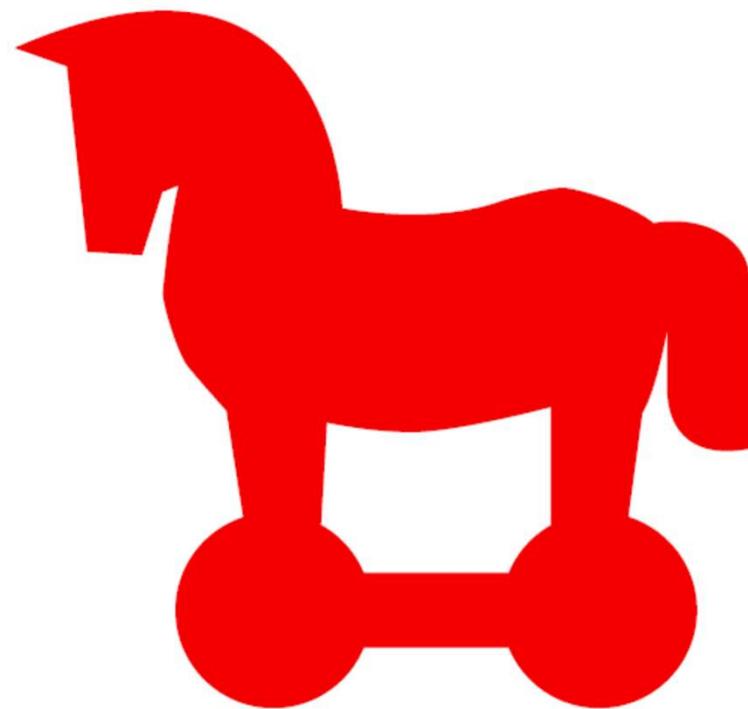


**GANTEK**  
INTEGRATING FUTURE

**GANTEK**  
INTEGRATING FUTURE

**GANTEK**  
INTEGRATING FUTURE



# **GANTEK ACADEMY**

**40 YILLIK TECRÜBE İLE**



**GITLAB**

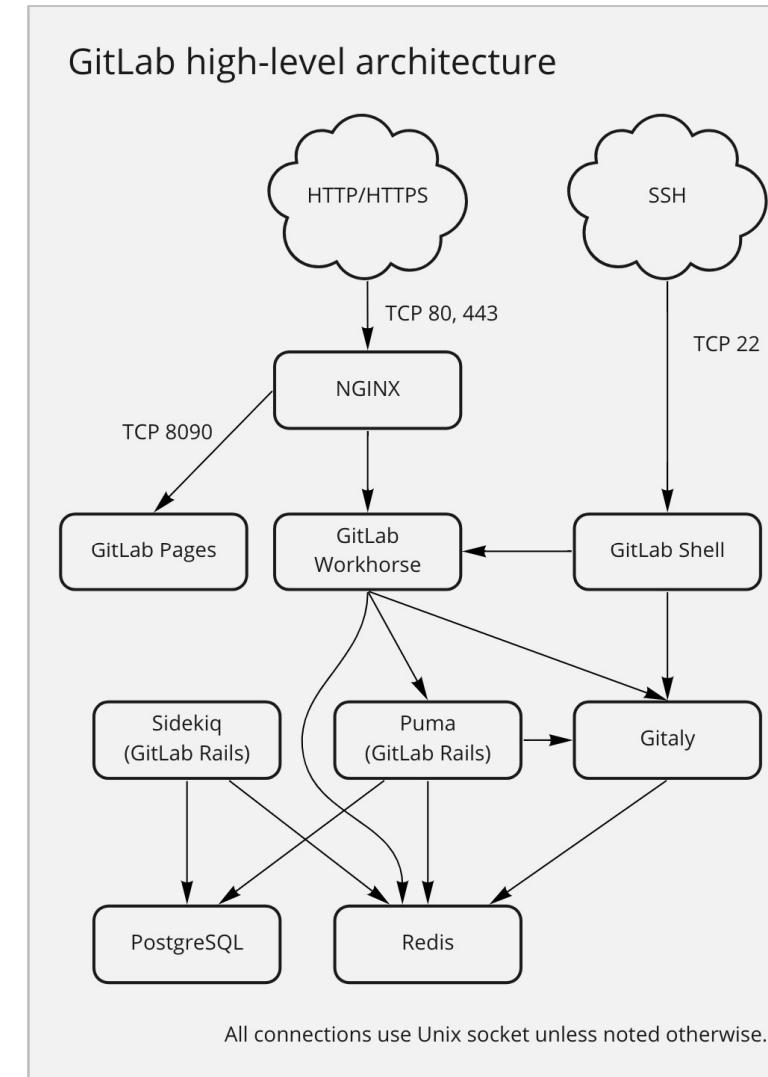
# Gitlab nedir?

- Github'a benzer bir sistemdir.
- Gitlab sunucunu, kendi sisteminizde lokalde veya konteyner olarak kurabilirsiniz.
- Açık kökenli projedir. Ücretsiz olarak «toplum» sürümü kullanılabilir. Daha çok hizmetin ve özelliğin sunulduğu «kurumsal» sürümü ücretlidir ve destek hizmeti sağlanmaktadır.
- Gitlab içinde şu sistemler bulunur:
  - Git
  - NGINX
  - PostgreSQL
  - Chef
  - Redis
- İçinde CI/CD araçları bulunur. Github için kendiniz kurmanız gereklidir.

<https://about.gitlab.com/install/?version=ce>

# Gitlab Mimarisi

- Gitlab, Ruby dilinde yazıldıktan daha sonra, önemli bir kısmı Go programlama dilinde yeniden yazılmıştır. Yine de Ruby dilinde yazılmış bileşenlere sahiptir.
- Web arayüzü NGINX üzerinden sunulmaktadır.
- **sidekiq** zamanlama işlerini yürütür.
- Veritabanı olarak **PostgreSQL**, açık kaynak kodlu veritabanı sunucusu kullanılır.
- **Redis**, bellek veritabanı ise **Sidekiq** hem de Gitlab motoru tarafından iş sıralarını düzenlemek için veri depolama amacıyla kullanılır.
- **Gitaly**, git depolarına okuma yazma imkanı sağlar.



<https://docs.gitlab.com/ee/development/architecture.html>

# Gitlab kurulumu

- Gitlab'ın kurumsal sürümünün lokal sunucuya kurulması için yapılacaklar aşağıdaki web sitesinde anlatılmaktadır:

`https://about.gitlab.com/install/`

- Toplum sürümünün kurulması için kullanılacak adres şudur:

`https://about.gitlab.com/install/?version=ce`

- Gitlab'ın kullandığı paketlerin kurulması gerekmektedir:

`$ sudo apt update`

`sudo apt install -y curl openssh-server ca-certificates tzdata perl`

`sudo apt-get install -y postfix`

- GitLab paketinin kod deposunu APT'ye ekliyoruz:

`$ curl -sS`

`https://packages.gitlab.com/install/repositories/gitlab/gitlab-ce/script.deb.sh | sudo bash`

# Gitlab kurulumu

- Gitlab'ın kuruluşu için geçerli bir konak ismine ihtiyaç duyulur. Bu nedenle aşağıdaki komutta «**gitlab.example.com**» adresi yerine kullanacağınız adresin bulunmasına dikkat edin. Ayrıca **???????** yazan yere güçlü bir şifre yazmayı unutmayın.
  - `sudo GITLAB_ROOT_PASSWORD="???????"  
EXTERNAL_URL="https://gitlab.example.com" apt-get  
install gitlab-ce`
- Gitlab'ın kurulumu sırasında «**alertmanager**», «**gitaly**», «**grafana**», «**nginx**», «**postgresql**», «**prometheus**», «**puma**», «**redis**», «**sidekiq**» gibi ek yazılımlara ihtiyaç duyulur. Bir ya da birkaç tanesinin kurulumunda arıza çıkması, Gitlab'ın kurulumunu engeller.

<https://docs.gitlab.com/omnibus/settings/configuration.html>

# Gitlab kurulumu



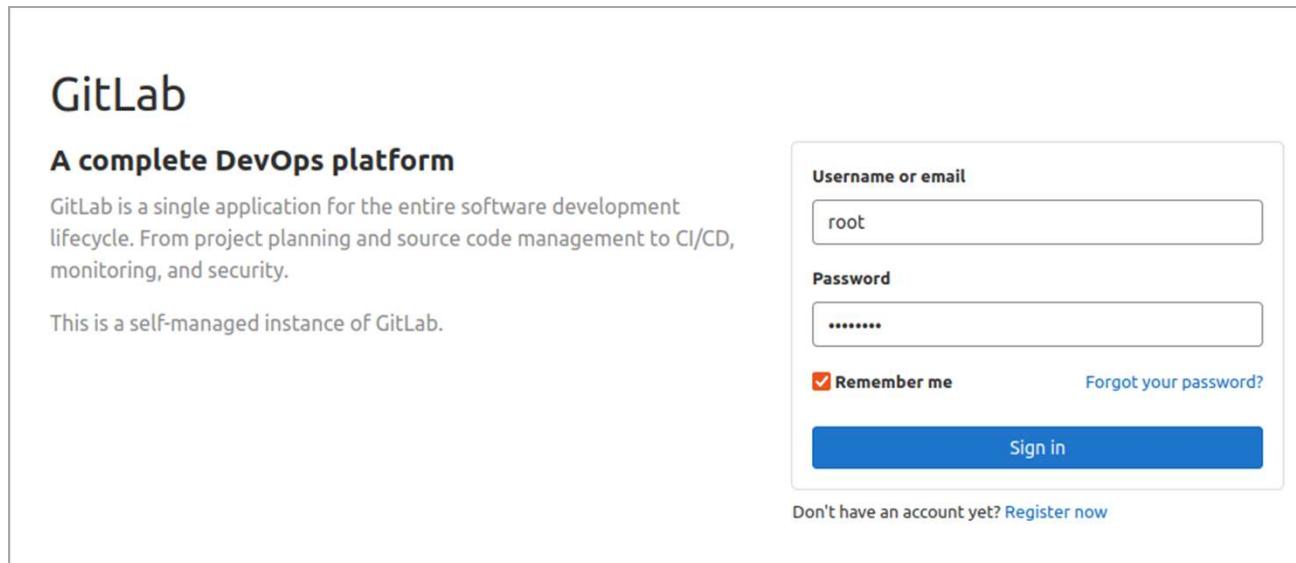
- Eğer kurulum sırasında bir hata çıkarsa, «**sudo gitlab-ctl reconfigure**» komutu verilerek yeniden yapılandırma yapılır ve **gitlab-ce**'nin kurulumu bir önceki komutla yeniden başlatılabilir.
  - Eğer Gitlab sunucusu çalışmıyorsa, «**sudo gitlab-ctl restart**» denilerek, Gitlab sunucusu yeniden başlatılabilir.
- ★ Kurulum sırasında «**root**» şifresi girilmesi unutulduysa veya hiç verilmediyse, şifrenin sıfırlanması ve yeniden girilmesi için aşağıdaki komut kullanılabilir:
- sudo gitlab-rake "gitlab:password:reset"**

<https://docs.gitlab.com/omnibus/settings/configuration.html>

[https://docs.gitlab.com/ee/security/reset\\_user\\_password.html#reset-your-root-password](https://docs.gitlab.com/ee/security/reset_user_password.html#reset-your-root-password)

# Gitlab kurulumu

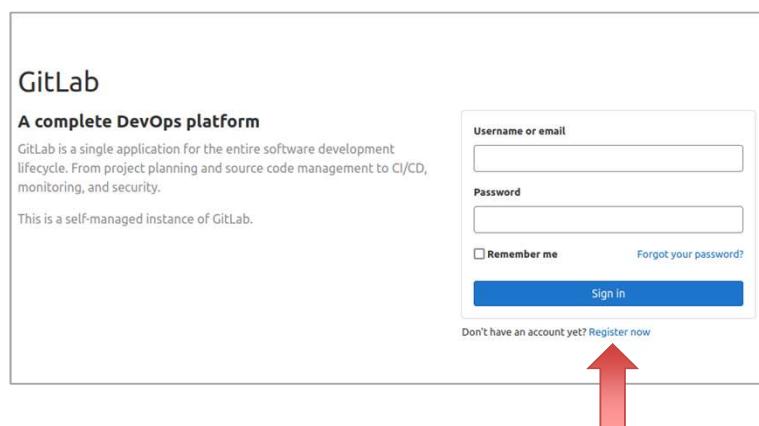
- Kurulum tamamlandıktan sonra Firefox'ta URL olarak «<https://127.0.0.1/>» adresi yazılmalıdır. Eğer SSL sertifikası bulunmuyorsa, Firefox güvenlik uyarısı verecektir. «**Accept the risk and Continue**» opsyonu seçilmelidir.
- Yönetici olan **root** ve şifresi yazıldıkten sonra panoya giriş yapılır.



