## *Practice instructions*

### Installation and configuration of Jenkins container

Please refer to the next article for the docker installation and configuration in case of issues:

[https://faun.pub/basic-ci-cd-for-python-projects-with-docker-and-jenkins-38eeb547fb28](https://faun.pub/basic-ci-cd-for-python-projects-with-docker-and-jenkins-38eeb547fb28%20)

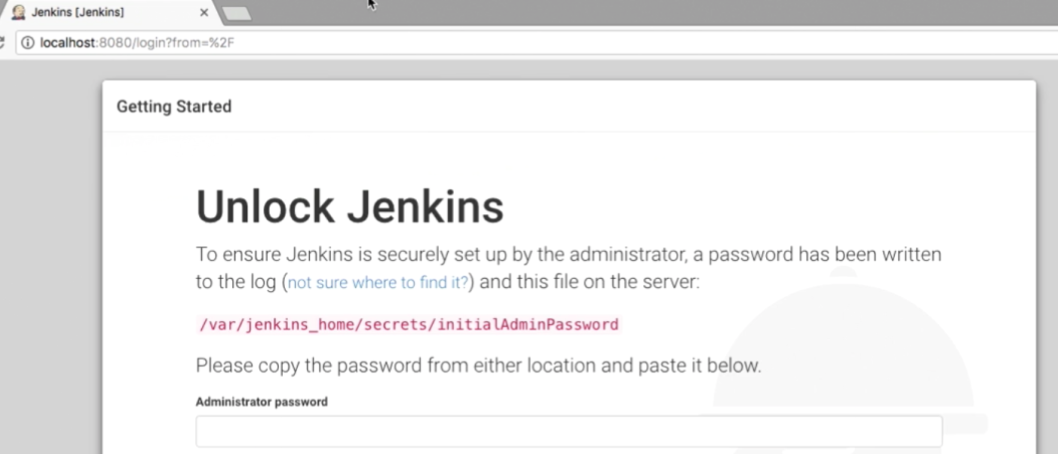
* Download and install Docker for your operating system
  + <https://www.docker.com/get-started>
* Make sure you have python and pip installed.
  + *python –version*
  + *pip –version*
* Find the Jenkins container in the Docker registry

<https://hub.docker.com/search?q=&type=image>

Graphical user interface, application

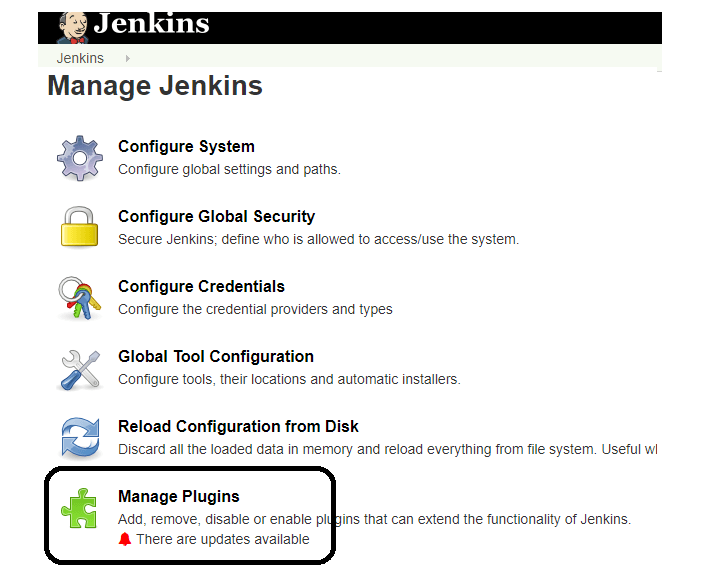
Description automatically generated

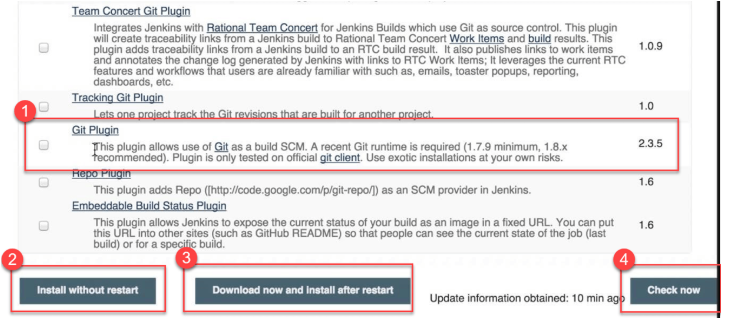
* Create and build the Dockefile based on jenkins/jenkins:latest image
* Run Docker image
* Jenkins setup: paste the password from the terminal

