ATTACK DEFENSE

by PentesterAcademy

Name	DevSecOps Pipeline: Nginx Software
URL	https://attackdefense.com/challengedetails?cid=2261
Type	DevSecOps

Important Note: This document illustrates all the important steps required to complete this lab. This is by no means a comprehensive step-by-step solution for this exercise. This is only provided as a reference to various commands needed to complete this exercise and for your further research on this topic. Also, note that the IP addresses and domain names might be different in your lab.

Challenge Description

DevOps practices are to combine software development (Dev) and IT operations (Ops) in order to improve the delivery process. DevOps pipelines are chained tasks and components that run in a sequence to cover different phases of software compilation, packaging, automated testing, and test deployment.

In this lab, we have a simple DevOps pipeline for Nginx webserver. The pipeline consists of the following components (and tasks):

- GitLab server (For hosting code)
- Jenkins server (For integrating all parts: building/testing Nginx, deploying with Ansible, and dynamic testing with Selenium)
- Test server (For test deployment)

It is suggested to play the DevOps focused lab before playing this lab.

DevSecOps refer to introducing security in different stages of the DevOps process. This is done to catch the vulnerabilities/insecurities as soon as possible in the pipeline. In this lab, the pipeline consists of the following components (and tasks):

- Automated Code Review: DevSkim
- Static Code Analysis: Flawfinder

Objective: Run the pipeline and observe/understand the DevSecOps process!

Instructions:

- The GitLab server is reachable with the name 'gitlab'
- Gitlab credentials:

Username	Password
root	welcome123

- The Jenkins server is reachable with the name 'jenkins'
- Jenkins credentials:

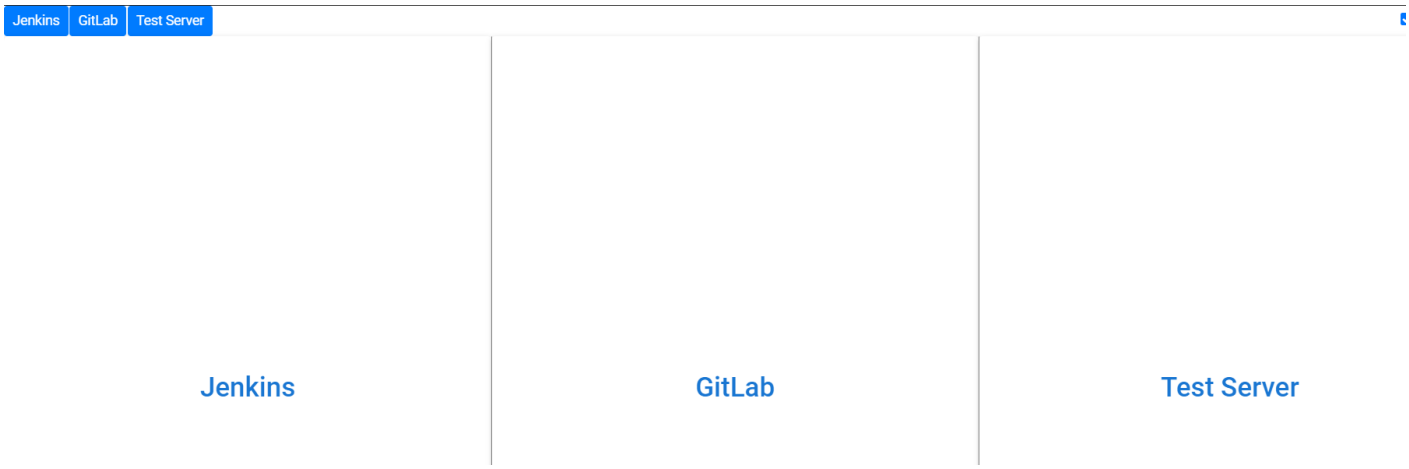
Username	Password
admin	welcome123

- The test deployment server is reachable by the name "test-server"
- Test server credentials:

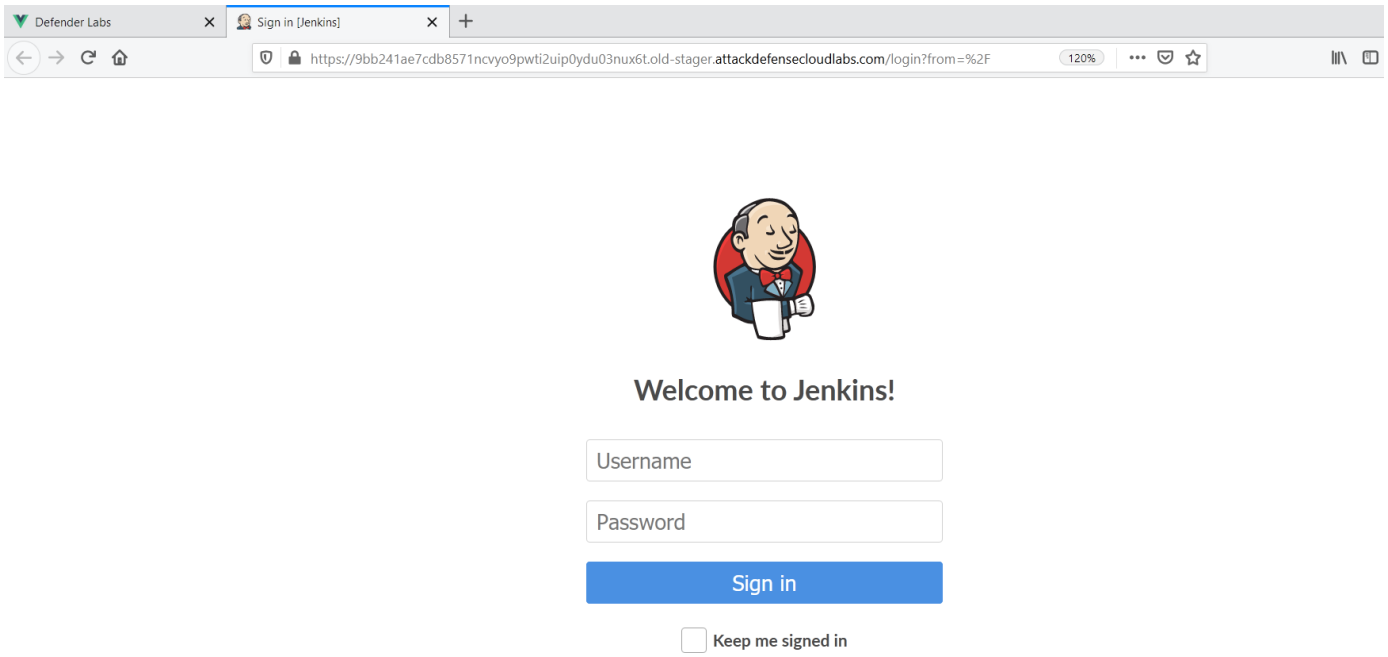
Username	Password
tomcat	password1

Lab Setup

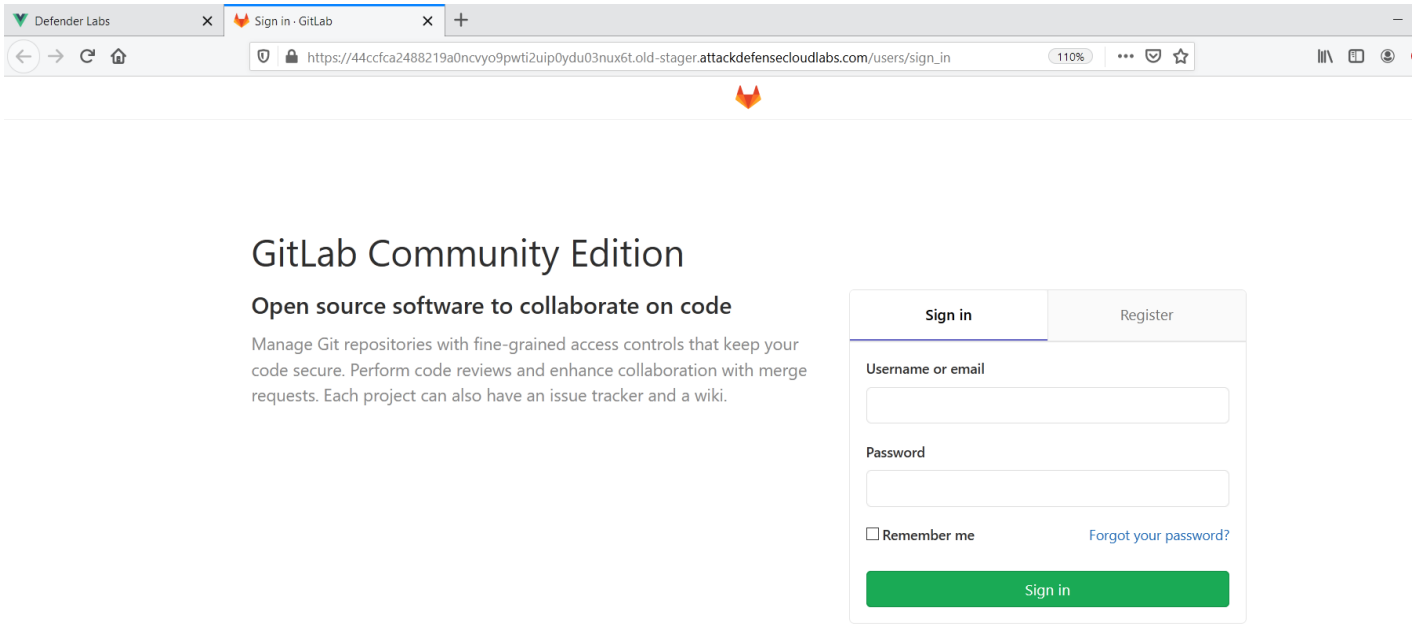
On starting the lab, the following interface will be accessible to the user.



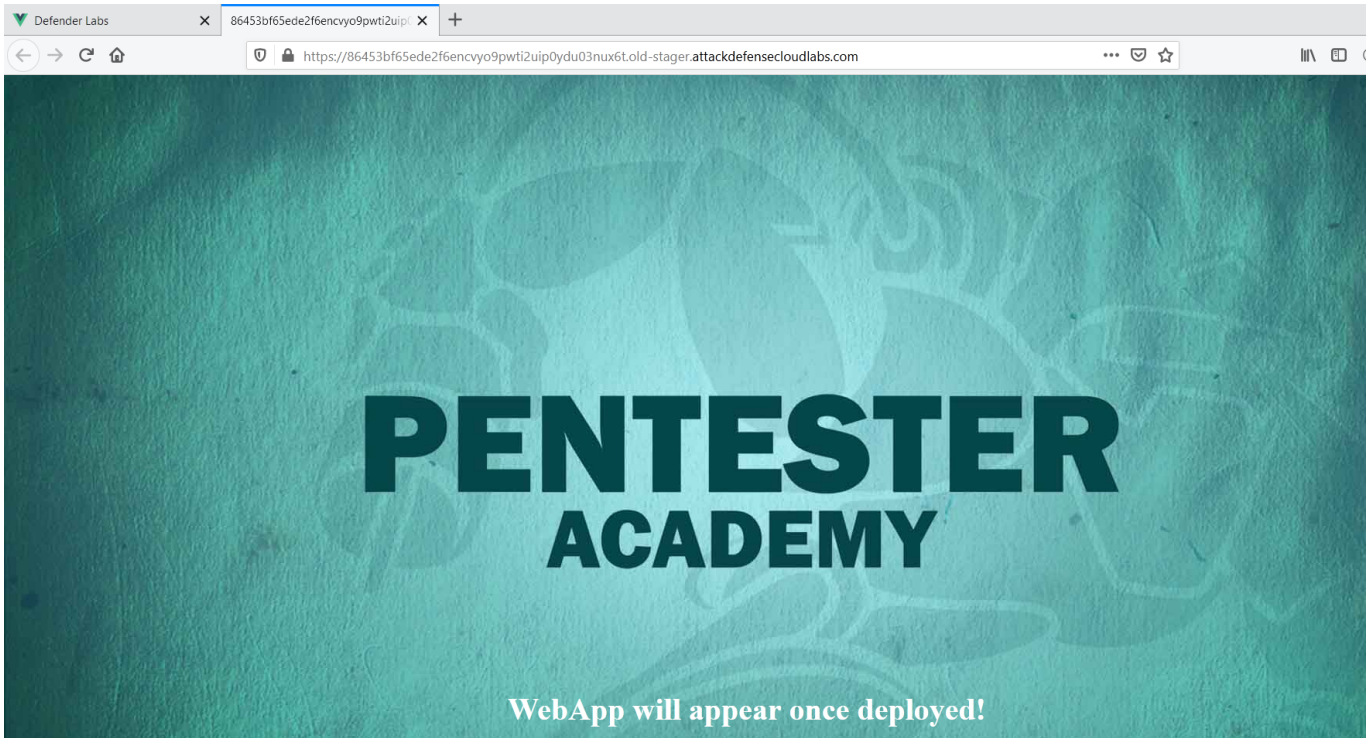
On choosing (clicking the text in the center) left panel, **Jenkins** will open in a new tab



On selecting the middle panel, a web UI of **Gitlab** will open in a new tab.