<https://docs.gitlab.com/omnibus/settings/configuration.html>

# Yeni Kullanıcı Oluşturma

- Gitlab, kullanıcıların sisteme kayıt yaptırmamasına izin vermektedir.
- Kullanıcı eğer isterse, giriş ekranında «**Sign up**» butonuna basarak kendi bilgilerini yazabilir. Böyle bir durumda, Gitlab «**root**» kullanıcısının onaylaması gerekmektedir. Onaylanana kadar kullanıcı giriş yapamayacaktır.
- Örnek olarak **hgsenel** isimli bir kullanıcıyı kayıt yaptıralım.

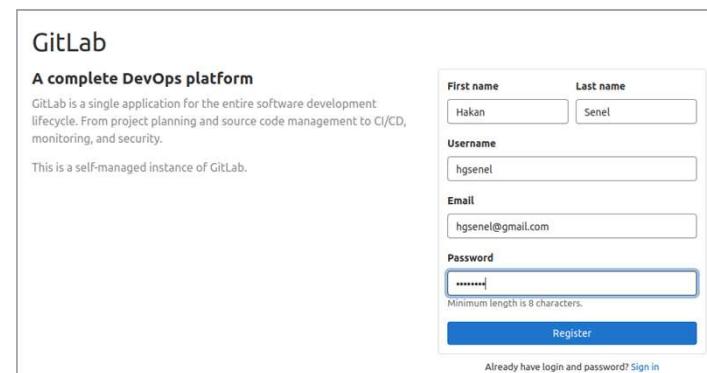


GitLab  
A complete DevOps platform  
GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.  
This is a self-managed instance of GitLab.

Username or email  
Password  
 Remember me      [Forgot your password?](#)

[Sign in](#)

Don't have an account yet? [Register now](#)



GitLab  
A complete DevOps platform  
GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security.  
This is a self-managed instance of GitLab.

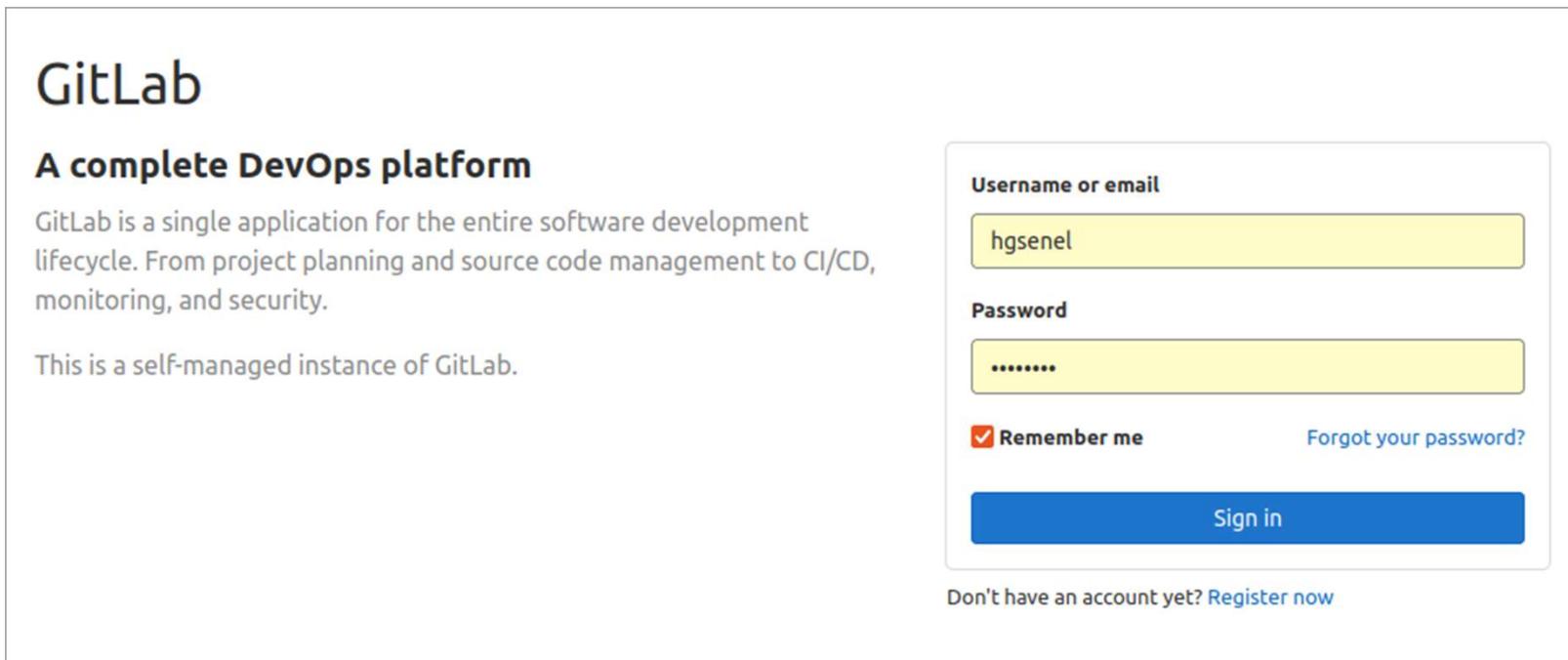
First name: Hakan      Last name: Senel  
Username: hgsenel  
Email: hgsenel@gmail.com  
Password:  Minimum length is 8 characters.

[Register](#)

Already have login and password? [Sign in](#)

# Yeni kullanıcı girişи

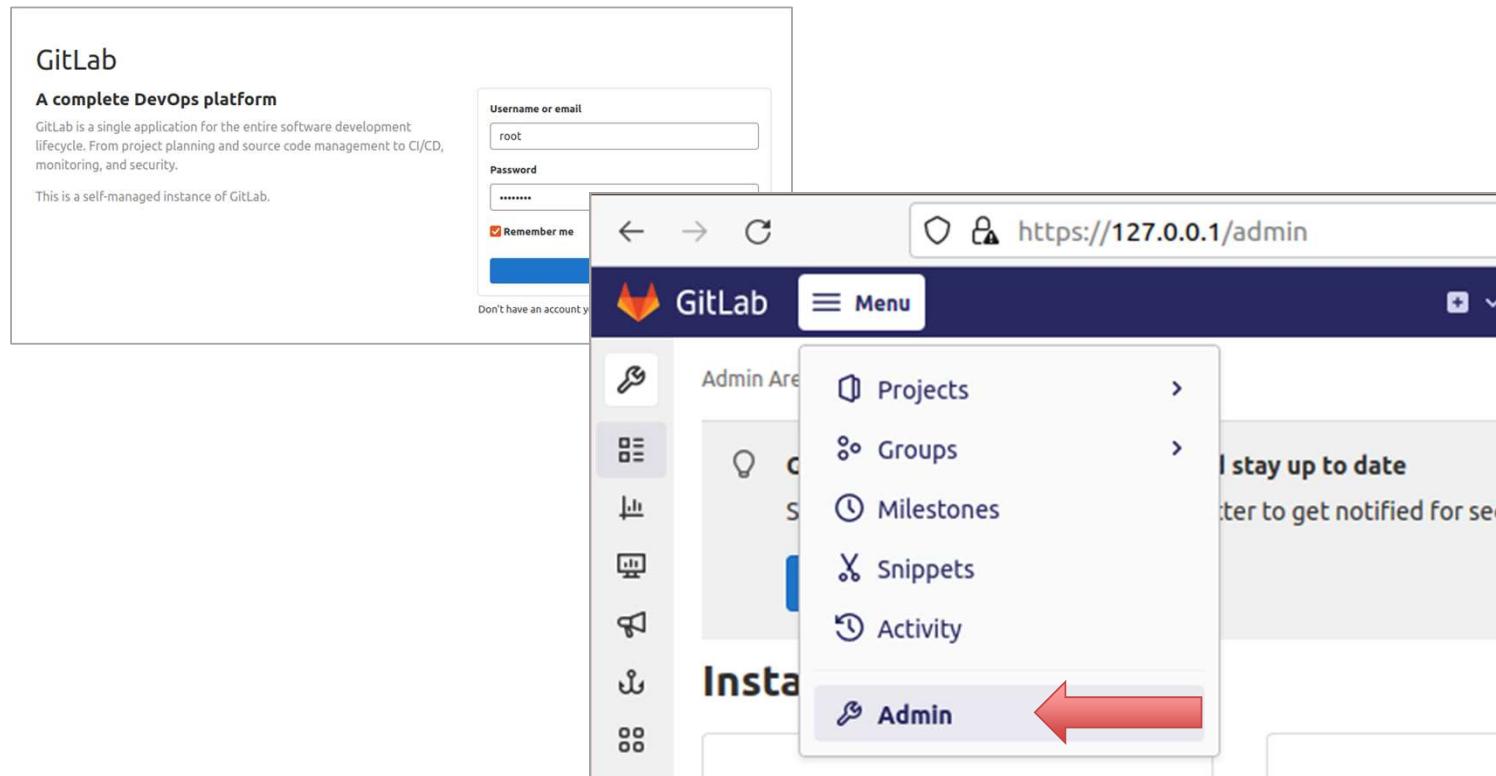
- Yeni kullanıcı **sign-up** yapar yapmaz, Gitlab'a giriş yapamayacaktır çünkü Gitlab «**root**»un onaylamasını ister.



The image shows the login page of a self-managed GitLab instance. The page has a header with the GitLab logo and the text "A complete DevOps platform". Below the header, there is a brief description of what GitLab is: "GitLab is a single application for the entire software development lifecycle. From project planning and source code management to CI/CD, monitoring, and security." A note below states, "This is a self-managed instance of GitLab." The main area contains a login form with fields for "Username or email" (containing "hgsenel") and "Password" (containing masked text). There are "Remember me" and "Forgot your password?" links, and a large blue "Sign in" button. At the bottom, there is a link for users who don't have an account yet: "Don't have an account yet? [Register now](#)".

# Yeni kullanıcının onaylanması

- Gitlab «**root**» kullanıcısı giriş yapmalı ve «**Menu**» bölümünde «**Admin**» seçeneğine tıklamalıdır.



# root «Admin» alanı

- Admin alanında giriş ekranında, tüm sistemle ilgili özet istatistik bilgiler bulunmaktadır.

The screenshot shows the 'Instance overview' page of a GitLab instance. On the left, there's a sidebar with various icons. The main area is divided into three main sections: 'Statistics', 'Features', and 'Components'.