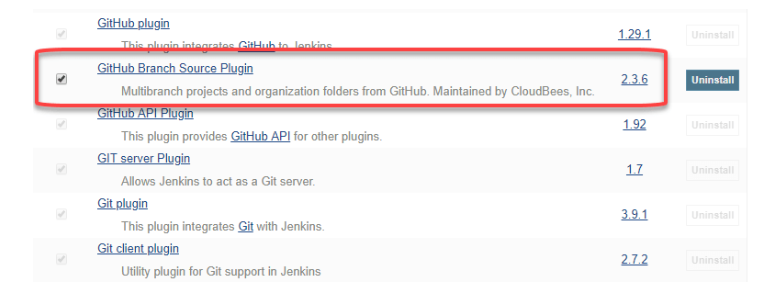


* Install Git on your local machine: go to https://git-scm.com/downloads. Use the same user as for Jenkins.
* Create and set up your repository in GitHub.
* Move .py scripts to the repository.
* Set up branches in the repository; Run examples of commits and PRs according to the chosen strategy.

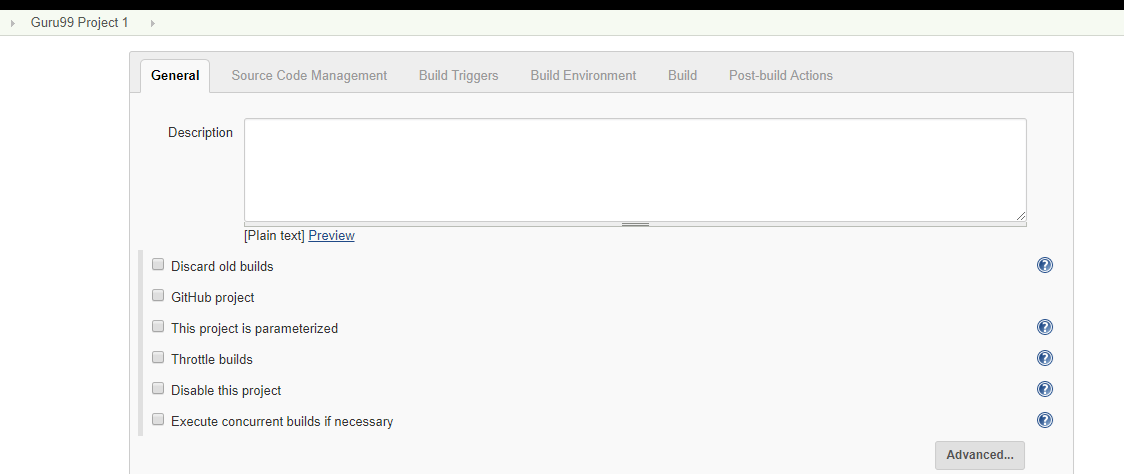
Install the GIT plugin to access the GitHub repository



