Jenkins Continuous Delivery Pre-Assignment

Start up your Jenkins AWS EC2 instance

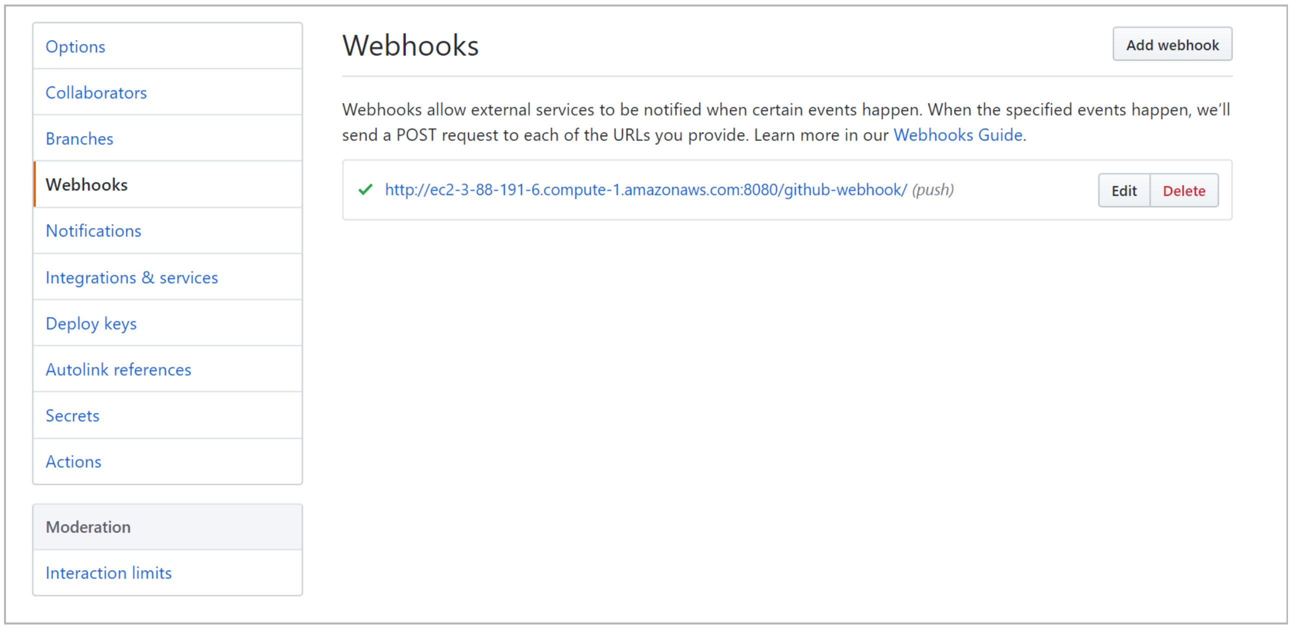
1. Navigate to the AWS EC2 Dashboard
2. Right click your Jenkins Instance
3. Click Instance State -> Start

Note: *There can be a cost associated with using EC2. AWS allows certain services to be run within a "free tier" for the first year of use of your AWS account, so most students can run an EC2 instance for free. However, if you have used up your free tier eligibility (by having created an AWS account more than a year ago, possibly for CS 260 or another class) you could be charged. However, the charges are small. If you create a EC2 t2.micro server for use in this lab, and leave it running for an entire week, you will be charged approximately $1.95 if you are not still eligible for the free tier. We recommend creating an EC2 t2.micro instance as you do this pre-assignment and then stopping (but not terminating) your instance when you are done. You can then restart it when you need for the next two weeks when you do the two Jenkins tutorials for this class.*

Set up Github Webhook

In the last tutorial we polled Git for changes once a minute. This can be inefficient. By setting up a Webhook, Github will notify the Jenkins server when changes are made to the repository. This way Jenkins builds the project every time a change is made. However, everytime you stop and restart your AWS instance, the IP address changes. ***If you stop the Jenkins instance and restart it, the EC2 IP address will probably change and you will need to update this webhook. There are AWS services that allow you to associate an unchanging domain name with an EC2 instance, but we don't want to pay for those for this tutorial.***

1. **Navigate to the github repository for the last tutorial's project**
2. Go to Settings
3. On the left side there is an option named Webhook. Click it
4. **Click add webhook**
    a. In the payload URL, paste the following: http://your_AWS_IP:8080/github-webhook/
    b. **Set content type to json**
    c. **Click add webhook**



Webhooks

Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.