**Statistics:**

Category	Value
Forks	0
Issues	0
Merge requests	0
Notes	0
Snippets	0
SSH Keys	0
Milestones	0
Active Users	1

**Features:**

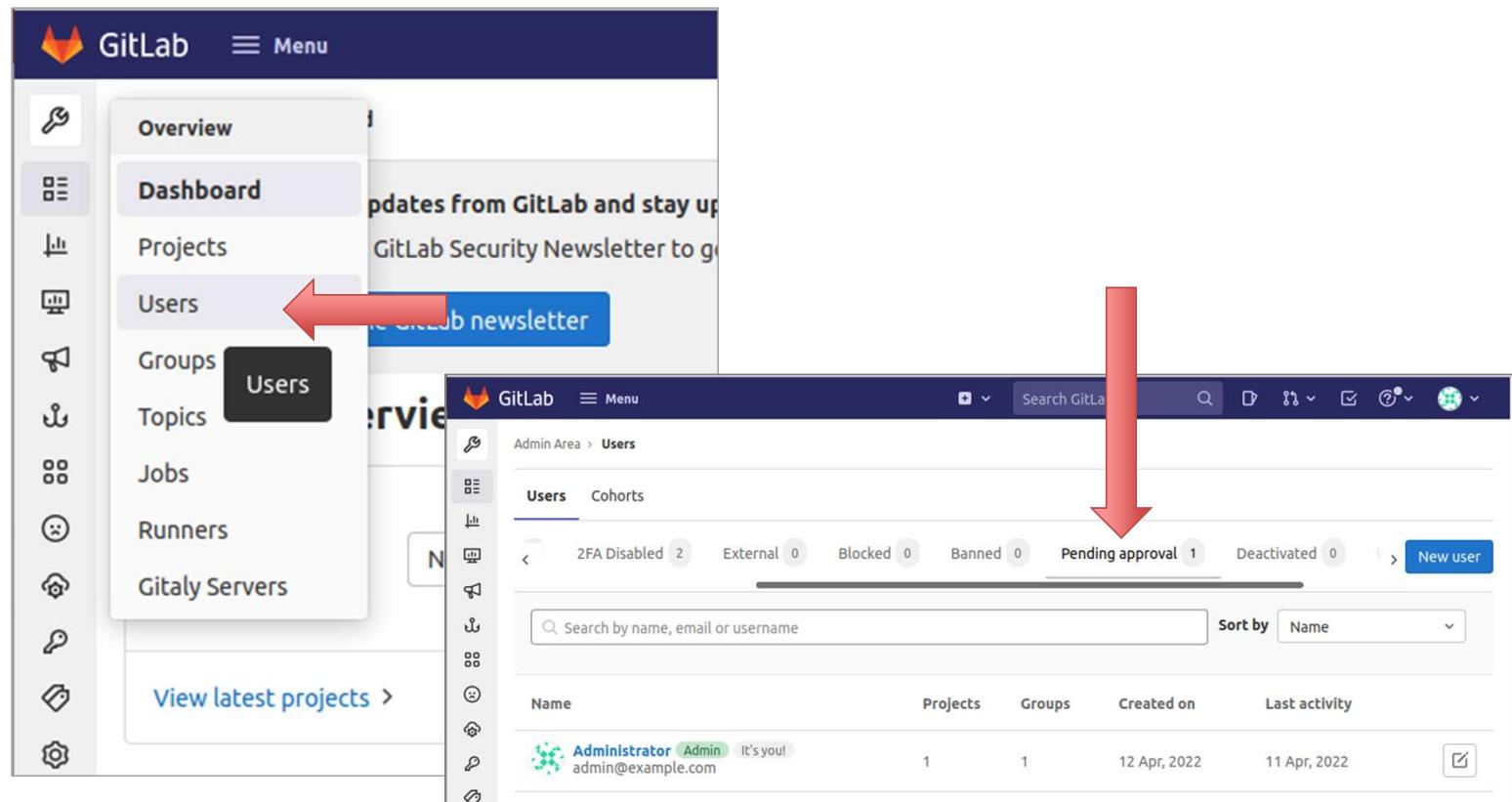
Feature	Status
Sign up	✓
LDAP	ⓘ
Gravatar	✓
OmniAuth	ⓘ
Reply by email	ⓘ
Container Registry	ⓘ
GitLab Pages	ⓘ
Shared Runners	✓

**Components:**

Component	Version
GitLab	14.9.2
GitLab Shell	13.24.0
GitLab Workhorse	v14.9.2
GitLab API	v4
GitLab KAS	14.9.0
Ruby	2.7.5p203
Rails	6.1.4.6
PostgreSQL	12.7
Redis	6.2.6
Gitaly Servers	

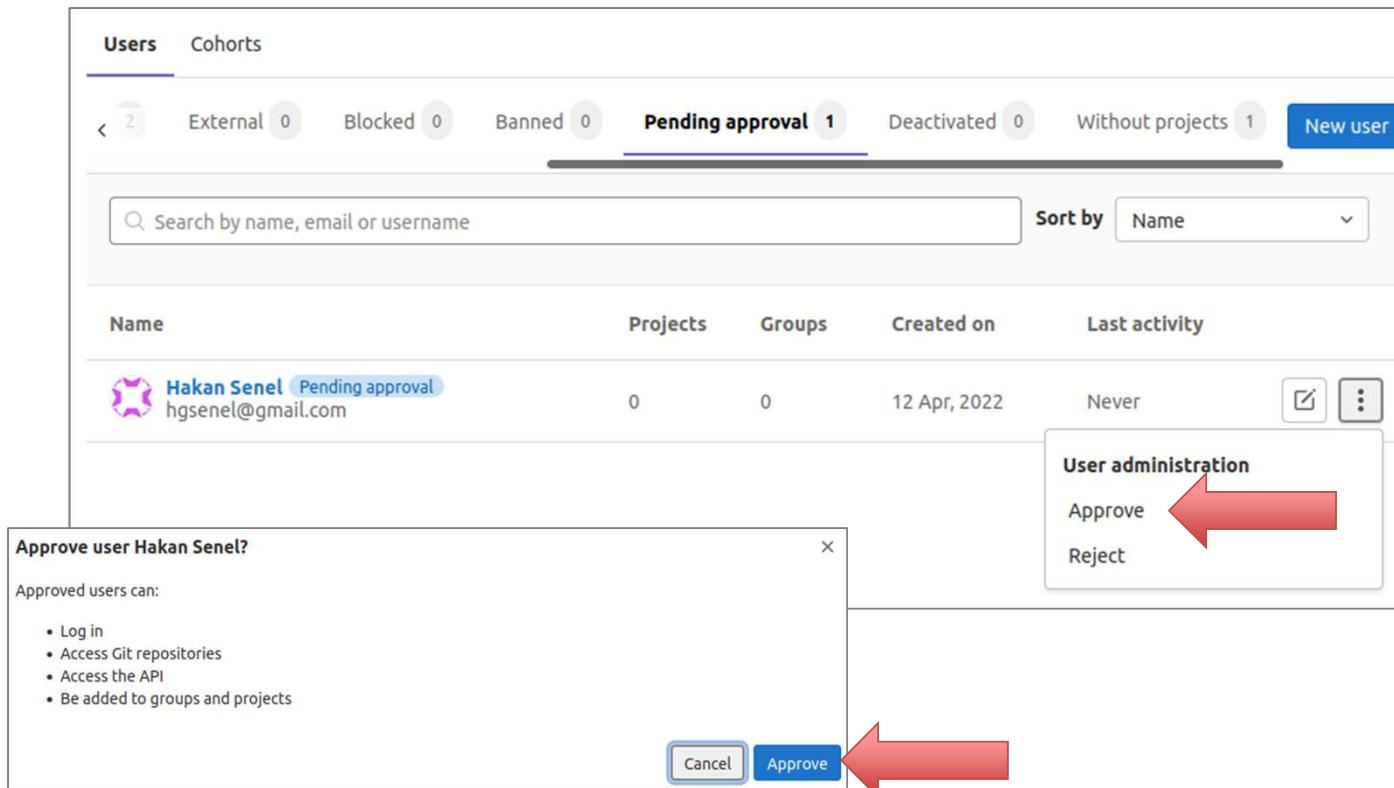
# root «admin/Users» alanı

- Sol üst köşede, «Overview» kısmında bulunan «Users» bölümü seçilmeli ve «Pending approval» tıklanmalıdır.



# root «admin/Users» alanı

- «Pending approval» bölümünde, bir kullanıcının onay beklediği görülebilir.
- Kullanıcı satırının yanındaki uzatma bölümüne tıklandığında çıkan menüde «Approve» veya «Reject» seçilebilir.

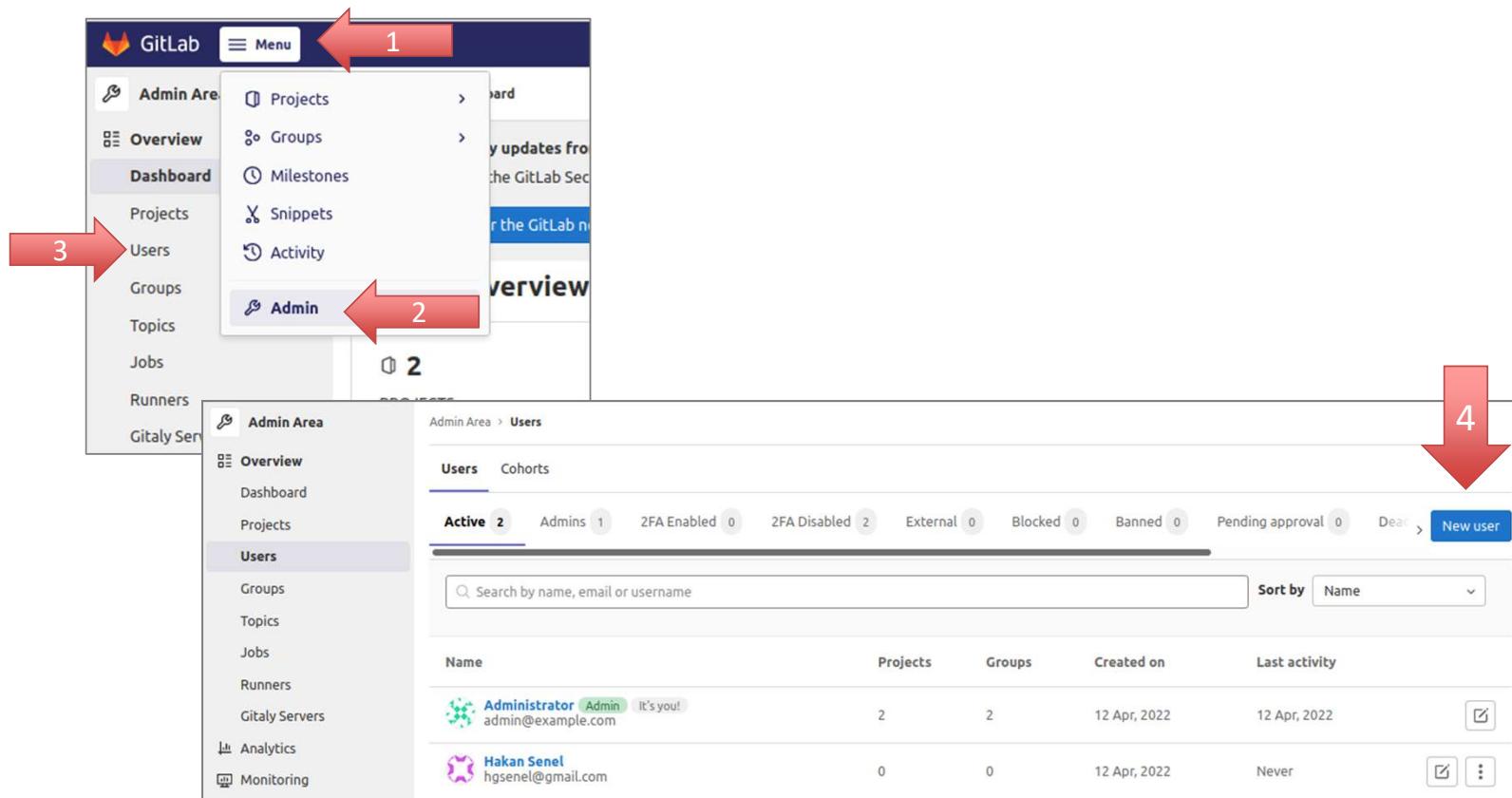


# Yeni Kullanıcı Oluşturma

- Gitlab, kullanıcıların kendi hesaplarını tanımlamalarına izin verdiği gibi ve **root** kullanıcısının kullanıcıları girmesini de sağlamaktadır.
- Kullanıcıların kendi hesaplarını oluşturmaları (**root** onayıyla tabii ki) güvenlik risk oluşturabilir. Zira, kullanıcılar için kimlik doğrulama imkanı bulunmamaktadır.
- Güvenlik açısından en doğru yaklaşım **root**'un kullanıcıları oluşturmasıdır. Kullanıcı **root** tarafından oluşturulduğunda (eğer **postfix** gibi e-posta sunucusu varsa) kullanıcı e-posta hesabına onay e-postası gönderilecektir.
- Kullanıcının e-postasına giden e-postayı onaylaması gerekmektedir.

