Installation GitLab Centos 7

### Step 1: Install and Configure Required Dependencies

**1.** First start by installing following necessary dependencies using [yum package manager](https://www.tecmint.com/20-linux-yum-yellowdog-updater-modified-commands-for-package-mangement/) as shown.

# yum install curl policycoreutils-python openssh-server

**2.** Next, install **Postfix** service to send notification emails, and enable it to start at system boot, then check if its up and running using following commands.

# yum install postfix

# systemctl start postfix

# systemctl enable postfix

# systemctl status postfix

During Postfix installation a configuration window may appear. Select ‘**Internet Site**‘ and use your server’s external DNS for ‘**mail name**‘ and press enter. If extra screens appear, continue to press enter to accept the defaults.

### Step 2: Add GitLab Repository and Install Package

**3.** Now add the **GitLab** package YUM repository to your system by running a following script.

$ curl https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.rpm.sh | sudo bash

**4.** Next, install the GitLab Community Edition package using following command and make sure to change ‘**http://gitlab.tecmint.com**‘ to the URL at which you want to access your GitLab instance from a web browser.

# EXTERNAL\_URL="http://gitlab.tecmint.com" yum install -y gitlab-ce

**Note**: If you want to change your main URL, you can configure it in the GitLab main configuration file **/etc/gitlab/gitlab.rb** in the **external\_url** section. Once changed, don’t forget to reconfigure gitlab to apply the recent changes in the configuration file using following command.

# gitlab-ctl reconfigure

**5.** If you have a system firewall enabled, you need to open port **80** (**HTTP**) and **443** (**HTTPS**) to allow connections in the system firewall.

# firewall-cmd --permanent --add-service=80/tcp

# firewall-cmd --permanent --add-service=443/tcp

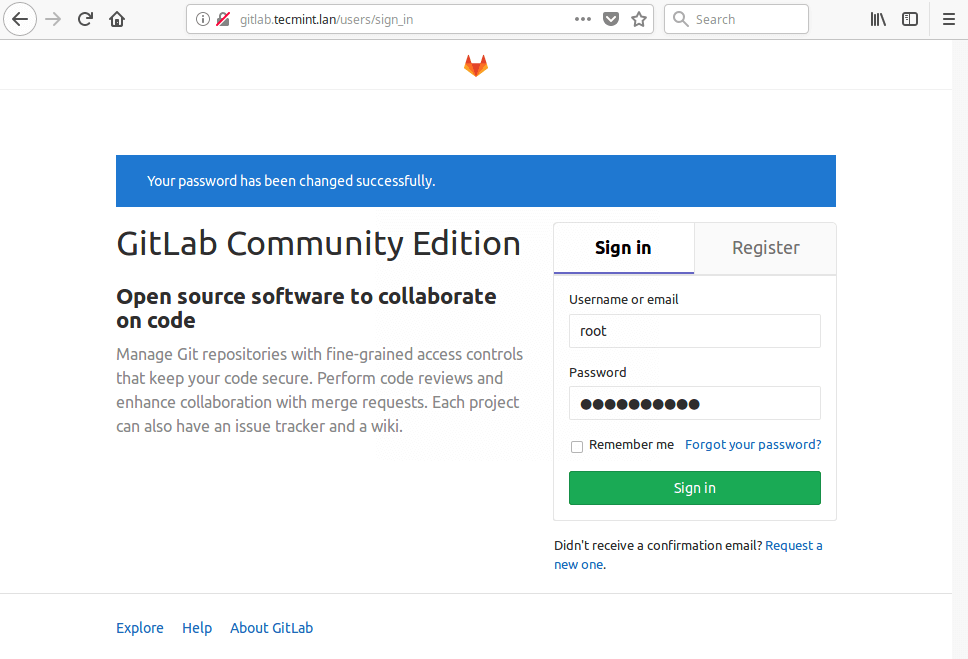
# systemctl reload firewalld

### Step 3: Perform Initial Gitlab Setup

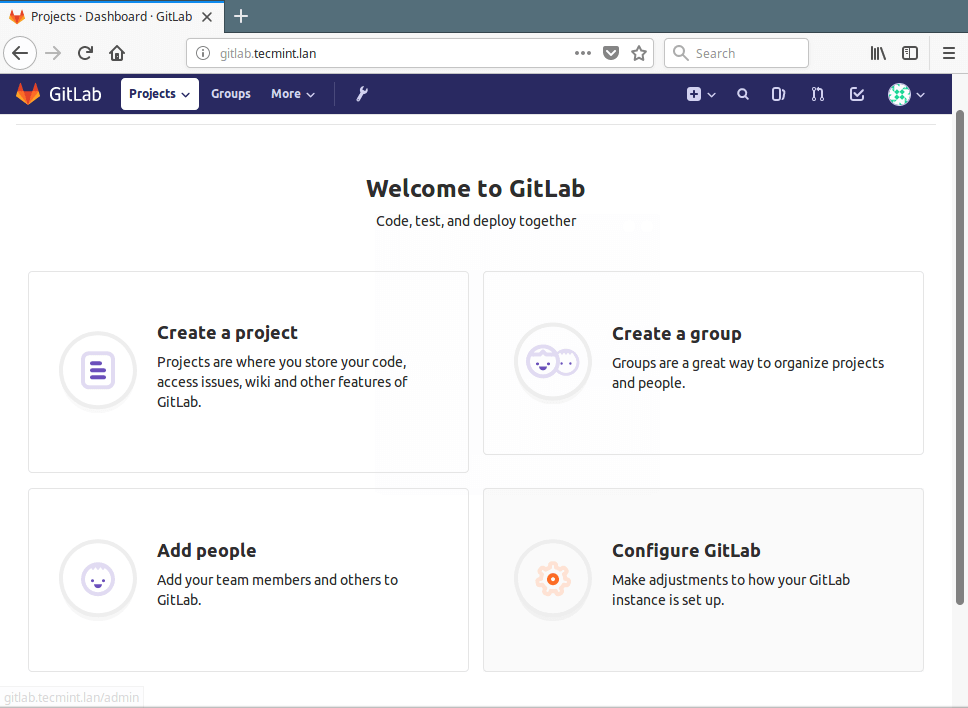
**6.** Now, open a web browser and access your gitlab instance using the following URL you set during installation.

http://gitlab.tecmint.com

**7.** On your first visit, you’ll be redirected to a password reset screen, create a new password for your new admin account and click “**Change your password**”. Once you set, it will be redirected back to the login screen and login with username **root** and the password you set.



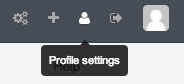
**8.** A successful login should take you into the admin user account as shown in the screenshot. From, here, you can create an object, create a group, add people or configure your gitlab instance as you wish. You can also edit your user profile, configure your email, and add SSH keys to your gitlab instance, and more.



## Step 4: How To Configure Your User Profile

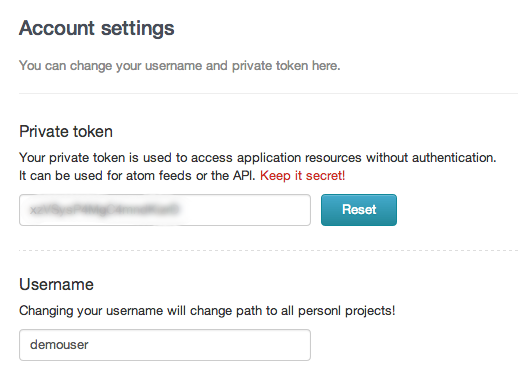
Before you get started using GitLab to handle your projects, it is a good idea to get your profile set up correctly. This will not only help your teammates identify you, it is also where you can manage your interface and the way that you connect to projects.

Begin by clicking on the "Profile settings" button in the upper-right corner:



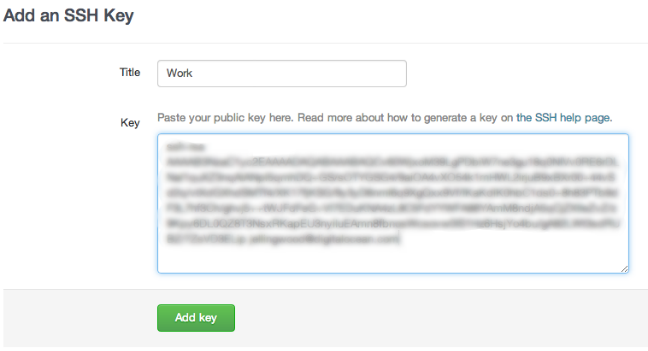
The initial screen will allow you to change the name and email associated with your account. You can also link your social media accounts, write a short bio, and upload a picture of your choice.

If you would also like to change your username, click the "Account" tab at the top. Here you can adjust the username that your repositories will be stored under. You will also have access to your access token:

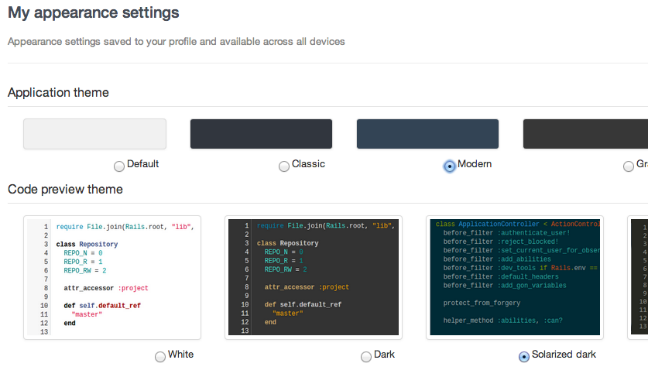


The other item that you definitely want to look into is the "SSH Keys" tab. In this tab, click the "Add SSH Key" button.

