intelligaia

Jenkins Installation and Setup

By
Ashutosh Gupta
DevOps Engineer
DevOps/Networking Department
Intelligaia Technologies

	Author	Version	Change Reference
21/08/2023	Ashutosh Gupta	1.0	Creation of document – Draft

Table Of contents

1. Introduction:

Jenkins

Jenkins is an open source continuous integration/continuous delivery and deployment (CI/CD) automation software DevOps tool written in the Java programming language. It is used to implement CI/CD workflows, called pipelines.

Pipelines automate testing and reporting on isolated changes in a larger code base in real time and facilitates the integration of disparate branches of the code into a main branch. They also rapidly detect defects in a code base, build the software, automate testing of their builds, prepare the code base for deployment (delivery), and ultimately deploy code to containers and virtual machines, as well as bare metal and cloud servers. There are several commercial versions of Jenkins. This definition only describes the upstream open source project.

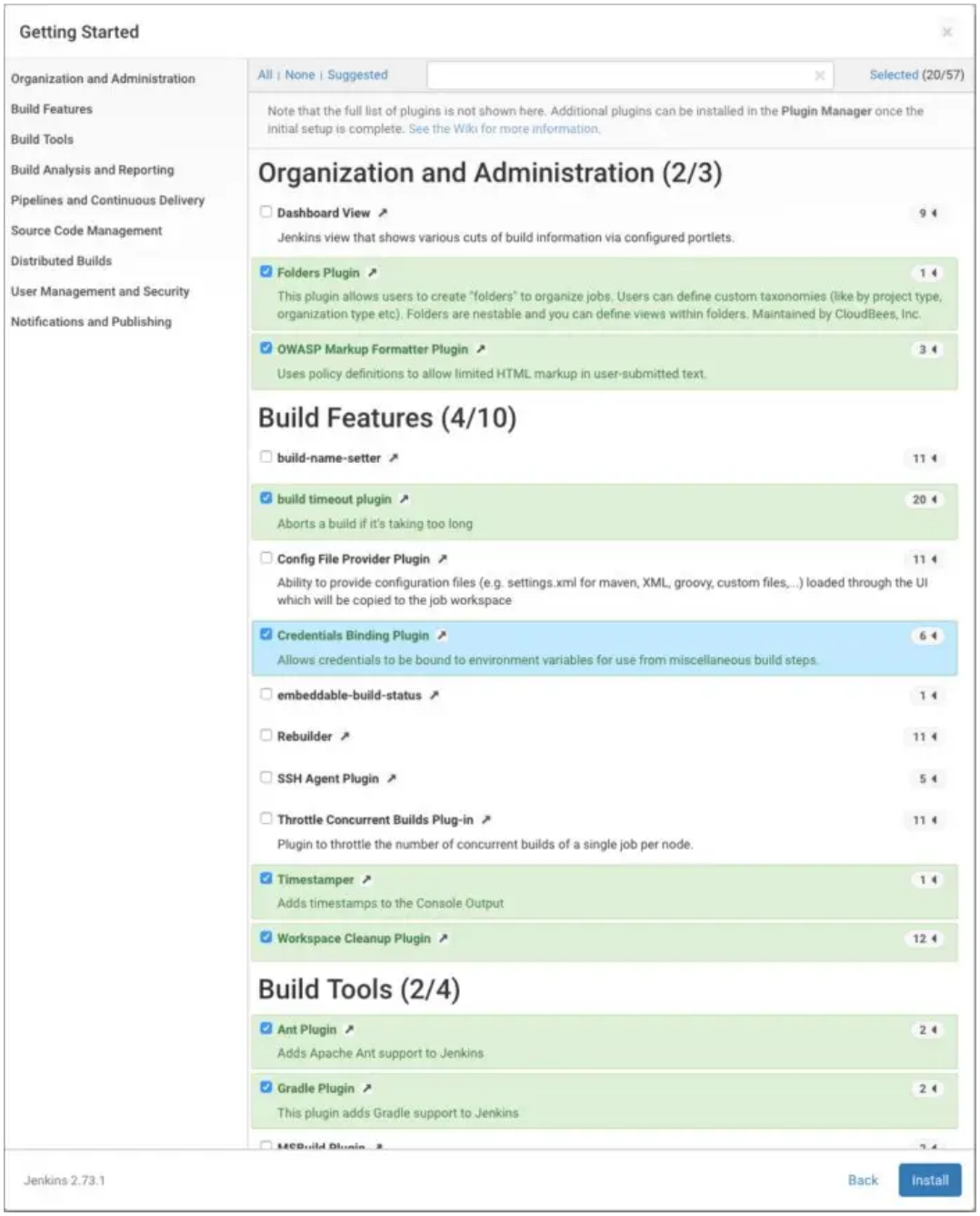
How Jenkins works

Jenkins is distributed as a WAR archive and as installer packages for the major operating systems, as a Homebrew package, as a Docker image, and as source code. Jenkins also supports installation and scaling on Kubernetes. The source code is mostly Java, with a few Groovy, Ruby, and Antlr files.

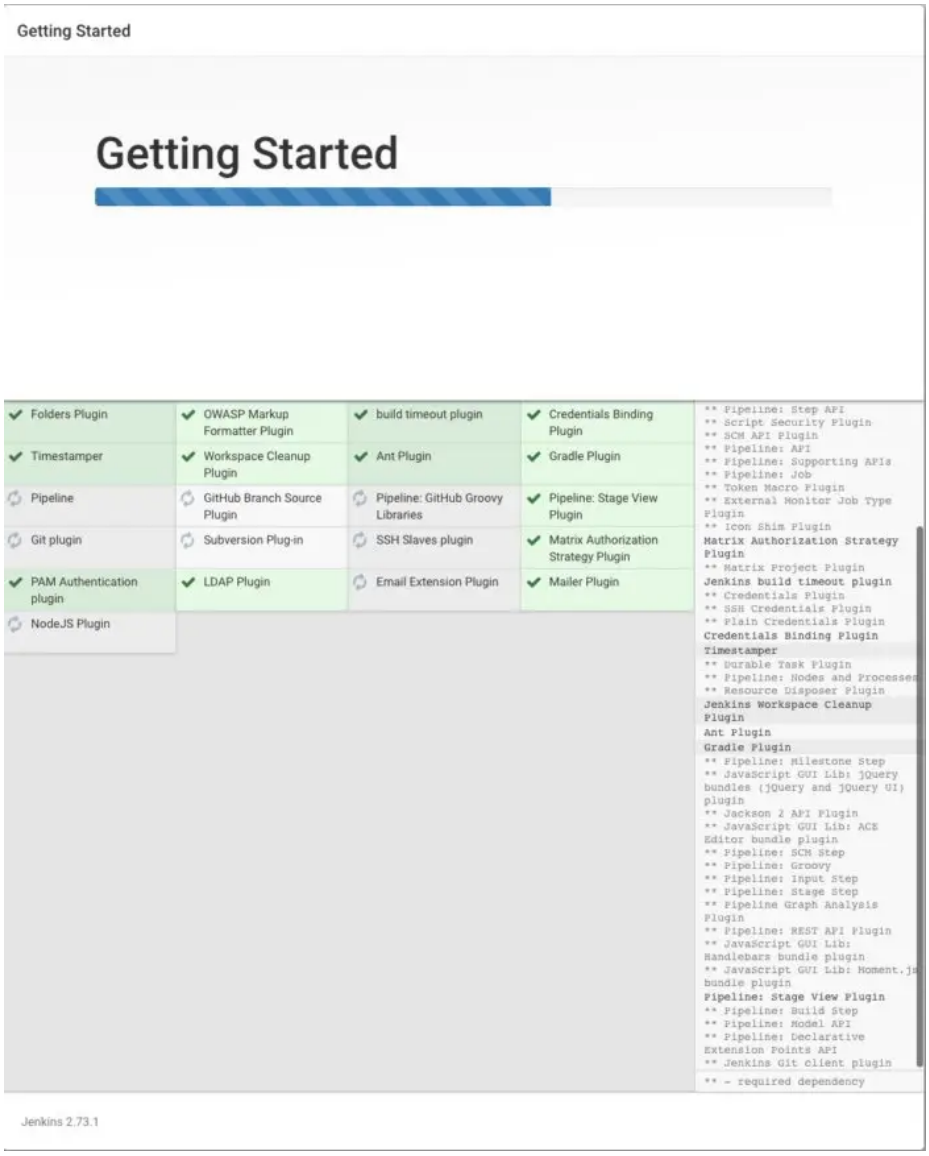
You can run the Jenkins WAR standalone or as a servlet in a Java application server such as Tomcat. In either case, it produces a web user interface and accepts calls to its REST API. When you run Jenkins for the first time, it creates an administrative user with a long random password, which you can paste into its initial web page to unlock the installation.

