392 KB. Full Log  
Downloading from logiclabs-maven-central: http://172.31.0.227:8081/repository/logiclabs-maven-group/org/apache/maven/doxia/doxia-module-fml/1.1.2/doxia-module-fml-1.1.2.jar  
Downloaded from logiclabs-maven-central: http://172.31.0.227:8081/repository/logiclabs-maven-group/org/apache/maven/doxia/doxia-site-renderer/1.1.2/doxia-site-renderer-1.1.2.jar (50 kB at 25 kB/s)  
Downloading from logiclabs-maven-central: http://172.31.0.227:8081/repository/logiclabs-maven-group/org/codehaus/plexus/plexus-i18n/1.0-beta-7/plexus-i18n-1.0-beta-7.jar  
Progress (2): 1.2 MB | 4.1/52 kB  
Progress (2): 1.2 MB | 7.8/52 kB  
Progress (2): 1.2 MB | 12/52 kB  
Progress (2): 1.2 MB | 16/52 kB  
Progress (2): 1.2 MB | 20/52 kB  
Progress (2): 1.2 MB | 24/52 kB  
Progress (2): 1.2 MB | 28/52 kB  
Progress (2): 1.2 MB | 32/52 kB  
Progress (2): 1.2 MB | 36/52 kB  
Progress (2): 1.2 MB | 41/52 kB  
Progress (2): 1.2 MB | 45/52 kB  
Progress (2): 1.2 MB | 49/52 kB  
Progress (2): 1.2 MB | 52 kB  
Progress (3): 1.2 MB | 52 kB | 4.1/15 kB  
Progress (3): 1.2 MB | 52 kB | 7.8/15 kB  
Progress (3): 1.2 MB | 52 kB | 12/15 kB  
Progress (3): 1.2 MB | 52 kB | 15 kB  
Downloaded from logiclabs-maven-central: http://172.31.0.227:8081/repository/logiclabs-maven-group/xerces/xercesImpl/2.8.1/xercesImpl-2.8.1.jar (1.2 MB at 577 kB/s)
```

After the build is successful, proceed to the next steps as given below.

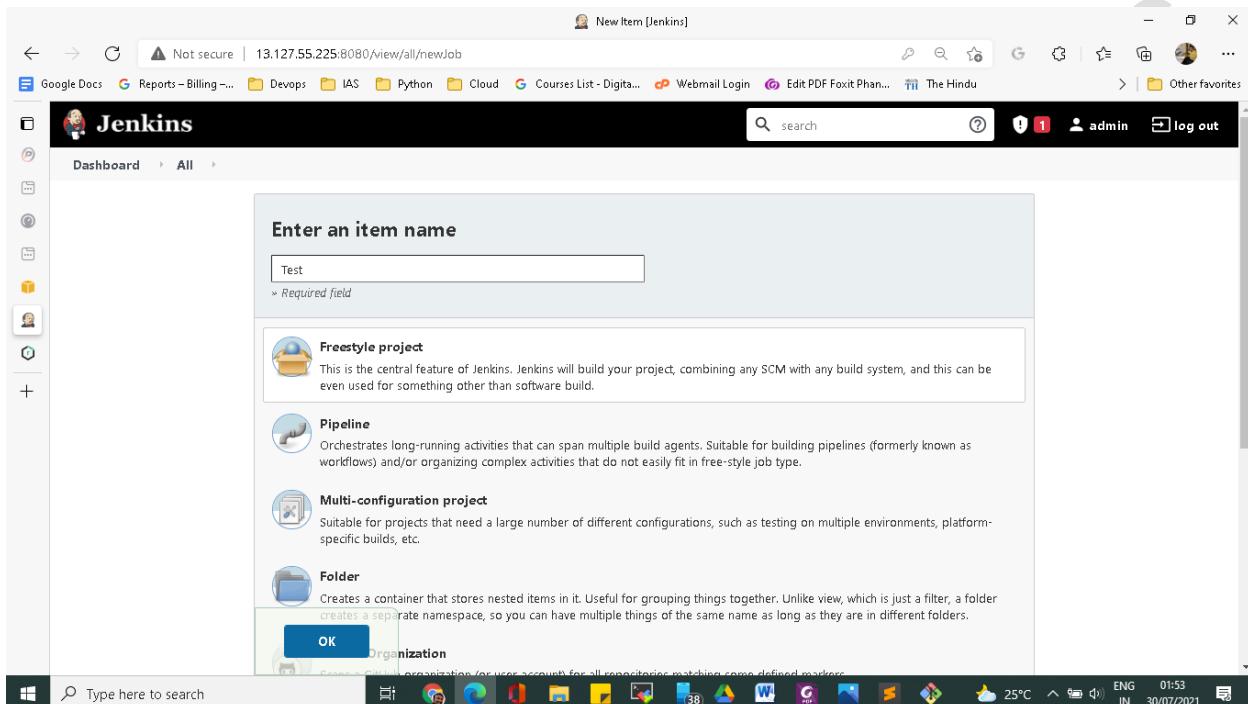
Verify whether the dependencies are getting created in the nexus repo :

The screenshot shows a browser window with the title "Browse - Nexus Repository Manager". The URL is "3.108.59.163:8081/#browse/browse/logiclabs-maven-group". The page displays the Nexus interface with a sidebar on the left and a main content area. The main content area is titled "Browse" and shows the following tree structure under "logiclabs-maven-group":

- antlr
- aopalliance
- backport-util-concurrent
- ch
- classworlds
- com
- commons-beanutils
- commons-cli
- commons-codec
- commons-collections
- commons-dbcp
- commons-digester
- commons-httpclient
- commons-lang
- commons-logging
- commons-pool
- commons-validator
- dom4j
- javax
- junit

Create the second job:

Name it as “Test” or “Unit-Test”.



Select freestyle and copy everything from Build job. For this select the last option in the screen and give the name of the job to be copied from.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

47

The screenshot shows the Jenkins 'New Item' creation dialog. It lists several item types: 'Multi-configuration project' (described as suitable for projects with many configurations), 'Folder' (described as a container for grouping items), 'GitHub Organization' (described as scanning a GitHub organization for repositories), and 'Multibranch Pipeline' (described as creating a set of Pipeline projects from detected branches). Below these, there's a note about creating new items from existing ones, with a 'Copy from' field containing 'CI-jenkins-Build1'. A large 'OK' button is at the bottom.

The screenshot shows the Jenkins 'Test Config' configuration page under the 'Build' tab. It includes fields for 'Goals' (set to 'test'), 'POM' (empty), and 'Properties' (containing environment variables like SNAP-REPO, NEXUS-USER, NEXUS-PASS, RELEASE-REPO, CENTRAL-REPO, NEXUS-GRP-REPO, NEXUSIP, and NEXUSPORT). A 'Save' and 'Apply' button are at the bottom.

Similarly create another job for performing “integration Tests”.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

48

The screenshot shows the Jenkins 'Integration' configuration page. The 'General' tab is selected. In the 'Description' section, the text 'Integration tests for CI-Jenkins project,' is entered. Below it, under 'Additional Behaviours', there are several checkboxes: 'Discard old builds', 'GitHub project', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', 'Execute concurrent builds if necessary', 'Quiet period', 'Retry Count', 'Block build when upstream project is building', and 'Block build when downstream project is building'. A checkbox for 'Use custom workspace' is also present. At the bottom of the form are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins 'Integration' configuration page with the 'Build Triggers' tab selected. Under 'Additional Behaviours', the 'Build after other projects are built' checkbox is checked. In the 'Build Triggers' section, the 'Build after other projects are built' checkbox is checked. Under 'Projects to watch', the text 'Test,' is entered. Below it, there are three radio button options: 'Trigger only if build is stable', 'Trigger even if the build is unstable', and 'Trigger even if the build fails'. There are also three checkboxes: 'Trigger builds remotely (e.g., from scripts)', 'Build periodically', and 'GitHub hook trigger for GITScm polling'. Under 'Build Environment', there are two checkboxes: 'Delete workspace before build starts' and 'Use secret text(s) or file(s)'. At the bottom of the form are 'Save' and 'Apply' buttons.