# Yeni Kullanıcı Oluşturma

- «Admin» yönetim panelindeki «Users» bölümünden «New User» butonuna basarak yeni bir kullanıcı oluşturabilir.



# Yeni Kullanıcı Oluşturma

- Kullanıcıyla ilgili bilgilerin girişi yapıldıktan sonra, kullanıcının erişim yetkileri belirlenmelidir.
- NOT: «**root**» dışında en az bir kullanıcının yönetim yetkisine sahip olması güvenlik açısından önerilir.

New user

Account

Name  \* required

Username  \* required

Email  \* required

Password

Password  Reset link will be generated and sent to the user.  
User will be forced to set the password on first sign in.

Access

Projects limit

Can create group

Access level  Regular  
Regular users have access to their groups and projects.  
 Administrator  
The user has unlimited access to all groups, projects, users, and features.

External  External users cannot see internal or private projects unless access is explicitly granted. Also, external users cannot create projects, groups, or personal snippets.

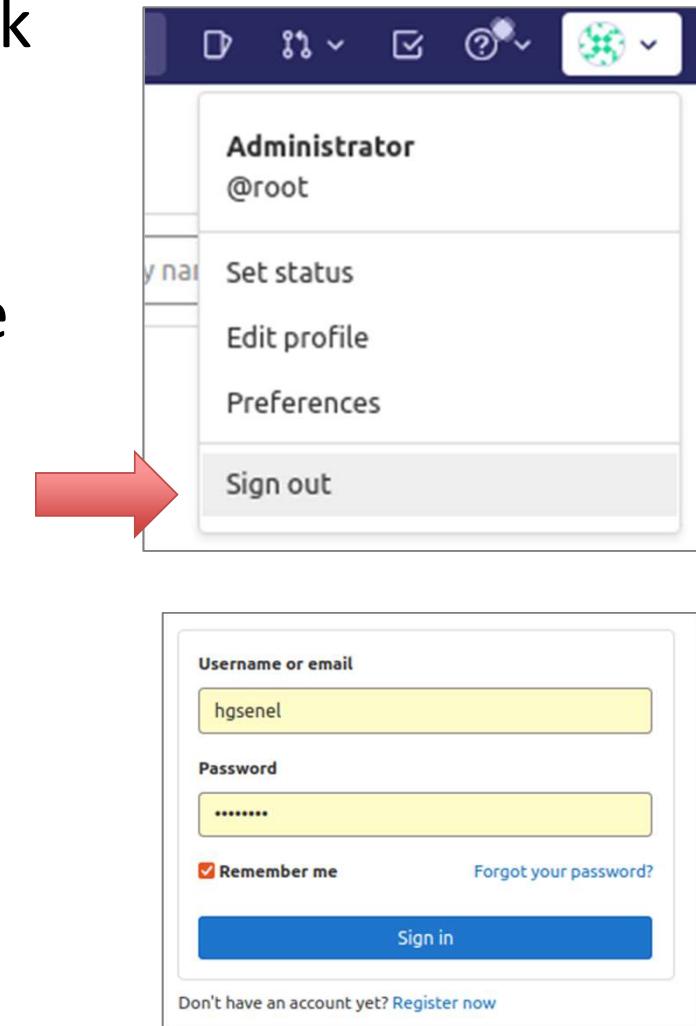
Validate user account  User is validated and can use free CI minutes on shared runners.  
A user can validate themselves by inputting a credit/debit card, or an admin can manually validate a user.



# **GITLAB'DA KULLANICI İŞLEMLERİ GRUP VE PROJE OLUŞTURMA**

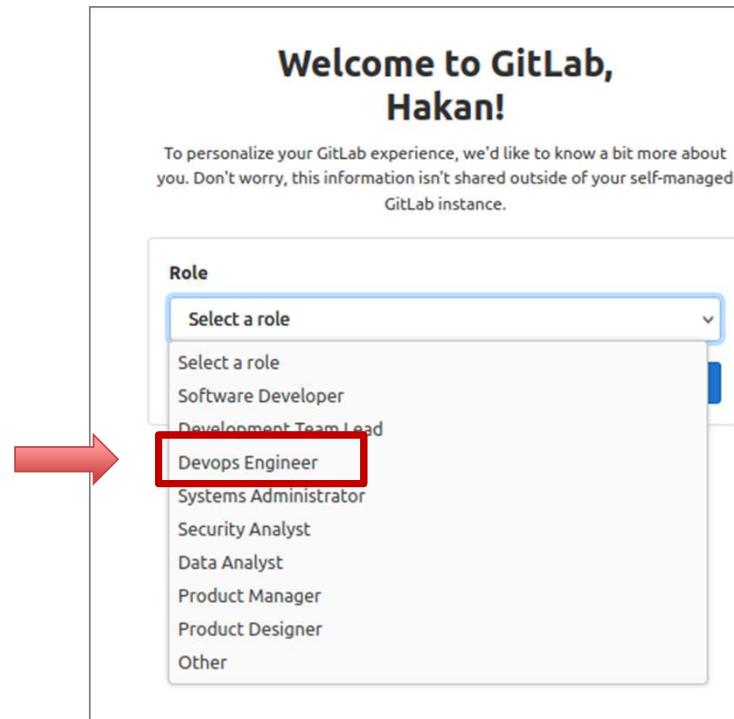
## Örnek:

- Şu aşamaya kadar **admin** olarak grup ve proje oluşturduk.
- Şimdi başka bir kullanıcı olarak giriş yaparak kullanıcı panelinde yapılabilecek işleri göreceğiz.
- «**root**» yani «**admin**» kullanıcısından çıkış yapalım ve oluşturduğumuz kullanıcı (**hgsenel** kullanıcıı oluşturmuştık) olarak giriş yapalım.



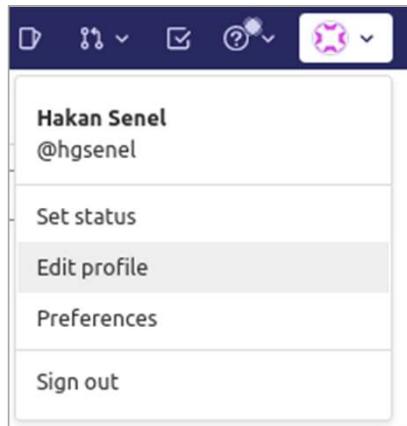
# Kullanıcı Rol girişi

- İlk giriş yapan kullanıcıya hangi rolle giriş yapacağı sorulmaktadır. Bu durum menülerin, role göre değiştirilmesini sağlayacaktır.



# Kullanıcı Profili

- Sağ üst köşede «Edit Profile» kısmına tıklayarak, kullanıcının profili görülebilir ve eksik bilgiler tamamlanabilir.



User Settings > **Edit Profile**

Q Search settings

**Public avatar**  
You can upload your avatar here or change it at [gravatar.com](http://gravatar.com)

**Upload new avatar**  
 No file chosen.  
The maximum file size allowed is 200KB.

**Current status**  
This emoji and message will appear on your profile and throughout the interface.

**Your status**  
 What's your status?   
 Busy  
An indicator appears next to your name and avatar

**Time settings**  
Set your local time zone

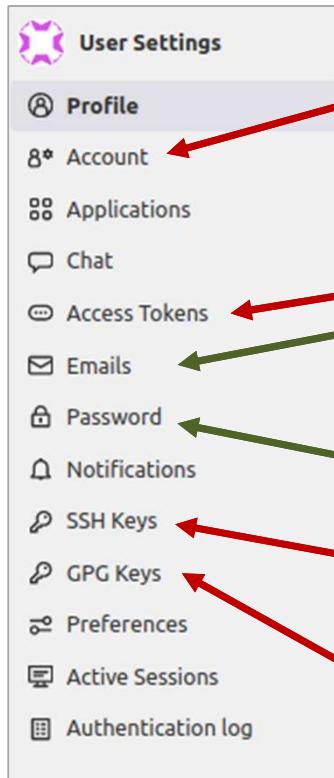
**Main settings**  
This information will appear on your profile

**Full name**  
Hakan Senel  
Enter your name, so people you know can recognize you

**Pronouns**  
  
Enter your pronouns to let people know how to refer to you

User ID  
2

# Kullanıcı ayarları



İlişkili e-posta hesabı  
girişi

Şifre değişikliği

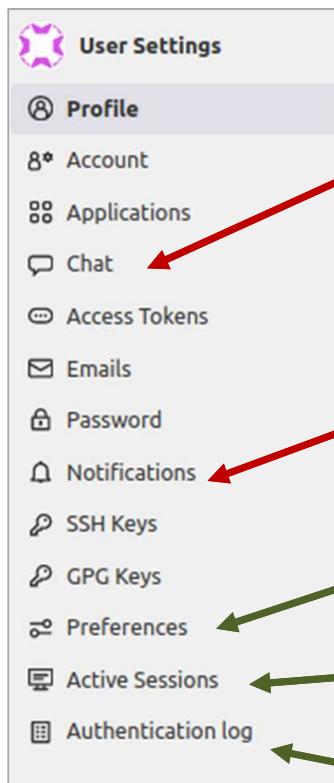
Kullanıcı isminin  
değiştirilmesi, hesabın  
silinmesi

«Access Token» hesabınıza  
erişmenizin bir yöntemidir.  
Komut satırı üzerinden  
yapılacak işlemlerde  
kullanılır. Diğer yöntem, SSH  
anahtar çifti kullanmaktadır.

**ssh-keygen** ile  
oluşturduğunuz açık anahtarını  
yükleyerek, şifre girişini  
yapmadan, güvenli şekilde kod  
yüklemesi yapabilirsiniz.

GnuPrivacy guard, Gitlab'da  
kod teslimlerinin (commit) ve  
etiketlerin imzalanması için  
kullanılır. SSH kimlik  
doğrulama içindir.

# Kullanıcı ayarları



Gitlab'ın farklı chat sistemleriyle entegrasyonu var. «slack» ve «mattermost» bunlardan ikisidir. «mattermost» Gitlab'la birlikte gelmektedir.

Kullanıcı, proje ve grup üzerinde oluşan olayları nasıl öğrenecek? e-posta ile.

Ekrana ilgili renk, font, vb yapılandırma

Aktif olan oturumları görme

Gitlab sunucusuna giriş logları.

<https://docs.gitlab.com/ee/integration/mattermost/>

# Açık SSH anahtarının girişи

- Terminalde **ssh-keygen** komutuyla **.ssh/** dizini içinde açık ve gizli anahtarlar oluşturulur. Açık anahtar **«cat id\_rsa.pub»** komutuyla görüntülenir ve kopyalanır.