And on selecting the right panel, a web UI of **Test Server** will open in a new tab.



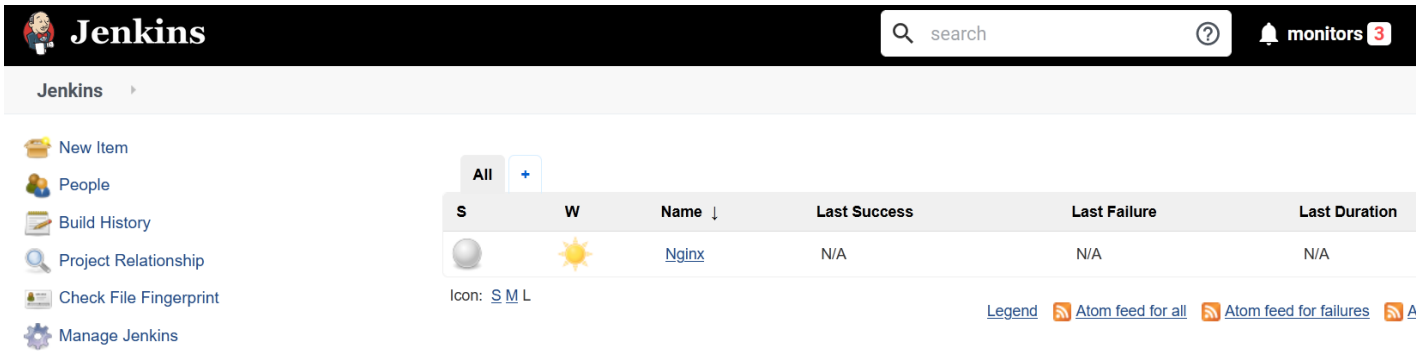
The page will reload until the test-server has started running the web service at port 80

Solution

Step 1: Login into Jenkins. The credentials are provided in the challenge description.

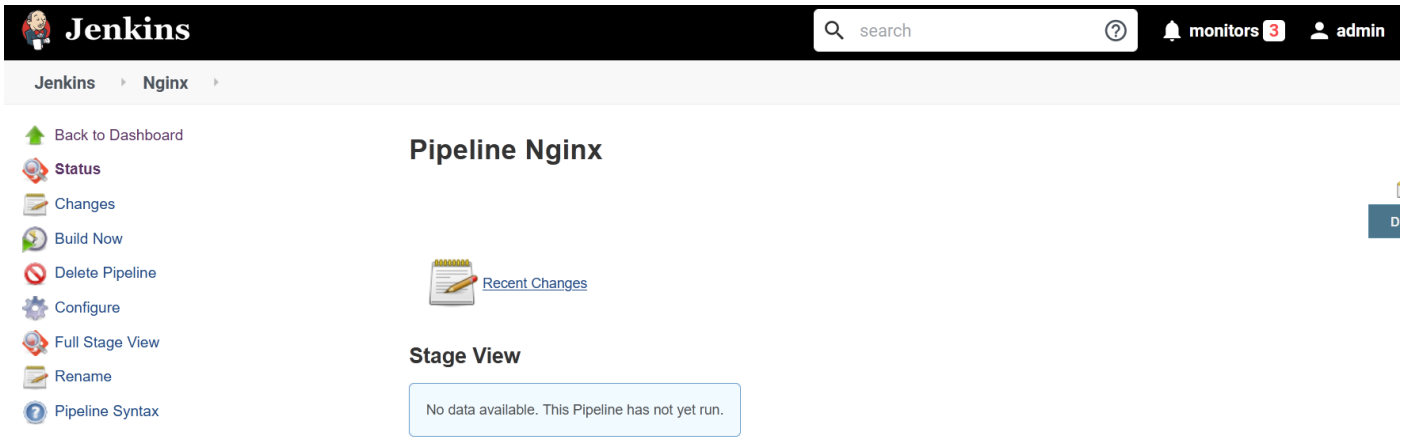
Credentials:

- **Username:** admin
- **Password:** welcome123



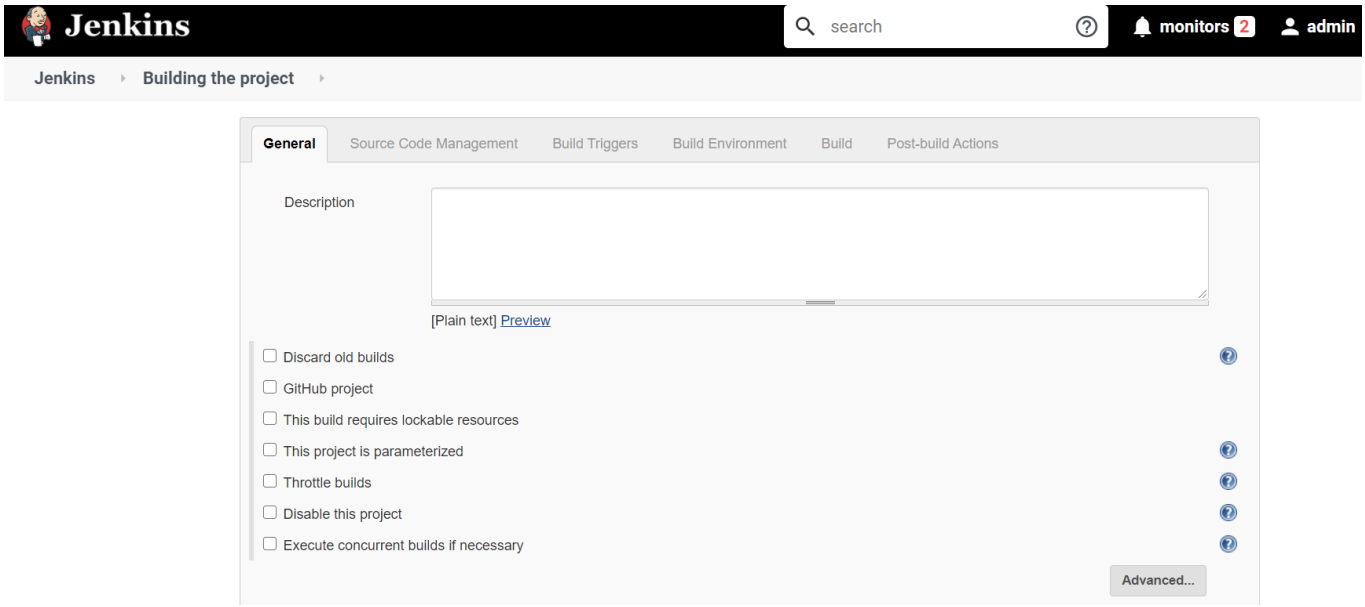
There is only one Job (Nginx) available in the Jenkins instance.