✓ http://ec2-3-88-191-6.compute-1.amazonaws.com:8080/github-webhook/ *(push)*

5. Go to your previous pipeline project -> configure -> click "**GitHub hook trigger for GITScm polling**"
6. Uncheck **Poll SCM**
7. Select Apply and Save
8. Jenkins should now trigger everytime you push file changes to your github repository
9. **Push your code** to makes sure your project automatically builds
   a. You may need to make a simple change in one of your project's files to be able to push it
   b. Make sure you don't have multiple pipelines that are using the same github repository. By default, Github seems to only send its push to one pipeline.

Install/Setup Docker on AWS EC2

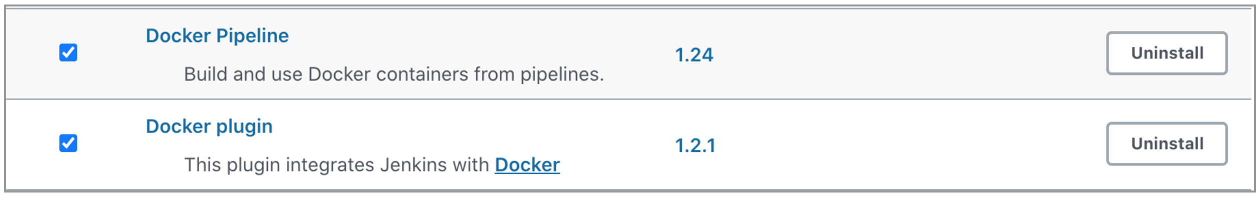
1. Install Docker
   a. **ssh into your AWS EC2 instance**
   b. In your AWS terminal install docker with the following command:
      i. `sudo yum install -y docker`
      ii. Verify installation with : `docker -v`
2. Set permissions
   In order for Jenkins to use Docker, you need to grant Jenkins permission to use it

   a. In your AWS terminal add the correct permissions to allow your ec2 user and Jenkins program to use Docker with the following commands :
      i. `sudo usermod -aG docker ec2-user`

        ii. `sudo usermod -a -G docker jenkins`

   b. Reboot the AWS device with the following command :
        i. `sudo reboot`
   c. After Rebooting AWS you will need to ssh back in to your server. You may need to restart Docker or Jenkins. Use the following commands to start those services:
        i. `sudo service docker start`
        ii. `sudo service jenkins start`

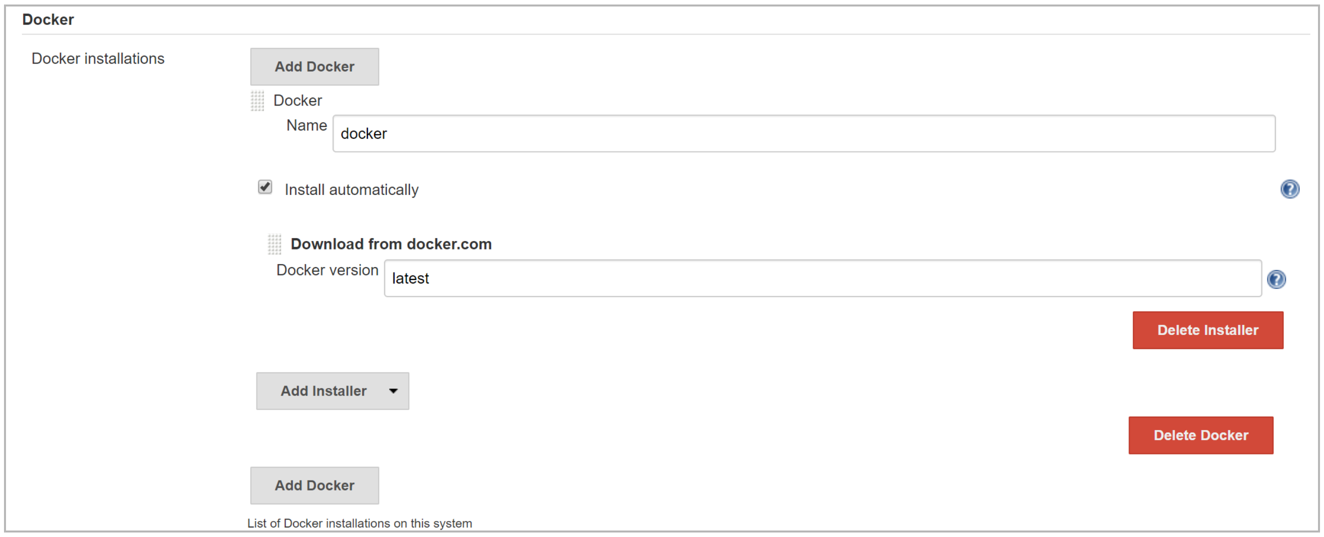
3. Verify that you have the correct Jenkins plugins installed
   a. **Navigate to you Jenkins** dashboard : your_aws_ip:8080
   b. Sign in
   c. Navigate to Manage Jenkins > manage plugins
   d. Click installed tabs
        i. Type in Docker Plugin and verify that it is there
        ii. Type in Docker Pipeline and verify that it is there.
        iii. These plugins should look like the screenshot below