Jenkins plugins

Once installed, Jenkins allows you to either accept the default plugin list or choose your own plugins.



Once you have picked your initial set of plugins, click the Install button and Jenkins will add them.



The Jenkins main screen displays the current build queue and Executor status, and offers links to create new items (jobs), manage users, view build histories, manage Jenkins, look at your custom views, and manage your credentials.

2. Prerequisites:

- Linux Server (Ubuntu 20.04 LTS)

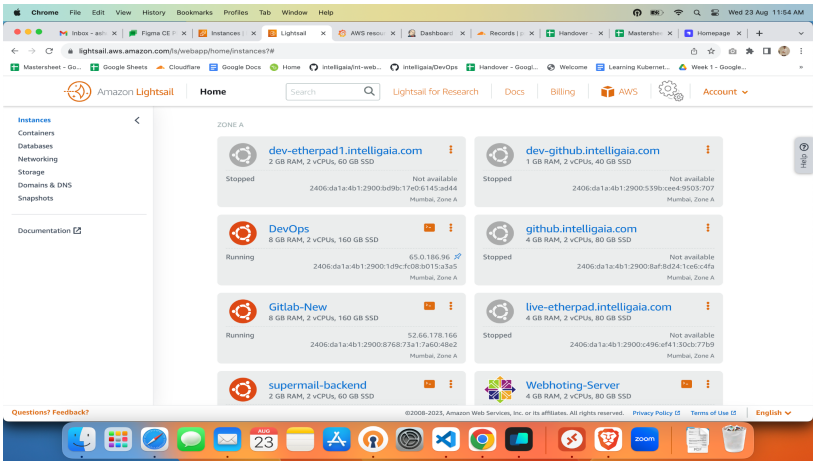
  We have create lightsail instance with the name DevOps in Mumbai regions with configuration

  RAM 8G

  2vCPUs

  160GB SSD storage

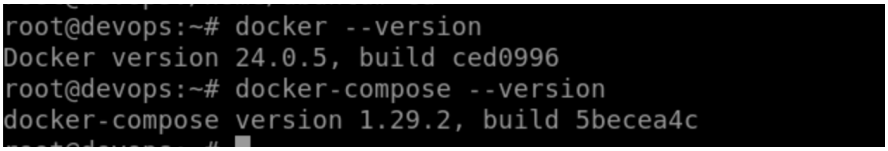
-



Here we have selected Ubuntu 20.04 LTS

```
root@devops:/home/ubuntu# cat /etc/os-release
NAME="Ubuntu"
VERSION="20.04 LTS (Focal Fossa)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 20.04 LTS"
VERSION_ID="20.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=focal
UBUNTU_CODENAME=focal
```

- Docker and Docker-compose installed on ubuntu 20.04

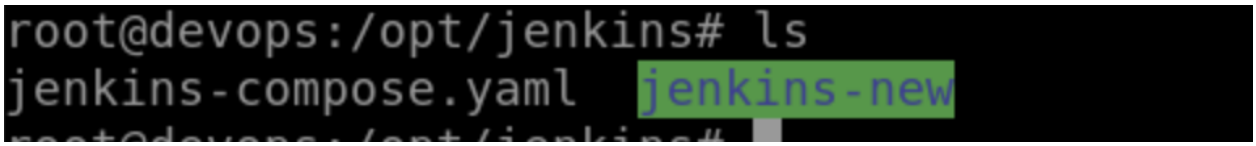


We can use below link to install docker and docker-compose on ubuntu

```
Unset
https://docs.docker.com/desktop/install/windows-install/

https://docs.docker.com/compose/install/
```

3. Installation

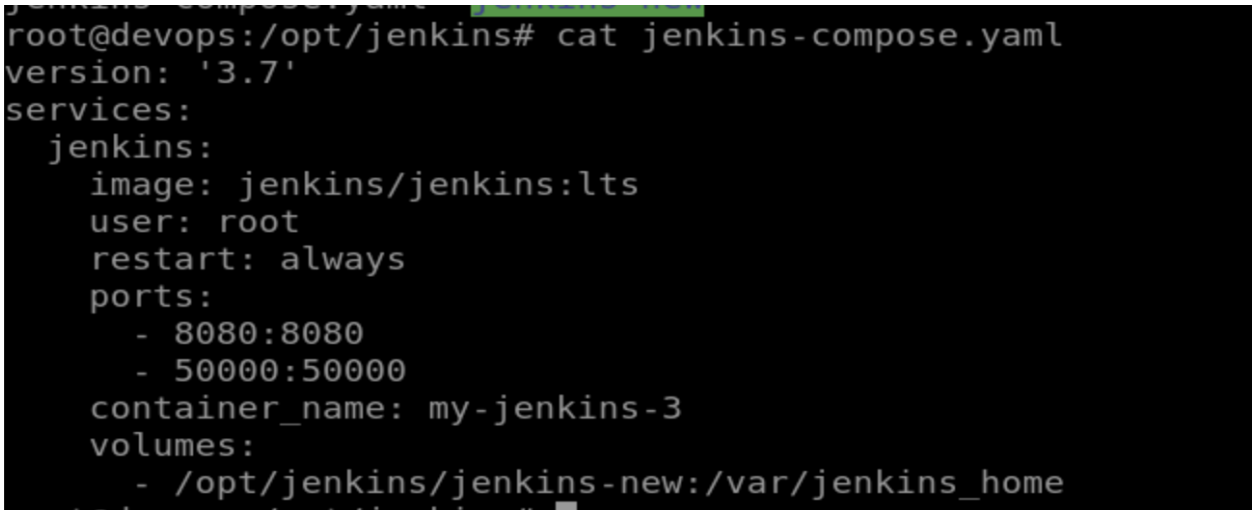
- For installation We need to take backup of **/var/lib/jenkins** which is mounted on efs **jenkins-worker-1**
- We have create one folder jenkins in **/opt** and whole data is from **/var/lib/jenkins** is kept in jenkins-new folder inside **/opt/jenkins**