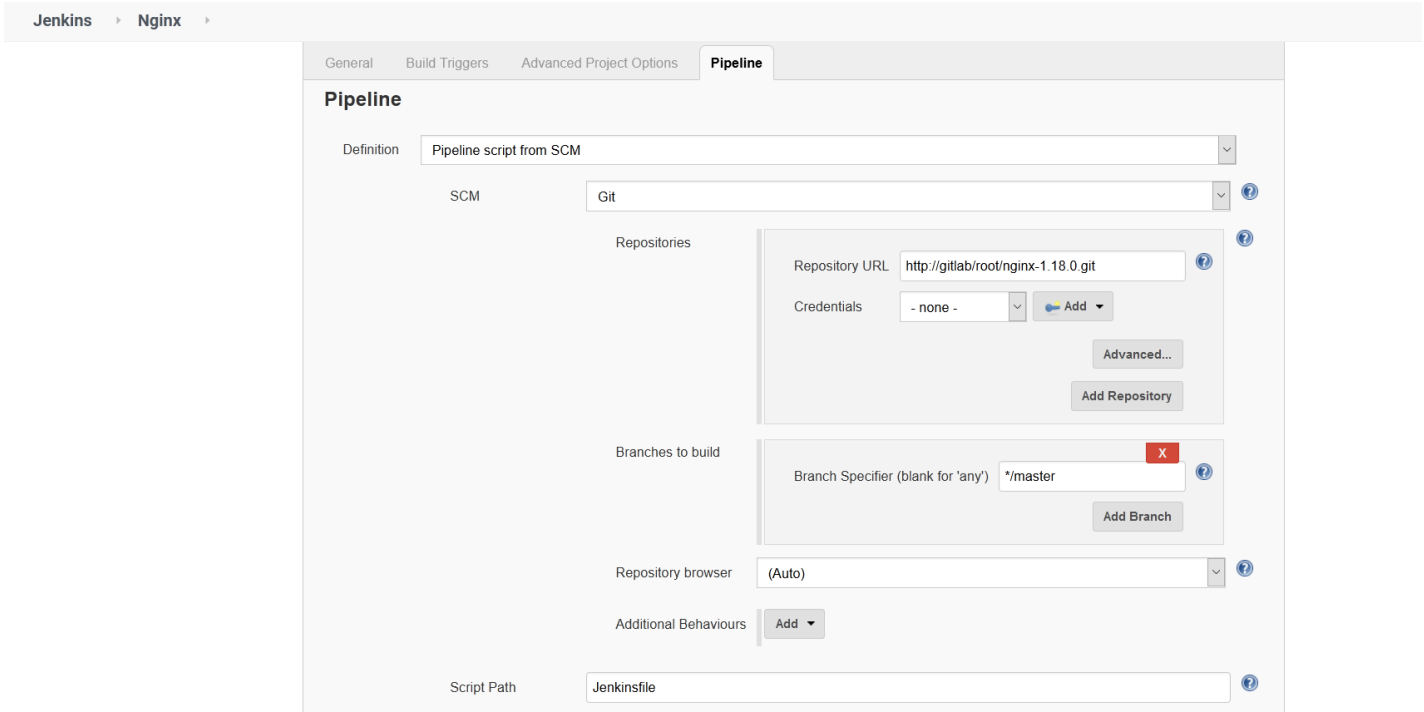
Step 2: Click on the "Nginx" job.



This page is for “Pipeline Nginx” job. The Pipeline is appended in front of the Job name because this is a “Pipeline” type job in which it accepts a ‘Jenkinsfile’ which has all the commands and configuration of the pipeline.

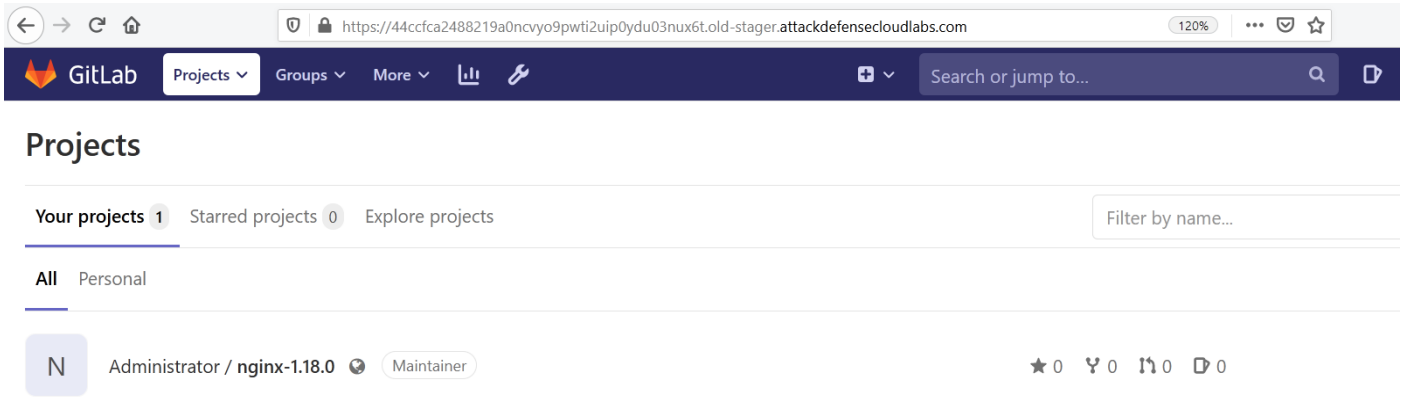
Step 3: Click on the “Configure” option to check the configuration of the Job.