The diagram illustrates the workflow for setting up an SSH key for GitLab access:

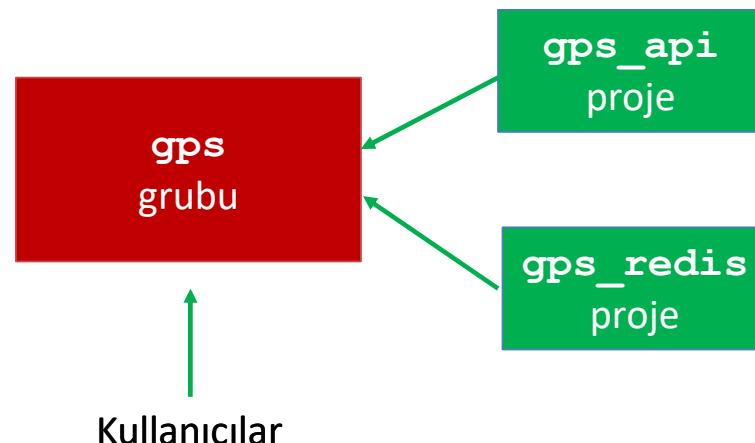
- SSH Key Generation:** A terminal window shows the command `ssh-keygen` being run, generating a new RSA key pair. The public key (`id_rsa.pub`) is saved to the `.ssh/` directory.
- Copy Public Key:** The terminal shows the command `cat .ssh/id_rsa.pub` being run to view the public key. A red arrow points from this terminal output to a context menu on a Linux desktop. The menu item **Copy** is highlighted, indicating the key should be copied to the clipboard.
- Paste into GitLab:** A screenshot of the GitLab "SSH Keys" settings page is shown. A red arrow points from the clipboard icon in the top right of the browser to the **Paste** field under the "Key" section. The copied public key is pasted into this field.
- Add Key:** After pasting the key, a red arrow points to the **Add key** button, which is then clicked to save the key to the account.
- Success:** The "Your SSH keys (0)" section shows a message: "There are no SSH keys with access to your account."
- Terminal Connection:** A terminal window shows the command `ssh -T git@127.0.0.1` being run. The authenticity of the host is being checked, and the user is prompted to continue connecting. The response "yes" is entered, and the connection is established successfully.

# SSH’la giriş neden önemli?

- SSH, lokalde git sayesinde oluşturduğumuz kod deposunu, şifresiz şekilde Gitlab'a yüklememizi sağlamaktadır.
- Kodu Gitlab'a veya Github'a yüklemek için kullanılan diğer yöntem, **https** üzerinden yüklemektir. Ancak bu yöntemde, terminal penceresinde kullanıcı adı ve şifrenin verilerek **https** adresini belirtmek gereklidir. Bu durum güvenlik riski oluşturabilir.
- En güvenli yöntem, kullanıcının SSH anahtar çiftini **ssh-keygen** ile oluşturması ve açık anahtarını Gitlab'da (veya Github) kullanıcı ayarları menüsündeki «**SSH Keys**» bölümünden girmesidir. Bu şekilde, SSH üzerinden «**git push**» yani lokal git deposundan Gitlab (veya Github) deposuna yükleme işlemi şifre girilmeden şifreli şekilde yapılabilir.

# Gitlab'da grup nedir?

- Grup, aynı çatı altında bulunan projelere verilen isimdir.
- Grup altındaki projelerin her biri farklı takımlar tarafından kullanılabilir.
- İlk önce grubu, sonra içindeki projeleri oluşturmak gereklidir.



# Grup oluşturma (gps grubu)

- Gitlab'da kullanıcı grupları, birbiriyle ilgili projelerin yönetimi için kullanılır ve gruplar projelerden oluşur.
- Aynı gruptaki kullanıcılar, grup içindeki projelere erişebilirler.

The image shows two screenshots of the GitLab interface. On the left, the 'Groups' page is displayed with a red box around the 'Groups' menu item in the sidebar. Below it, a red box highlights the 'Create group' button. On the right, a 'Create group' dialog box is open, also with a red box around the 'Create group' button at the bottom. The dialog form includes fields for 'Group name' (set to 'gps'), 'Group URL' (set to 'https://127.0.0.1/gps'), and 'Visibility level' (set to 'Private'). A red arrow points from the 'Create group' button on the Groups page to the 'Create group' button in the dialog box.

**Private:** sadece bu grubun üyeleri Projelere erişebilir.

**Internal:** Sisteme giriş yapabilen Bütün kullanıcılar grup içindeki projelere erişebilir.

**Public:** herkes projelere erişebilir.

# Gitlab'da proje nedir?

- Kodlarınızı depolamak için kullanılan nesnelerdir.
- Projeler üzerinde, sorunların takibini ve kod üzerinde takım olarak ortak çalışma gerçekleştirebilir ve CI/CD özellikleriyle yazılımınızın devreye alınmasını (deploy) sağlayabilirsiniz.
- Github'daki repo'lara (depolarla) benzer özelliktidir. Gitlab'da kod depo yönetiminden daha fazla özelliği olduğundan repo yerine proje ismi tercih edilmiştir.

# Grup Altında Proje oluşturma

- Grup içinde yeni proje oluşturulabilir. Ayrıca oluşturulan alt gruplar da yönetim açısından faydalı bir özelliktir.

The screenshot shows the GitLab 'Subgroups and projects' page for a group named 'gps'. A red arrow points from a callout box on the left to the 'Create new project' section. Another red arrow points from a callout box on the right to the 'Create from template' option. A third red arrow points from the 'Import project' section at the bottom to a callout box on the left.

**Create new project**

- Create blank project**  
Create a blank project to house your files, plan your work, and collaborate on code, among other things.
- Create from template**  
Create a project pre-populated with the necessary files to get you started quickly.
- Import project**  
Migrate your data from an external source like GitHub, Bitbucket, or another instance of GitLab.

You can also create a project from the command line. [Show command](#)

**G gps**  
Group ID: 11

**Subgroups and projects** Shared projects Archived projects

Search by name Name

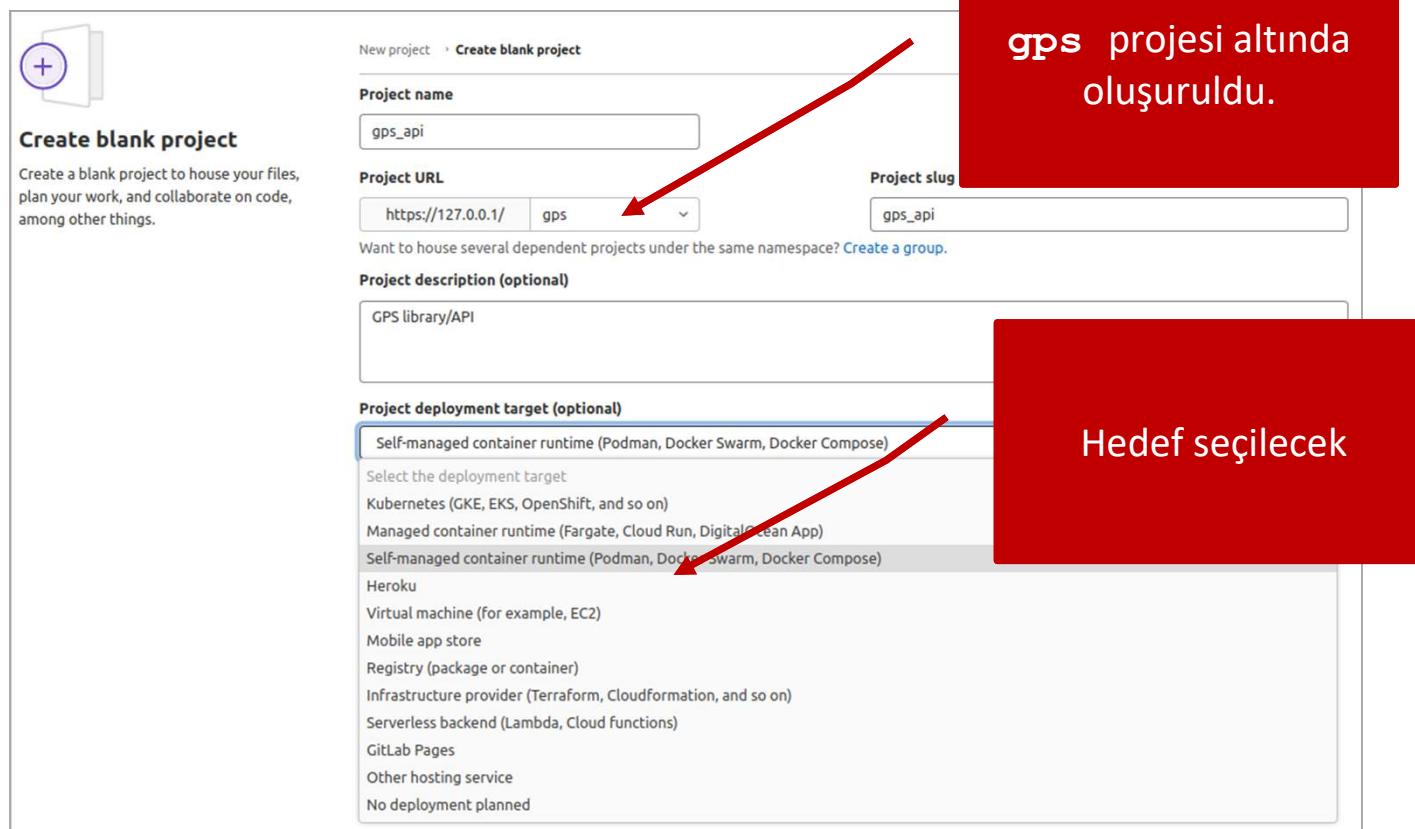
**Import project**

Önceden belirlenen dosyalarla beraber oluşturulan projeler

Github, bitbucket gibi sistemlerden projeler alınabilir.

# Proje Bilgilerinin Girişi

- Proje oluşturduğumuz zaman, projeyi kendi başına veya bir gruba bağlı olarak oluşturabiliriz.



**Create blank project**  
Create a blank project to house your files, plan your work, and collaborate on code, among other things.

New project > Create blank project

**Project name**

**Project URL**  
 **gps**

**Project slug**

Want to house several dependent projects under the same namespace? [Create a group](#).