Change maven build goals as follows:

For Test job - test

For Integration -> verify -DskipUnitTests

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

49

The screenshot shows the Jenkins 'Integration' configuration page. The 'Build' tab is selected. Under 'Build', there is a section titled 'Invoke top-level Maven targets' with a 'Goals' input field containing 'verify -DskipUnitTests'. Below this is an 'Add build step' button. Under 'Post-build Actions', there is an 'Add post-build action' button. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins Dashboard. On the left, there is a sidebar with links like 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', and 'New View'. The main area displays a table of active builds:

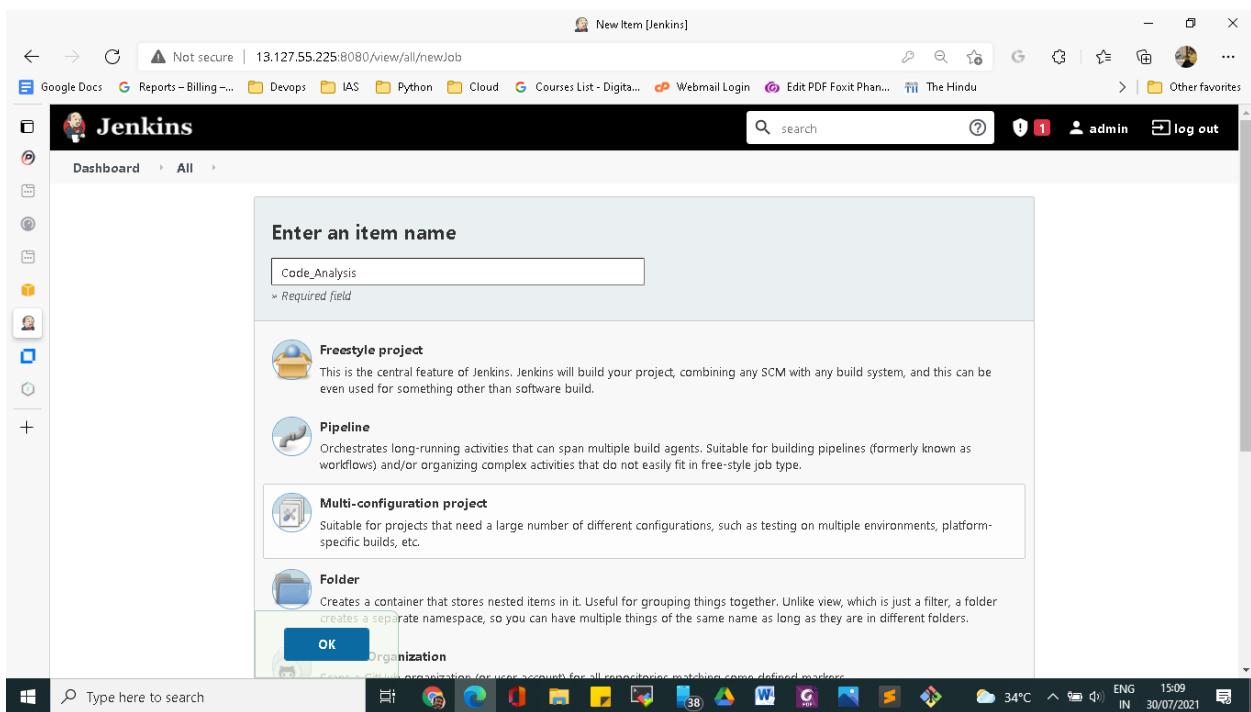
All	S	W	Name	Last Success	Last Failure	Last Duration
+	✓	⌚	CI-jenkins-Build!	2 min 5 sec - #3	N/A	8.8 sec
+	✓	⌚	Integration	1 min 27 sec - #4	N/A	13 sec
+	✓	⌚	Test	1 min 47 sec - #8	N/A	11 sec

At the bottom, there are links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

Inorder to perform static code analysis, we create a new step and name it code-analysis (you can name it as you wish).

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

50



Select copying settings from build pipeline and Click on ok

Install jenkins plugins :

→ Warnings next generation plugin

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

51

The screenshot shows the Jenkins Plugin Manager interface. The left sidebar has links for Dashboard, Manage Jenkins, and Update Center. The main area has a search bar with 'warnings' typed in. Below it, tabs for 'Updates', 'Available', 'Installed', and 'Advanced' are shown, with 'Available' selected. A table lists three plugins:

Name	Version	Released
Violations .NET Development, Maven, Build Reports This plugin does reports on checkstyle, cslint, pmd, cpd, fxcop, pylint, jcReport, findbugs, and perlcritic violations. This plugin is deprecated. In general, this means that it is either obsolete, no longer being developed, or may no longer work. Learn more .	0.7.11	8 yr 10 mo ago
Analysis Model API api-plugin Static Analysis Model and Parsers API used by the Warnings plugin.	10.2.5	2 mo 24 days ago
Warnings Next Generation This plugin collects compiler warnings or issues reported by static analysis tools and visualizes the results. It has built-in support for numerous static analysis tools (including several compilers), see the list of supported report formats .	9.4.0	11 days ago

Buttons at the bottom include 'Install without restart', 'Download now and install after restart', 'Update information obtained: 22 hr ago', and 'Check now'.

The screenshot shows the Jenkins Update Center interface. The left sidebar has links for Dashboard, Manage Jenkins, and Manage Plugins. The main area has a heading 'Installing Plugins/Upgrades'. Under 'Preparation', there is a bulleted list:

- Checking internet connectivity
- Checking update center connectivity
- Success

Below this, a table lists various Jenkins components with their status:

Component	Status
SSH server	Success
Folders	Success
Trilead API	Success
OWASP Markup Formatter	Success
Struts	Success
Pipeline: Step API	Success
Token Macro	Success
Build Timeout	Success
Credentials	Success
Plain Credentials	Success
SSH Credentials	Success
Credentials Binding	Success
SCM API	Success
Pipeline: API	Success
Timestamper	Success
Caffeine API	Success
Script Security	Success

Add a post build action to record compiler warnings and static analysis results

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

52

The screenshot shows the Jenkins configuration interface for a job named 'Code_Analysis'. The 'Build' tab is selected. A context menu is open over the 'Record compiler warnings and static analysis results' option. The menu includes options like 'Archive the artifacts', 'Build other projects', 'Discover reference build', 'Mine SCM repository', 'Publish JUnit test result report', 'Publish Javadoc', 'Record fingerprints of files to track usage', 'Report Violations', 'Git Publisher', 'E-mail Notification', 'Editable Email Notification', 'Set GitHub commit status (universal)', 'Set build status on GitHub commit [deprecated]', and 'Delete workspace when build is done'. At the bottom of the menu, there is an 'Add post-build action' button. Below the menu, there are 'Save' and 'Apply' buttons.

In the tools option, select Checkstyle.

The screenshot shows the Jenkins configuration interface for a job named 'Code_Analysis'. The 'Post-build Actions' tab is selected. Under the 'Record compiler warnings and static analysis results' section, the 'Tool' dropdown is set to 'CheckStyle'. The 'Report File Pattern' field contains the value '*/checkstyle-result.xml'. There is a checked checkbox for 'Skip symbolic links when searching for files'. Below these settings are fields for 'Report Encoding' and 'Custom ID'. At the bottom, there are 'Save' and 'Apply' buttons.

Run the job with other settings same. After success observe the checkstyle results as shown below.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

53

The screenshot shows the Jenkins 'CheckStyle Warnings' dashboard for a project named 'Code_Analysis'. The left sidebar has a 'CSI CheckStyle Warnings' section selected. The main area features a large red circle with a white center, labeled 'Overview', and a bell icon. Below it is a legend: New (red), Outstanding (yellow), and Fixed (green). To the right is a 'History' chart showing the number of issues over time, with a red shaded area representing 'New' issues and a blue bar at the bottom representing 'Fixed' issues. The chart spans from build #1 to #10.

The screenshot shows the Jenkins 'Dashboard' page. The left sidebar lists various Jenkins management options like 'New Item', 'Build History', and 'Manage Jenkins'. The main area displays a table of active Jenkins projects. Each row includes a status icon (green checkmark), a name (e.g., CI-jenkins-Build1, Code_Analysis, Integration, Test), the last success time (e.g., 43 min - #6, 34 min - #12, 42 min - #7, 43 min - #11), the last failure time (N/A or 1 hr 12 min - #7), the last duration (e.g., 8.3 sec, 7.9 sec, 13 sec, 11 sec), and the number of issues (e.g., 0, 96, 0, 0). A legend at the bottom indicates icons for S (Success), M (Failure), and L (Last Failure). At the bottom of the dashboard, there are links for 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

SONAR QUBE configuration:

Login to sonarQube with public IP. Usually Sonar Qube works on port 9000, however as we are installing it along with Nginx, we need not provide any port number after public ip in the browser to access sonar.

The image shows two screenshots of the SonarQube web application running on port 9000.

Screenshot 1: SonarQube Home Page

- The URL in the address bar is `65.0.73.49/about`.
- The page title is "SonarQube".
- The header menu includes: Google Docs, Reports - Billing, Devops, IAS, Python, Cloud, Courses List - Digital Marketing, Webmail Login, Edit PDF Foxit PhantomJS, The Hindu, and Other favorites.
- The main content area displays "Continuous Code Quality" with a "Log in" button and a "Read documentation" link.
- On the right, there is a summary of project metrics: 0 Bugs, 0 Vulnerabilities, 0 Code Smells, and 0 Security Hotspots.
- The bottom section shows supported programming languages: Java, C/C++, C#, COBOL, ABAP, HTML, RPG, JavaScript, TypeScript, Objective C, XML, VB.NET, PL/SQL, T-SQL, Flex, Python, Groovy, PHP, Swift, Visual Basic, and PL/I.
- The taskbar at the bottom shows various pinned icons and the system status: 30°C, ENG IN, 17:49, and the date 30/07/2021.

Screenshot 2: Log In to SonarQube

- The URL in the address bar is `65.0.73.49/sessions/new`.
- The page title is "SonarQube".
- The header menu is identical to the first screenshot.
- The main content area displays the "Log In to SonarQube" form with two input fields: one for "admin" and another for a password, which has a clear icon next to it.
- Below the form are "Log in" and "Cancel" buttons.