The “Pipeline” sections accept Jenkinsfile directly or a source such as Gitlab where the code and Jenkinsfile are stored for the project.

The code is hosted on GitLab instance at this path “http://gitlab/root/nginx-1.18.0.git”



Step 3: Open the project on Gitlab and check the Jenkinsfile to build the pipeline. (Login into Gitlab using the credentials provided in the description)

Name	Last commit
auto	ADD files
conf	ADD files
contrib	ADD files
html	ADD files
man	ADD files
src	ADD files
CHANGES	ADD files
CHANGES.ru	ADD files
Jenkinsfile	Update Jenkinsfile
LICENSE	ADD files
README	ADD files
configure	ADD files
nginx.yml	Update nginx.yml
selenium_checks.py	Add new file

```
        stage ('Devskim - Scan') {
                        // Shell build step
        sh """
            devskim analyze .
        """
        }

        stage ('Flawfinder - Scan') {
                        // Shell build step
        sh """
            flawfinder .
        """
        }
```

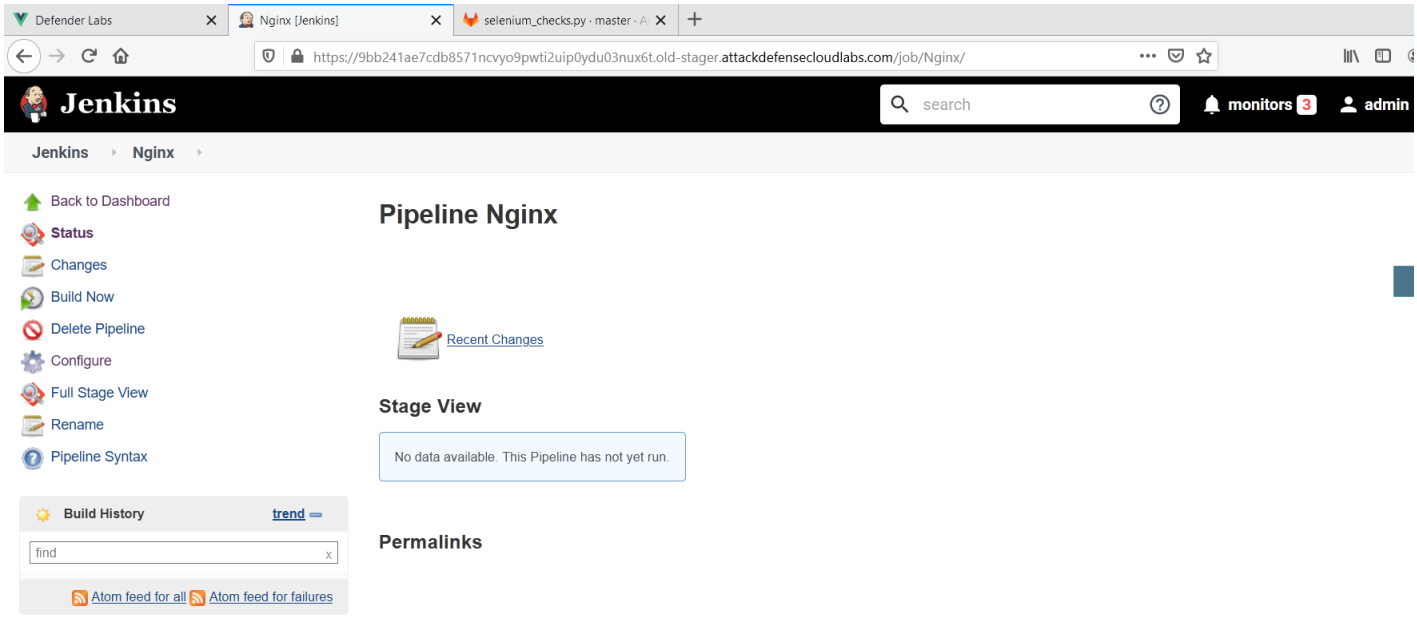
There are 5 stages in the Jenkinsfile, We will take one stage (DevSecOps only) at a time to study as DevOps stages are already covered in the DevOps pipeline lab. Please check that first if you haven't already.

Jenkinsfile Stages:

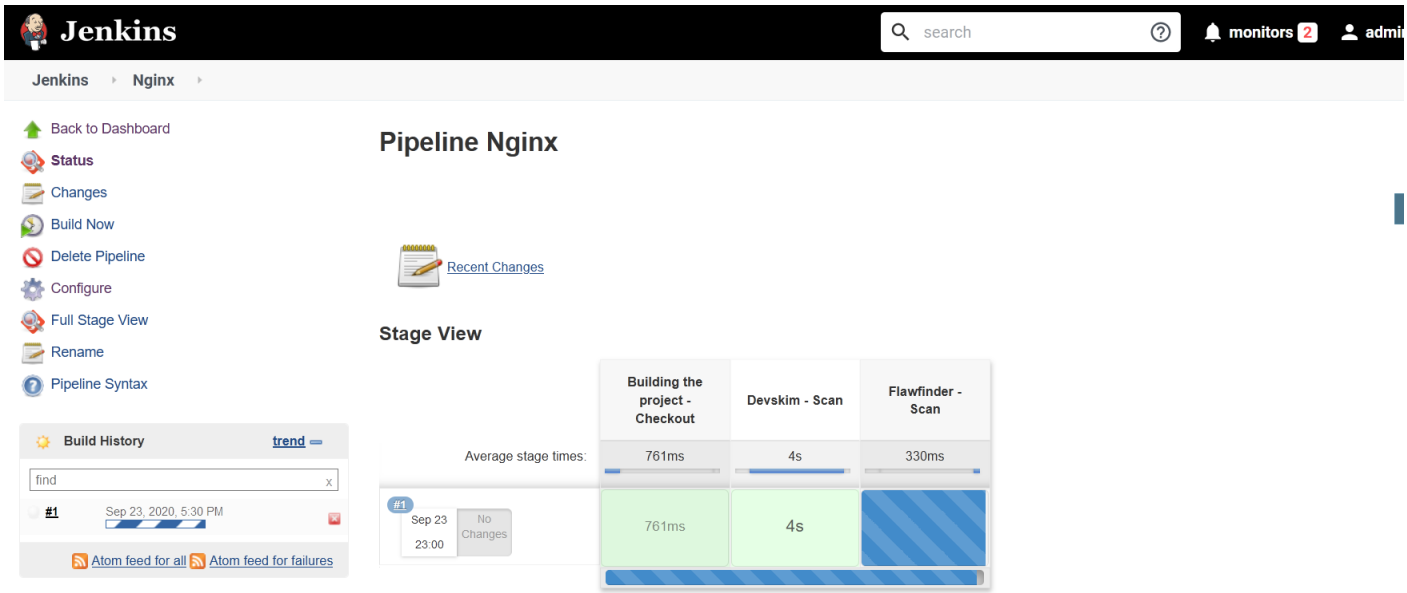
- **DevSkim - Scan:** In this stage, the source code of the application will be reviewed for any vulnerabilities.
- **Flawfinder:** In this stage, the flawfinder will perform static code analysis on the source code of the application to find vulnerabilities in the code.