Here, you can add the SSH key that you will use to communicate with the server through git. Multiple keys can be added for each user. Add your SSH key and click "Add Key":

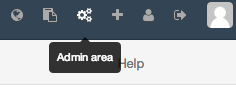


This is also a good opportunity to change the application and code preview themes, if you so desire. Click on the "Design" tab. You can choose from five application themes and four popular code preview themes:



## Step 5: How To Manage Users and Groups

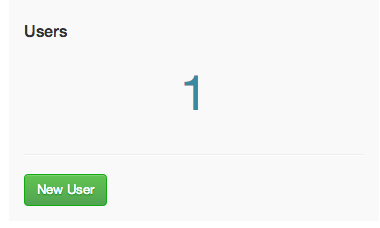
## GitLab gives you the tools to manage projects, users, and groups from one screen. Access it by clicking the "Admin area" button in the top-right corner:



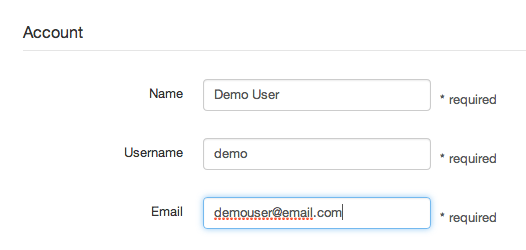
The interface is divided into six sections: projects, latest projects, users, latest users, groups, and stats. The main points of contact that we will be using are the projects, users, and groups.

### Step 6: Create a New User

## Currently, there is only a single user and no projects of groups. Let's start by creating a demonstration user that we can use later. Click on the "New User" button in the middle column:



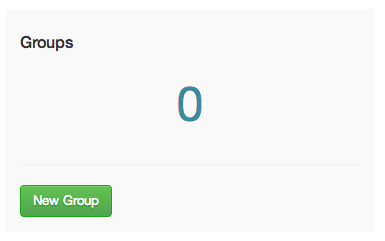
We will need to fill out the regular contact details. When you are done, click the "Create User" button at the bottom:



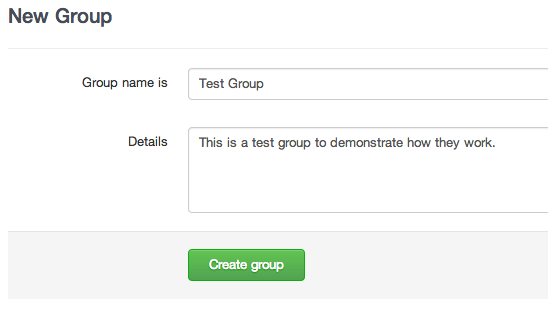
You will be taken to the new user's account page. An email with an initial password will be sent to the account email.

### Step 7: Create a New Group

### Let's add a new group now. Click on the "Admin area" button again in the top-right corner. Click the "New Group" button in the right column:



The only thing you need to do to create a group is fill out a name and description. Click the "Create group" button:

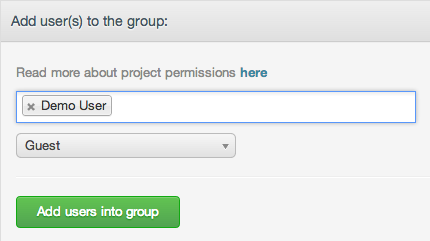


The group will be created and you will automatically be added to the group as an owner. Let's add our demonstration account to this group as well.

Search for the second username in the box titled "Add user(s) to the group:". You will need to choose an access level for the user as well. For an explanation of the different permission levels available on your GitLab, go here: **your\_domain.com/help/permissions**

This is also available as a link in the box under "Read more about project permissions here".

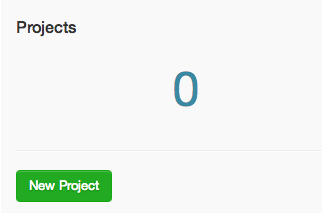
For now, it doesn't matter which level you choose. Click "Add users into group" to add their access:



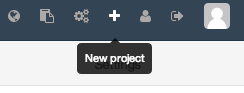
If you would like to change a user's permission level, you can add the user again with the new permissions. It will update accordingly.

## Step 8: Initialize a Project

Now, we can initialize a project. Return to the admin dashboard by clicking on the "Admin area" button as before. Click on the "New Project" button on the left side:

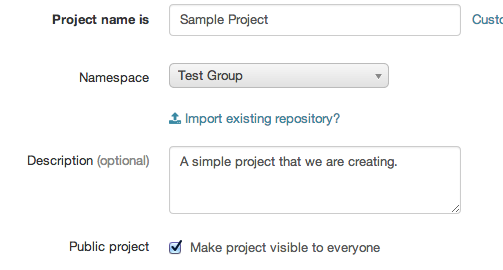


You can also get to this area by clicking on the "New project" button in the top navigation bar:



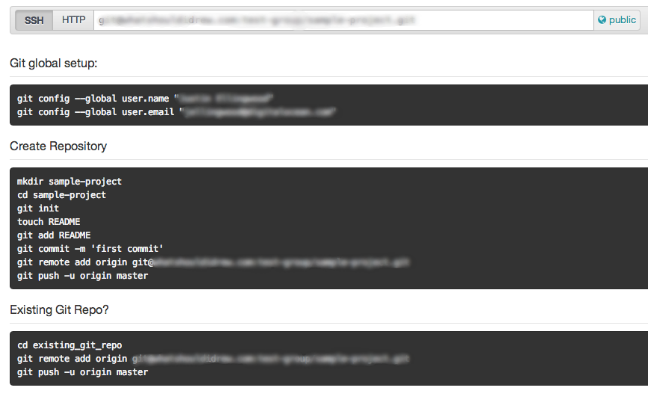
Pick a name and a namespace. A namespace is who will own the project. We will give the group we created ownership of our project. You should also fill out a description and choose whether the repo should be public.

It is also possible to import a repository from another site. You can click the "Import existing repository" link to get access to this functionality.



You will be taken to a page that will show you how to clone the project to your local computer, or push a local project to your new repository on GitLab.

You will be given a repository link that you can use to clone the repository and share with group members (or other people if it is public).



If you follow the instructions, you should be able to push your first commit to your GitLab repository.

## Step 9 : Manage Projects with GitLab

We will add a sample project to our GitLab repository so we can see some of the features available when a project has a history and many commits.

Go through the procedure to create a new project. Call it "Rails". This time, click "Import existing repository" link. We will be using Michael Hartl's Ruby on Rails tutorial sample application.

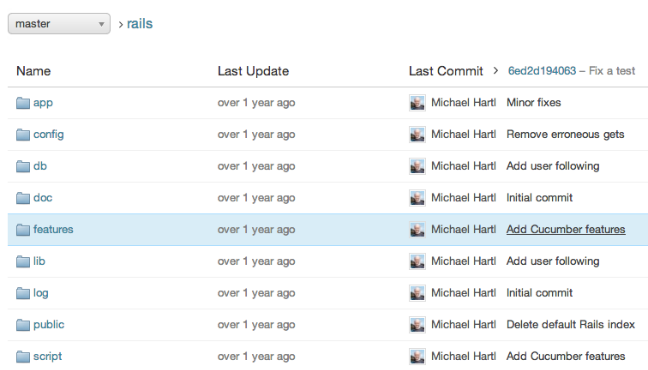
In the Import existing repo field, type:

https://github.com/mhartl/sample\_app.git

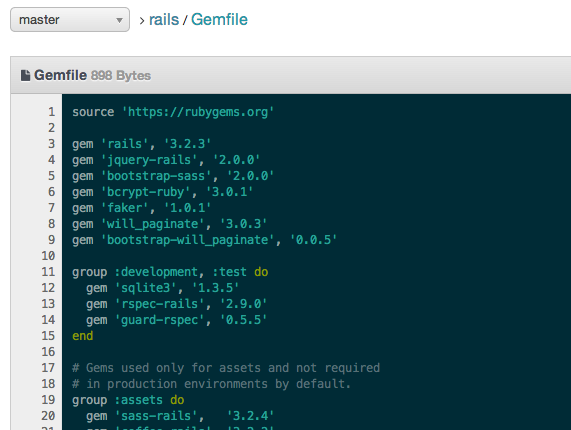
Click "Create project".