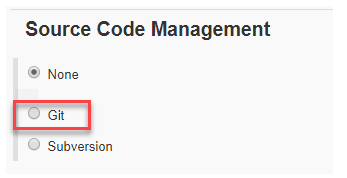
* 

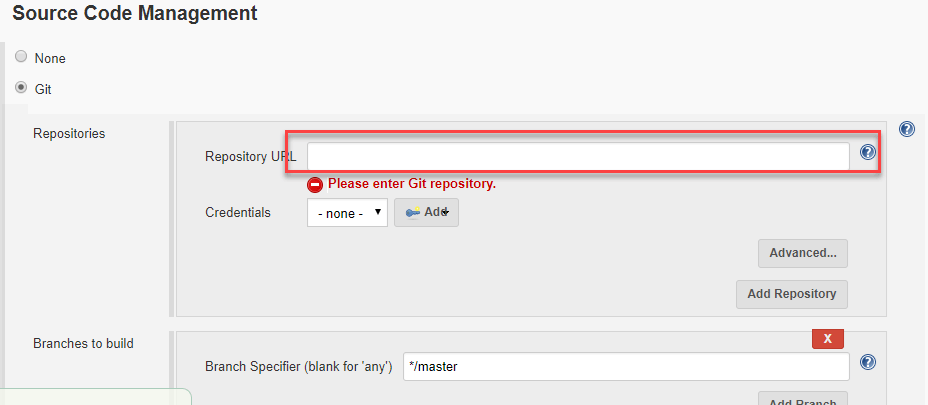
After installing the plugins, go to the "Manage Jenkins" section in the Jenkins UI. You should see your plugins listed among the others.

Jenkins GitHub Integration: Create a new Jenkins Job. Project type is Pipeline project and enter project info:



On the second tab, you will see an option **Git in section** « **Source code management»,** if your Git plugin was succecfully installed in Jenkins:



Enter your Git repository URL to get the code from GitHub 

* Develop Jenkins File and add it to your GitHub repository.
* Commands and examples are available here: <https://www.jenkins.io/doc/book/pipeline/>

pipeline {

agent any

stages {

stage('Test') {

steps {

echo 'Hello World ...'

}

}

}

}

######################

*pipeline {*

*agent any*

*stages {*

*stage ('GIT Checkout'){*

*steps {*

*git changelog: false, poll: false, url: 'https://github.com/...git'*

*}*

*}*

*stage('build') {*

*steps {*

*sh 'pip install -r requirements.txt'*

*}*

*}*

*stage ('Test'){*

*steps {*

*sh 'python unit-test.py'*

*}*

*}*

*}*

*}*

* Configure Git plugin trigger to run jenkinsfile when push / pr to repository.

Generate github Personal Access Token (PAT)

In the GitHub repository, set the URL of the web hook to jenkinswebhook: *jenkinspath /github-webhook/*

Chose ‘Let me select individual events’ ->‘Pull Request’

### Configuration connection to the Database from the container

During the preparation for the execution of your scripts you can encounter several issue with connection to the database from the container. Please find below some useful articles and tips for potential issues

#### Connection library for python to connect from the docker

Use **pymssql** library instead of pyodbc.

#### Configuring SQL Server browser service

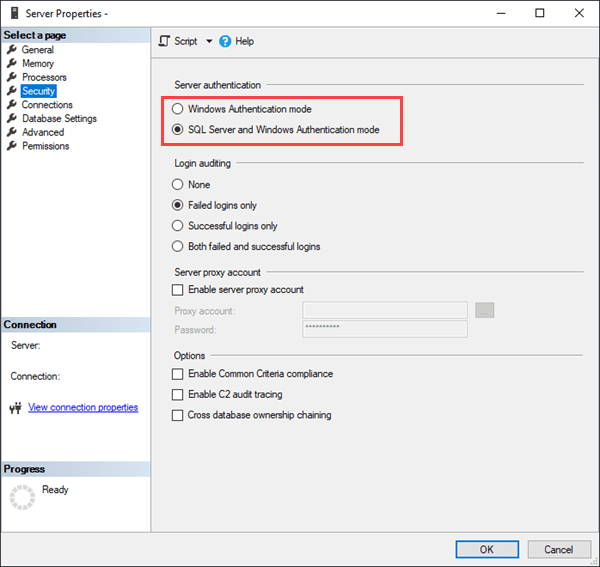
To be able to connect externally you need to have SQL Server browser service enabled.

If it is not enabled please go to the *Control Panel->Administrative Tools->Services*. Find SQL *Server Browser service> Properties > General tab*, change Startup type to Automatic, click Apply button, then click Start button in Service Status area**.**