**Project description (optional)**

**Project deployment target (optional)**

Select the deployment target

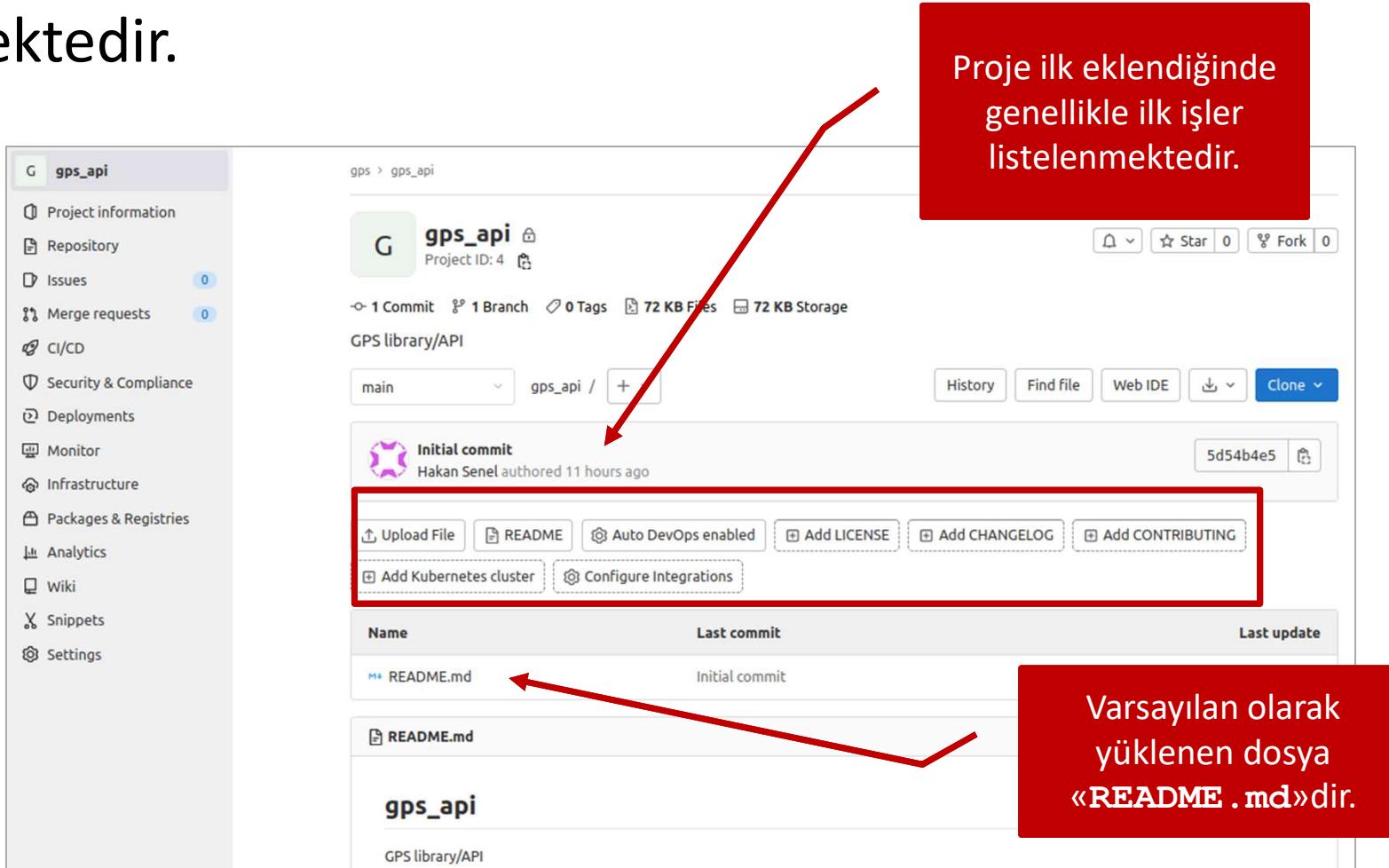
- Self-managed container runtime (Podman, Docker Swarm, Docker Compose)**
- Kubernetes (GKE, EKS, OpenShift, and so on)
- Managed container runtime (Fargate, Cloud Run, DigitalOcean App)
- Heroku
- Virtual machine (for example, EC2)
- Mobile app store
- Registry (package or container)
- Infrastructure provider (Terraform, Cloudformation, and so on)
- Serverless backend (Lambda, Cloud Functions)
- GitLab Pages
- Other hosting service
- No deployment planned

gps projesi altında oluşturuldu.

Hedef seçilecek

# Proje Ekranı

- Proje oluşturuluduktan sonra, proje bilgi ekranı karşımıza gelmektedir.



The screenshot shows a GitHub project page for 'gps\_api'. The sidebar on the left contains links like Project information, Repository, Issues, Merge requests, CI/CD, Security & Compliance, Deployments, Monitor, Infrastructure, Packages & Registries, Analytics, Wiki, Snippets, and Settings. The main content area shows the repository details: 1 Commit, 1 Branch, 0 Tags, 72 KB Files, 72 KB Storage. Below this is a commit history section with a single entry: 'Initial commit' by Hakan Senel, authored 11 hours ago. A red arrow points from this commit to a red box containing buttons for 'Upload File', 'README', 'Auto DevOps enabled', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', and 'Configure Integrations'. Another red arrow points from the 'README' button in this box to the 'README.md' file listed in the repository files table below. The table has columns for Name, Last commit, and Last update. It lists 'README.md' with a last commit of 'Initial commit' and 'gps\_api' with a last update of 'GPS library/API'.

Proje ilk eklendiğinde genellikle ilk işler listelenmektedir.

Varsayılan olarak yüklenen dosya «**README . md**»dir.

# Proje Ekranı

- «gps\_api» proje sayfasında, ilk eklendiğinde yapılan işler listelenmektedir.

The screenshot shows the 'gps\_api' project page on GitLab. It includes sections for 'Getting started', 'Add your files', 'Integrate with your tools', and 'Collaborate with your team'. A red box highlights the 'Add your files' section, which contains a command-line snippet for cloning a repository. Another red box highlights the 'Collaborate with your team' section, which lists options for team collaboration like inviting members and creating merge requests. Red arrows point from the explanatory text boxes to these specific sections.

Lokaldeki git reposunun https yöntemiyle nasıl bu projeye yükleneceğini belirtir.

Takımla ortak çalışma için yapılacak işler. (Yardım sayfaları açılır)

gps\_api

GPS library/API

**Getting started**

To make it easy for you to get started with GitLab, here's a list of recommended next steps.

Already a pro? Just edit this README.md and make it your own. Want to make it easy? [Use the template at the bottom](#).

**Add your files**

Create or upload files  
 Add files using the command line or push an existing Git repository with the following command:

```
cd existing_repo
git remote add origin https://gitlab.example.com/gps/gps_api.git
git branch -M main
git push -uf origin main
```

**Integrate with your tools**

Set up project integrations

**Collaborate with your team**

Invite team members and collaborators  
 Create a new merge request  
 Automatically close issues from merge requests  
 Enable merge request approvals  
 Automatically merge when pipeline succeeds

# Lokal Git Reposu Oluşturmak

- Size verilen «gps . tar» ve «gps \_ data . tar» dosyalarını ana ev dizinimize yükleyin.
- Gitlab'la çalışabilmek için, yeni bir dizin (örneğin **gps \_ api**) açalım ve dizinin içinde lokal bir repo'yu «git init» ile oluşturalım.
- Tar dosyalarını **gps \_ api** dizininin içinde «tar xvf» ile açalım. İki dizinde ~/gps \_ api/gps ve ~/gps \_ api/gps \_ data çeşitli dosyalar oluşturulacaktır.
- Derleme sırasında oluşacak EXE ve object dosyaları için ~/gps \_ api/build dizini oluşturalım.

```

adminpc@ubuntu:~$ ls
Desktop  Downloads  gps.tar  Pictures  Templates
Documents  gps_data.tar  Music  Public  Videos
adminpc@ubuntu:~$ mkdir gps_api
adminpc@ubuntu:~$ cd gps_api
adminpc@ubuntu:~/gps_api$ git init
Initialized empty Git repository in /home/adminpc/gps_api/.git/
adminpc@ubuntu:~/gps_api$ tar xvf ../gps.tar
gps/
gps/util.h
gps/util.c
gps/gps.h
gps/gps.c
gps/config.h
gps/main.c
gps/setup.h
adminpc@ubuntu:~/gps_api$ tar xvf ../gps_
gps_api/
gps_data.tar
adminpc@ubuntu:~/gps_api$ tar xvf ../gps_data.tar
gps_data/
gps_data/gps_data.txt
adminpc@ubuntu:~/gps_api$ git add gps/*
adminpc@ubuntu:~/gps_api$ git add gps_data/*
adminpc@ubuntu:~/gps_api$ mkdir build
adminpc@ubuntu:~/gps_api$ cat > .gitignore
build/*
adminpc@ubuntu:~/gps_api$ git add .gitignore
adminpc@ubuntu:~/gps_api$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:  .gitignore
  new file:  gps/config.h
  new file:  gps/gps.c
  new file:  gps/gps.h
  new file:  gps/main.c
  new file:  gps/setup.h
  new file:  gps/util.c
  new file:  gps/util.h
  new file:  gps_data/gps_data.txt

```

Repo için **gps\_api** dizini oluşturarak içine girin.

**«git init»** ile repoyu oluşturun

Ev dizinize yüklediğiniz tar dosyalarını açınız.  
**gps** ve **gps\_data** dizinleri açılacak

**gps** ve **gps\_data** dizinlerindeki dosyaları repoya ekleyin.

**build** dizinini oluşturun ve bunu repodan **.gitignore** dosyası sayesinde çıkartın

# Lokal git reposu

- git reposunda teslim (commit) işlemini hangi kullanıcının yaptığı önemli bir bilgidir. Bu nedenle öncelikli olarak aşağıdaki komutlarla kullanıcı ismimizi ve e-posta adresimi «**git config --global**» komutuyla git'e kaydettmeliyiz.

```
git config --global user.name "Hakan Senel"
```

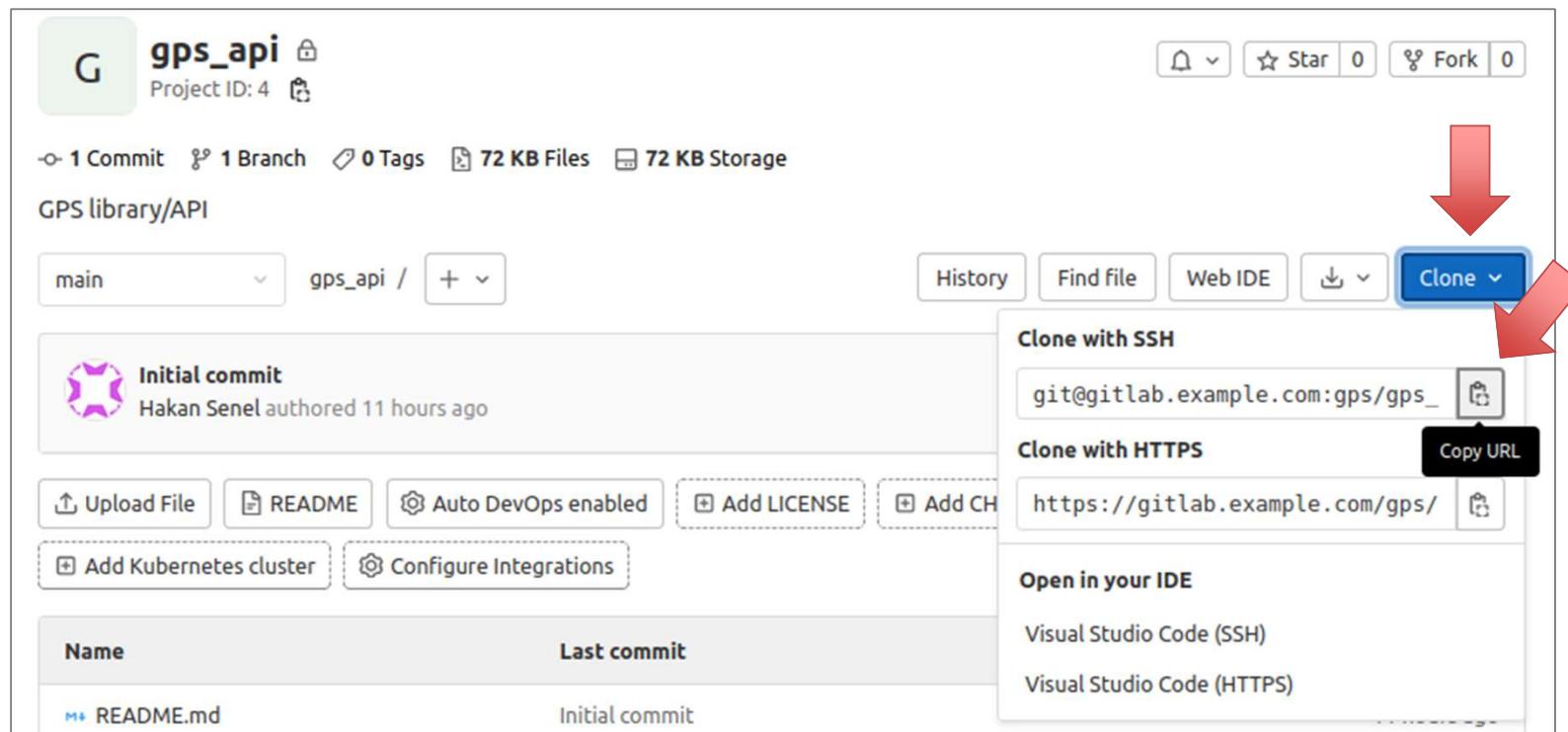
```
git config --global user.email "hgsenel@gmail.com"
```

- Repomuz Gitlab'a yüklenmeden önce, «**git commit**» komutuyla hazırlık (staging) alanına aktarılması gereklidir.

```
adminpc@ubuntu:~/gps_api$ git config --global user.name "Hakan Senel"
adminpc@ubuntu:~/gps_api$ git config --global user.email "hgsenel@gmail.com"
adminpc@ubuntu:~/gps_api$ git commit -m "ilk commit denememiz"
[master (root-commit) b977252] ilk commit denememiz
 9 files changed, 890 insertions(+)
   create mode 100644 .gitignore
   create mode 100755 gps/config.h
   create mode 100755 gps/gps.c
   create mode 100755 gps/gps.h
   create mode 100644 gps/main.c
   create mode 100755 gps/setup.h
   create mode 100755 gps/util.c
   create mode 100755 gps/util.h
   create mode 100755 gps_data/gps_data.txt
```

# Lokal reponun Gitlab'a yüklenmesi

- Öncelikli olarak repoyu Gitlab'a yüklemek için, oluşturduğumuz projenin git adresini bulmamız gereklidir. Bu bilgi için proje bilgi sayfasındaki mavi «Clone» butonuna basılabilir.