We need to use below compose file to install and setup

Unset

```
version: '3.7'
services:
 jenkins:
  image: jenkins/jenkins:lts
  user: root
  restart: always
  ports:
   - 8080:8080
   - 50000:50000
  container_name: my-jenkins-3
  volumes:
   - /opt/jenkins/jenkins-new:/var/jenkins_home
```



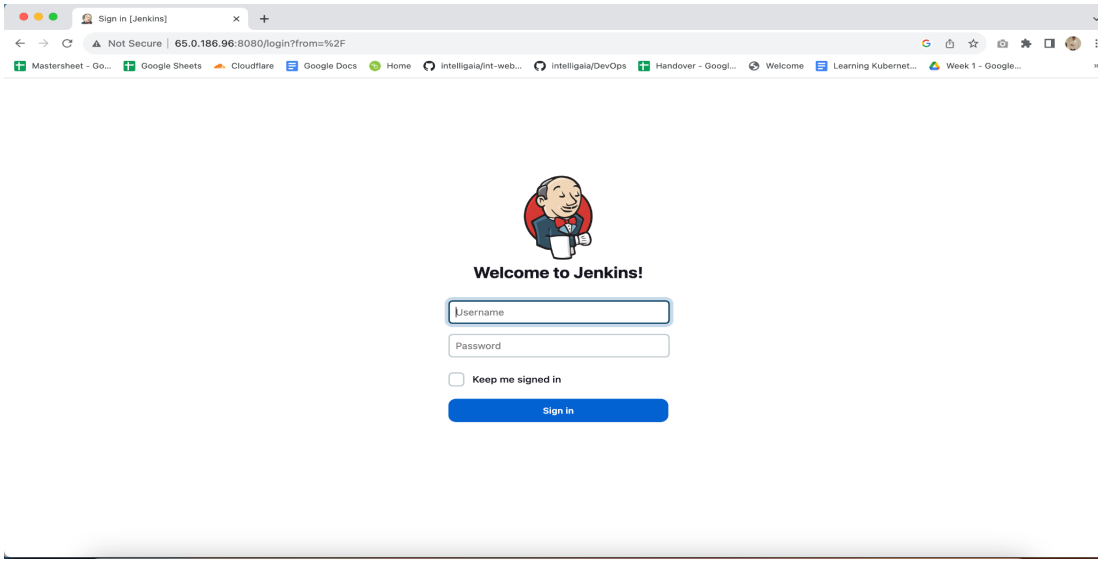
- After creating docker-compose.yaml then we can run below command to install and run jenkins

```
Unset
docker-compose -f /opt/jenkins/jenkins-compose.yaml up -d
```

```
a67f44684d26   jenkins/jenkins:lts                "/usr/bin/tini -- /u…"   6 days ago   Up 6
days   0.0.0.0:8080->8080/tcp, :::8080->8080/tcp, 0.0.0.0:50000->50000/tcp, :::50000->50000
/tcp   my-jenkins-3
```

- We need to open port 8080 in security group

4. Configuration

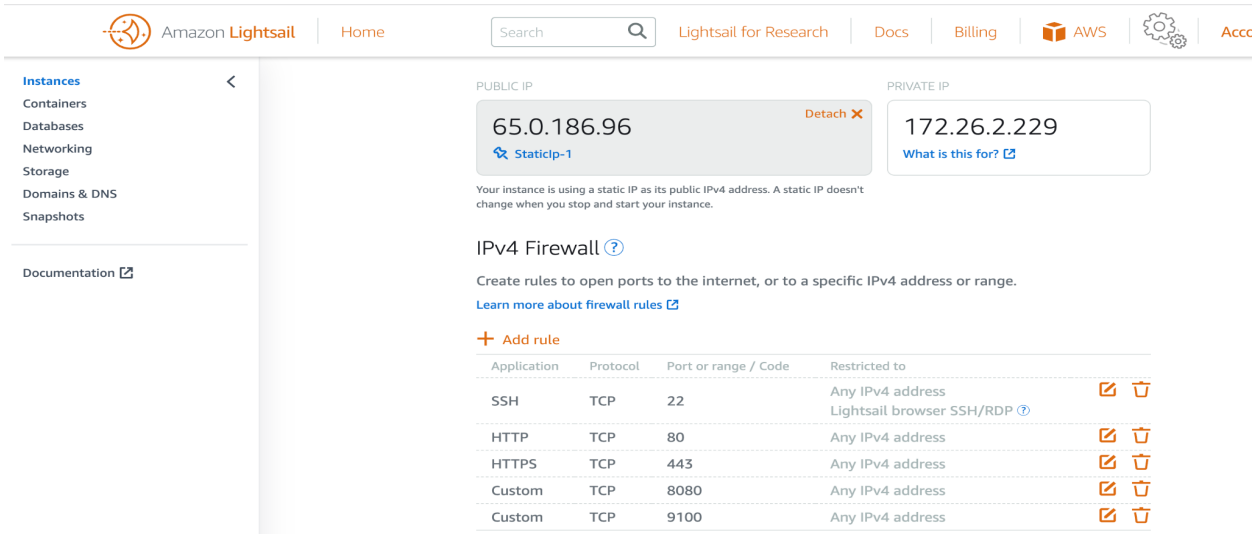


- At this point we're able to login to jenkins with active directory credentials because we had already integrated jenkins with sonarqube
- For accessing the jenkins we need to use reverse proxy to jenkins.intelligaia.com domain
- For that we need to install nginx
- Nginx can be installed using below command

```
Unset
apt update
apt install nginx -y
```

- After successfully installing create nginx configuration file we need to open 80 and 443 ports so that we could access nginx service