- The bottom section contains the footer text: "SonarQube™ technology is powered by SonarSource SA", "LGPLv3 - Community - Documentation - Get Support - Plugins", and the same system status information as the first screenshot.
- The taskbar at the bottom is identical to the first screenshot.

Change the default password afterwards.

The screenshot shows the SonarQube web interface. At the top, there's a navigation bar with links like 'Projects', 'Issues', 'Rules', 'Quality Profiles', 'Quality Gates', and 'Administration'. Below the navigation bar, there's a search bar and a button to 'Create new project'. On the left, there's a sidebar with 'My Favorites' and a 'Filters' section. The main area displays metrics for 'Quality Gate' (Passed: 0, Failed: 0), 'Reliability' (A-E: 0), and 'Security' (A-E: 0). A message says 'Once you analyze some projects, they will show up here.' At the bottom, it says 'SonarQube™ technology is powered by SonarSource SA' and lists 'Community Edition - Version 8.3 (build 34182) - LGPLv3 - Community - Documentation - Get Support - Plugins - Web API - About'.

Go to Administrator → my account → security→ Generate tokens

The screenshot shows the 'Security - My Account' page. At the top, there's a navigation bar with links like 'Profile', 'Security' (which is underlined, indicating it's active), 'Notifications', and 'Projects'. Below the navigation bar, there's a section titled 'Tokens' with a note about using User Tokens for security. There's a 'Generate Tokens' form with a 'Generate' button. A success message says 'New token "sonartoken" has been created. Make sure you copy it now, you won't be able to see it again!' with a 'Copy' button and the token value '7b2500e58fcf6e2221fb1959e0c13cce94ad979'. Below this, there's a table showing the token details:

Name	Last use	Created
sonartoken	Never	July 30, 2021

Create a token by the name sonartoken. Copy the token and use it in jenkins.

Get sonar plugin in Jenkins:

Jenkins → manage jenkins → manage plugins → Available

The screenshot shows the Jenkins Plugin Manager interface. A search bar at the top contains the text 'sonar'. Below the search bar, there are four tabs: 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A table lists available plugins. The first plugin listed is 'SonarQube Scanner', which has a checked checkbox next to its name. It includes a brief description: 'This plugin allows an easy integration of SonarQube, the open source platform for Continuous Inspection of code quality.' To the right of the plugin details, the 'Version' (2.13.1) and 'Released' date ('3 mo 1 day ago') are shown. The second plugin listed is 'Sonar Quality Gates', also with a checked checkbox. It includes a warning message: 'Warning: This plugin version may not be safe to use. Please review the following security notices: • Credentials transmitted in plain text'. To the right of this plugin, the 'Version' (1.3.1) and 'Released' date ('2 yr 11 mo ago') are shown. At the bottom of the table, there are three buttons: 'Install without restart', 'Download now and install after restart', and 'Check now'. The Jenkins logo is visible in the top left corner of the browser window.

Note: If these plugins are deprecated by the time this document is used, please look for alternatives...

Configure sonar in jenkins:

Global tool configuration→

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

57

The screenshot shows two consecutive screenshots of the Jenkins Global Tool Configuration page. Both screenshots are identical, displaying the 'Global Tool Configuration' section with tabs for 'Dashboard' and 'Global Tool Configuration'. The main content area lists various tools: 'SonarScanner for MSBuild', 'SonarQube Scanner', 'Ant', and 'Maven'. Under 'SonarQube Scanner', there is a sub-section for 'SonarQube Scanner installations' with a 'Add SonarQube Scanner' button. In the second screenshot, a new 'SonarQube Scanner' entry has been added, showing the configuration details: 'Name' set to 'Sonar-4.4', 'Install automatically' checked, and 'Install from Maven Central' selected with 'Version' set to 'SonarQube Scanner 4.4.0.2170'. Buttons for 'Save' and 'Apply' are visible at the bottom.

Goto jenkins→ configuration→ add sonar and enable environmental variables

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

58

The screenshot shows the Jenkins 'Configure System' page under the 'SonarQube servers' section. A checkbox for 'Environment variables injection of SonarQube server configuration as build environment variables' is checked. Below it, a 'SonarQube installations' section is shown with a single entry: 'sonar-logidbs' for Name, 'http://172.31.5.9' for Server URL, and '- none -' for Server authentication token. Buttons for 'Save', 'Apply', 'Advanced...', and 'Delete SonarQube' are at the bottom.

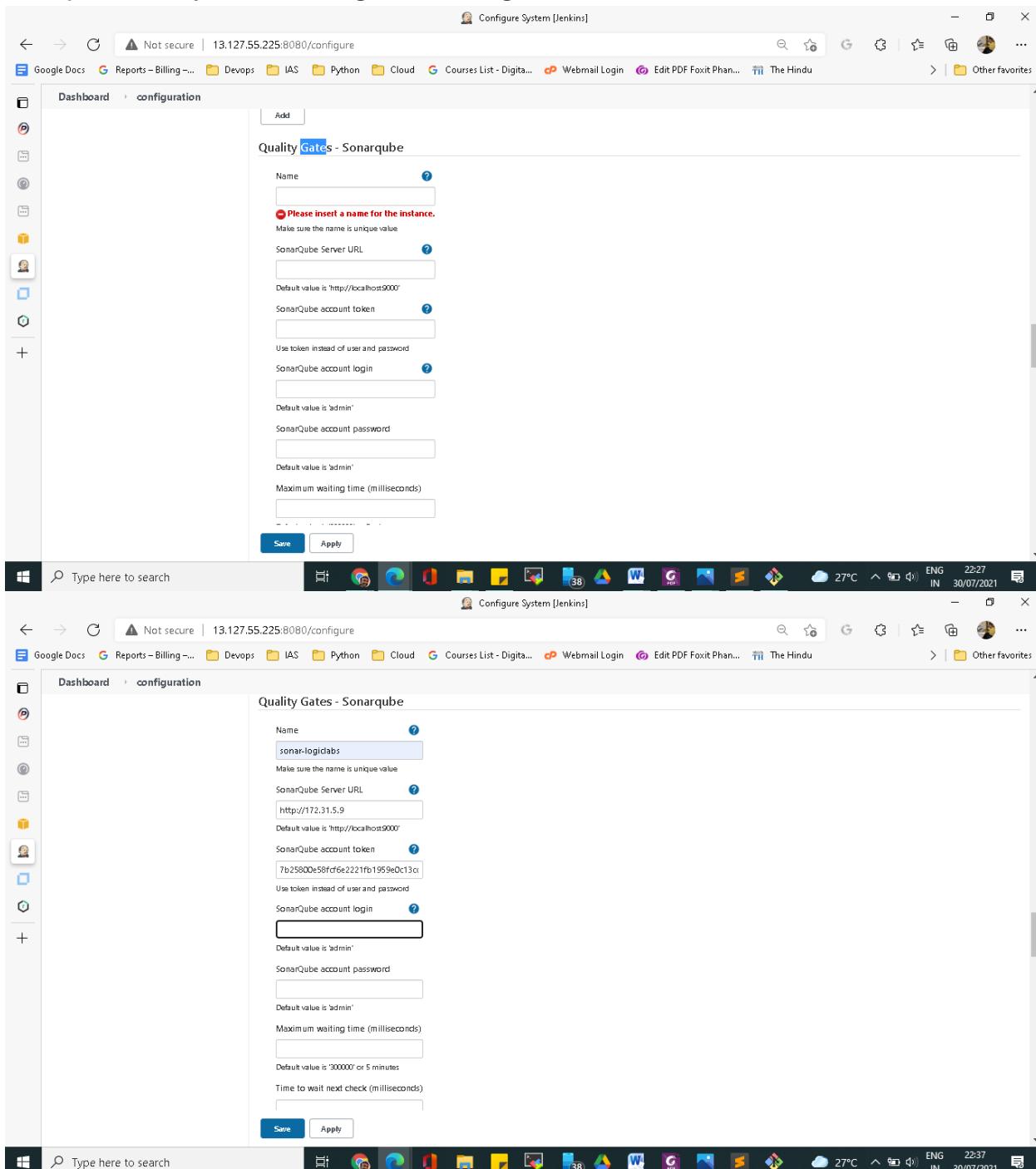
Select add credentials to add sonar token. If it doesn't open, save without adding a token and later reopen the configuration page and check again.

The screenshot shows the Jenkins 'Add Credentials' dialog box. Under 'Domain', 'Global credentials (unrestricted)' is selected. Under 'Kind', 'Secret text' is selected. In the 'Scope' dropdown, 'Global (Jenkins, nodes, items, all child items, etc.)' is chosen. The 'Secret' field contains a redacted password. The 'ID' field is set to 'sonartoke'. A 'Saved data' section shows a key-value pair: 'sonartoken'. Buttons for 'Add' and 'Cancel' are at the bottom.

save the configuration after adding the sonar token at the sonar authentication token.

Quality gates give us the option to pass the build to next stages if only certain criteria are fulfilled.

Sample Quality Gate configuration is given below.