✓	Docker Pipeline Build and use Docker containers from pipelines.	1.24	Uninstall
✓	Docker plugin This plugin integrates Jenkins with Docker	1.2.1	Uninstall

        iv. If the plugins are not installed you will need to install them
        v. Navigate to the available tab in the manage plugin menu
        vi. Type in “Docker Plugin” install it and restart
        vii. Type in “Docker Pipeline” install and restart

4. Add docker credentials to Jenkins
   a. Navigate to Manage Jenkins > Global Tools Configuration >Docker
        i. Note: If Docker does not appear in the Global Tools configuration, then it probably didn’t install the docker plugin.
            1. Click Manage Jenkins > Plugins > available : Search for “Docker”
            2. If Docker is not installed, click on available and type in “Docker”
            3. Several Docker related plugins will appear.
            4. Download and install the plugin that just says “Docker”
            5. Start back at Step #3
   b. Add docker
        i. Give it the name docker
        ii. Check Install Automatically
        iii. Add Installer > Download from docker.com
        iv. Select latest version of docker



Docker

Docker installations

Docker

Name: docker

Install automatically: ✓

Download from docker.com

Docker version: latest

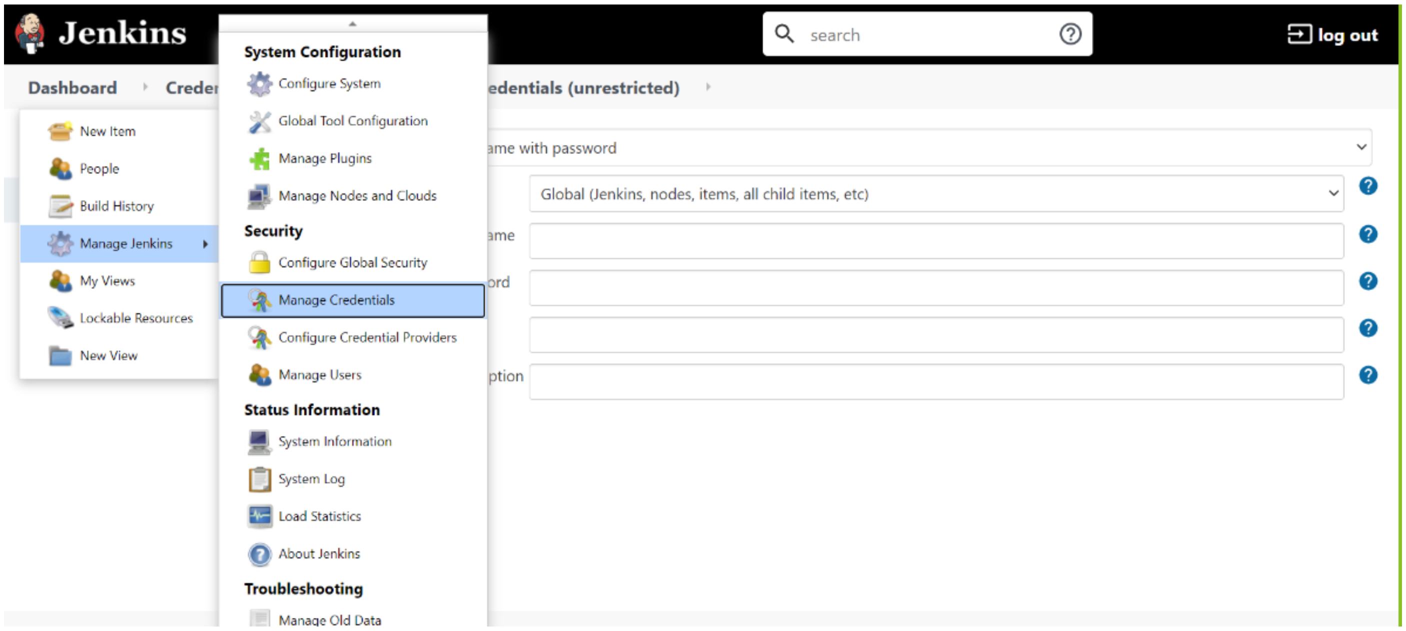
List of Docker installations on this system