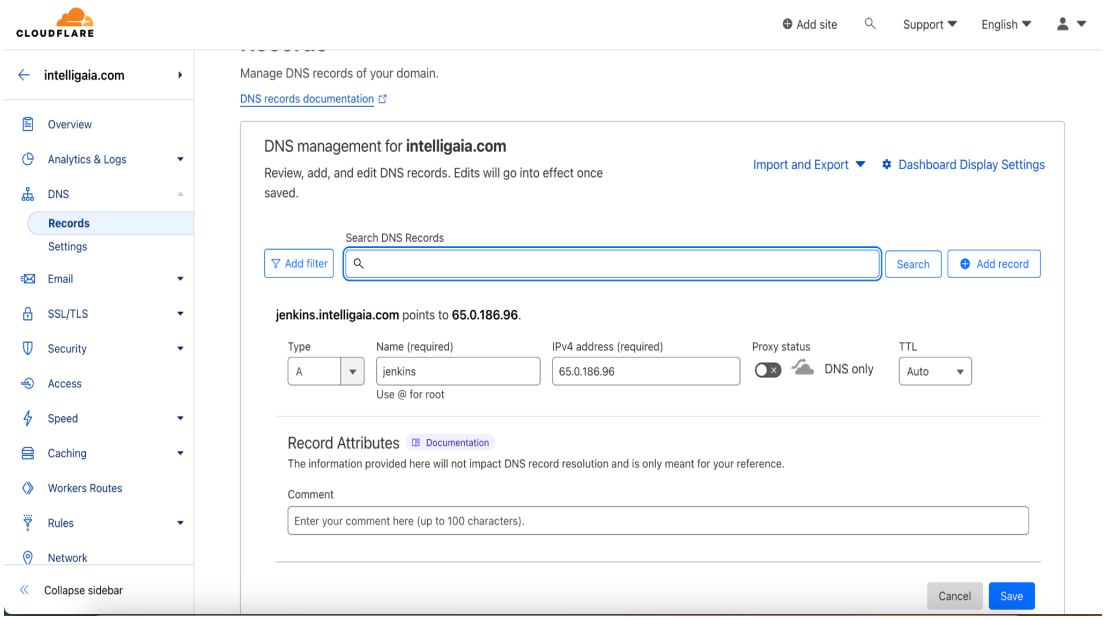
```
root@devops:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2023-08-18 05:10:21 UTC; 5 days ago
       Docs: man:nginx(8)
    Process: 193117 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (c>
    Process: 193138 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exite>
   Main PID: 193140 (nginx)
      Tasks: 3 (limit: 9422)
     Memory: 19.0M
     CGroup: /system.slice/nginx.service
             ├─193140 nginx: master process /usr/sbin/nginx -g daemon on; master_process o>
             ├─193199 nginx: worker process
             └─193200 nginx: worker process

Aug 18 05:10:21 devops systemd[1]: Starting A high performance web server and a reverse pr>
Aug 18 05:10:21 devops systemd[1]: Started A high performance web server and a reverse pro>
```



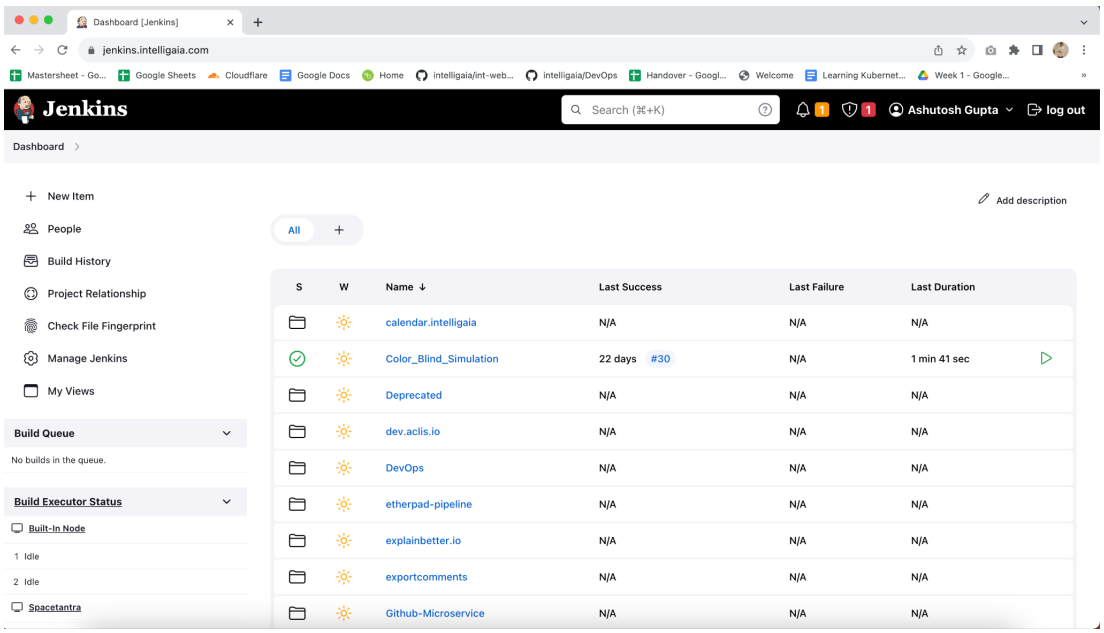
- cat /etc/nginx/sites-available/default

```
Unset
server {
  server_name jenkins.intelligaia.com;
  access_log off;

  location / {
    proxy_pass    http://127.0.0.1:8080;

    proxy_set_header  Host      $host;
    proxy_set_header  X-Real-IP    $remote_addr;
    proxy_set_header  X-Forwarded-For $proxy_add_x_forwarded_for;
```

```
    proxy_set_header  X-Forwarded-Proto http;
    proxy_max_temp_file_size 0;
    proxy_connect_timeout   150;
    proxy_send_timeout    100;
    proxy_read_timeout    100;
    proxy_buffer_size     8k;
    proxy_buffers       4 32k;
    proxy_busy_buffers_size  64k;
    proxy_temp_file_write_size 64k;

  }
  listen 443 ssl; # managed by Certbot
  ssl_certificate /etc/letsencrypt/live/jenkins.intelligaia.com/fullchain.pem; #
managed
by Certbot
  ssl_certificate_key /etc/letsencrypt/live/jenkins.intelligaia.com/privkey.pem;
# manage
d by Certbot
  include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
  ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot

}
server {
  if ($host = jenkins.intelligaia.com) {
    return 301 https://$host$request_uri;
  } # managed by Certbot

  listen 80;
  server_name jenkins.intelligaia.com;
  return 404; # managed by Certbot
}
```



- After adding dns rules to cloudflare we can access the URL jenkins.intelligaia.com



5. Integration

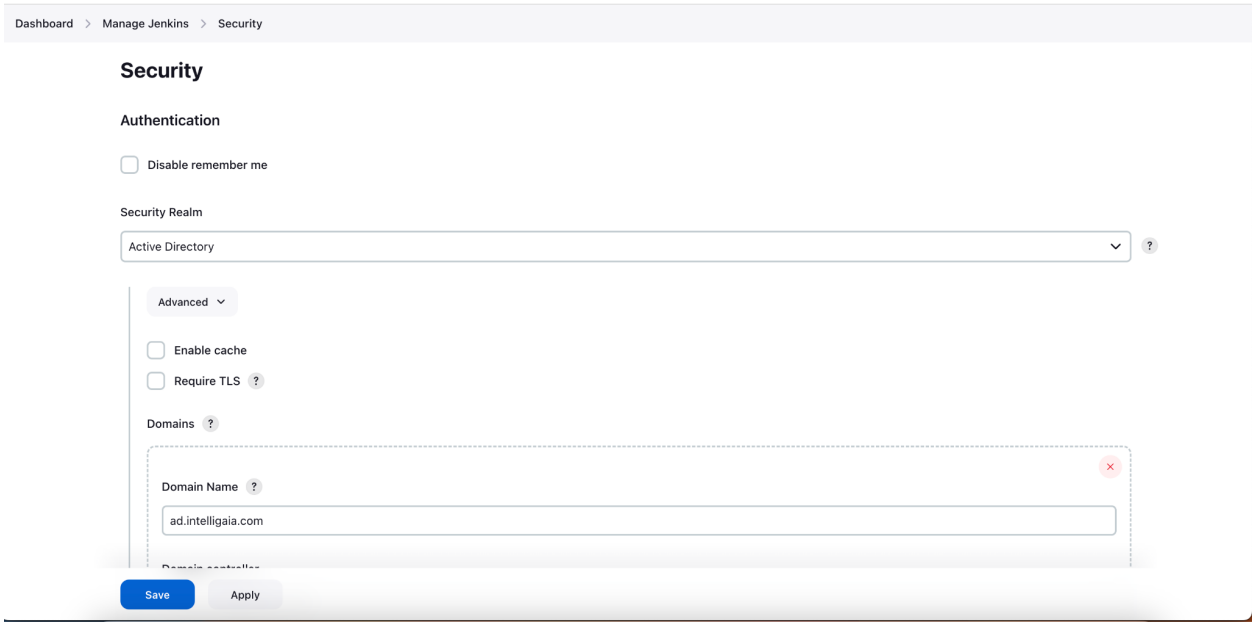
- Jenkins is integrated with active directory and sonarqube
- For integration with Jenkins we need to setup active directory server
- For setting up active directory please refer below link