The screenshot shows the Jenkins Configuration System interface. The left sidebar has 'Dashboard' and 'configuration' selected. A modal window titled 'Quality Gates - Sonarqube' is open. It contains the following fields:

- Name: (highlighted with a red border)
- SonarQube Server URL: (Default value is 'http://localhost:9000')
- SonarQube account token:
- SonarQube account login: (Default value is 'admin')
- SonarQube account password: (Default value is 'admin')
- Maximum waiting time (milliseconds): (Default value is '300000' or 5 minutes)
- Time to wait next check (milliseconds):

At the bottom of the modal are 'Save' and 'Apply' buttons.

Save quality gates configuration

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

60

F:\Devops projects\vprofile-project\userdata\sonar-analysis-properties (vprofile-project) - Sublime Text (UNREGISTERED)

File Edit Selection Find View Goto Tools Project Preferences Help

FOLDERS

- vprofile-project
 - ansible
 - src
 - target
 - userdata
 - jenkins-setup.sh
 - nexus-installation.sh
 - nexus-setup.sh
 - sonar-analysis-properties
 - sonar-analysis-properties
 - sonar-setup.sh
 - vagrant
 - vprofile-project
 - Jenkinsfile
 - kdfjal
 - pom.xml
 - properties-maven.xml
 - README.md
 - settings.xml

sonar.projectKey=logiclabs
sonar.projectName=logiclabs-repo
sonar.projectVersion=1.0
sonar.sources=src/
sonar.java.binaries=target/test-classes/com/visualpathit/account/controllerTest/
sonar.junit.reportsPath=target/surefire-reports/
sonar.jacoco.reportsPath=target/jacoco.exec
sonar.java.checkstyle.reportPaths=target/checkstyle-result.xml

Line 2, Column 28

Type here to search

ci-jenkins (74) Tab Size: 4 Plain Text

27°C ENG IN 22:56 30/07/2021

copy the sonar analysis properties from the file in the source directory with name 'sonar-analysis-properties'.

Add a new job in jenkins for sonar scanner code analysis.

Sonarscanner-CodeAnalysis Config [Jenkins]

Not secure | 13.127.55.225:8080/job/Sonarscanner-CodeAnalysis/configure

Google Docs G Reports – Billing ... Devops IAS Python Cloud Courses List - Digital ... Webmail Login Edit PDF Foxit Phan... The Hindu > Other favorites

Jenkins Dashboard Sonarscanner-CodeAnalysis

General Source Code Management Build Triggers Build Environment Build Post-build Actions

Description
CI-Jenkins project sonarscanner codeAnalysis

[Plain text] Preview

Discard old builds
 GitHub project
 This build requires lockable resources
 This project is parameterized
 Throttle builds
 Disable this project
 Execute concurrent builds if necessary

Advanced...

Source Code Management

None
 Git
Repositories

Repository URL: http://30.0.0.100:8080/Logilabs-cidemo.git

Save Apply

Type here to search

27°C ENG IN 22:59 30/07/2021

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

61

The screenshot shows the Jenkins configuration page for the 'Sonarscanner-CodeAnalysis' job. The 'Build' tab is selected. In the 'Path to project properties' section, the following properties are defined:

```
sonar.sources=src  
sonar.java.binaries=target/test-classes/com/visualpathlib/account/controller/test/  
sonar.junit.reportsPath=target/surefire-reports/  
sonar.jacoco.reportsPath=target/jacoco.exec  
sonar.java.checkstyle.reportPaths=target/checkstyle-result.xml
```

Below this, there is a 'Additional arguments' field which is currently empty. Under 'JVM Options', there is also an empty field. In the 'Post-build Actions' section, a 'Build other projects' step is configured to build the 'Test' project. At the bottom of the configuration page, there are 'Save' and 'Apply' buttons.

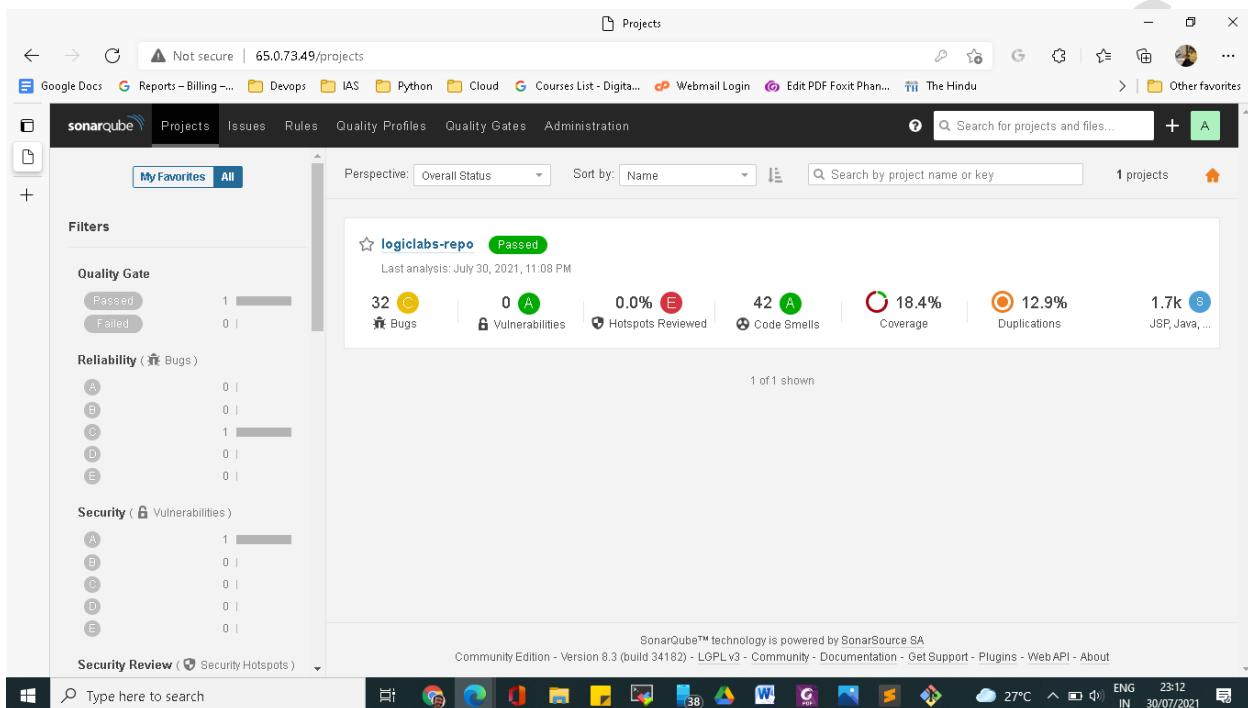
copy paste the variables from the file above mentioned.

The screenshot shows the Jenkins dashboard for the 'Sonarscanner-CodeAnalysis' project. The left sidebar shows various project management options like Status, Changes, Workspace, and SonarQube. The 'SonarQube' option is currently selected and highlighted with a blue border. The main content area displays the project details and its status. It shows the project name 'Project Sonarscanner-CodeAnalysis' and the CI-Jenkins project 'sonarscanner codeAnalysis'. Below this, there are links for SonarQube, Workspace, and Recent Changes. A 'SonarQube Quality Gate' section indicates that the 'logilabs-repo' build has passed and the 'server-side processing' build has also passed. The 'Permalinks' section provides a link to the last successful build. The status bar at the bottom shows the system is waiting for 13.127.55.225...

If the build fails because of checkstyle issues, it can be removed from maven targets step for this demonstration.

Verification of Static code Analysis from sonar server:

After Jenkins job is complete, login to sonar server to verify and obtain the static code analysis results.



(Optional)For experimentation purposes, the name of the branch in the jenkins job can be changed to vp-rem and check the sonar results after running the job.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

63

The screenshot shows the Jenkins configuration page for a job named "Sonarscanner-CodeAnalysis". The "Source Code Management" tab is selected. Under "Branches to build", the "Branch Specifier" field contains the value "*/ip-realm". Below it, the "Repository browser" dropdown is set to "(Auto)". In the "Build Triggers" section, there are three checkboxes: "Trigger builds remotely (e.g., from scripts)", "Build after other projects are built", and "Build periodically". A "GitHub hook trigger for GitHub Polling" option is also present. At the bottom of the configuration page, there are "Save" and "Apply" buttons.

The screenshot shows the SonarQube interface for the project "logiclabs-repo". The dashboard displays various quality metrics: 82 E Bugs, 3 B Vulnerabilities, 0.0% E Hotspots Reviewed, 138 A Code Smells, 6.1% Coverage, 10.0% Duplications, and 3.3k S JSP, Java, ... files. On the left, there are filters for Quality Gate (Passed: 1, Failed: 0), Reliability (A: 0, B: 0, C: 0, D: 0, E: 1), Security (A: 0, B: 1, C: 0, D: 0, E: 0), and a Security Review section. The bottom of the page includes links for SonarQube™ technology information and community resources.