### Step 10 : View Project Code

You should be taken to the project's landing page. Click on the "Files" tab to see the actual repository files:



Click on the "Gemfile" file. As you can see, there is great syntax highlighting included for recognized languages:

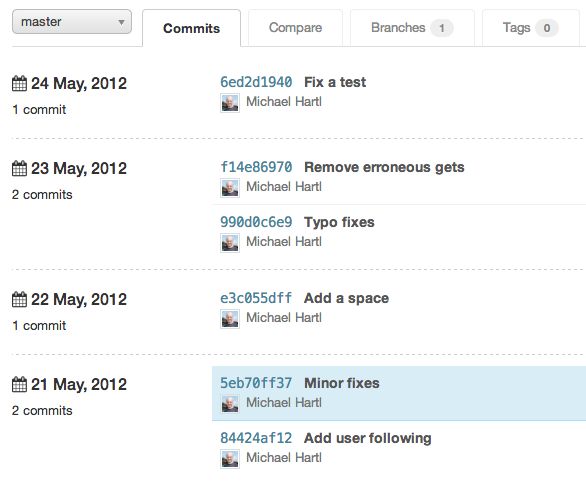


If your project has multiple branches, you can change the branch view by clicking on the master drop-down and switch to a different branch:

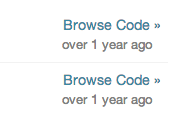


### Step 11 : Review Commits

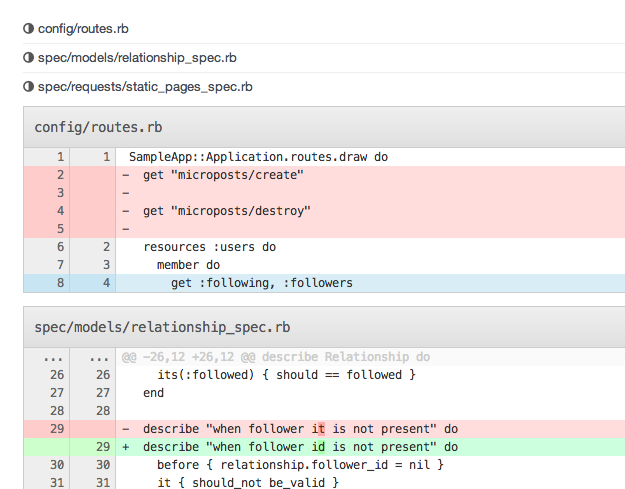
Click on the "Commits" tab to see the list of commits for the current branch:



You can click on any commit to see the diff that was produced by the commit. If you would like to see the entire project at any commit point, click the "Browse Code" link associated with that commit:



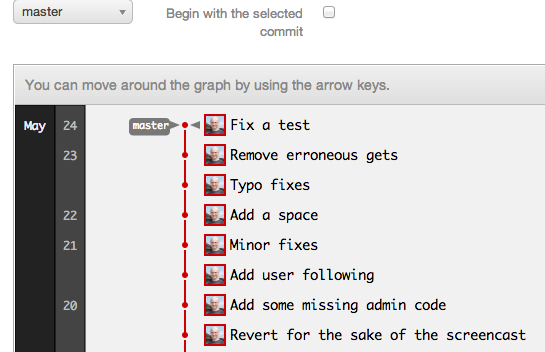
To see the diff between two commits, you can click the "compare" tab at the top. If we type the name of one of the commits in the left-hand box (in this case, we'll use e3c055dff), we can get a complete diff between the two:



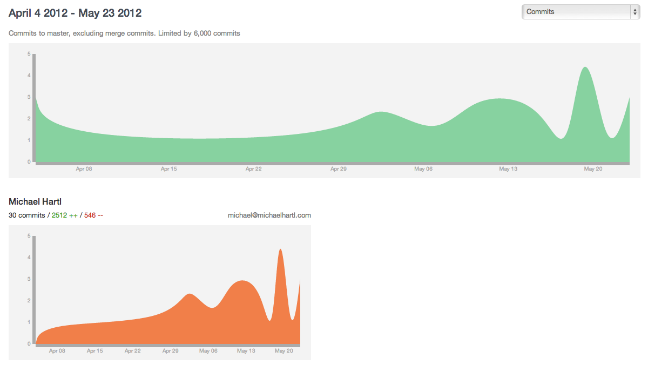
### Step 12 : Graphic Representations of your Project

You can view your project graphically in two different ways.

First, if you click on the "Network" tab, you can see a commit tree of your project. This allows you to see branches, merges, and commits. It will use the user's picture next to each commit as well, for easy identification:



The other way to get a graphical picture of your project is with the "Graphs" tab. You will see a large graph with the number of the project's commits vs time displayed. Below it will be a matching graph for each contributor:



You can select a portion of the project graph and the user's graphs will adjust accordingly. You can also choose to display additions or deletions instead of commits through the drop-down menu.

## Conclusion

There are other tools included within GitLab that you should check out if you plan on using it with a team. This includes an embedded wiki for each project, an issue tracking system, and merge requests. For sharing small pieces of code outside of any specific project, GitLab also has snippet support.

For most teams, a properly configured GitLab will be more than adequate to handle projects. It provides simple user control and easy project creation and sharing. Explore the interface to see if it could be an easy solution for your project.

Installation Nexus Centos 7

This tutorial shows you how to install Nexus on your computer.  No prior knowledge is expected, and instead of giving instructions such as “create a user”, I will list exactly how to type each command.

The server which I am using for this tutorial is named “quartz”.  Anywhere you see this name, you will need to change it to match your hostname.

Type the following commands into the terminal.  Everything after the # is a comment for your better understanding, and does not need to be entered with the command, I have colored the comments green for clarity.

wget <http://download.sonatype.com/nexus/oss/nexus-2.11.2-06-bundle.tar.gz> # download the nexus tarball.

sudo cp nexus-2.11.2.-06-bundle.tar.gz /usr/local # copy the tarball to the /usr/local directory

cd /usr/local # change directory to the location of the tarball

sudo tar xvzf nexus-2.11.2-06-bundle.tar.gz # extract the tarball

sudo ln -s nexus-2.11.2-06 nexus # create a symbolic link “nexus” to the versioned directory

sudo adduser nexus # Create a user that will be used to run Nexus

sudo sh –c ‘echo “export NEXUS\_HOME=/usr/local/nexus” >> /etc/profile.d/nexus.sh’

sudo cp NEXUS\_HOME/bin/nexus  /etc/init.d/nexus

sudo chmod 755 /etc/init.d/nexus # allow everyone to read and execute this file but only the owner can write to the file.

sudo chown root /etc/init.d/nexus # change the owner of startup script to be root

sudo chown –h nexus:nexus NEXUS\_HOME # change the owner of the nexus home directory including symbolic links to be the nexus user we created above

sudo chown -RH nexus:nexus NEXUS\_HOME # recursively change the owner of the nexus home directory to be the nexus user we created above

sudo mkdir /etc/local/sonatype-work # create a directory for Sonatype products to use for working files

sudo mkdir /etc/local/sonatype-work/nexus # create a directory for Nexus to use for working files

sudo chown –R nexus:nexus /etc/local/sonatype-work # make the nexus user created above the owner of this directory

sudo nano /etc/init.d/nexus # edit this file to point to the directory and user created above.

NEXUS\_HOME=”/usr/local/nexus”

RUN\_AS\_USER=”nexus”

cd /etc/init.d # change to the directory which controls starting Linux subsystems on boot

sudo chkconfig –add nexus # add nexus as a system service

sudo chkconfig –levels 345 nexus on # start the nexus service on boot for run levels 3-5 (Multi-User Mode with Networking up to X11)

sudo service nexus start # start the Nexus service

sudo firewall-cmd –permanent –zone=public –add-port=8081/tcp # permanently open the port 8081 in the CentOS firewall

sudo firewall-cmd –reload # reload the firewall settings to make them active

You can now open a browser and go to <http://quartz:8081/nexus> to explore Nexus.  The default username is deployment and default password is deployment123. Hopefully this tutorial made it simple and easy to install Nexus. Stay tuned for more installation tutorials.

Installation Jenkins Centos 7

[Jenkins](https://jenkins.io/) is an open source, Java-based automation server that offers an easy way to set up a continuous

integration and continuous delivery (CI/CD) pipeline.

Continuous integration (CI) is a DevOps practice in which team members regularly commits their code changes to the version control repository, after which automated builds and tests are run. Continuous delivery (CD) is a series of practices where code changes are automatically built, tested and deployed to production.

This tutorial will walk you through the steps of installing Jenkins on a CentOS 7 system using the official Jenkins repository.

## [Prerequisites](https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#prerequisites)

Before continuing with this tutorial, make sure you are logged in as a [user with sudo privileges](https://linuxize.com/post/create-a-sudo-user-on-centos/).

## [Installing Jenkins](https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#installing-jenkins)

To install Jenkins on your CentOS system, follow the steps below:

1. Jenkins is a Java application, so the first step is to install Java. Run the following command to install the OpenJDK 8 package:

sudo yum install java-1.8.0-openjdk-devel

The current version of Jenkins does not support Java 10 (and Java 11) yet. If you have multiple versions of Java installed on your machine [make sure Java 8 is the default Java version](https://linuxize.com/post/install-java-on-centos-7/#set-the-default-version).