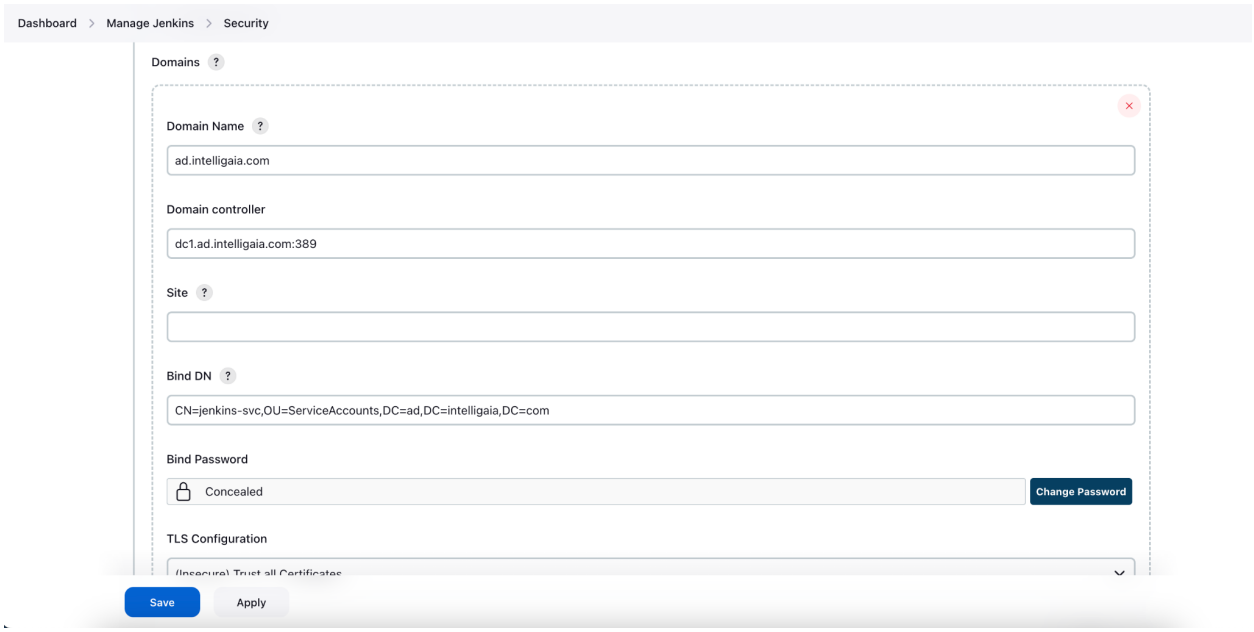
```
Unset
https://github.com/intelligaia/DevOps-Docs/tree/main/Active%20Directory
```

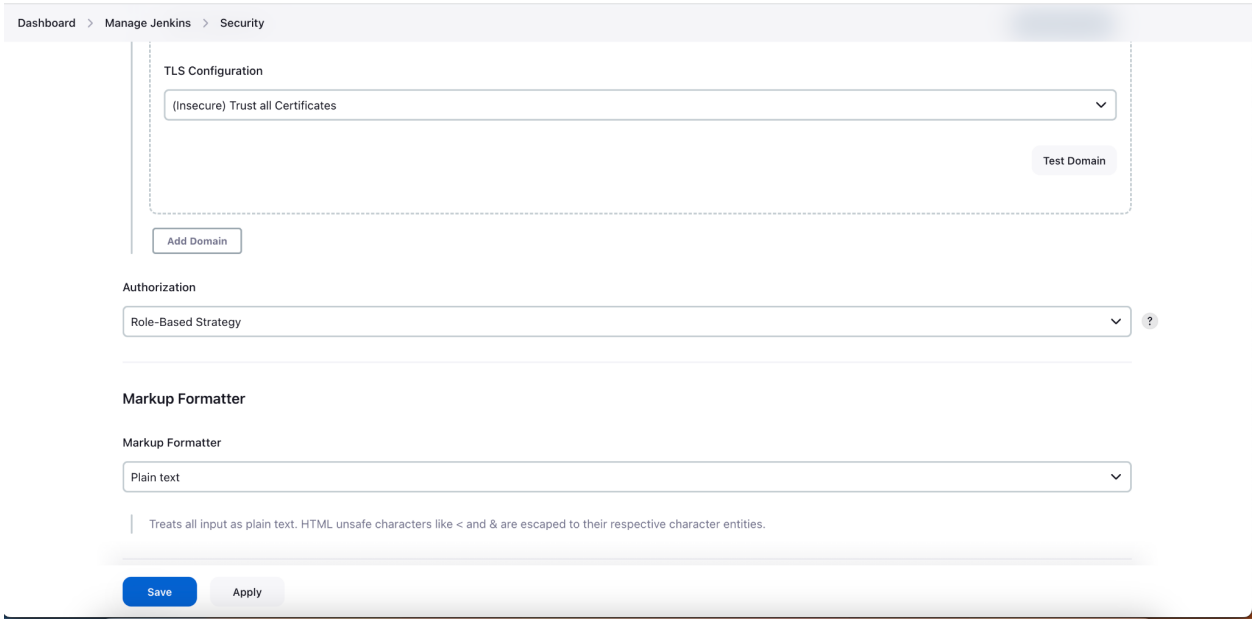
For creating new users or manage user you can use below link