c. Apply, Save, and Navigate back to the Jenkins Dashboard

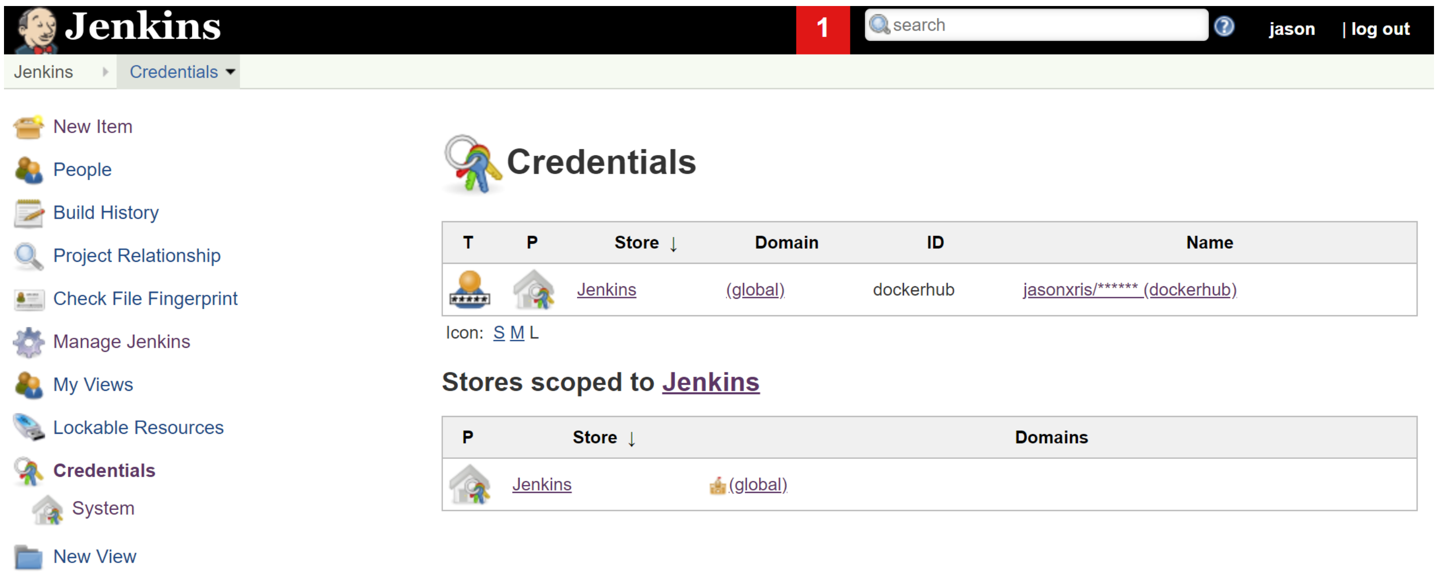
d. Navigate to **Manage Jenkins >Credentials > (Global) > Add Credentials** (start from Credentials in the left sidebar menu)

i. Enter your username and password for DockerHub
Note: You should have a DockerHub account from a previous lab. If not, you will need to create one.

ii. Give it the ID and description : dockerhub



iii. When you go back to the credential dashboard, screen should look like the screenshot below



Credentials

T	P	Store ↓	Domain	ID	Name
		Jenkins	(global)	dockerhub	jasonxris/****** (dockerhub)

Icon: S M L

Stores scoped to Jenkins

P	Store ↓	Domains
	Jenkins	(global)

Setup Jenkins to send a notification when a build fails

Jenkins needs to be configured to use a specific email inorder to send notifications. **To make this step easier, make sure you have a gmail account.**