Pipeline Execution

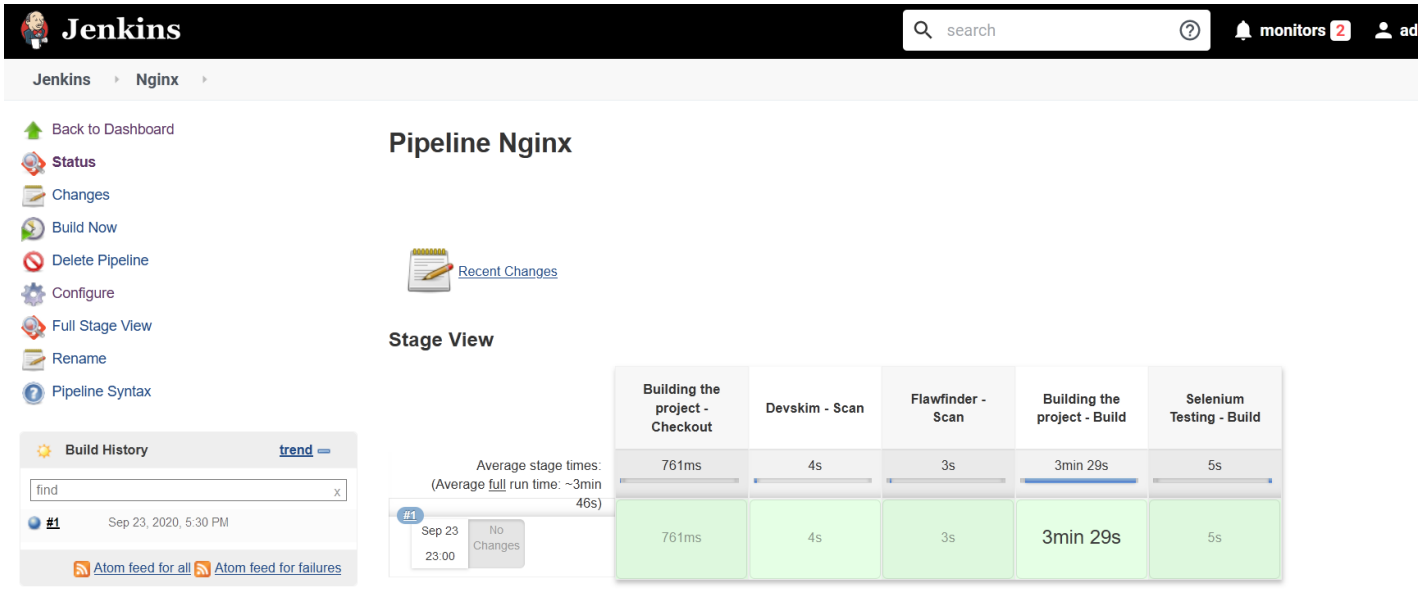
Step 1: Navigate back to the Pipeline tab.



Step 2: Click on the “Build Now” button to start the Pipeline.

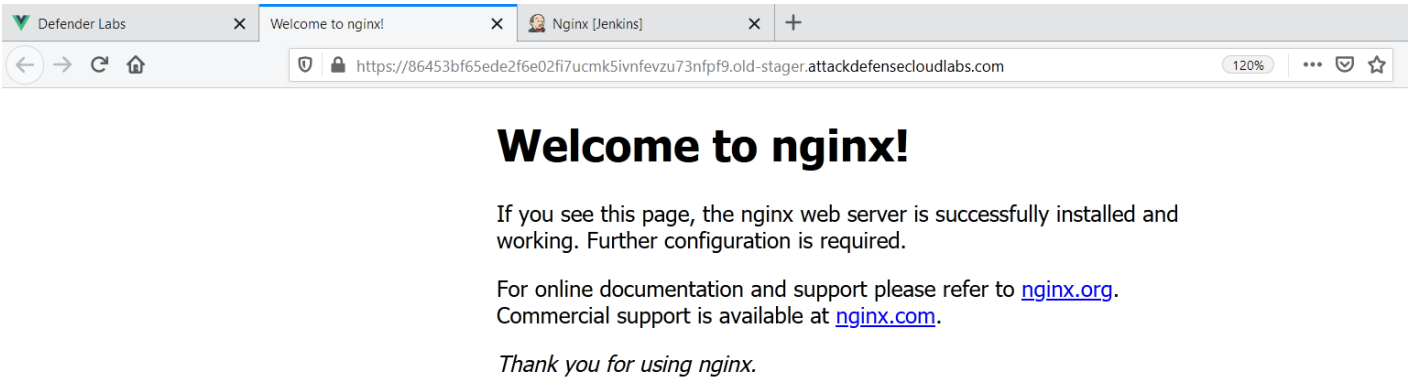


The page will automatically update and show the latest build information about the test-server.



The pipeline completed the execution successfully.