1. The next step is to enable the Jenkins repository. To do that, import the GPG key using the following [curl](https://linuxize.com/post/curl-command-examples/)command:

curl --silent --location http://pkg.jenkins-ci.org/redhat-stable/jenkins.repo | sudo tee /etc/yum.repos.d/jenkins.repo

And add the repository to your system with:

sudo rpm --import https://jenkins-ci.org/redhat/jenkins-ci.org.key

1. Once the repository is enabled, install the latest stable version of Jenkins by typing:

sudo yum install jenkins

After the installation process is completed, start the Jenkins service with:

sudo systemctl start jenkins

To check whether it started successfully run:

systemctl status jenkins

You should see something similar to this:

● jenkins.service - LSB: Jenkins Automation Server

Loaded: loaded (/etc/rc.d/init.d/jenkins; bad; vendor preset: disabled)

Active: active (running) since Thu 2018-09-20 14:58:21 UTC; 15s ago

Docs: man:systemd-sysv-generator(8)

Process: 2367 ExecStart=/etc/rc.d/init.d/jenkins start (code=exited, status=0/SUCCESS)

CGroup: /system.slice/jenkins.service

Finally enable the Jenkins service to start on system boot.

sudo systemctl enable jenkins

jenkins.service is not a native service, redirecting to /sbin/chkconfig.

Executing /sbin/chkconfig jenkins on

## [Adjust the Firewall](https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#adjust-the-firewall)

If you are installing Jenkins on a remote CentOS server that is [protected by a firewall](https://linuxize.com/post/how-to-setup-a-firewall-with-firewalld-on-centos-7/) you need to port 8080.

Use the following commands to open the necessary port:

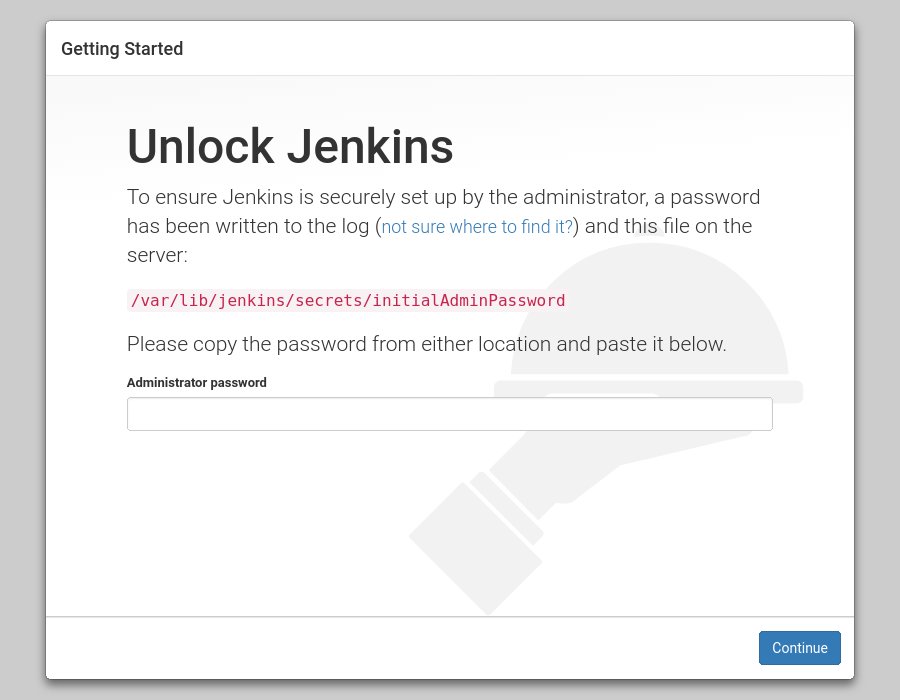
sudo firewall-cmd --permanent --zone=public --add-port=8080/tcpsudo firewall-cmd --reload

## [Setting Up Jenkins](https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#setting-up-jenkins)

To setup your new Jenkins installation, open your browser and type your domain or IP address followed by port 8080:

http://your\_ip\_or\_domain:8080

A screen similar to the following will appear, prompting you to enter the Administrator password that is created during the installation:



Use the following command to print the password on your terminal:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

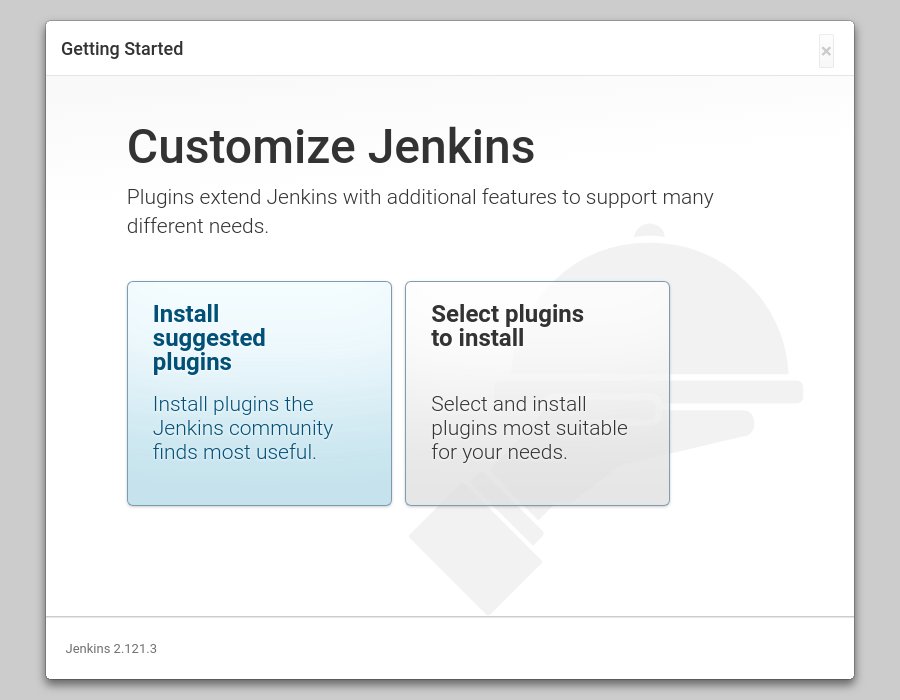
Copy

You should see a 32-character long alphanumeric password as shown below:

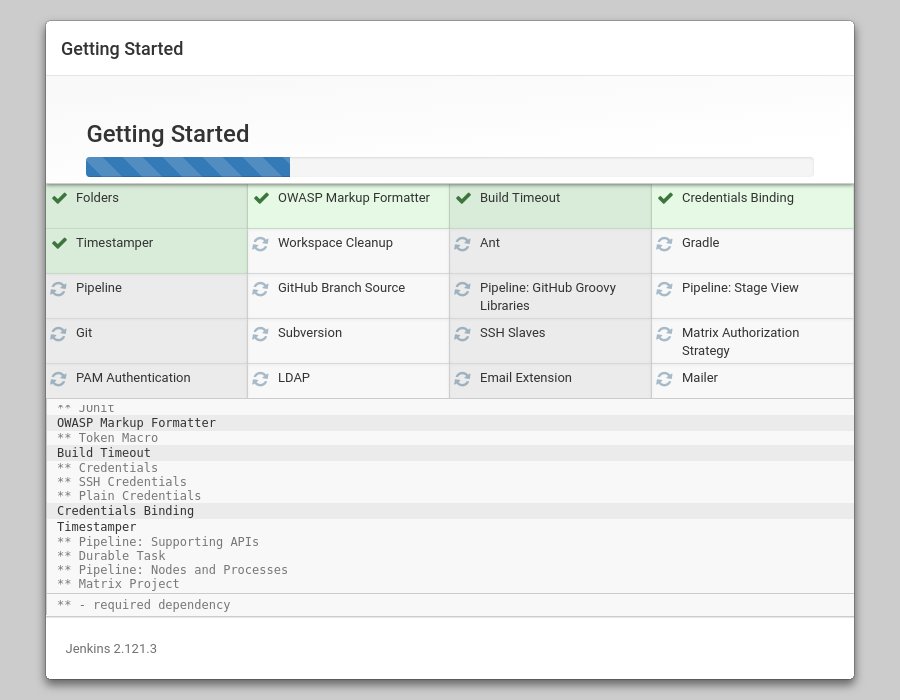
2115173b548f4e99a203ee99a8732a32

Copy

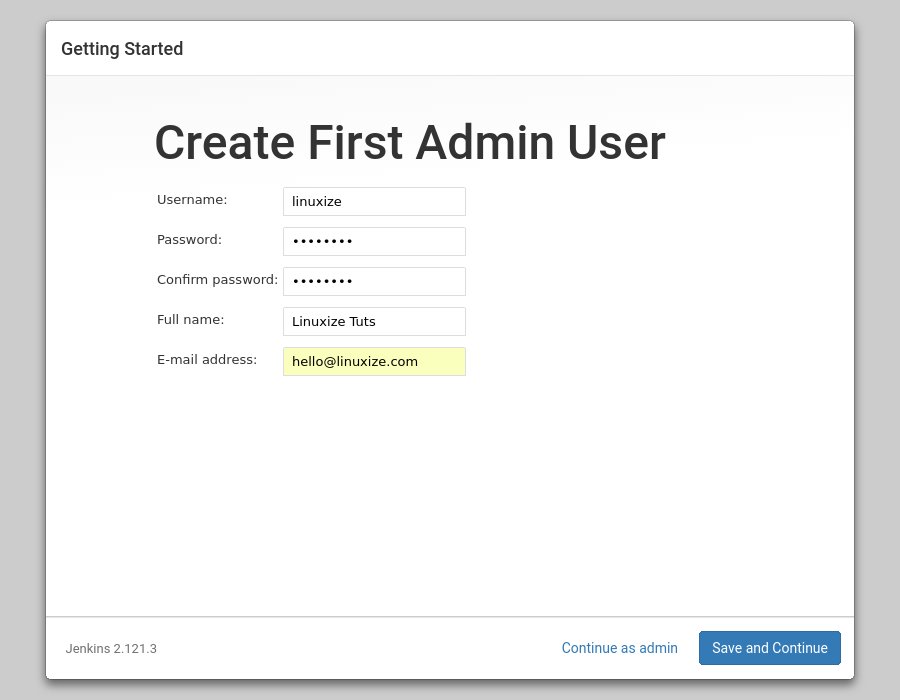
Copy the password from your terminal, paste it into the Administrator password field and click Continue.