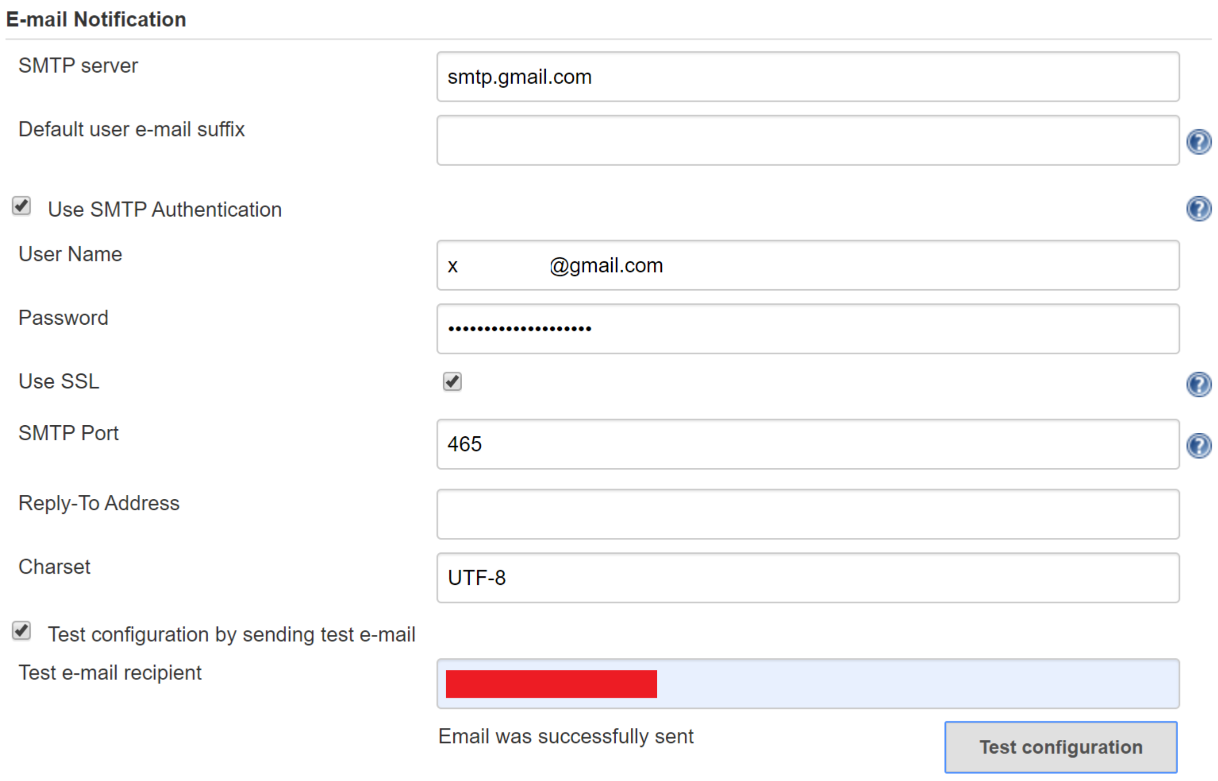
1. Navigate to Manage Jenkins> Configure System > E-mail Notification (at the bottom of the page)
   a. **DO NOT CONFIGURE EXTENDED EMAIL, WE WILL BE USING THE NORMAL E-mail Notification service**
   b. The SMTP server for gmail is : smtp.gmail.com
   c. Click “Advanced”
   d. **Check : Use SMTP authentication**
   e. Put your username information for your Gmail account
   f. **Check: Use SSL**
   g. **Set the SMTP port to the default : 465**
   h. **Check “Test configuration”**
   i. **Enter in a valid email you have access to**
   j. **Save**
   k. **You will put in the gmail password created for you by gmail in the following step**

2. Set up Email Authentication
   For Jenkins to be able to use your email, it needs to authenticate itself. While some Email services may allow your app to sign in using just your username and password, Google has additional security.

   a. **Follow the tutorial,** [Jenkins: Email Authentication](), in order to **set-up the correct authentication** for your Jenkins server

   b. Send a test e-mail to see if the e-mail notification settings were configured correctly.

If everything was set up correctly, your settings should look like the screenshot below



E-mail Notification

SMTP server	smtp.gmail.com
Default user e-mail suffix	
✓ Use SMTP Authentication	
User Name	x @gmail.com
Password	••••••••••••••••••••
Use SSL	✓
SMTP Port	465
Reply-To Address	
Charset	UTF-8
✓ Test configuration by sending test e-mail	
Test e-mail recipient	[redacted]

Email was successfully sent

If your test email works, then your email notification is set up and you are done with the Pre-assignment