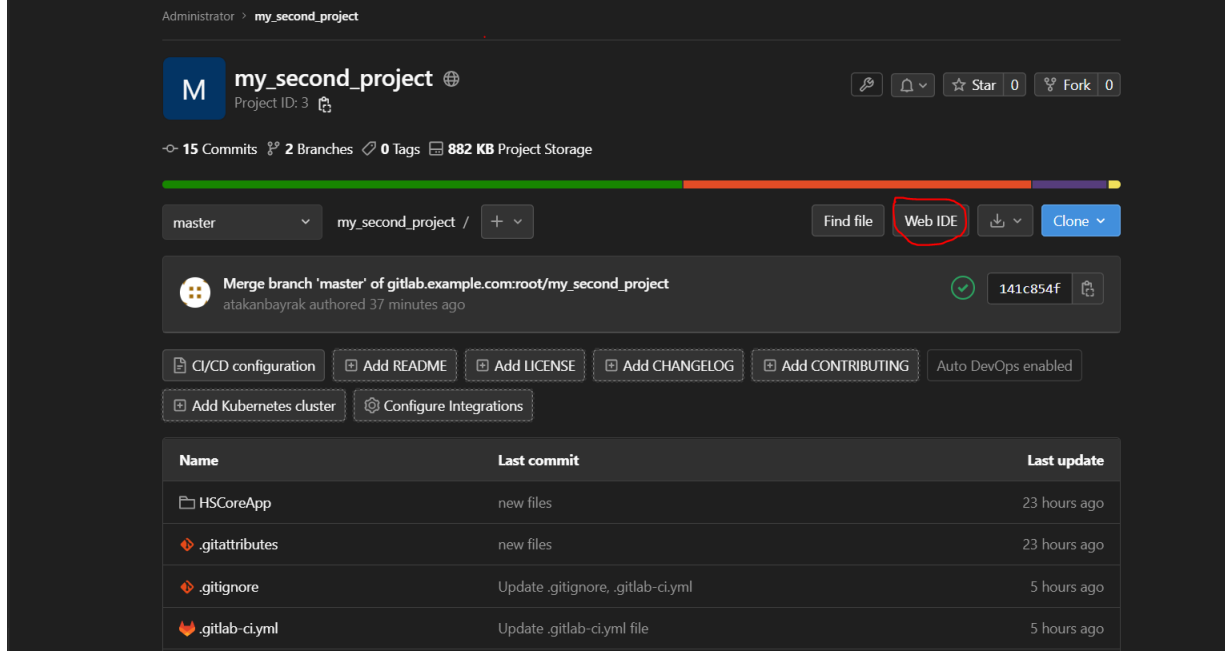
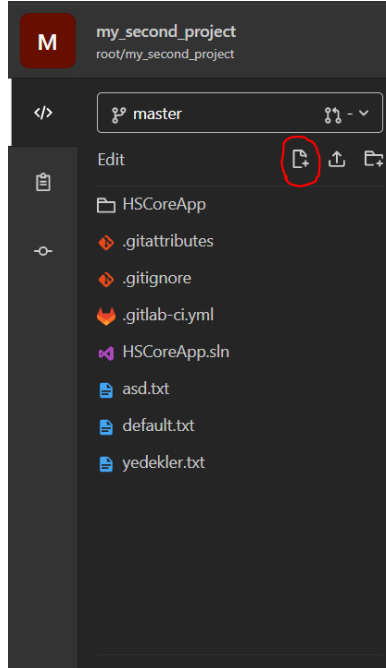


GitLab Pipeline Kullanımı (Ubuntu 20.04 Server)

Herhangi bir projeyi server üzerine başarılı bir şekilde yükledikten sonra projemizi WEB IDE olarak açıyoruz.



Burada sol taraftan “New File” diyerek bir “.gitlab-ci.yml” dosyası oluşturuyoruz.



Bu adımda dikkat edilmesi gereken en önemli nokta bu dosyanın proje dosyalarınızla aynı dizinde olması. Eğer proje dosyanızın dışında bırakırsanız pipeline ekranında **“out of workflow”** hatası ile karşılaşabilirsiniz.

Bu oluşturduğumuz dosya bizim pipeline kurallarını yazacağımız temel dosyamız.

Temel olarak ekran görüntüsündeki bir tanımlama yapabiliriz.

```
1  ∨ stages:
2    | - test
3
4  ∨ test_stage:
5    stage: test
6
7  ∨ tags:
8    - testing
9
10 ∨ script:
11   - echo "Proje"
12   - ls
13
14
15
16
17
18
19
```

Bu tanımlama bize temel olarak bir test yaptırmış olacak. Ve ekrana proje çıktısını verdikten sonra dosyaların bir listesini yazacak.