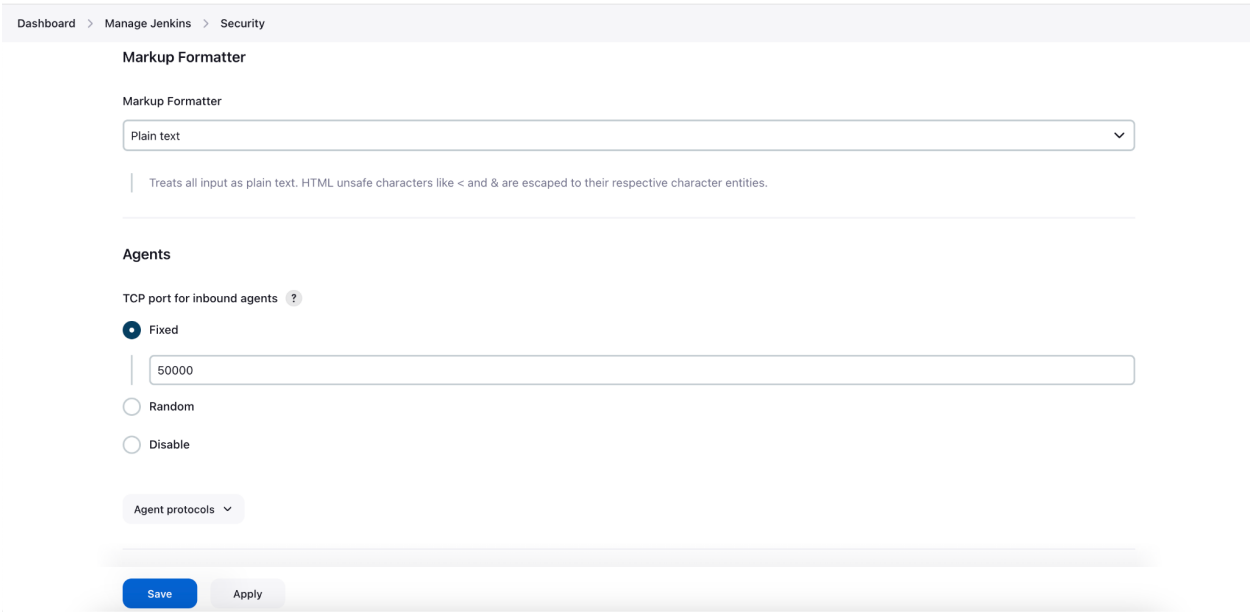
```
Unset
https://github.com/intelligaia/DevOps-Docs/tree/main/Active%20Directory/Add-user-in-Active-directory
```

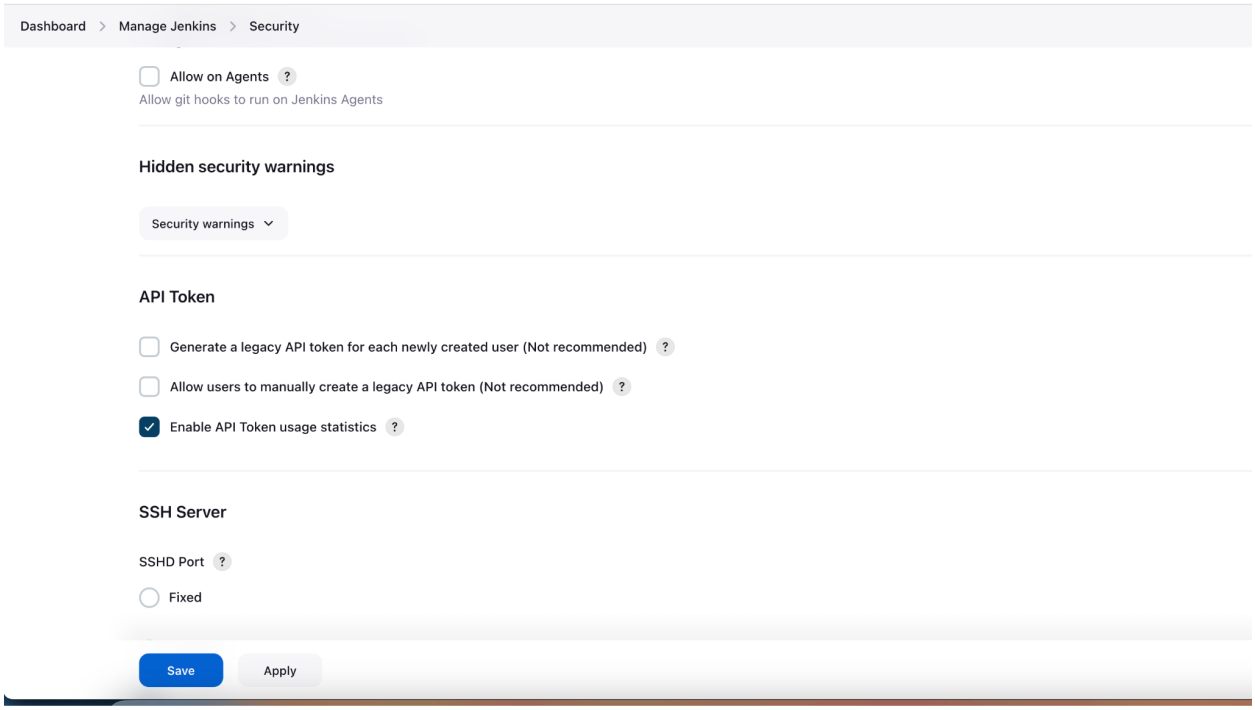
- For integration with Jenkins we need to install plugin for Active directory in jenkins
- For installing plugin you can navigate to Manage Jenkins > Plugins > Available Plugins
- Search for Active Directory plugin
- Click install without restart
- Navigate to manage jenkins > Security
- Configure the authentication with Active directory