Creating Quality Gates:

The below screenshot shows how to configure Quality gates.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

64

Sonar way - Quality Gates

Not secure | 65.0.73.49/quality_gates/show/1

Google Docs Reports - Billing ... Devops IAS Python Cloud Courses List - Digital ... Webmail Login Edit PDF Foxit Phan... The Hindu Other favorites

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration

Quality Gates Create

Sonar way BUILT-IN

Sonar way DEFAULT BUILT-IN

Conditions Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Projects Every project not specifically associated to a quality gate will be associated to this one by default.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 8.3 (build 34182) - LGPLv3 - Community - Documentation - Get Support - Plugins - Web API - About

Type here to search

27°C ENG IN 23:27 30/07/2021

Sonar way - Quality Gates

Not secure | 65.0.73.49/quality_gates/show/1

Google Docs Reports - Billing ... Devops IAS Python Cloud Courses List - Digital ... Webmail Login Edit PDF Foxit Phan... The Hindu Other favorites

sonarcube Projects Issues Rules Quality Profiles Quality Gates Administration

Quality Gates Create

Sonar way DEFAULT BUILT-IN

Create Quality Gate

Name* sonar-qualitygate

Save Cancel

Conditions Conditions on New Code

Metric	Operator	Value
Coverage	is less than	80.0%
Duplicated Lines (%)	is greater than	3.0%
Maintainability Rating	is worse than	A
Reliability Rating	is worse than	A
Security Hotspots Reviewed	is less than	100%
Security Rating	is worse than	A

Projects Every project not specifically associated to a quality gate will be associated to this one by default.

SonarQube™ technology is powered by SonarSource SA
Community Edition - Version 8.3 (build 34182) - LGPLv3 - Community - Documentation - Get Support - Plugins - Web API - About

Type here to search

27°C ENG IN 23:27 30/07/2021

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

65

The screenshot shows the SonarQube interface for managing Quality Gates. A modal window titled "Add Condition" is open, allowing the configuration of a new quality gate. The "On New Code" option is selected. The "Quality Gate fails when" dropdown is set to "Bugs". The "Operator" is "is greater than" and the "Value" is "50". The background shows a list of existing quality gates, including "sonar-qualitygate" which is currently selected.

The screenshot shows the SonarQube interface for the "Projects" perspective. The main view displays the analysis results for the "logiclabs-repo" project, which is marked as "Passed". Key metrics shown include: 82 E Bugs, 3 B Vulnerabilities, 0.0% E Hotspots Reviewed, 138 A Code Smells, 6.1% Coverage, 10.0% Duplications, and 3.3k S JSP, Java, ... files. On the left, there are filters for Quality Gate (Passed: 1, Failed: 0), Reliability (A: 0, B: 0, C: 0, D: 0, E: 1), and Security (A: 0, B: 1, C: 0, D: 0). The bottom of the page includes copyright information for SonarQube™ technology and links to the Community Edition version 8.3 documentation and support.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

66

The screenshot shows the SonarQube Quality Gates interface. The URL is 65.0.73.49/quality_gates/show/2. The page title is "sonar-qualitygate - Quality Gates". The left sidebar has icons for Projects, Issues, Rules, Quality Profiles, Quality Gates (selected), and Administration. The main area shows a table for "Conditions on New Code" with one row:

Metric	Operator	Value	Edit	Delete
Bugs	is greater than	50		

Below the table is a section titled "Projects" with tabs: With, Without, All. A search bar is also present.

Copying Artifact Stage:

The build which passes through these stages and ready to be deployed is called an Artifact in SDLC lifecycle.

The artifacts are stored and versioned using Artifacts repository.

For this demonstration purpose, **Sonatype Nexus-3** is being used.

Before that let's install a relevant plugin in the Jenkins .

1. Copy Artifact plugin
2. Zentimestamp plugin

The screenshot shows the Jenkins Plugin Manager interface. The URL in the address bar is 13.127.55.225:8080/pluginManager/available. The search bar contains the text 'copy'. The 'Available' tab is selected. A table lists several plugins, with the 'Copy Artifact' plugin highlighted. The 'Copy Artifact' plugin details are as follows:

Install	Name	Version	Released
<input checked="" type="checkbox"/>	Copy Artifact	1.46.1	2 mo 1 day ago
<input type="checkbox"/>	Scriptler	3.3	1 mo 15 days ago
<input type="checkbox"/>	SVN 1.4 Compatibility	1.1	9 yr 8 mo ago
<input type="checkbox"/>	Copy data to workspace	1.0	10 yr ago

Below the table are buttons for 'Install without restart' and 'Download now and install after restart'. A status message says 'Update information obtained: 1 day 6 hr ago'. There is also a 'Check now' button. The bottom of the screen shows the Windows taskbar with various icons and system status.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

68

The screenshot shows the Jenkins Plugin Manager interface. A search bar at the top contains the text 'zentimestamp'. Below it, there are tabs for 'Updates', 'Available' (which is selected), 'Installed', and 'Advanced'. A single result is listed: 'Zentimestamp' under the 'Build Wrappers' category. The description states: 'Plugin that allows the customization of the date and time pattern for the Jenkins BUILD_TIMESTAMP variable.' It includes a note: 'This plugin is up for adoption! We are looking for new maintainers. Visit our [Adopt a Plugin](#) initiative for more information.' The version is 4.2, and it was released 6 years ago. At the bottom, there are buttons for 'Install without restart' and 'Download now and install after restart', along with a 'Check now' button.

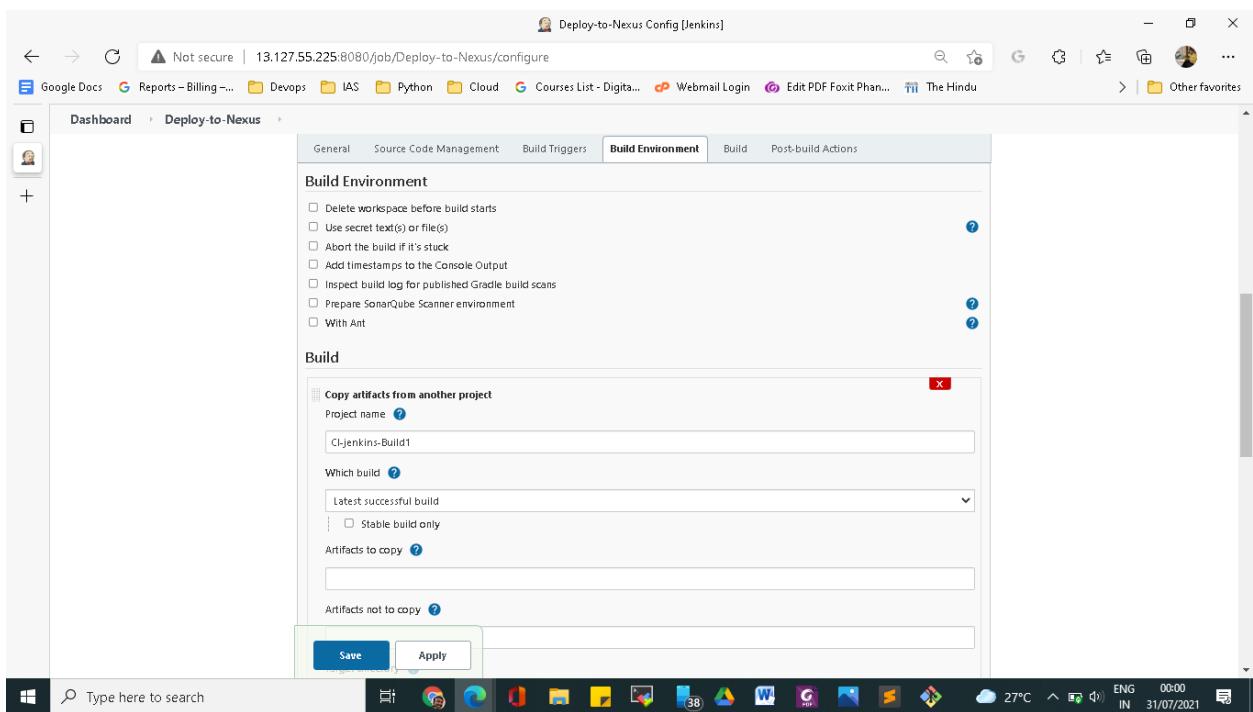
Create a new step in Jenkins for Deploying artifact to nexus.

The screenshot shows the Jenkins 'New Item' creation page. The title bar says 'New Item [Jenkins]'. The main area has a heading 'Enter an item name' with a text input field containing 'Deploy-to-Nexus'. Below the input field is a note: 'Required field'. There are five job type options listed: 'Freestyle project', 'Maven project', 'Pipeline', 'External Job', and 'Multi-configuration project'. Each option has a brief description. At the bottom of the list is a blue 'OK' button. The status bar at the bottom shows system information: Windows 10, Type here to search, various pinned icons, 27°C, ENG IN, 23:59, and 30/07/2021.