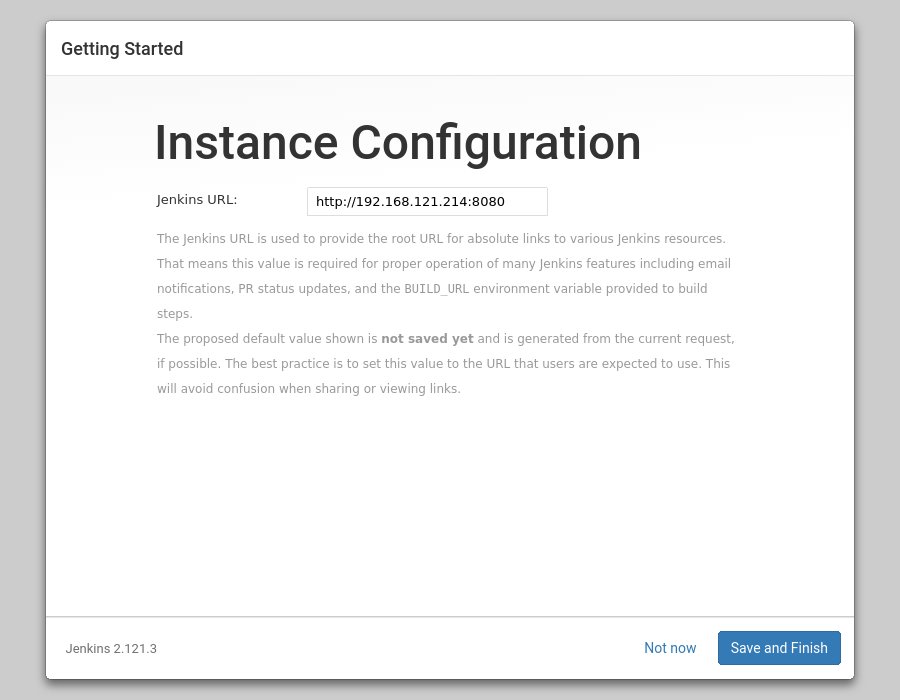
On the next screen you will be asked whether you want to install the suggested plugins or to select specific plugins. Click on the Install suggested plugins box, and the installation process will start immediately.



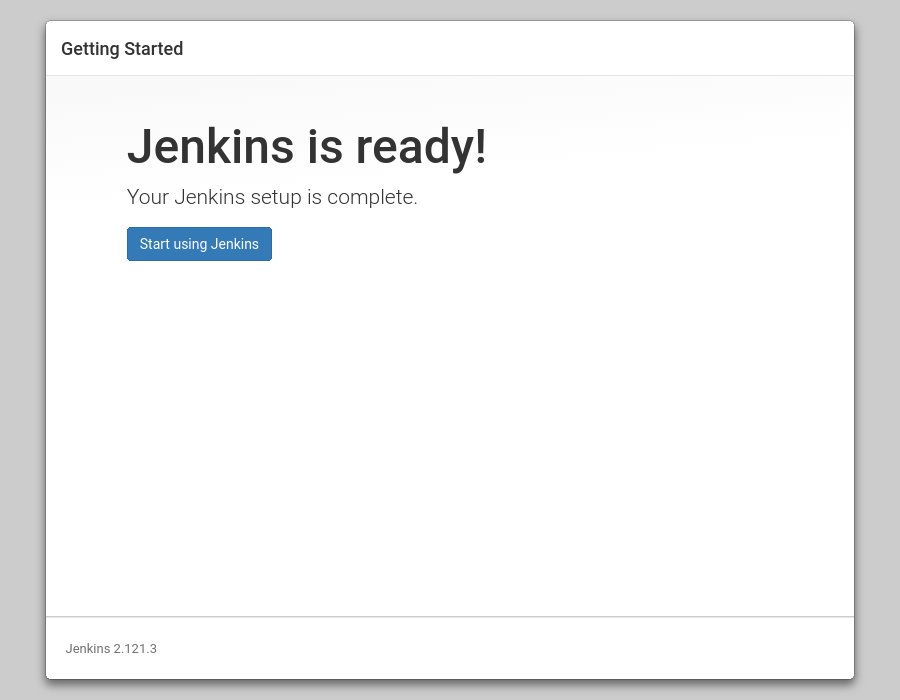
Once the installation is complete, you will be prompted to set up the first administrative user. Fill out all required information and click Save and Continue.



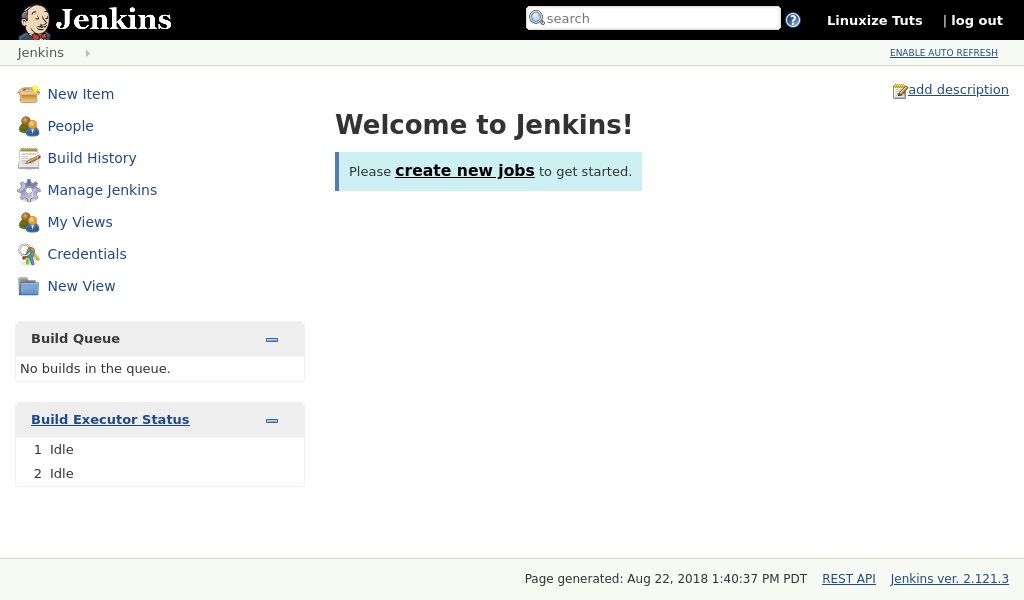
On the next page you will be asked to set the URL for the Jenkins instance. The URL filed will be populated with an automatically generated URL.



To complete the setup confirm the URL by clicking on the Save and Finish button.



Finally, click on the Start using Jenkins button and you will be redirected to the Jenkins dashboard logged in as the admin user you have created in one of the previous steps.



If you’ve reached this point, you’ve successfully installed Jenkins on your CentOS system.

## [Conclusion](https://linuxize.com/post/how-to-install-jenkins-on-centos-7/#conclusion)

In this tutorial, you have learned how to install and complete the initial configuration of Jenkins on CentOS/RHEL based systems. You can now visit the official [Jenkins documentation](https://jenkins.io/doc/) page and start exploring Jenkins’s workflow and plug-in model.

If you have any question, please leave a comment below.

Installation Ansible Centos 7

**Red Hat Ansible** permet la configuration et la gestion centralisée des serveurs et également des applications. En clair, lors du déploiement d’un ou plusieurs serveurs, vous devez les configurer, installer les packages nécessaires, mettre à jour une application particulière, supprimer une application…etc. Vous risquez donc d’y passer un bon moment.