# Lokal reponun Gitlab'a yüklenmesi

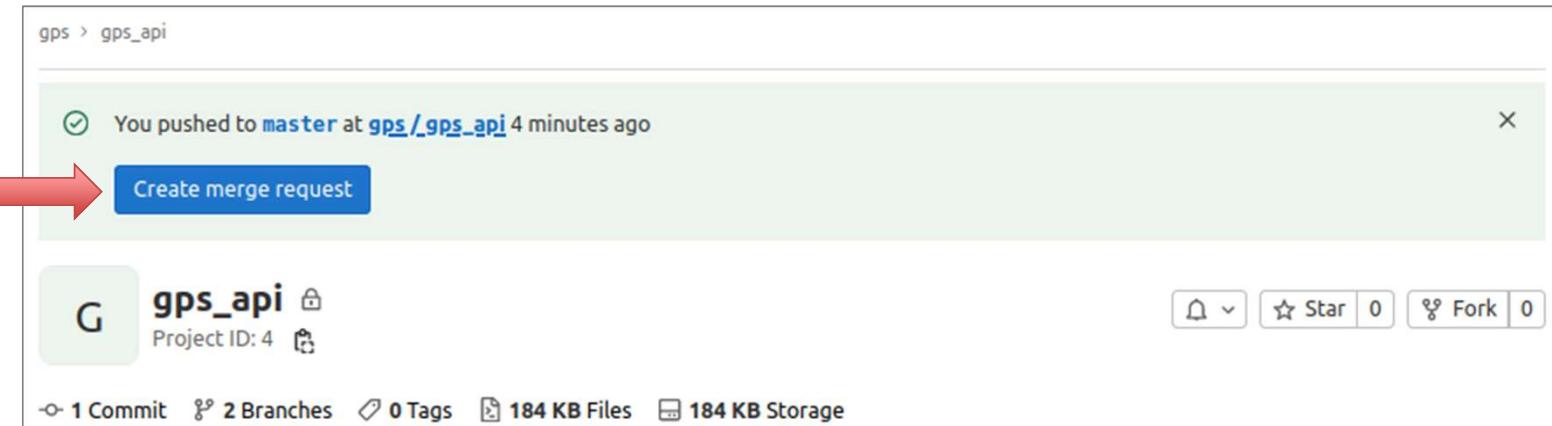
- Komut satırında, projenin Gitlab adresinin ne olduğunu «**git remote add origin**» komutuyla belirtmemiz ve ardından «**git push**» komutuyla, Gitlab'daki «**master**» dala yüklememiz gereklidir.

```
$ git remote add origin git@127.0.0.1:gps/gps_api  
$ git push --set-upstream origin master
```

```
adminpc@ubuntu:~/gps_api$ git remote add origin git@127.0.0.1:gps/gps_api  
adminpc@ubuntu:~/gps_api$ git push --set-upstream origin master  
Enumerating objects: 13, done.  
Counting objects: 100% (13/13), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (11/11), done.  
Writing objects: 100% (13/13), 8.32 KiB | 266.00 KiB/s, done.  
Total 13 (delta 0), reused 0 (delta 0)  
remote:  
remote: To create a merge request for master, visit:  
remote: https://gitlab.example.com/gps/gps_api/-/merge_requests/new?merge_requ  
est%5Bsource_branch%5D=master  
remote:  
To 127.0.0.1:gps/gps_api  
 * [new branch]      master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

# Lokal Reponun Gitlab'a yüklenmesi

- «**git push**» komutuyla, lokal reponun kopyası bir ara durumda ele alınmaktadır. Projede bu iş için görevlendirilenlerin «**merge request**» oluşturmasını istemektedir.
- «**merge request**» sayesinde kodlar, proje sahibi tarafından incelenerek onay verildikten sonra projeye yüklenmesi gerçekleşmektedir.
- «**push**» işlemi gerçekleştiğinde proje sayfasında bir bilgi notu gelmekte ve kullanıcının «**merge request**» oluşturması istenmektedir



# «merge request»

The screenshot shows a 'New merge request' interface. A red arrow points from the first annotation box to the 'Title' field. Another red arrow points from the second annotation box to the 'Assignee' dropdown. A third red arrow points from the fourth annotation box to the 'Labels' dropdown. A large red arrow points from the fifth annotation box to the 'Create merge request' button at the bottom.

**New merge request**

From master into main Change branches

Title \*

Start the title with **Draft:** to prevent a merge request draft from merging before it's ready.  
Add [description templates](#) to help your contributors to communicate effectively!

Description [Write](#) [Preview](#)

Describe the goal of the changes and what reviewers should be aware of.

Markdown and quick actions are supported

Assignee  [Assign to me](#)

Reviewer

Milestone

Labels

Merge options  Delete source branch when merge request is accepted.  
 Squash commits when merge request is accepted. (?)

[Create merge request](#) [Cancel](#)

Yapılan değişikliğin açıklamasının yapılması beklenir.

Hangi kullanıcılar koddaki bu talebi değerlendirecek ve onay verecek?

Bu değişiklik proje için bir kilometre taşı mı?

Değişiklige etiket vereceğimiz? RC1, v1.2, vb

# «merge request» nedir?

- «**merge request**» lokal repodan Gitlab'a yüklenenlerin, «**main**» kodla birleştirilmesi için gerekli onay mekanizmasıdır. Bu şekilde kodun ancak incelendikten sonra birleştirilmesi sağlanır.
- Lokal repodan gelen yükleme dışında, projedeki üyeleri, yaptıkları değişiklikler için «**merge request**» ile **değişiklik önerisi** yapabilirler.
- «**merge request**» atanın kullanıcıda gelecek ekranda, «**merge**» butonuna basılarak, değişikliklerin Gitlab'daki projeye aktarılmasını sağlanabilir (not: görevli kişilerin tamamının onay vermesi gereklidir)

# «merge request»

The screenshot shows a GitLab merge request interface. At the top, it says "Open" and "Created just now by Hakan Senel (Owner)". There are "Edit" and "Mark as draft" buttons. The title of the merge request is "ilk commit denememiz". Below the title, there's an "Overview" section with "0 Commits" and "9 Changes". A message says "ilk teslimimiz. Daha önce değişiklik yok." (Our first delivery. No changes yet.).

The main content area shows a "Request to merge master into main". It states "The source branch is 1 commit behind the target branch". There are "Open in Web IDE", "Check out branch", and a download button. Below this, there's an "Approve" button with a count of 8 and a note that approval is optional.

