

CONTINUOUS INTEGRATION OF PIPELINE FOR TOOLING WEBSITE

Step 1 Install Jenkins server

1. Create an AWS EC2 server based on Ubuntu Server 20.04 LTS and name it "Jenkins"
2. Install **JDK** (since Jenkins is a Java-based application)

```
sudo apt update
```

```
sudo apt install default-jdk-headless
```

3. Install Jenkins

```
wget
```

```
https://raw.githubusercontent.com/Tonybesto/Installation\_Scripts/main/jenkins.sh
```

Once downloaded, make it executable `sudo chmod +x <file name>` and then `./<filename>`

4. Make sure jenkins is up and running `sudo systemctl status jenkins`
5. By default Jenkins server uses TCP port 8080 – open it by creating a new Inbound Rule in your EC2 Security Group
6. Perform initial Jenkins setup by going to your browser
`http://<Jenkins-Server-Public-IP-Address-or-Public-DNS-Name>:8080`

You will be prompted to provide a default admin password, Retrieve it from your server. `sudo cat /var/lib/jenkins/secrets/initialAdminPassword`

7. Then you will be asked which plugins to install – choose suggested plugins.
8. Once plugins installation is done – create an admin user and you will get your Jenkins server address.

Step 2 Configure Jenkins to retrieve source codes from GitHub using Webhooks.

1. Enable webhooks in your GitHub repository settings and then add and paste in your jenkins dns, should be like this `http://<dns:8080>github-webhook/`, then under content type select “application/json” and “select just the push event”

2. Go to Jenkins web console, click "New Item" and create a "Freestyle project"
3. To connect your GitHub repository, you will need to provide its URL, you can copy from the repository itself by clicking on codes and getting the link.
4. In configuration of your Jenkins freestyle project scroll to source code management, choose Git repository, provide there the link to your Tooling GitHub repository and credentials (user/password) so Jenkins could access files in the repository.

IF YOU ARE GETTING ERRORS ON YOUR JENKINS KINDLY GO TO MANAGE JENKINS AND THEN CLICK ON CONFIGURE GLOBAL SECURITY AND THEN SCROLL DOWN TO CSRF PROTECTION AND ENABLE PROXY COMPATIBILITY.

5. Try to run the build. For now we can only do it manually. Click "Build Now" button, if you have configured everything correctly, the build will be successful and you will see it under #1.
6. You can open the build by clicking on the build number #1 and check in "Console Output" if it has run successfully. But this build does not produce anything and it runs only when we trigger it manually. Let us fix it.
7. Click "Configure" your job/project and add these two configurations
 - a. Under build trigger, Configure triggering the job from GitHub webhook by selecting github build trigger for GITScm polling.
 - b. Under Post build actions, select archive the artifacts and type '**' under files to archive
8. Now, go ahead and make some changes in any file in your GitHub repository (e.g. README.MD file) and push the changes to the master branch. You will see that a new build has been launched automatically (by webhook) and you can see its results – artifacts, saved on Jenkins server.

You have now configured an automated Jenkins job that receives files from GitHub by webhook trigger (this method is considered as 'push' because the changes are being 'pushed' and files transfer is initiated by GitHub). There are also other methods: trigger one job (downstream) from another (upstream), poll GitHub periodically and others.

By default, the artifacts are stored on Jenkins server locally `ls`

```
/var/lib/jenkins/jobs/tooling_github/builds/<build_number>/archive/
```

Step 3 – Configure Jenkins to copy files to NFS server via SSH

Now we have our artifacts saved locally on Jenkins server, the next step is to copy them to our NFS server to /mnt/apps directory. Jenkins is a highly extendable application and there are 1400+ plugins available. We will need a plugin that is called "Publish Over SSH".