Server’ımıza Gitlab Runner Ekleme ve Register İşlemleri

Bu adımda ilk önce ayarlarınızdan **“Settings”**, **“CI/CD”** ardından **“Runners”** kısmını **“Expand”** yapmalısınız.

Specific runners

These runners are specific to this project.

Set up a specific runner for a project

1. Install GitLab Runner and ensure it's running.
2. Register the runner with this URL:
<http://gitlab.example.com/>

And this registration token:
`GR1348941mz-p1VFajTgMa2qgWAr`

[Reset registration token](#)

[Show runner installation instructions](#)

Available specific runners

#2 (snrrvnu)

runner2

testing

[Remove runner](#)

Shared runners

These runners are shared across this GitLab instance.

The same shared runner executes code from multiple projects, unless you configure autoscaling with [MaxBuilds](#) set to 1 (which it is on GitLab.com).

Enable shared runners for this project

☒ This GitLab instance does not provide any shared runners yet. Instance administrators can register shared runners in the admin area.

Group runners

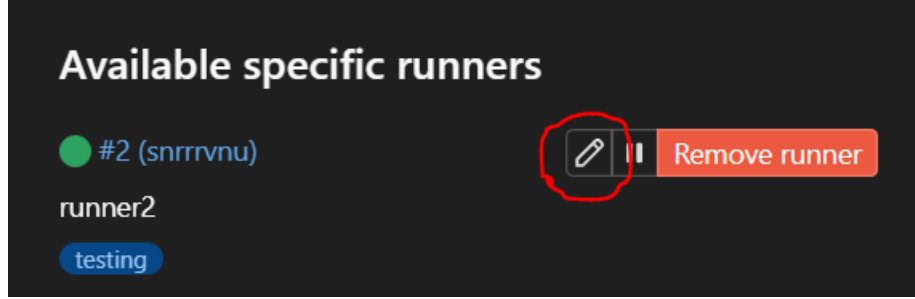
These runners are shared across projects in this group.

Group runners can be managed with the [Runner API](#).

This project does not belong to a group and cannot make use of group runners.

Burada GitLab bizim için zaten bir açıklama kısmı ayırmış. Buradaki adımları izleyerek `**"Runner**"` konfigürasyonumuzu server içerisine yapıyoruz.

Bu adımdan sonra ilk olarak runner'ın kenarındaki küçük kalem ifadesine basarak ayarlar sekmesini açıyoruz.



Burada ayarları gösterildiği gibi ayarlamamız gerek:

- 1 Runner'ımızın aktif olduğundan emin olmamız lazım.
- 2 Diğer tikli olan ifade ise herhangi bir pipeline'a tag atarsakta atamasakta runner'ın bu pipeline ile çalışabileceğini ifade ediyor.
- 3 En altta istersek runner için bir tag atayabiliriz. Bu tag runner'ın spesifik pipeline'lar ile çalışması noktasında önem arz ediyor.

This runner is associated with specific projects.
You can set up a specific runner to be used by multiple projects but you cannot make this a shared runner. [Learn more.](#)

Active

☒ Paused runners don't accept new jobs

Protected

☐ This runner will only run on pipelines triggered on protected branches

Run untagged jobs

☒ Indicates whether this runner can pick jobs without tags

Lock to current projects

☐ When a runner is locked, it cannot be assigned to other projects

IP Address

127.0.0.1

Description

runner2

Maximum job timeout

Enter the number of seconds, or other human-readable input, like "1 hour". This timeout takes precedence over lower timeouts set for the project.

Tags

testing

You can set up jobs to only use runners with specific tags. Separate tags with commas.

Save changes

Bu adımları tamamladıktan sonra PuTTY konsolumuz üzerinden runner'ın register edilmiş ve çalışır durumda olmasına dikkat ediyoruz.

```
$ sudo gitlab-runner register \  
  --non-interactive \  
  --url "BURAYA URL ADRESİ YAZILACAK" \  
  --registration-token "BURAYA TOKEN YAZILACAK" \  
  --description "docker-runner" \  
  --executor "docker" \  
  --docker-image ubuntu:latest
```

Register ekranında doldurulacak bilgileri GitLab bizim için Runner ekranında veriyor.