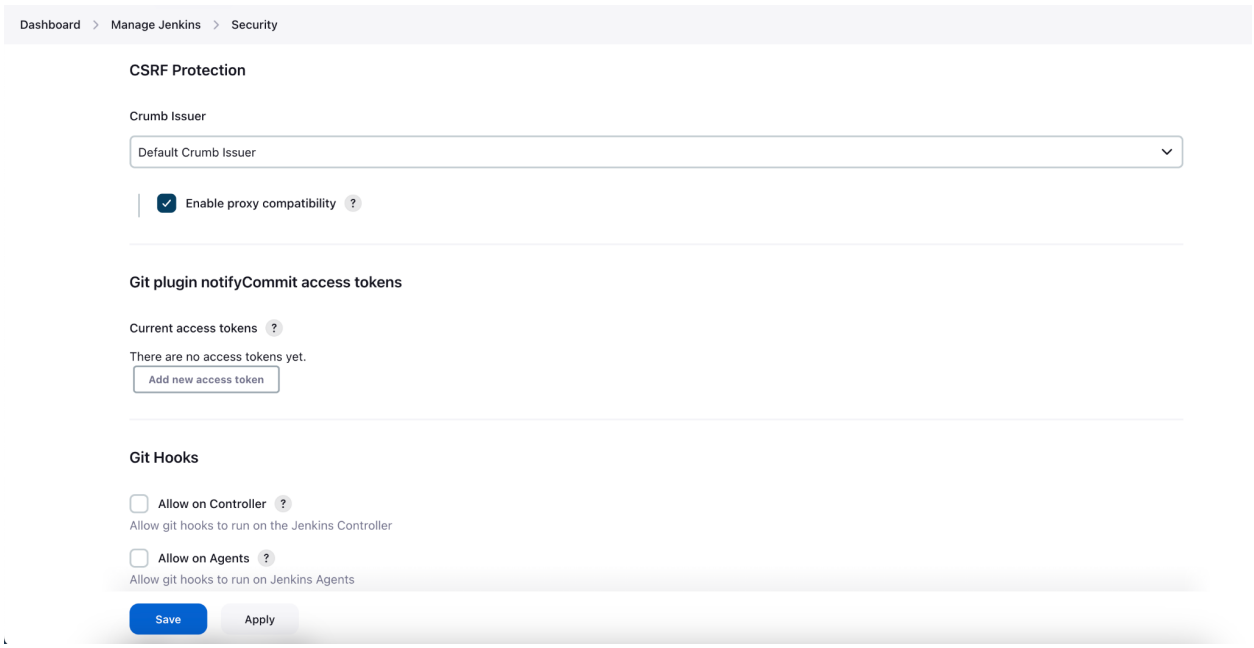


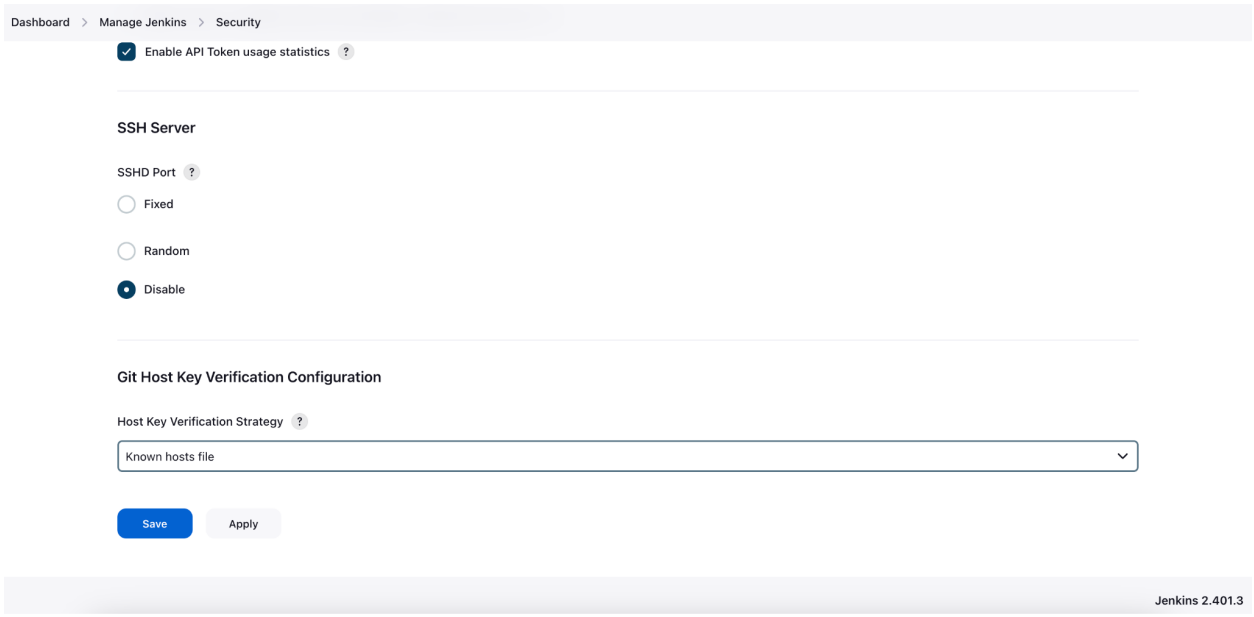

Dashboard > Manage Jenkins > Security
Domains
Domain Name
ad.intelligaia.com
Domain controller
dc1.ad.intelligaia.com:389
Site
Bind DN
CN=jenkins-svc,OU=ServiceAccounts,DC=ad,DC=intelligaia,DC=com
Bind Password
Concealed
Change Password
TLS Configuration
Save
Apply


Dashboard > Manage Jenkins > Security
TLS Configuration
(Insecure) Trust all Certificates
Test Domain
Add Domain
Authorization
Role-Based Strategy
Markup Formatter
Markup Formatter
Plain text
Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.
Save
Apply


Dashboard
Manage Jenkins
Security
Markup Formatter
Markup Formatter
Plain text
Treats all input as plain text. HTML unsafe characters like < and & are escaped to their respective character entities.
Agents
TCP port for inbound agents
Fixed
50000
Random
Disable
Agent protocols
Save
Apply


Dashboard > Manage Jenkins > Security
Allow on Agents
Allow git hooks to run on Jenkins Agents
Hidden security warnings
Security warnings
API Token
Generate a legacy API token for each newly created user (Not recommended)
Allow users to manually create a legacy API token (Not recommended)
Enable API Token usage statistics
SSH Server
SSHD Port
Fixed
Save
Apply

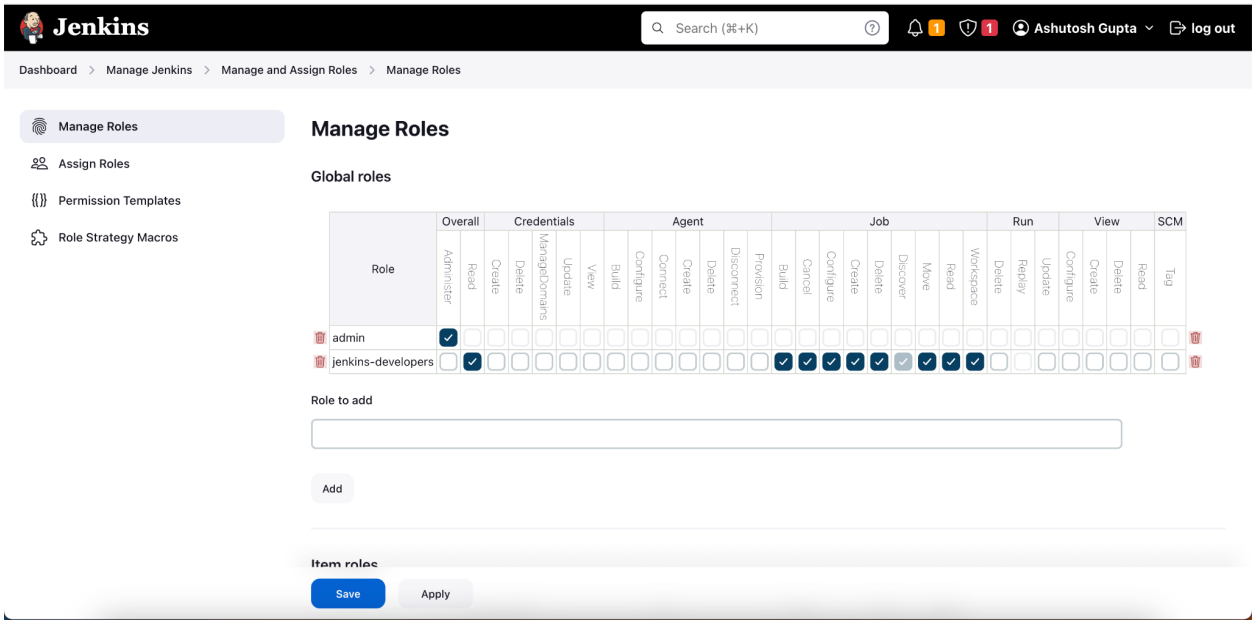

Dashboard > Manage Jenkins > Security
CSRF Protection
Crumb Issuer
Default Crumb Issuer
Enable proxy compatibility
Git plugin notifyCommit access tokens
Current access tokens
There are no access tokens yet.
Add new access token
Git Hooks
Allow on Controller
Allow git hooks to run on the Jenkins Controller
Allow on Agents
Allow git hooks to run on Jenkins Agents
Save
Apply