Avec **Ansible**, vous allez réaliser ces tâches répétitives automatiquement d’un seul coup et sur tous les serveurs. *C’est cool non ?*

**Voici ce que vous devez savoir sur Ansible**

* Installation simple et rapide*(RPM, APT, PIP, YUM, GIT…)*
* Il utilise un langage simple pour gérer les conf : **YAML**
* Pas d’infrastructure à gérer pour Ansible
* Pas de serveur Central (master) ni d’agent
* Pas besoin d’une grosse machine pour tourner Ansible
* Peut être installé sur tous les systèmes ou presque *(Unix, FreeBSD et MacOS)*
* Peut être installé sous Windows grâce à **Microsoft WSL***(Windows Subsystem for Linux)*

**Ce qu’il vous faut :**

* **Une VM sous Linux avec un interpréteur : Python 2.7+**
* **Une ou plusieurs VM de tests**
* **Savoir mettre en place les clés SSH**
* **Connaissance langage YAML**
* **Python pour les modules**

**Les copains d’Ansible :**

* **Puppet**
* **Chef**
* **Vagrant**
* **SaltStack**
* **Rudder**

**Liens Ansible utiles :**

* **Ansible** : [ansible.com/](https://www.ansible.com/)
* Github **Ansible :** [/ansible](https://github.com/ansible)
* **Ansible Galaxy** : [explore#/](https://galaxy.ansible.com/explore#/)
* **Documentation Center :** [docs.ansible.com/](https://docs.ansible.com/)
* **Guide d’installation** : [installation\_guide/](https://docs.ansible.com/ansible/latest/installation_guide/" \t "_blank)
* **YAML Syntax** : [#yaml-syntax](https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html#yaml-syntax)
* **Playbooks** : [playbooks/](https://docs.ansible.com/ansible/latest/user_guide/playbooks.html" \t "_blank)
* **Les modules :** [list\_of\_all\_modules](https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html" \t "_blank)

**Ansible Github Repo :**

* **Ansible Release :**[ansible/releases](https://github.com/ansible/ansible/releases" \t "_blank)
* **Ansible exemple Playbooks :** [/ansible-examples](https://github.com/ansible/ansible-examples)
* **Playbooks** by : [snowplow/ansible-playbooks](https://github.com/snowplow/ansible-playbooks" \t "_blank)
* **Playbooks** by : [mgcrea/ansible-web-playbooks](https://github.com/mgcrea/ansible-web-playbooks" \t "_blank)
* **Playbooks** by : [sfromm/ansible-playbooks](https://github.com/sfromm/ansible-playbooks" \t "_blank)
* **Playbooks** by : [nickjj/ansible-playbooks](https://github.com/nickjj/ansible-playbooks" \t "_blank)
* **Playbooks** by : [francisbesset/ansible-playbooks](https://github.com/francisbesset/ansible-playbooks" \t "_blank)

**Les fichiers de configuration Ansible :**

* **/etc/ansible/hosts**
* **/etc/ansible/ansible.cfg**

**Machines virtuelles nécessaires :**

* **CentOS 7 :** [**Création de VM**](https://pixelabs.fr/machine-virtuelle-pixelabs/#VM_CentOS_Red_Hat_7) sous VirtualBox.
* Installation **par défaut version minimale**

 **Une machine de test Linux : Centreon (CentOS)**

# **Install EPEL-Release**

Mettez à jour votre système.

1. [root@ansible ]# yum -y update

Installez EPEL :

1. [root@ansible ~]# yum -y install epel-release

C’est pour moi :

1. [root@ansible ~]# yum -y install nano git htop mlocate wget bind-utils tree

***bind-utils*** : c’est pour avoir la commande nslookup

# **Install Ansible 2.5.5**

Pour cherche un package : ansible

1. [root@ansible ~]# yum search ansible
2. ...
3. ...
4. epel 12590/12590
5. ==================================================== N/S matched: ansible ====================================================
6. ansible-doc.noarch : Documentation for Ansible
7. ansible-inventory-grapher.noarch : Creates graphs representing ansible inventory
8. ansible-lint.noarch : Best practices checker for Ansible
9. ansible-openstack-modules.noarch : Unofficial Ansible modules for managing Openstack
10. ansible-review.noarch : Reviews Ansible playbooks, roles and inventory and suggests improvements
11. python2-ansible-runner.noarch : A tool and python library to interface with Ansible
12. python2-ansible-tower-cli.noarch : A CLI tool for Ansible Tower
13. # ansible.noarch : SSH-based configuration management, deployment, and task execution system
14. kubernetes-ansible.noarch : Playbook and set of roles for seting up a Kubernetes cluster onto machines
15. loopabull.noarch : Event loop driven Ansible playbook execution engine
16. standard-test-roles.noarch : Standard Test Interface Ansible roles
17. Correspondance avec le nom ou le résumé uniquement, utilisez « search all » pour une recherche complète.
18. [root@ansible ~]#

Lancez l’installation à l’aide de la commande suivante :

1. [root@ansible ~]# yum -y install ansible
2. ...
3. ...
4. ...
5. ==========================================================================================================================
6. Package Architecture Version Dépôt Taille
7. ==========================================================================================================================
8. Installation :
9. ansible noarch 2.5.5-1.el7 epel 9.0 M
10. Installation pour dépendances :
11. PyYAML x86\_64 3.10-11.el7 base 153 k
12. libtomcrypt x86\_64 1.17-26.el7 extras 224 k
13. libtommath x86\_64 0.42.0-6.el7 extras 36 k
14. libyaml x86\_64 0.1.4-11.el7\_0 base 55 k
15. python-babel noarch 0.9.6-8.el7 base 1.4 M
16. python-backports x86\_64 1.0-8.el7 base 5.8 k
17. python-backports-ssl\_match\_hostname noarch 3.5.0.1-1.el7 base 13 k
18. python-cffi x86\_64 1.6.0-5.el7 base 218 k
19. python-enum34 noarch 1.0.4-1.el7 base 52 k
20. python-httplib2 noarch 0.9.2-1.el7 extras 115 k
21. python-idna noarch 2.4-1.el7 base 94 k
22. python-ipaddress noarch 1.0.16-2.el7 base 34 k
23. python-jinja2 noarch 2.7.2-2.el7 base 515 k
24. python-keyczar noarch 0.71c-2.el7 epel 218 k
25. python-markupsafe x86\_64 0.11-10.el7 base 25 k
26. python-paramiko noarch 2.1.1-4.el7 extras 268 k
27. python-ply noarch 3.4-11.el7 base 123 k
28. python-pycparser noarch 2.14-1.el7 base 104 k
29. python-setuptools noarch 0.9.8-7.el7 base 397 k
30. python-six noarch 1.9.0-2.el7 base 29 k
31. python2-crypto x86\_64 2.6.1-15.el7 extras 477 k
32. python2-cryptography x86\_64 1.7.2-2.el7 base 502 k
33. python2-jmespath noarch 0.9.0-3.el7 extras 39 k
34. python2-pyasn1 noarch 0.1.9-7.el7 base 100 k
35. sshpass x86\_64 1.06-2.el7 extras 21 k
36. Résumé de la transaction
37. ===========================================================================================================================
38. Installation 1 **Paquet** (+25 Paquets en dépendance)
39. Taille totale des téléchargements : 14 M
40. Taille d'installation : 68 M
41. Is this ok [y/d/N]: y

# **Version Ansible**

Afficher la version installée :

1. [root@ansible ~]# ansible --version
2. [root@ansible ~]# ansible --version
3. ansible 2.5.5
4. config file = /etc/ansible/ansible.cfg
5. configured module search path = [u'/root/.ansible/plugins/modules', u'/usr/share/ansible/plugins/modules']
6. ansible python module location = /usr/lib/python2.7/site-packages/ansible
7. executable location = /usr/bin/ansible
8. python version = 2.7.5 (default, Apr 11 2018, 07:36:10) [GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]
9. [root@ansible ~]#

Version Python :**Python2.7.5**

1. [root@ansible ~]# python --version
2. Python 2.7.5
3. [root@ansible ~]#

Pour afficher les commandes Ansible :

1. [root@ansible ~]# ansible <tabulation 2x>
2. ansible ansible-console ansible-doc-2.7 ansible-playbook ansible-pull-2.7
3. ansible-2 ansible-console-2 ansible-galaxy ansible-playbook-2 ansible-vault
4. ansible-2.7 ansible-console-2.7 ansible-galaxy-2 ansible-playbook-2.7 ansible-vault-2
5. ansible-config ansible-doc ansible-galaxy-2.7 ansible-pull ansible-vault-2.7
6. ansible-connection ansible-doc-2 ansible-inventory ansible-pull-2
7. [root@ansible ~]# ansible

## **Add Linux Host**

Naturellement, Ansible ne connait pas les serveurs que vous voulez administrés. Il faut donc renseigner vos hôtes dans le fichier **/etc/ansible/hosts.** C’est dans ce fichier que vous devez mettre tous vos serveurs (fichier d’inventaire par défaut).

Je souhaite donc administrer mon serveur**Centreon (CentOS)** avec Ansible.

* **Centreon Server : 192.168.1.100**

Ouvrir le fichier hosts et ajouter l’adresse de votre hôte :

1. [root@ansible ~]# nano /etc/ansible/hosts
2. 192.168.1.100

* Avec l’éditeur nano :
  + Enregistrer : **Ctrl+O** et **Entrée**
  + Quitter : **Ctrl + X**

Maintenant que nous avons ajouté l’hôte, il faut maintenant donner l’autorisation à Ansible d’administrer le serveur.