click on add build step and select artifact upload option to give artifact details

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

69



click on add build step and select nexus artifact uploader option to give artifact details

Add the credentials of Nexus server in the Jenkins credentials provider.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

70

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The 'Build' tab is selected. A modal window titled 'Jenkins Credentials Provider: Jenkins' is open, showing the 'Add Credentials' form. The 'Domain' dropdown is set to 'Global credentials (unrestricted)'. The 'Kind' dropdown is set to 'Username with password'. The 'Scope' dropdown is set to 'Global (Jenkins, nodes, items, all child items, etc.)'. The 'Username' field contains 'admin' and the 'Password' field contains '*****'. The 'ID' field contains 'nexuslogin'. Below the form, a message says 'Repository must not be empty'. At the bottom of the modal are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The 'General' tab is selected. The 'Description' field contains 'deploy artifact'. Under the 'Preview' section, there is a note: '[Plain text] Preview' and a checked checkbox for 'Change date pattern for the BUILD_TIMESTAMP (build timestamp) variable'. Below this, there is a 'Date and Time Pattern' input field containing 'yyyyMMddHHmmss'. There are several other unchecked checkboxes: 'Discard old builds', 'GitHub project', 'Permission to Copy Artifact', 'This build requires lockable resources', 'This project is parameterized', 'Throttle builds', 'Disable this project', and 'Execute concurrent builds if necessary'. At the bottom of the 'General' tab are 'Save' and 'Apply' buttons.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

71

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Nexus' job. The 'Source Code Management' tab is selected, showing 'Git' as the provider. The 'Build Triggers' section has 'Build after other projects are built' checked. The 'Build Environment' section includes options like 'Delete workspace before build starts' and 'Add timestamps to the Console Output'. At the bottom are 'Save' and 'Apply' buttons.

The screenshot shows the Jenkins configuration page for the 'Deploy-to-Nexus' job. The 'Build' tab is selected, showing the 'Copy artifacts from another project' section. It specifies 'Project name: CI-Jenkins-Build1' and 'Which build: Latest successful build'. Under 'Artifacts to copy', the pattern '**/*.war' is listed. The 'Target directory' field is empty. At the bottom are 'Save' and 'Apply' buttons.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

72

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The URL in the browser is `13.127.55.225:8080/job/Deploy-to-Nexus/configure`. The page title is 'Deploy-to-Nexus Config [Jenkins]'. The left sidebar has a 'Dashboard' icon and a '+' icon. The main content area has a 'Deploy-to-Nexus' heading and a 'Nexus artifact uploader' section. The 'Nexus Details' tab is selected. The configuration includes:

- Nexus Version: NEXUS3
- Protocol: HTTP
- Nexus URL: 172.31.0.227:8081
- Credentials: admin/******** (nexuslogin) (dropdown menu)
- GroupID: QA
- Version: L\$BUILD_ID
- Repository: logiclabs-release

At the bottom of the configuration form are 'Save' and 'Apply' buttons. Below the configuration form is a Windows taskbar with various icons and a search bar.

Note: Make sure you don't repeat http while giving the nexus pvt URL.

Give the time format as given in the screenshot below.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

73

The screenshot shows two stacked Jenkins configuration pages for a job named "Deploy-to-Nexus".

Top Configuration Page (General Tab):

- Description:** deploy artifact
- Date and Time Pattern:** yyyyMMddHHmmss (highlighted in red)
- Message:** You must provide a pattern value
- Buttons:** Save, Apply

Bottom Configuration Page (Build Tab):

- Repository:** logilabs-release
- Artifacts:**
 - ArtifactId:** \$BUILD_TIMESTAMP
 - Type:** war
 - Classifier:** (empty)
 - File:** target\profile-v2.war
- Buttons:** Add, Save, Apply

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

74

The screenshot shows the Jenkins 'Deploy-to-Nexus' configuration page. The URL is <http://13.127.55.225:8080/job/Deploy-to-Nexus/configure>. The page has tabs for General, Source Code Management, Build Triggers, Build Environment, Build, Nexus Details, and Post-build Actions. Under Artifacts, there is a section for 'Artifact'. The 'ArtifactId' field contains '\$BUILD_TIMESTAMP'. The 'Type' field is set to 'war'. The 'Classifier' field is empty. The 'File' field contains 'target\profile-v2.war'. There is an 'Add' button and an 'Add build step' dropdown. Under Post-build Actions, there is a 'Save' button and an 'Apply' button. The status bar at the bottom shows the date as 31/07/2021.

VALIDATE Artifact Deployment to Nexus repository:

Goto Nexus server → Browse repo → release repository

The screenshot shows the Sonatype Nexus Repository Manager 'Browse' page. The URL is <http://3.108.59.163:8081/#/browse/browse/logiclabs-release>. The page has a sidebar with 'Browse' selected. The main area shows a tree view under 'HTML View': 'QA' → '20210730185301' → 'L3' → '20210730185301-L3.war', '20210730185301-L3.war.md5', and '20210730185301-L3.war.sha1'. There is a 'Search components' input field and a 'Sign In' button. The status bar at the bottom shows the date as 31/07/2021.

BUILDING Pipeline:

In jenkins install the pipeline plugin as shown below. It helps in visually appealing pipeline view in Jenkins.

Click on the + symbol besides All in the jobs dashboard to add a new view.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

76

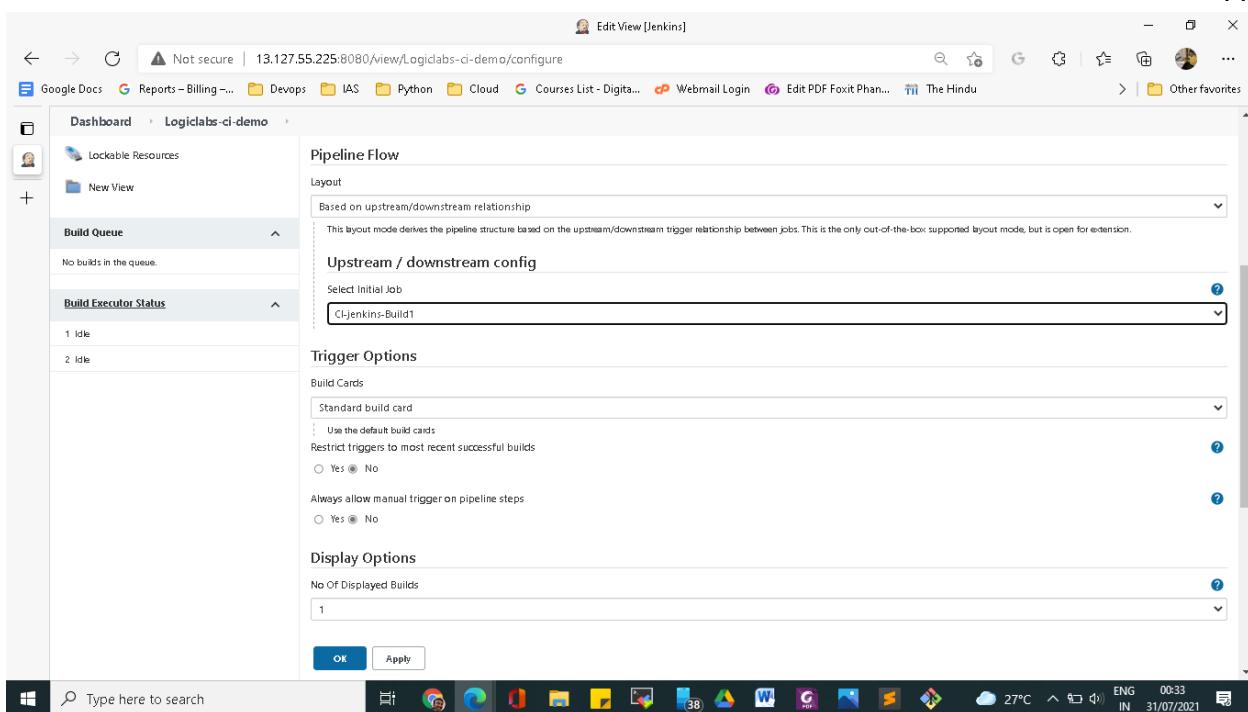
The screenshot shows the Jenkins dashboard with the 'New View' option selected in the sidebar. A modal window is open for creating a new view, with the name 'Logilabs-ci-demo' entered in the 'View name' field. The 'Build Pipeline View' radio button is selected. The Jenkins interface includes a sidebar with various management options like 'New Item', 'People', and 'Manage Jenkins', and a main area showing build queue and executor status.

Give the name of the first job to be started .