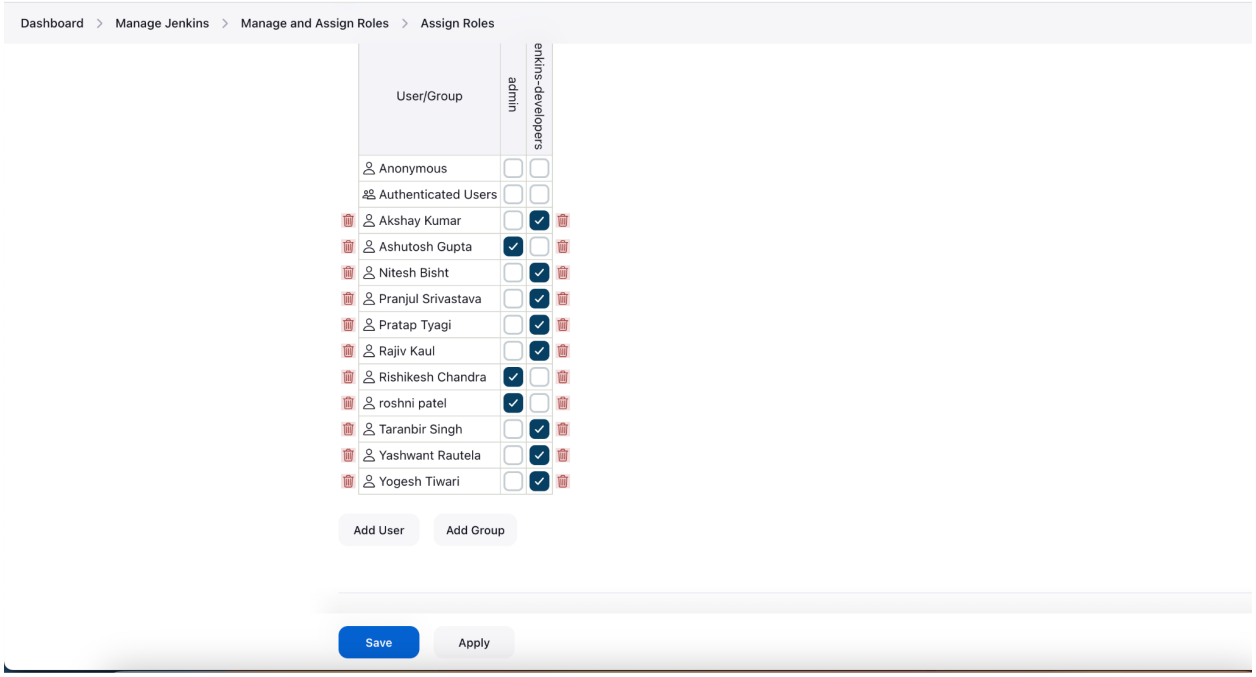
- You need to add necessary parameters such as
  Security Realm : Active Directory
  Domain Name : ad.intelligaia.com
  Domain controller : dc1.ad.intelligaia.com:389
  Bind DN : CN=jenkins-svc,OU=ServiceAccounts,DC=ad,DC=intelligaia,DC=com
  Bind Password : In+3lli6@ia2022

  Authorization : Role-Based Strategy (You might need to install plugin) (Plugin : Role-based Authorization Strategy)
  TCP port for inbound agents: 50000
  CSRF Protection : Default Crumb issuer

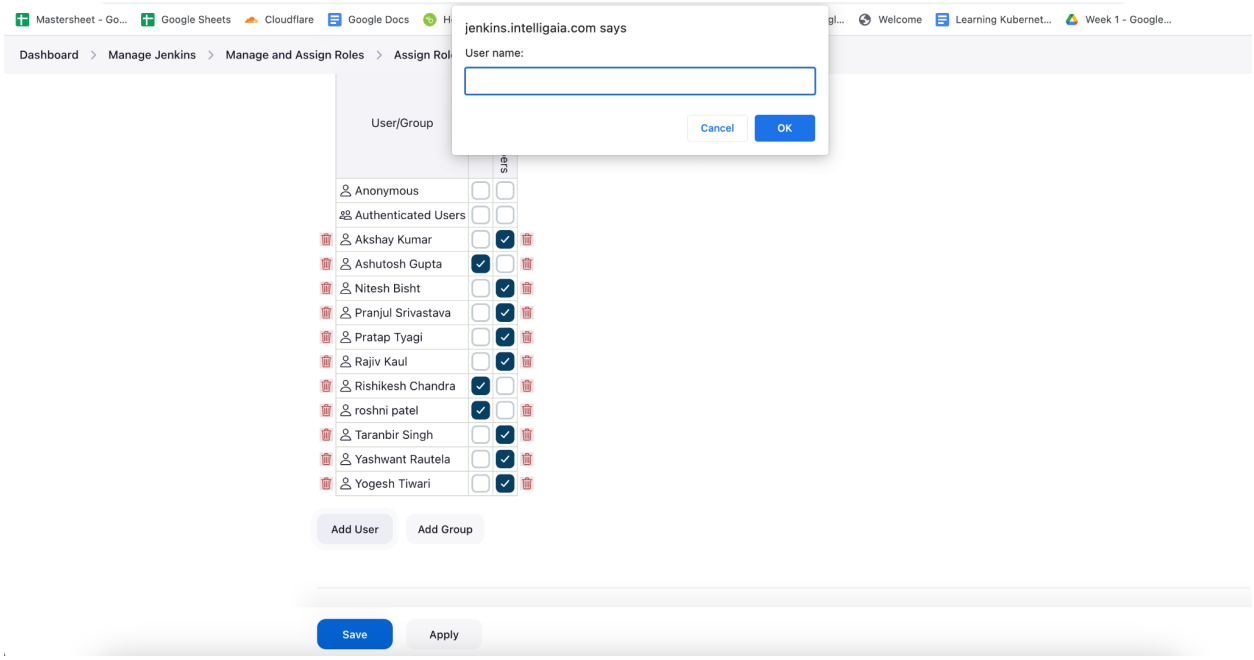
- Apply and **Save** all the configuration
- Now for creating new users in the active directory link is mentioned above.
- After adding the user into **jenkins-group** we can give access to the user to Jenkins
- Then the mandatory step is to Assign a role to the user's we have created Role and that role has been assigned to the users.
- You can navigate to Manage **Jenkins > Security > Manage and Assign Roles > manage Roles**
- Here we have created 2 Roles **Admin and Developers**, developers will have only access to the Jobs while admins have all access.



- You can navigate to **Manage Jenkins > Security > Manage and Assign Roles > Assign Roles**



- Created users in AD needs to be added into assign roles so we can click on Add user which will give you below pop



- You can add the username **same created in the Active directory** in the **pop up** box
- Then users will be listed and we can check the box to assign the roles click save.
- Now the added user can be authenticated via active directory.

6. Backup

- Jenkins Backup is must for production env as all the jobs are dependent on the jenkins if jenkins fails we can restore the jenkins if we have backup
- So we have created weekly backup and we're storing that to s3 storage
- We have created simple bash script so that we could upload the jenkins backup
- The script path is /root/jenkins-backup.sh
- We have scheduled this in the crontab of DevOps server in lightsail so that it could take backup of /opt/jenkins directory.
- The cronjob will run on saturday 4:05am.
- Below is the script which creates tar in /tmp directory and then aws s3 cp command to copy tar file to s3 storage

```
Unset
#!/bin/bash
rm -rf /tmp/Jenkins*.tar.gz | exit 0
tar -zcvf "/tmp/Jenkins-$(date '+%Y-%m-%d').tar.gz" /opt/jenkins
aws s3 cp /tmp/Jenkins*.tar.gz s3://jenkins-backup-pipeline
```