1. Install "Publish Over SSH" plugin. On the main dashboard select "Manage Jenkins" and choose "Manage Plugins" menu item. On "Available" tab search for "Publish Over SSH" plugin and install without restart
2. Configure the job/project to copy artifacts over to the NFS server, On the main dashboard select "Manage Jenkins" and choose the "Configure System" menu item. Scroll down to Publish over SSH plugin configuration section and configure it to be able to connect to your NFS server:
 - a. Provide a private key (content of .pem file that you use to connect to NFS server via SSH/Putty)
 - b. Arbitrary name - NFS SERVER
 - c. Hostname – can be private IP address of your NFS server
 - d. Username – ec2-user (since NFS server is based on EC2 with RHEL 8)
 - e. Remote directory – /mnt/apps since our Web Servers use it as a mounting point to retrieve files from the NFS server
 - f. Test the configuration and make sure the connection returns Success.
Remember, that TCP port 22 on the NFS server must be open to receive SSH connections.
3. Save the configuration, open your Jenkins job/project configuration page and add another one "Post-build Action", select 'send build artifacts over ssh' Configure it to send all files produced by the build into our previously defined remote directory. In our case we want to copy all files and directories – so we use '**' under source files. If you want to apply some particular pattern to define which files to send – [use this syntax](#).

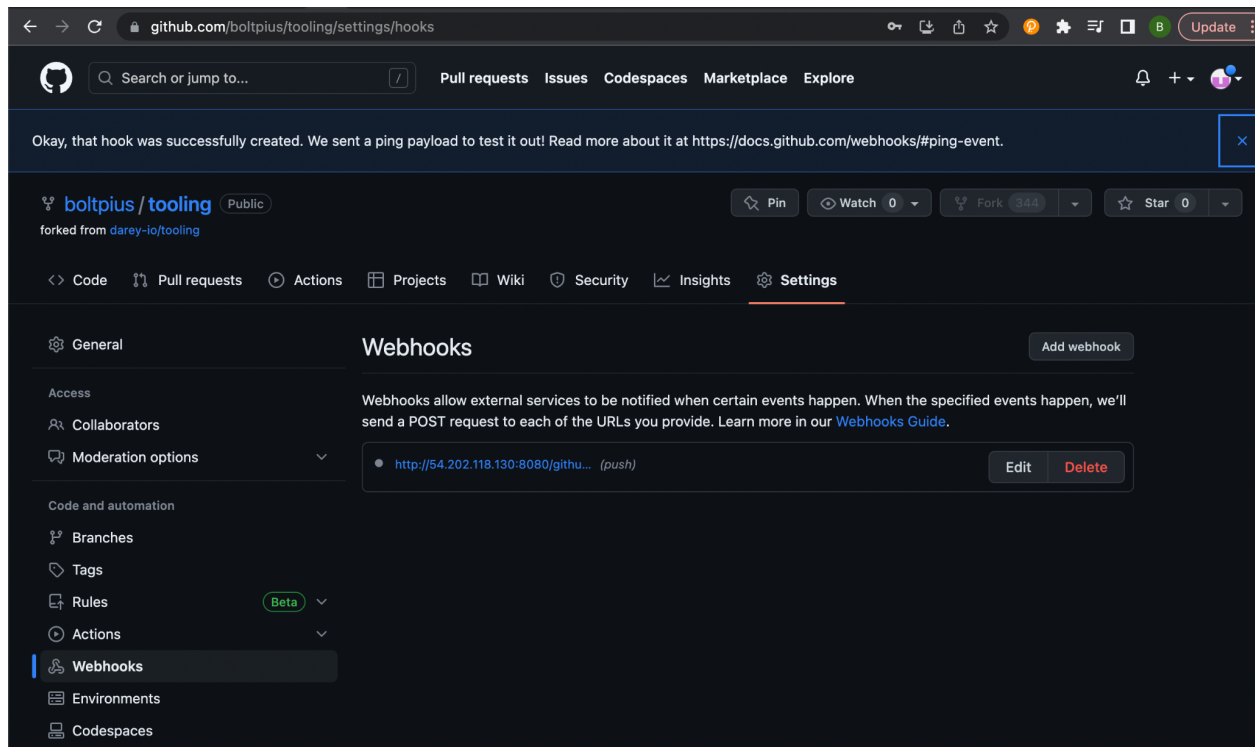
Save this configuration and go ahead, change something in the README.MD file in your GitHub Tooling repository. Webhook will trigger a new job and in the "Console Output" of the job you will find something like this:

```
SSH: Transferred 25 file(s)
```

Finished: SUCCESS

4. To make sure that the files in /mnt/apps have been updated – connect via SSH/Putty to your NFS server and check README.MD file `cat /mnt/apps/README.md`

If you see the changes you had previously made in your GitHub – the job works as expected.