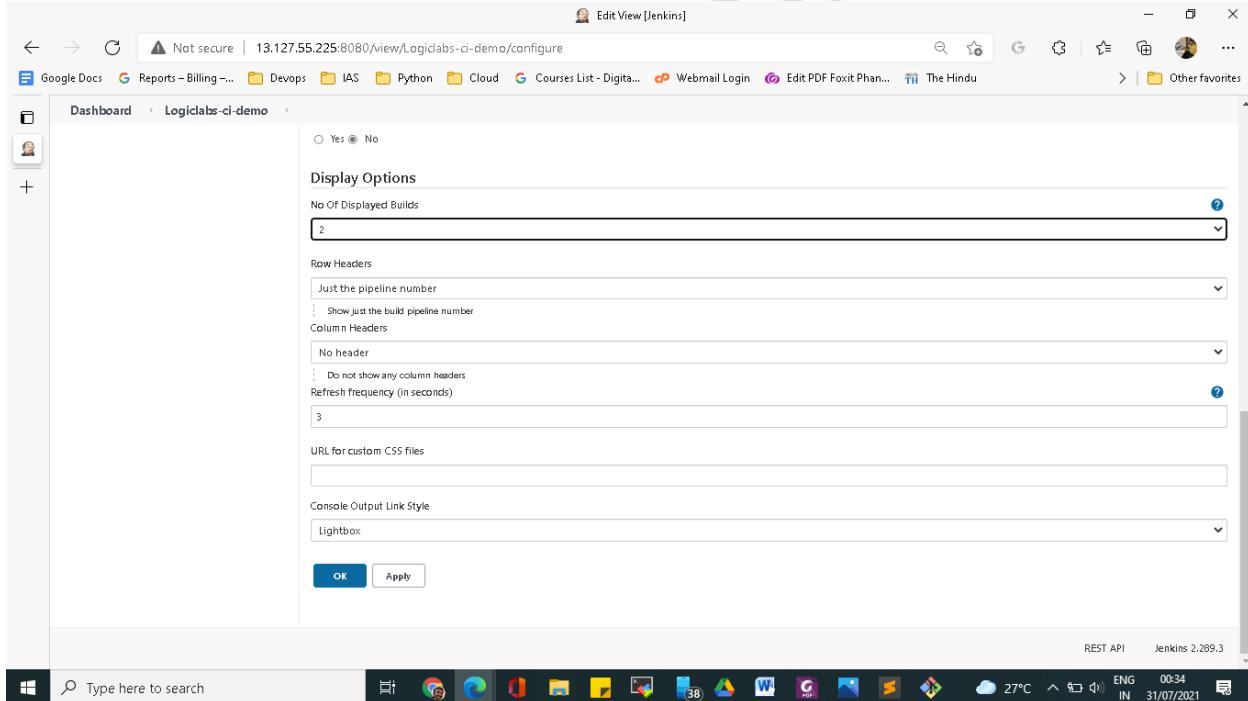
The screenshot shows the 'Edit View' configuration page for the 'Logilabs-ci-demo' view. The 'Name' field is set to 'Logilabs-ci-demo'. Under 'Pipeline Flow', the 'Layout' is set to 'Based on upstream/downstream relationship'. In the 'Upstream / downstream config' section, the 'Select Initial Job' dropdown is populated with 'CI-jenkins-Build1'. The Jenkins interface includes a sidebar with options like 'Edit View', 'Delete View', and 'New View', and a main area showing build queue and executor status.

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

77



The screenshot shows the Jenkins Pipeline Flow configuration page for the 'Logilabs-ci-demo' view. The 'Upstream / downstream config' section is expanded, showing the 'Select Initial Job' dropdown set to 'CI-jenkins-Build1'. The 'Trigger Options' section includes 'Standard build card' and 'Always allow manual trigger on pipeline steps' options. The 'Display Options' section has 'No Of Displayed Builds' set to 1. At the bottom are 'OK' and 'Apply' buttons.



The screenshot shows the Jenkins Pipeline Flow configuration page for the 'Logilabs-ci-demo' view. The 'Display Options' section is expanded, showing 'No Of Displayed Builds' set to 2. Other settings include 'Row Headers' (set to 'Just the pipeline number'), 'Column Headers' (set to 'No header'), and 'Refresh frequency (in seconds)' set to 3. At the bottom are 'OK' and 'Apply' buttons.

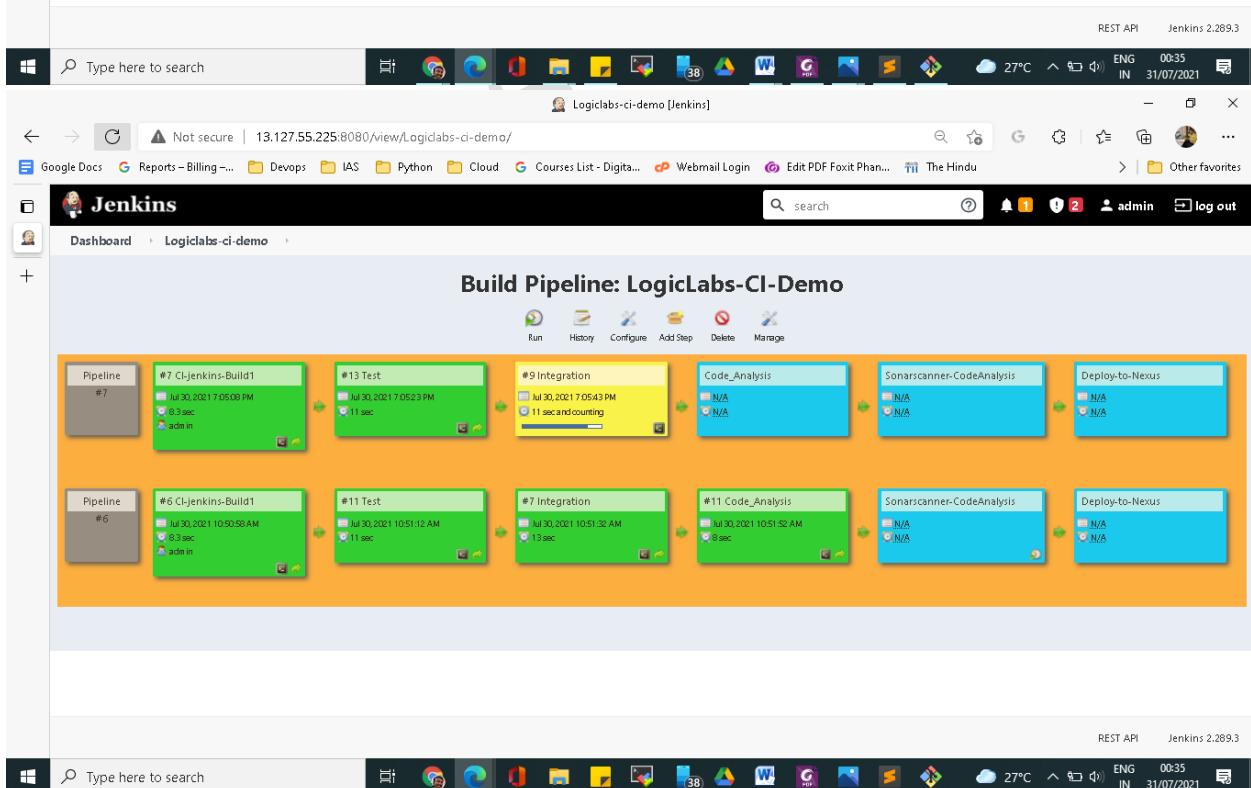
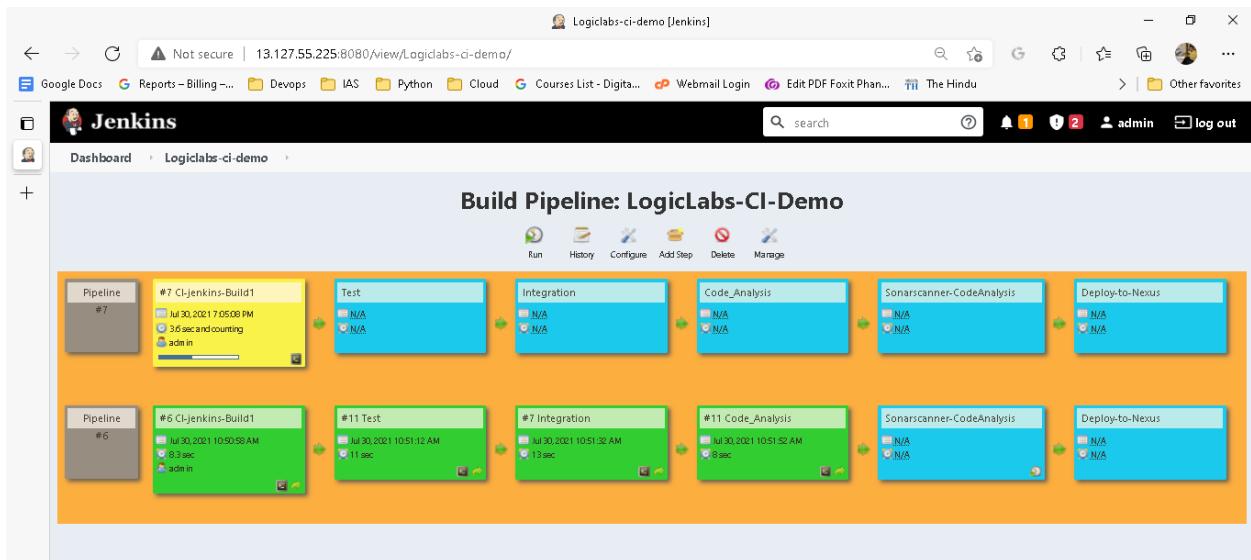
Click on Run to start the pipeline. Observe that all the downstream Jobs are performed sequentially.

Verify that all the jobs are properly configured for downstream jobs/upstream jobs.

Please make sure the pipeline flow is as shown below.

CI jenkins Build → UnitTest→ Integration Test→ CodeAnalysis → SonarScanner

Code Analysis → Deploy to Nexus



A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

79

The screenshot shows the Jenkins Pipeline dashboard for the project "LogicLabs-ci-demo". It displays two parallel pipelines, Pipeline #7 and Pipeline #6, each consisting of several stages. The stages are color-coded: Pipeline steps are grey, Test steps are green, Integration steps are green, Code Analysis steps are yellow, SonarScanner analysis steps are blue, and Deploy steps are light blue. Pipeline #7 has completed its stages successfully. Pipeline #6 is currently running, with the last stage, "Deploy-to-Nexus", still pending.