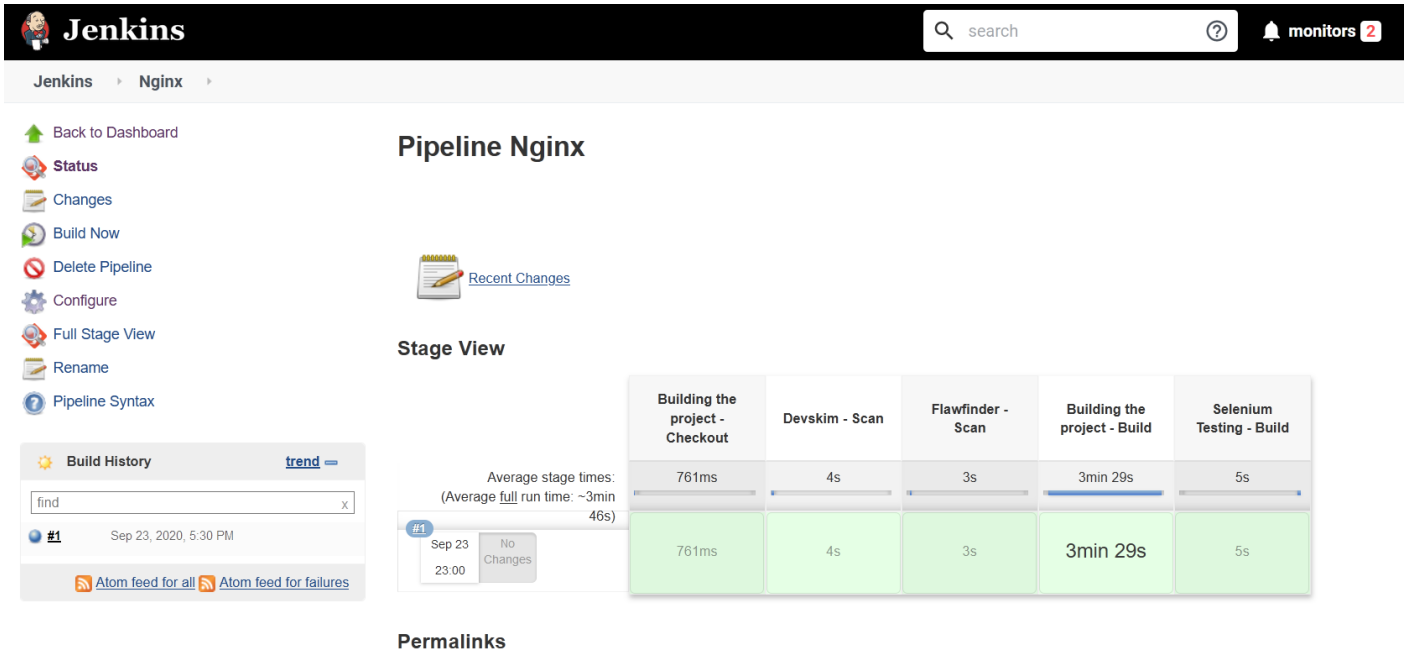
Step 3: Check the Test server.



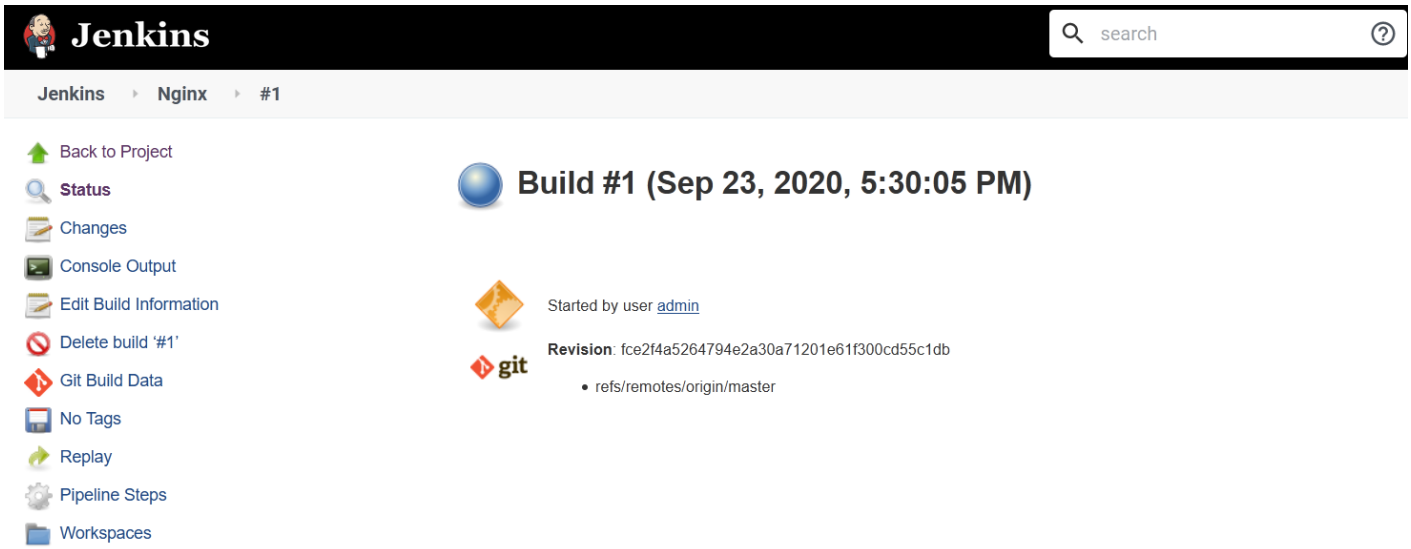
The default page of Nginx is displayed on the test-server which means the installation of Nginx was successful.

Log Review:

Step 1: Navigate to the Nginx Job Panel in Jenkins.