## **Configuration SSH**

Nous allons donc générer une clé **SSH**sur le serveur Ansible. Vous connaissez surement cette histoire de clé hein ? Clé privée, clé publique ? Il faut tout un article pour expliquer le mécanisme d’authentification par clé (une autre fois peut-être).

Passons à la pratique :

1. [root@ansible ~]# ssh-keygen -t rsa -b 2048
2. [root@ansible ~]# ssh-keygen -t rsa -b 2048
3. Generating public/private rsa key pair.
4. Enter file in which to save the **key** (/root/.ssh/id\_rsa): #laissez vide <Entrée>
5. Enter **passphrase** (empty for no passphrase): #laissez vide <Entrée>
6. Enter same passphrase again: #laissez vide <Entrée>
7. Your identification has been saved in /root/.ssh/id\_rsa.
8. Your public key has been saved in /root/.ssh/id\_rsa.pub.
9. The key fingerprint is:
10. SHA256:wyPIsGjtVP9jMjGr3m3q6JTPunFq2U8NELBs9Hf98/k root@ansible
11. The key's randomart image is:
12. +---[RSA 2048]----+
13. | o.. |
14. | o o . . |
15. | . = o . . . |
16. | ..+o...o . . |
17. |...oo .+S. o |
18. |. o ..=oo +|
19. | . +o= = . ..|
20. | .oX.\*.. .|
21. | +O+\*+o E|
22. +----[SHA256]-----+
23. [root@ansible ~]#

Maintenant que nous avons généré les clés SSH. Il faut copier la**clé publique** sur le serveur sur lequel vous souhaitez donc déployer une configuration avec Ansible. L’objectif c’est de permettre à Ansible de se connecter sur le serveur automatiquement.

J’ai donc deux machines :

* **Ansible Server : 192.168.1.103**
* **Centreon Server : 192.168.1.100**

Je vais donc copier la clé publique depuis mon serveur **Ansible** vers le serveur **Centreon**.

1. [root@ansible ~]# ssh-copy-id root@192.168.1.100
2. /usr/bin/ssh-copy-id: INFO: Source of **key**(s) to be installed: "/root/.ssh/id\_rsa.pub"
3. The authenticity of host '192.168.1.100 (192.168.1.100)' can't be established.
4. RSA key fingerprint is SHA256:vL55i7RpHXCUczkiSorPMph8x9MiNb5L4Z8r4ElaCAg.
5. RSA key fingerprint is MD5:42:69:37:89:01:49:ca:12:d6:cb:69:9c:38:b9:7c:06.
6. Are you sure you want to continue **connecting** (yes/no)?# yes
7. /usr/bin/ssh-copy-id: INFO: attempting to log in with the new **key**(s), to filter out any that are already installed
8. /usr/bin/ssh-copy-id: INFO: 1 **key**(s) remain to be installed -- if you are prompted now it is to install the new keys
9. root@192.168.1.100's password:# mot de passe root de Centreon Server
10. Number of **key**(s) added: 1
11. Now try logging into the machine, with: "ssh 'root@192.168.1.100'"
12. and check to make sure that only the **key**(s) you wanted were added.
13. [root@ansible ~]#

Je peux maintenant me connecter sur le serveur **Centreon** depuis le serveur **Ansible** sans taper le mot de passe :

1. [root@ansible ~]# ssh root@192.168.1.100
2. Last login: Fri Jun 22 16:05:06 2018 from 192.168.1.94
3. [root@centreon ~]#
4. [root@centreon ~]#
5. [root@centreon ~]# exit
6. logout
7. Connection to 192.168.1.100 closed.
8. [root@ansible ~]#

## **Ansible Ping Host**

Commençons par les choses simples. On peut effectuer un Ping avec Ansible :

1. [root@ansible ~]# ansible -m ping 192.168.1.100
2. 192.168.1.100 | SUCCESS => {
3. "changed": false,
4. "ping": "pong"
5. }
6. [root@ansible ~]#

**192.168.1.100 = Centreon Server**

Pour afficher le résultat sur une seule ligne :

1. [root@ansible ~]# ansible -m ping 192.168.1.100 --one-line
2. 192.168.1.100 | SUCCESS => {"changed": false, "ping": "pong"}
3. [root@ansible ~]#

## **Ansible Ping Group**

Il y a mieux encore. On peut créer des groupes d’hôtes pour faciliter la configuration avec Ansible. Retournez dans le fichier**/etc/ansible/hosts** :

1. [root@ansible ~]# nano /etc/ansible/hosts

Voici un exemple :

1. [Centreon-Server]
2. 192.168.1.100
3. 192.168.1.101
4. 192.168.1.102
5. [VMware-Server]
6. 192.168.1.103
7. 192.168.1.104
8. 192.168.1.105
9. ...
10. ...

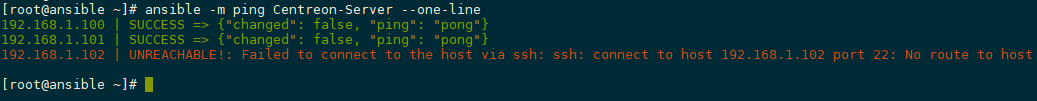
Vous pouvez maintenant réaliser un Ping sur tous les hôtes d’un groupe :

1. [root@ansible ~]# ansible -m ping Centreon-Server --one-line
2. 192.168.1.100 | SUCCESS => {"changed": false, "ping": "pong"}
3. 192.168.1.101 | SUCCESS => {"changed": false, "ping": "pong"}
4. 192.168.1.102 | UNREACHABLE!: Failed to connect to the host via ssh: ssh: connect to host 192.168.1.102 port 22: No route to host
5. [root@ansible ~]#

Le dernier c’est normal, j’ai que deux machines.

* **Centreon Server : 192.168.1.100**
* **CentOS Server : 192.168.1.101**

Remarque : n’oubliez pas de copier la clé publique sur les nouveaux serveurs.



C’est terminé.  **Ansible** est prêt à faire son job sur les serveurs Linux. Explorez et récupérez les modules sur le site Ansible (voir les liens au début de l’article). Dans le prochain chapitre, nous allons voir comment configurer un hôte Windows avec Ansible en utilisant le protocole **CredSSP** (Credential Security Support Provider).

Bonne journée et à bientôt.

Build Jenkins Gitlab Nexus Centos 7

## **WHY CI?**

When you are building software for multiple clients, multiple configurations and architectures you need a common way to deal with software building. The usual way of doing is not useful anymore.

We found project were you have to build a system for JBOSS 7 AS with EJB and J2EE compliance while maintaining a version for Tomcat 6, built on WARs and Spring. The system can have two flavors, JPA one and pure Hibernate and must run over Mysql, Oracle or Postgresql. Maybe others.

When a customer requires a release of a new version, you must provide a clean environment no matter what happened on development side.

This is why we built our CI environment based on this tools:

* Git: We need to track every version of the files over time. And ensure that nothing breaks between changes.
* Maven 3: Our build system must track dependencies, be able to build different types of packages and allow customization on each build.
* Sonatype Nexus: We must track every version that’s deployed to the client. And also make available to all team every new version of each library. Additionally it also stores a clean version of each product, ready to be deployed.
* Jenkins: Is the loyal majordomo that joins all together.
* GitLab: It eases software management, let us to write tutorials about how to configure the software, and the most important. The “Hook” feature let’s Jenkins know when a repository has changed.

With all this in place our day to day is as follows:

* Every project is built as needed. When code changes or developer requires a build.
* Every dependent on the built project is built to see if nothing breaks.
* A version of the “ready to use” packages are made available for the rest on Nexus.

And developer even doesn’t knows that this happened.

JUST GREAT!

## **PREMISES**

We suppose that you have every product installed and running. We are going to join every piece together.

So we are going to:

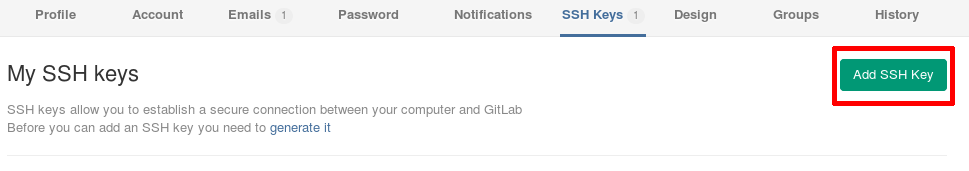
1. Jenkins needs access to GitLab repository so it can download (clone) the latest repo when it’s changed.
2. GitLab must inform Jenkins over repo changes.
3. Jenkins also need access to Sonatype Nexus to upload new versions when everything is compiled and ready to use.
4. Jenkins maven need to be access to company repos.