← → ↻ Not Secure 54.202.118.130:8080/view/all/newJob


Jenkins Search (⌘+K) ? 1 1 Pius log out

Dashboard > All >


Enter an item name

Project 9


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder is a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

OK

← → ↻ github.com/boltpius/tooling/edit/master/README.md

```
48 ***
49 git push --set-upstream origin feature/[Your branch name]
50 ***
51
52 ## Validate your changes have been triggered by gitlab-ci in
53 [propitix-scm] (https://gitlab.com/propitix/microservices/frontend-propitix)
54
55 ## Check the image have been pushed to
56 [Google Container Registry] (https://console.cloud.google.com/gcr/images/non-prod-pdz/EU/frontend-propitix?project=non-prod-pdz&authuser=1&gcr1)
57 (Depending on the environment. Either non-prod or prod)
58
59 ## pulling the image
60 ***
61 docker pull eu.gcr.io/$environment/frontend-propitix:$tag-version
62 ***
63
64 ## Running (You can do this step without the pulling the above as it will put down if not found locally)
65 To run the container:
66 ***
67 $ docker run -d eu.gcr.io/$environment/frontend-propitix:$tag-version
68 ***
69
70 Default web root:
71 ***
72 /usr/share/nginx/html
73 ***
74
75 ## If you require permissions to GCP, or Gitlab resources, please talk to dare@propitix.com
76
77 This is nform project 9
78
```

Attach files by dragging & dropping, selecting or pasting them.

Dashboard > Project 9 > #4

 Status

 Changes

 Console Output

 Edit Build Information

 Delete build '#4'

 Polling Log

 Git Build Data

 Previous Build

Build #4 (29 May 2023, 21:57:26)

Keep this build for

Started 44 sec ago

 Add description

Took **3.4 sec**



Build Artifacts

[Expand all](#) [Collapse all](#)

 View

- .dockerignore
- apache-config.conf
- Dockerfile
- html
 - admin_tooling.php
 - create_user.php
 - functions.php
- img
 - .gitkeep
 - grafana.png
 - jenkins.png
 - jfrog.png
 - kibana.png
 - kubernetes.png
 - logo-propitix.png
 - prometheus.png
 - rancher.png
- index.php
- login.php

Dashboard > Project 9 >

Status

</> Changes

Workspace

Build Now

Configure

Delete Project

GitHub Hook Log

Rename

Project Project 9

Add description

Disable Project

Last Successful Artifacts

Expand all Collapse all

View

Permalinks

- Last build (#4), 1 day 22 hr ago
- Last stable build (#4), 1 day 22 hr ago
- Last successful build (#4), 1 day 22 hr ago
- Last completed build (#4), 1 day 22 hr ago

Build History

trend

Filter builds...

#4

29 May 2023, 21:57

#3

29 May 2023, 21:50

#2

29 May 2023, 21:50

#1

29 May 2023, 21:49

Atom feed for all

Atom feed for failures

```
## How to use this repository
The build is automatically triggered by a git push to your feature/[branch]

## First clone the repository to your workstation
...
$ git clone https://gitlab.com/propitix/microservices/php-frontend.git
$ cd frontend-propitix
...

Create a feature branch. # Always start with feature/(name of your branch)
...
git branch -b feature/add-css-style-to-about-us-page

Update the application code in
...
./html/
...

Then add/commit/push to gitlab
...
git status # to see your changes
...

git add --all # If you are satisfied with your changes and willing to push everything. Otherwise, select only the files to add
...

git commit -m "Put some message about this push here"
...

## Push your changes to gitlab, and merge to dev branch
...
git push --set-upstream origin feature/(Your branch name)

### Validate your changes have been triggered by gitlab-ci in
(propitix-scm) (https://gitlab.com/propitix/microservices/frontend-propitix)

### Check the image have been pushed to
[Google Container Registry] (https://console.cloud.google.com/gcr/images/non-prod-pdz/EU/frontend-propitix?project=non-prod-pdz&authuser=1&gcrImageListsize=30) (Depending on the environment. Either non-prod or prod)

## pulling the image
...
docker pull eu.gcr.io/$environment/frontend-propitix:$tag-version
...

## Running (You can do this step without the pulling the above as it will put down if not found locally)
To run the container:
...
$ docker run -d eu.gcr.io/$environment/frontend-propitix:$tag-version

Default web root:
...
/usr/share/nginx/html
...

## If you require permissions to GCP, or Gitlab resources, please talk to dare@propitix.com

Just testing
[ec2-user@ip-172-31-22-132 ~]$
```