Stage	Pipeline	Status	Duration	Last Run
CI-Jenkins-Build1	#7	Success	8.3 sec	Jul 30, 2021 7:05:08 PM
Test	#7	Success	11 sec	Jul 30, 2021 7:05:23 PM
Integration	#7	Success	13 sec	Jul 30, 2021 7:05:43 PM
Code Analysis	#7	In Progress	66 second counting	Jul 30, 2021 7:06:03 PM
Sonarscanner-CodeAnalysis	#7	Success	N/A	N/A
Deploy-to-Nexus	#7	Success	N/A	N/A
CI-Jenkins-Build1	#6	In Progress	8.3 sec	Jul 30, 2021 10:50:56 AM
Test	#6	In Progress	11 sec	Jul 30, 2021 10:51:12 AM
Integration	#6	In Progress	13 sec	Jul 30, 2021 10:51:32 AM
Code Analysis	#6	In Progress	3 sec	Jul 30, 2021 10:51:52 AM
Sonarscanner-CodeAnalysis	#6	Success	N/A	N/A
Deploy-to-Nexus	#6	Success	N/A	N/A

This screenshot is identical to the one above, showing the Jenkins Pipeline dashboard for the project "LogicLabs-ci-demo". It displays two parallel pipelines, Pipeline #7 and Pipeline #6, with the same stages and current status. Pipeline #7 is completed, while Pipeline #6 is still in progress, with the final stage "Deploy-to-Nexus" pending.

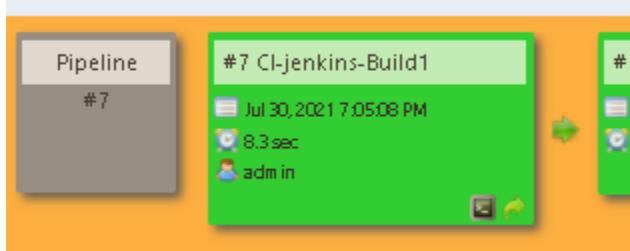
Stage	Pipeline	Status	Duration	Last Run
CI-Jenkins-Build1	#7	Success	8.3 sec	Jul 30, 2021 7:05:08 PM
Test	#7	Success	11 sec	Jul 30, 2021 7:05:23 PM
Integration	#7	Success	13 sec	Jul 30, 2021 7:05:43 PM
Code Analysis	#7	In Progress	66 second counting	Jul 30, 2021 7:06:03 PM
Sonarscanner-CodeAnalysis	#7	Success	N/A	N/A
Deploy-to-Nexus	#7	Success	N/A	N/A
CI-Jenkins-Build1	#6	In Progress	8.3 sec	Jul 30, 2021 10:50:56 AM
Test	#6	In Progress	11 sec	Jul 30, 2021 10:51:12 AM
Integration	#6	In Progress	13 sec	Jul 30, 2021 10:51:32 AM
Code Analysis	#6	In Progress	3 sec	Jul 30, 2021 10:51:52 AM
Sonarscanner-CodeAnalysis	#6	Success	N/A	N/A
Deploy-to-Nexus	#6	Success	N/A	N/A

A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

80

The screenshot shows the Jenkins dashboard for the project 'LogicLabs-ci-demo'. The main title is 'Build Pipeline: LogicLabs-CI-Demo'. Below it, there are two rows of Jenkins jobs. The first row contains: 'Pipeline #7 CI-jenkins-Build1' (status: green), '#13 Test' (status: green), '#9 Integration' (status: green), '#14 Code_Analysis' (status: green), '#5 Sonarscanner-CodeAnalysis' (status: green), and '#4 Deploy-to-Nexus' (status: green). The second row contains: 'Pipeline #6 CI-jenkins-Build1' (status: green), '#11 Test' (status: green), '#7 Integration' (status: green), '#11 Code_Analysis' (status: green), 'Sonarscanner-CodeAnalysis' (status: blue), and 'Deploy-to-Nexus' (status: blue). The Jenkins interface includes a search bar, navigation links, and a top menu bar.

Click on the console button on each of the jobs to see the logs



A Project Demonstration on Continuous Integration using Jenkins, SonarQube and Nexus

81

The screenshot shows a browser window with the URL 13.127.55.225:8080/view/Logiclabs-ci-demo/job/CI-jenkins-Build1/. The page title is "Logiclabs-ci-demo [Jenkins]". The main content is the "Console Output" for build #7, which displays the command-line logs of the build process. The logs show the Jenkins user "admin" starting the build, cloning the repository from GitHub, and committing changes to Bitbucket Pipelines. The Jenkins interface includes a sidebar with Pipeline #7 and Pipeline #6, and a top navigation bar with links like "Dashboard", "Status", "Changes", "Console Output", "Edit Build Information", "Delete build #7", "Git Build Data", and "See Fingerprints". The Jenkins version is 2.289.3.

Automating the build pipeline triggering:

Enable poll scm or github webhooks and observe that the pipeline is triggered automatically whenever any commit happens in the SCM tool (Github)

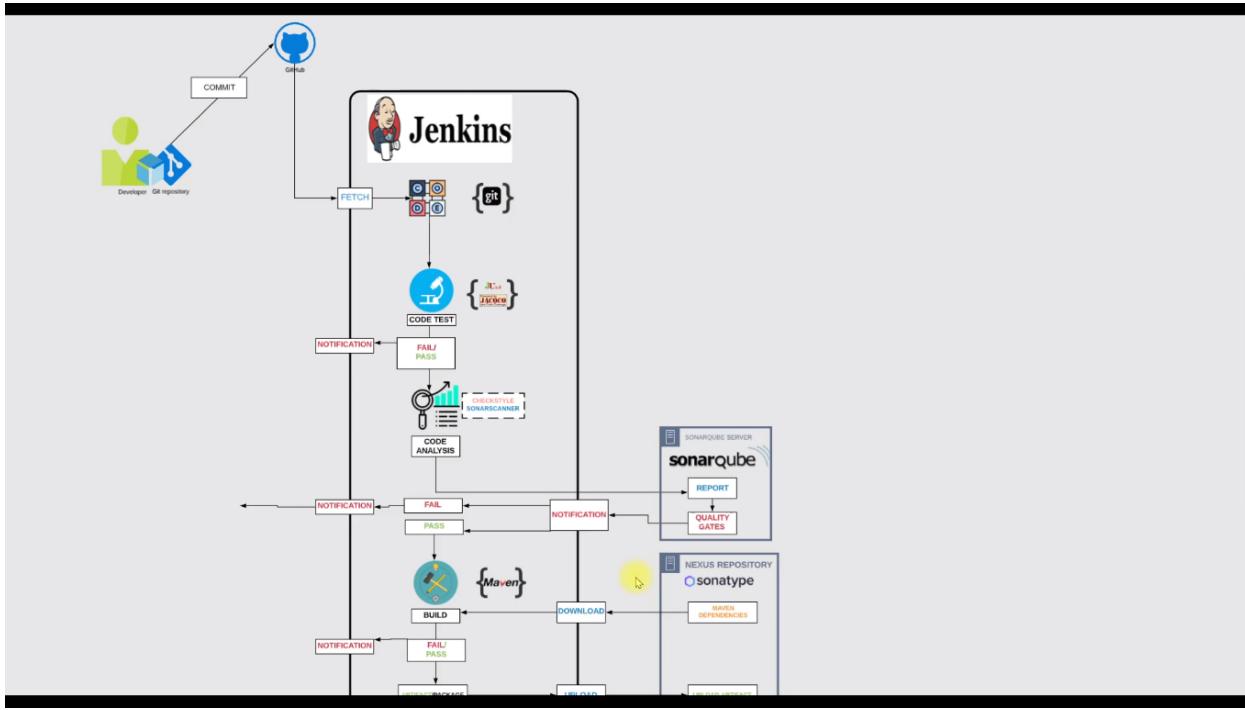
The screenshot shows the Jenkins configuration page for the "CI-jenkins-Build1" job. The URL is 13.127.55.225:8080/job/CI-jenkins-Build1/configure. The "Build Triggers" tab is selected. Under "Build Triggers", the "Poll SCM" option is checked, and the "Schedule" field contains the cron expression "1 * * * *". There is also an unchecked checkbox for "Ignore post-commit hooks". Other tabs include "General", "Source Code Management", "Build Environment", "Build", and "Post-build Actions". At the bottom are "Save" and "Apply" buttons. The Jenkins version is 2.289.3.

Eg: 1 */5 * * * = check for commits and start building at every 5 hour interval of every day.

Note: Inorder to verify, create your own repo which is cloned from the original repo and experiment with this automated pipeline triggering mechanism.

LogicLab Technologies

Summary:



In this Demonstration, we have witnessed how a C.I. pipeline can be built using tools such as jenkins, sonarQube and Nexus. We have observed how the source code commit by developer triggers a pipeline which starts with build job followed by Unit test, Integration test, Code analysis using SonarQube scanner and finally a well tested artifact is uploaded into Nexus repository.