## **GIVE A GRANT TO GITLAB**

An user must be created in GitLab, this can be done as administrator in the GitLab Panel:

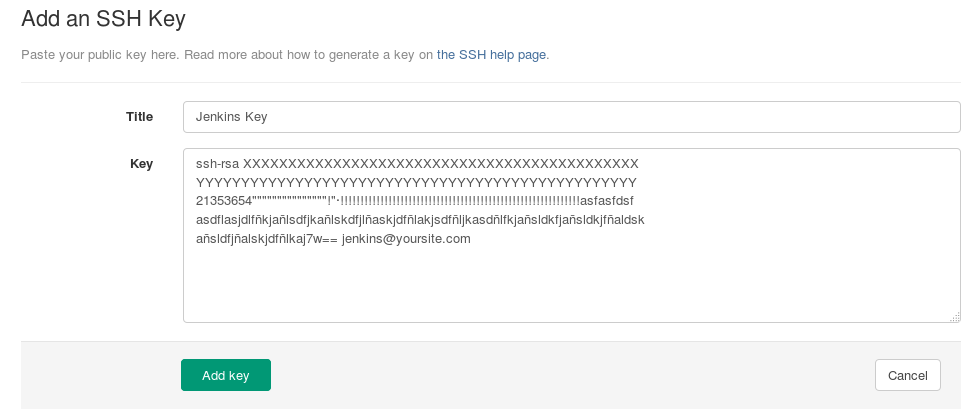
[](http://www.level2crm.com/wp-content/uploads/2015/01/GitlabJenkins.png)

Now we must create a public and private key so it can access to the repos.

[](http://www.level2crm.com/wp-content/uploads/2015/01/GitlabCreateKey.png)

$> ssh-keygen -t rsa -C "jenkins@yoursite.com"

$> cat ~/.ssh/id\_rsa.pub

And what it shows is what you have to put in the key.  
[](http://www.level2crm.com/wp-content/uploads/2015/01/GitlabKey.png)

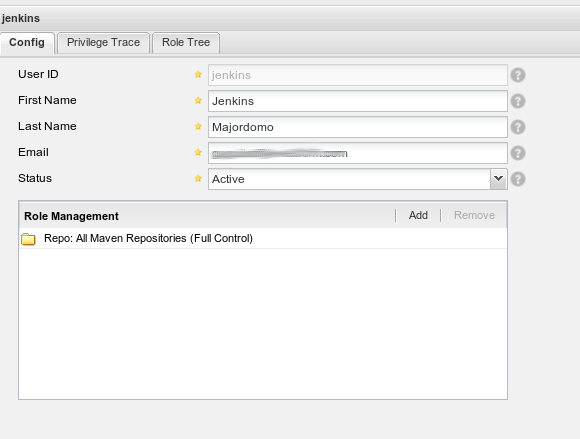
And now you have to copy the keys to the jenkins server under the jenkins home. In my case from /etc/passwd:

jenkins:x:115:119:Jenkins,,,:<strong>/mnt/webapps/jenkins</strong>:/bin/bash

So I created a directory there called .ssh where I put both keys, public and private. Is this a security risk? Maybe I have to check and surely change the home of jenkins in the OS.

## **GIVE A GRANT TO NEXUS REPOS**

Now we must give access to Jenkins to Sonatype Nexus. This is done in the interface.

[](http://www.level2crm.com/wp-content/uploads/2015/01/NexusUser.png)

Remember the password you set to Jenkins user because you will need it for configuring access to the repos. This is done by editing ~/.m2/settings.xml of the jenkins user.

 <?xml version="1.0" encoding="UTF-8"?>

<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0 http://maven.apache.org/xsd/settings-1.0.0.xsd">

<servers>

<server>

<id>nexus-snapshots</id>

<username>jenkins</username>

<password>yourpasswordhere</password>

</server>

</servers>

</settings>

If you need to give it access to another repos, please add it here. I only added support for nexus-snapshots repo. And you will ask. What’s the nexus-snapshot repo? Well this depends on the project jenkins it’s building. More on this later.

## **CREATE A TASK FOR JENKINS**

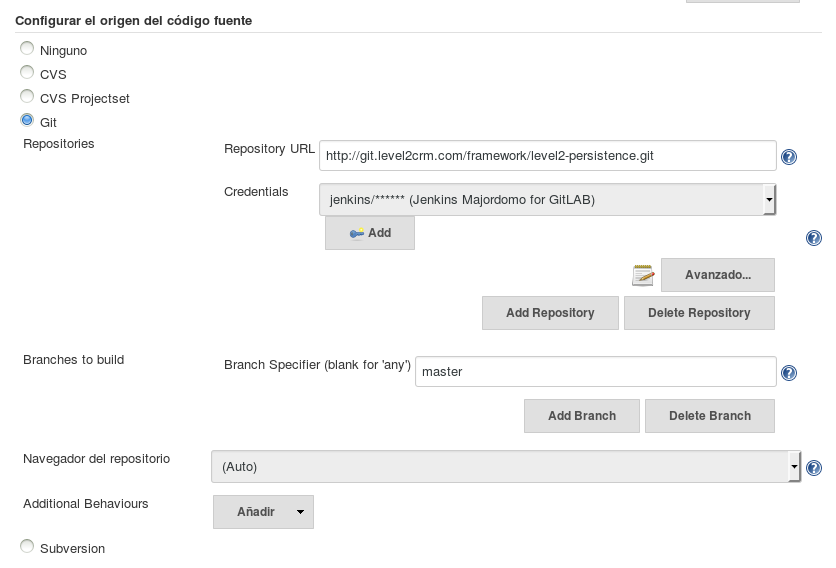
Well, now our Jenkins majordomo is ready to build. Let’s build a task. But first we must tell Jenkins to use the SSH credentials created on the server under his user.

[](http://www.level2crm.com/wp-content/uploads/2015/01/JenkinsCredential.png)

We are telling Jenkins to use the private key to access any resource it needs. Remember we configured and copied the keys on the GitLab step.

### **CONFIGURE JENKINS ACCESS TO GIT REPO**

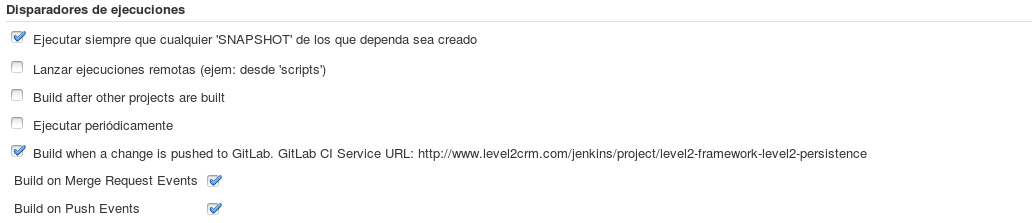
We tell Jenkins to access GitLab to clone the repo on building. And also we tell it must compile the master branch. In our case we will not use master, we prefer to use develop because we use another methodology called gitflow. You can read more about this [here](http://nvie.com/posts/a-successful-git-branching-model/" \o "GitFlow Branching Model).

[](http://www.level2crm.com/wp-content/uploads/2015/01/JenkinsClone1.png)

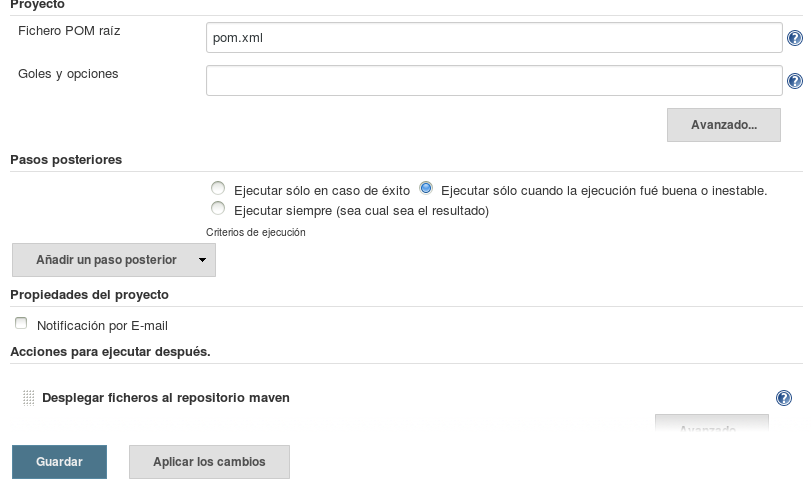
You can make Jenkins to compile only the branch where you did commit by setting the “Branches to build” to origin/${gitlabTargetBranch}. But it will only work when GitLab launches the compilation so make sure to launch always from GitLab by using “Test Hook”.

### **CONFIGURE WHEN JENKINS SHOULD BUILD IT AND DEPLOY**

We tell Jenkins to build it whenever a dependency changes or when GitLab informs it that the code changed. To see the GitLab stuff you must have the GitLab plugins installed on Jenkins.

[](http://www.level2crm.com/wp-content/uploads/2015/01/JenkinsTrigger.png)

And also we want to build the project and if it compiles upload it to Nexus. It’s great that Jenkins now about Nexus because the only think you have to do it is configure credentials as done before, and let the system perform as expected.

[](http://www.level2crm.com/wp-content/uploads/2015/01/JenkinsDeployMaven.png)

I tell Jenkins execute after build steps only when inestable of good. This means that we will have a version if it compiles. The action after is “Deploy to maven repository”. And it knows what to do.

DONE!

## **EXPECTED RESULTS**

When we upload a new change to Gitlab, it will tell Jenkins that something changed. It will use maven to compile the code and if everything went okay, automagically it will upload the version to Nexus. It’s magic!!!

I will post some pictures of the result when I do a change…