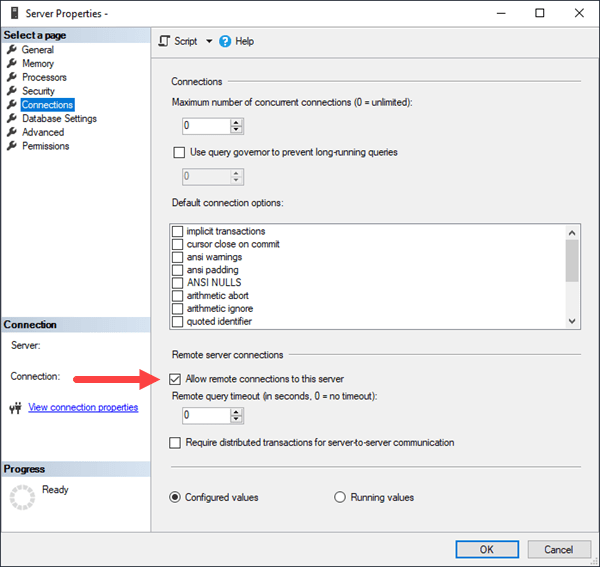
#### Configuring SQL Express database to external connection from container

Security and connection

1. Open SQL Server Management Studio (SSMS)
2. Connect to your server
3. Right-click on your server name and click 'Properties'.
4. Go to the Security page for Server Authentication, and select 'SQL Server and Windows Authentication' mode.



1. Then, go to the Connections page and ensure that "Allow remote connections to this server" is checked, and click OK.

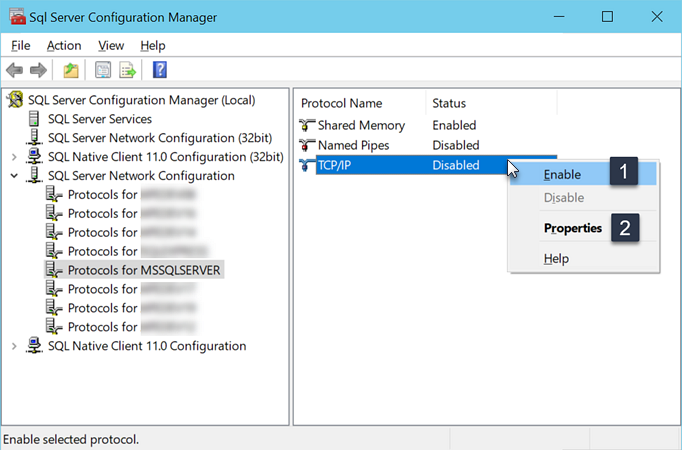


*NB: Take in mind that user used for connecting should have enough permissions for the external connect*

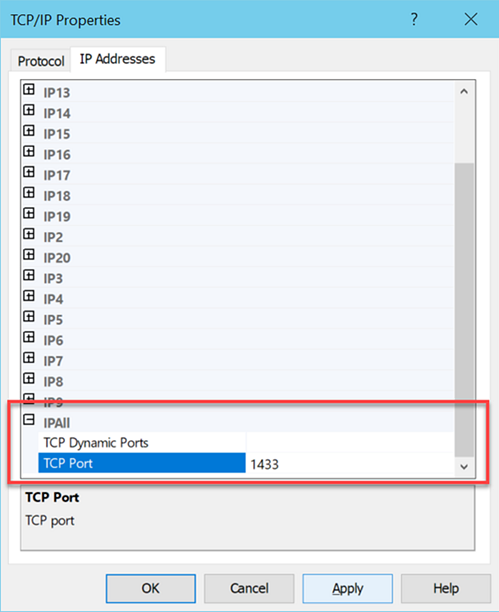
SQL Server Configuration

Your server is set up to allow remote connections with a SQL Server login but now you must enable TCP/IP protocols for your server.

1. Open SQL Server Configuration Manager
2. Expand SQL Server Network Configuration and Protocols for {Your server name}.
3. Right-click 'TCP/IP' and select Enable. Then click OK on the message that the service needs to be restarted before changes take effect.



1. SQL Server Configuration Enable TCP/IP
2. Right-click 'TCP/IP' again and select Properties. View the IP Addresses tab and locate 'IPAll', (all IP Addresses).



1. Enter the value '1433' directly in the TCP Port field. Click OK to apply the change, and click OK on the message that the service needs to be restarted before changes take effect.
2. TCP Properties
3. Back in the SQL Server Services dialog, right-click on your server name and select Restart. Alternatively, you can do this from SSMS by right-clicking the server name and clicking Restart.
4. Restart SQL Server and SQL Server browser services