Bu adımdan sonra:

```
$ sudo gitlab-runner run
```

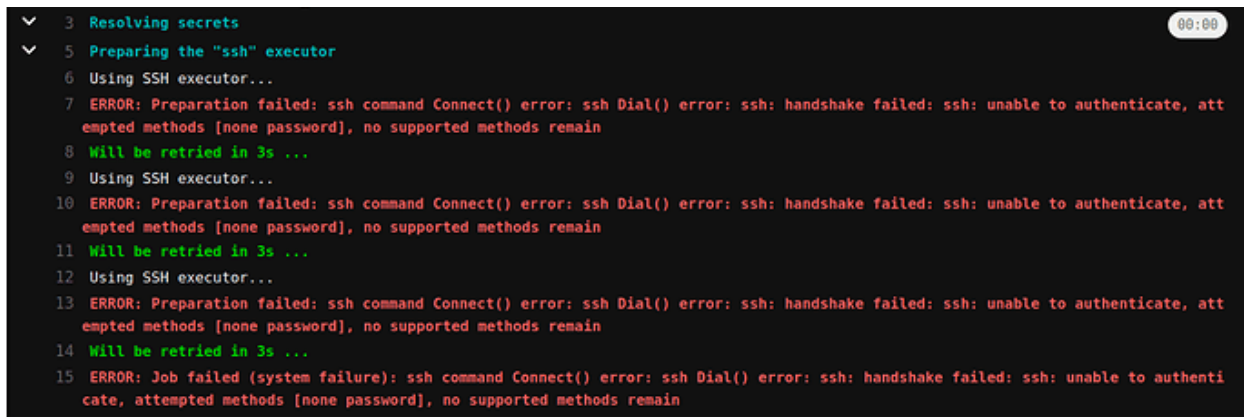
komutunu çalıştırarak runnerımızı aktif hale getiriyoruz. GitLab üzerinde runner ekranında eğer runner'ın yanında yeşil renk varsa runner çalışıyor demektir. Artık pipeline ekranına geçebiliriz.

“Pipelines” ekranında **“Run Pipeline”** dedikten sonra pipeline'ımızı çalıştırıyoruz.

Eğer;

```
ERROR: Preparation failed: ssh command Connect() error: ssh Dial() error: ssh: handshake failed: ssh: unable to authenticate, attempted methods [none password publickey], no supported methods remain
```

hatası alırsanız;



Yapmanız gereken şey server üzerinden config.toml dosyasını bularak içerisinde username ve password kısmını kontrol etmek.

Genellikle;

```
$ /etc/gitlab-runner/config.toml
```

dizinde bulunuyor.

Eğer;

ERROR: Job failed: prepare environment: Process exited with status 1. Check <https://docs.gitlab.com/runner/shells/index.html#shell-profile-loading> for more information

hatası alırsanız;

```
1 Running with gitlab-runner 13.1.1 (6fbc7474)
2   on test-runner XnPwKDMR
3   ✓ Preparing the "virtualbox" executor 00:15
4   Using VirtualBox version 6.1.12r139181 executor...
5   Restoring VM from snapshot...
6   Starting VM...
7   Waiting VM to become responsive...
8   Starting SSH command...
9   ✓ Preparing environment 00:00
10  Running on cuinjune-VirtualBox via Zackui-MacBook-Pro.local...
11  13 ERROR: Job failed (system failure): prepare environment: Process exited with status 1. Check https://docs.gitlab.com/runner/shells/index.html#shell-profile-loading for more information
```

Yapmanız gereken şey

```
$ find / -name .bash_logout
```

komutunu çalıştırıp ardından

```
$ sudo rm -r /home/gitlab-runner/.bash_logout
```

```
$ sudo rm -r /home/<username>/.bash_logout
```

komutlarını çalıştırmak. Bu adımları da bitirdikten sonra test uygulamanız görüntüdeki gibi çalışıyor olması gerek.

```
1 Running with gitlab-runner 15.2.1 (32fc1585)
2   on runner2 snrrrvnu
3   ✓ Preparing the "ssh" executor 00:00
4   Using SSH executor...
5   ✓ Preparing environment 00:01
6   Running on atakan via atakan...
7   ✓ Getting source from Git repository 00:01
8   Fetching changes with git depth set to 20...
9   Reinitialized existing Git repository in /home/atakan/builds/snrrrvnu/0/root/my_second_project/.git/
10  Checking out 141c854f as master...
11  Skipping Git submodules setup
12  ✓ Executing "step_script" stage of the job script 00:00
13  $ echo "Proje"
14  Proje
15  $ ls
16  asd.txt
17  default.txt
18  HSCoreApp
19  HSCoreApp.sln
20  yedekler.txt
21  Job succeeded
```

Bu işlemle beraber **“master”** üzerine yapılan bütün commitlerden sonra bu pipeline çalışarak içerisinde bulunan işlemleri yapacaktır.