Step 2: Click on the latest build which in this case is #1



Click on the Console Output

Devskim Scan

```
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Devskim - Scan)
[Pipeline] sh
17:30:08  + devskim analyze .
17:30:09  file:./selenium_checks.py
17:30:09        region:6,15,6,20 - DS137138 [Moderate] - Insecure URL
17:30:09
17:30:10  file:./src/http/ngx_http_parse.c
17:30:10        region:713,22,713,27 - DS137138 [Moderate] - Insecure URL
17:30:10        region:1707,17,1707,22 - DS137138 [Moderate] - Insecure URL
17:30:10
17:30:10  file:./src/http/modules/ngx_http_grpc_module.c
17:30:10        region:232,19,232,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:233,19,233,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:234,19,234,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:235,19,235,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:236,19,236,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:237,19,237,24 - DS169125 [Important] - Do not use outdated SSL/TLS protocols
17:30:10        region:232,19,232,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
17:30:10        region:233,19,233,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
17:30:10        region:234,19,234,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
17:30:10        region:235,19,235,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
17:30:10        region:236,19,236,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
17:30:10        region:237,19,237,24 - DS169126 [Important] - An Outdated or Banned SSL/TLS Protocol is Used
```

```
17:30:12  file:./src/mail/ngx_mail_imap_module.c
17:30:12        region:30,50,30,53 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:40,27,40,30 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12
17:30:12  file:./src/mail/ngx_mail.h
17:30:12        region:292,28,292,31 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:300,28,300,31 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12
17:30:12  file:./src/mail/ngx_mail_handler.c
17:30:12        region:567,63,567,66 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:585,39,585,42 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:585,61,585,64 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:592,41,592,44 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12
17:30:12  file:./src/mail/ngx_mail_imap_handler.c
17:30:12        region:385,29,385,32 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12        region:387,55,387,58 - DS126858 [Critical] - Weak/Broken Hash Algorithm
17:30:12
17:30:12  file:./src/os/unix/ngx_errno.c
17:30:12        region:59,23,59,34 - DS161085 [BestPractice] - Problematic C function detected (malloc)
17:30:12        region:68,13,68,24 - DS161085 [BestPractice] - Problematic C function detected (malloc)
17:30:12        region:83,24,83,35 - DS161085 [BestPractice] - Problematic C function detected (malloc)
17:30:12        region:65,15,65,23 - DS154189 [Moderate] - Banned C function detected
17:30:12        region:83,64,83,72 - DS154189 [Moderate] - Banned C function detected
17:30:12
17:30:12  file:./src/os/unix/ngx_time.c
17:30:12        region:28,9,28,15 - DS154189 [Moderate] - Banned C function detected
17:30:12        region:47,9,47,18 - DS154189 [Moderate] - Banned C function detected
17:30:12        region:64,9,64,18 - DS154189 [Moderate] - Banned C function detected
17:30:12        region:83,9,83,18 - DS154189 [Moderate] - Banned C function detected
17:30:12        region:99,9,99,15 - DS154189 [Moderate] - Banned C function detected
17:30:12
17:30:12  file:./src/os/unix/ngx_alloc.c
17:30:12        region:21,9,21,21 - DS161085 [BestPractice] - Problematic C function detected (malloc)
17:30:12        region:24,24,24,35 - DS161085 [BestPractice] - Problematic C function detected (malloc)
17:30:12
```

Issues Detected:

- Outdated SSL/TLS protocols
- Insecure URL
- Weak/Broken Hash Algorithm
- Banned C function
- Problematic C function (malloc)

Flawfinder Scan

```
17:30:16  FINAL RESULTS:
17:30:16
17:30:16  ./src/core/ngx_connection.c:635:  [5] (race) chmod:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchmod( ) instead.
17:30:16  ./src/core/ngx_cycle.c:1216:  [5] (race) chown:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchown( ) instead.
17:30:16  ./src/core/ngx_cycle.c:1235:  [5] (race) chmod:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchmod( ) instead.
17:30:16  ./src/core/ngx_file.c:632:  [5] (race) chown:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchown( ) instead.
17:30:16  ./src/core/ngx_file.c:645:  [5] (race) chmod:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchmod( ) instead.
17:30:16  ./src/os/unix/ngx_files.h:165:  [5] (race) chmod:
17:30:16    This accepts filename arguments; if an attacker can move those files, a
17:30:16    race condition results. (CWE-362). Use fchmod( ) instead.
17:30:16  ./src/core/ngx_file.c:115:  [4] (race) access:
17:30:16    This usually indicates a security flaw. If an attacker can change anything
17:30:16    along the path between the call to access() and the file's actual use
17:30:16    (e.g., by moving files), the attacker can exploit the race condition
17:30:16    (CWE-362/CWE-367!). Set up the correct permissions (e.g., using setuid())
17:30:16    and try to open the file directly.

17:30:16    Statically-sized arrays can be improperly restricted, leading to potential
17:30:16    overflows or other issues (CWE-119!/CWE-120). Perform bounds checking, use
17:30:16    functions that limit length, or ensure that the size is larger than the
17:30:16    maximum possible length.
17:30:16  ./src/core/ngx_connection.c:1049:  [1] (buffer) read:
17:30:16    Check buffer boundaries if used in a loop including recursive loops
17:30:16    (CWE-120, CWE-20).
17:30:16  ./src/core/ngx_connection.c:1052:  [1] (buffer) read:
17:30:16    Check buffer boundaries if used in a loop including recursive loops
17:30:16    (CWE-120, CWE-20).
17:30:16  ./src/core/ngx_connection.c:1131:  [1] (buffer) read:
17:30:16    Check buffer boundaries if used in a loop including recursive loops
17:30:16    (CWE-120, CWE-20).

17:30:16  ./src/core/ngx_string.h:61:  [1] (buffer) strlen:
17:30:16    Does not handle strings that are not \0-terminated; if given one it may
17:30:16    perform an over-read (it could cause a crash if unprotected) (CWE-126).
17:30:16  ./src/event/modules/ngx_devpoll_module.c:262:  [1] (buffer) read:
17:30:16    Check buffer boundaries if used in a loop including recursive loops
17:30:16    (CWE-120, CWE-20).
17:30:16  ./src/event/modules/ngx_devpoll_module.c:278:  [1] (buffer) read:
17:30:16    Check buffer boundaries if used in a loop including recursive loops
17:30:16    (CWE-120, CWE-20).
17:30:16  ./src/event/modules/ngx_devpoll_module.c:494:  [1] (buffer) read:
```

```
17:30:16  Hits = 303
17:30:16  Lines analyzed = 192172 in approximately 2.87 seconds (66888 lines/second)
17:30:16  Physical Source Lines of Code (SLOC) = 137449
17:30:16  Hits@level = [0]  31 [1] 176 [2]  51 [3]   9 [4]  61 [5]   6
17:30:16  Hits@level+ = [0+] 334 [1+] 303 [2+] 127 [3+]  76 [4+]  67 [5+]   6
17:30:16  Hits/KSLOC@level+ = [0+] 2.42999 [1+] 2.20445 [2+] 0.923979 [3+] 0.552932 [4+] 0.487454 [5+] 0.0436526
17:30:16  Dot directories skipped = 1 (--followdotdir overrides)
17:30:16  Minimum risk level = 1
17:30:16  Not every hit is necessarily a security vulnerability.
17:30:16  There may be other security vulnerabilities; review your code!
17:30:16  See 'Secure Programming HOWTO'
17:30:16  (https://dwheeler.com/secure-programs) for more information.
```

Issues Detected:

- Race Condition (CWE-362)
- Buffer Overflows (CWE-120)

Learning

Working of a simple DevSecOps pipeline consisting of different components.