A red arrow points to the "Merge" button, which is checked. There are also checkboxes for "Delete source branch" and "Squash commits". A note below says ">Adds 1 commit and 1 merge commit to main. Modify merge commit".

At the bottom, there are "Write" and "Preview" buttons, a rich text editor toolbar, and a comment input field with placeholder "Write a comment or drag your files here...". It also says "Markdown and quick actions are supported" and has an "Attach a file" button. Finally, there are "Comment" and "Close merge request" buttons at the very bottom.

# Birleştirme (merge) sonucu

Merged Created just now by  Hakan Senel Owner Edit Mark as draft ▾

## ilk commit denememiz

Overview 0 Commits 1 Changes 9

ilk teslimimiz. Daha önce değişiklik yok.

 Request to merge master  into main 

 Approval is optional

 Merged by  Hakan Senel 27 seconds ago Revert Cherry-pick

The changes were merged into main with 34a1acf0 

 0  0 

Oldest first ▾ Show all activity ▾

 Hakan Senel @hgsenel mentioned in commit 34a1acf0 just now

 Hakan Senel @hgsenel merged just now

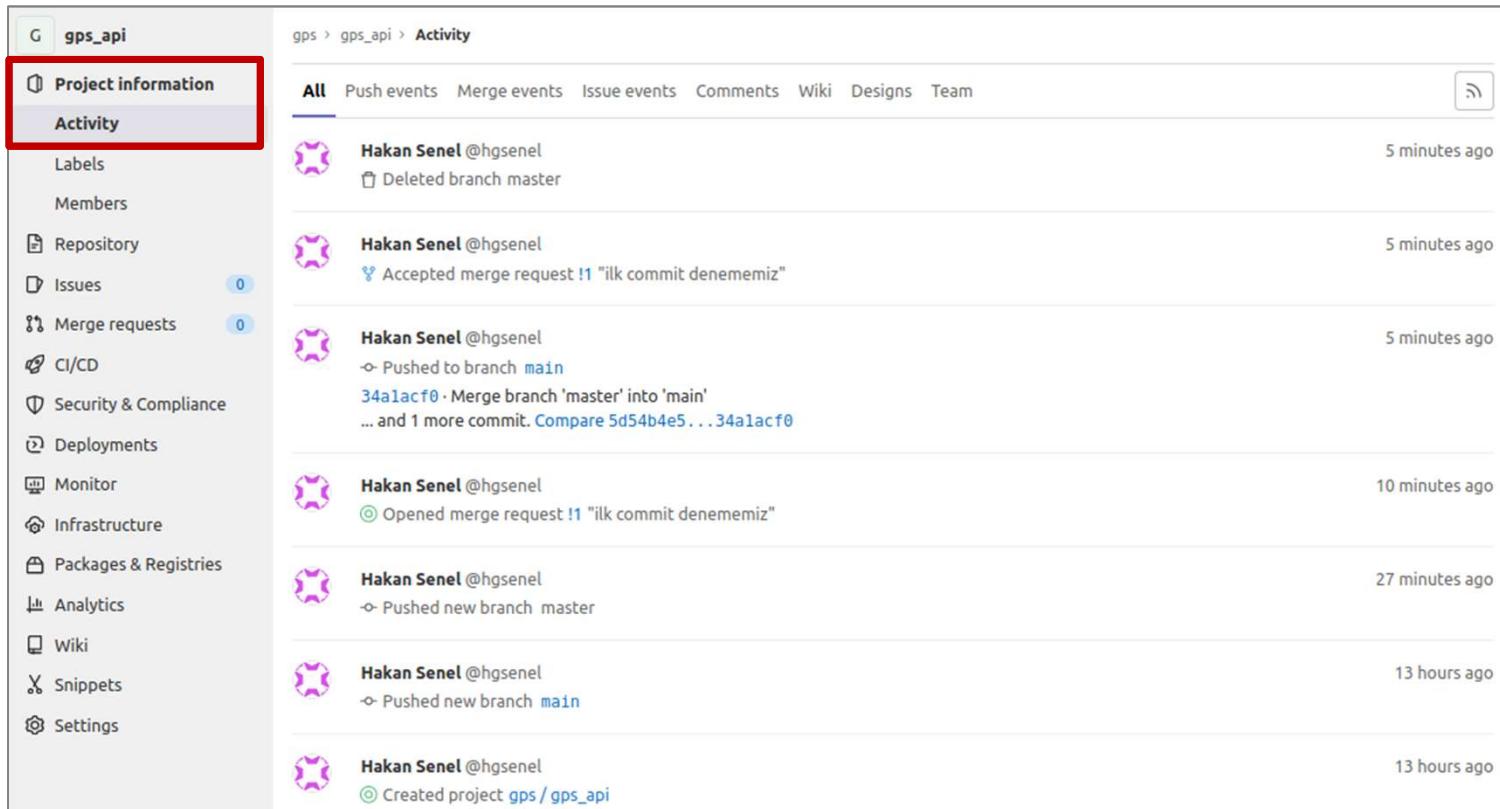
**Write** Preview 

Write a comment or drag your files here...

Markdown and quick actions are supported 

# Proje bilgisi: Aktivite

- Proje menüsünde, «**Project Information -> Activity**» bölümünde, projede gerçekleşen bütün olaylar listelenebilir.

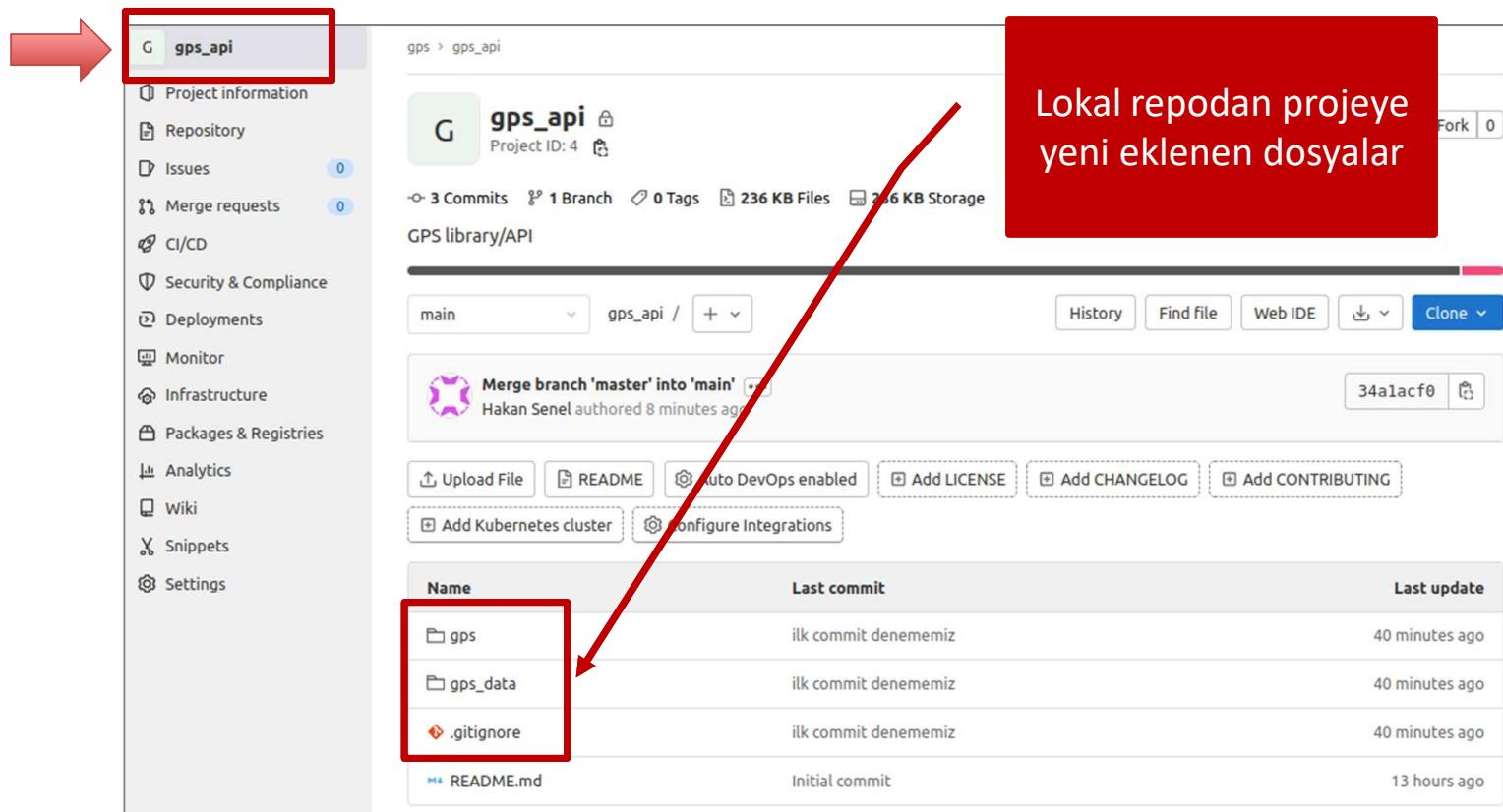


The screenshot shows a project interface with a sidebar and a main activity feed. The sidebar on the left has a red box around the 'Activity' tab, which is currently selected. The main area displays a list of recent activities:

Activity	Time Ago
Hakan Senel @hgsenel Deleted branch master	5 minutes ago
Hakan Senel @hgsenel Accepted merge request !1 "ilk commit denememiz"	5 minutes ago
Hakan Senel @hgsenel Pushed to branch main 34a1acf0 · Merge branch 'master' into 'main' ... and 1 more commit. Compare 5d54b4e5...34a1acf0	5 minutes ago
Hakan Senel @hgsenel Opened merge request !1 "ilk commit denememiz"	10 minutes ago
Hakan Senel @hgsenel Pushed new branch master	27 minutes ago
Hakan Senel @hgsenel Pushed new branch main	13 hours ago
Hakan Senel @hgsenel Created project gps / gps_api	13 hours ago

# Yüklenen Dosyalar

- Solda, proje ismi (**gps\_api**) tıklandığında, projeye birleştirilen yeni dosyalar ekranda görülebilir.



A screenshot of a Git repository interface, likely GitHub, showing the contents of the 'gps\_api' project. A red arrow points from the left sidebar to the project name 'gps\_api', which is highlighted with a red box. Another red arrow points from the bottom of the sidebar to the list of files in the main content area, also highlighting the file names with a red box. A large red callout box on the right side contains the text: 'Lokal repodan projeye yeni eklenen dosyalar' (New files added to the project from the local repository).

Name	Last commit	Last update
gps	ilk commit denememiz	40 minutes ago
gps_data	ilk commit denememiz	40 minutes ago
.gitignore	ilk commit denememiz	40 minutes ago
README.md	Initial commit	13 hours ago

# Gitlab'dan Lokal git reposuna

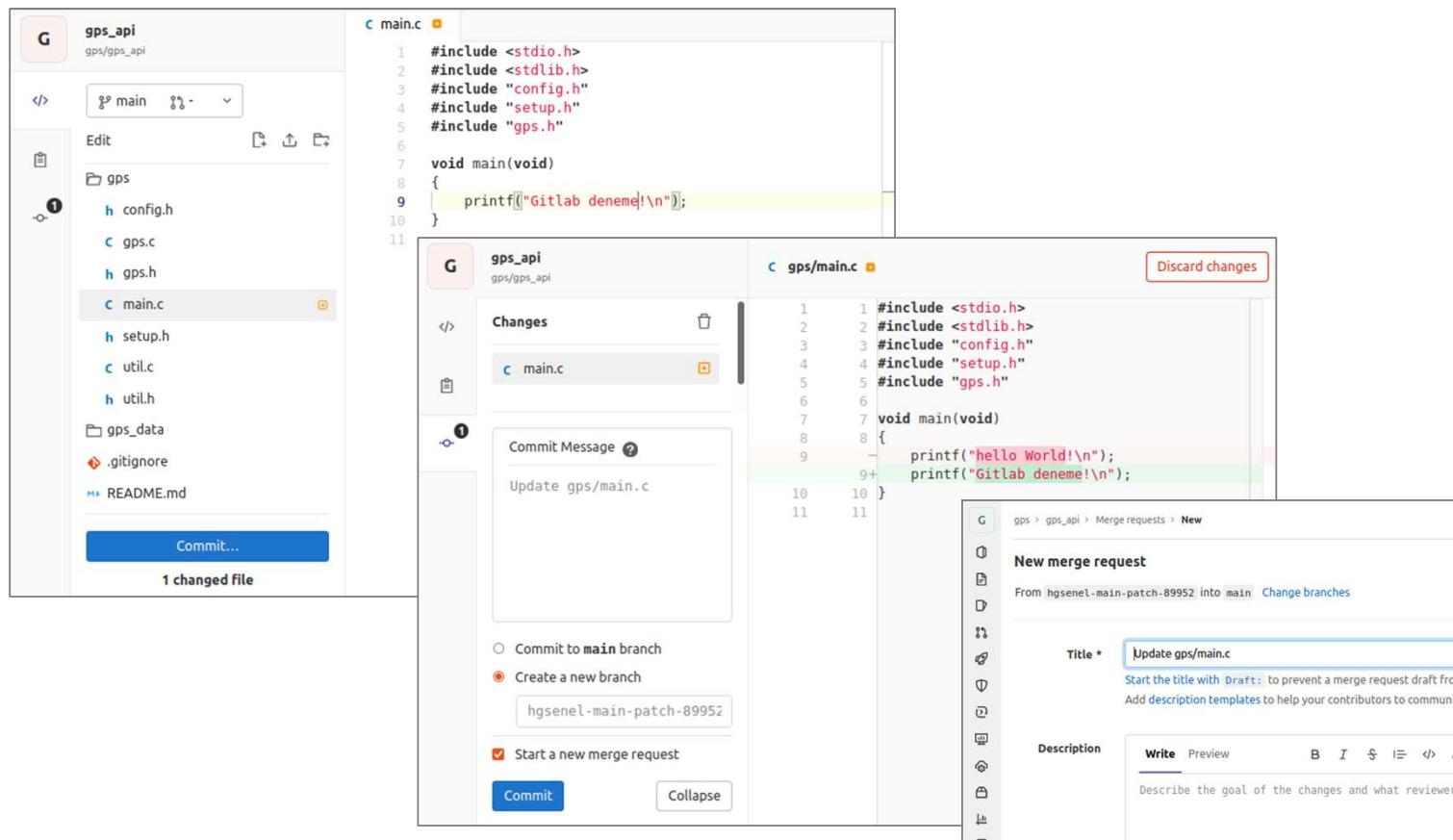
- Takım halinde yapılan çalışmalarında, birden fazla kişi Gitlab'a kod teslimi (commit) yapabilir.
- Lokalde oluşturduğunuz git reposunun Gitlab'daki değişiklikleri alması için, lokalde «**git pull git@127.0.0.1:gps/gps\_api**» komutunun kullanılması gereklidir.

```
adminpc@ubuntu:~/gps_api$ git pull git@127.0.0.1:gps/gps_api
From 127.0.0.1:gps/gps_api
 * branch            HEAD      -> FETCH_HEAD
Updating b977252..8fc8115
Fast-forward
 README.md |  92 ++++++++++++++++++++++++++++++++++++++++++++++++++++++
 gps/main.c |   2 ++
 2 files changed, 93 insertions(+), 1 deletion(-)
 create mode 100644 README.md
adminpc@ubuntu:~/gps_api$ cat gps/main.c
#include <stdio.h>
#include <stdlib.h>
#include "config.h"
#include "setup.h"
#include "gps.h"

void main(void)
{
    printf("Gitlab deneme!\n");
}
```

# Web IDE

- Kod değişikliğini, Gitlab arayüzünden «Web IDE» ile gerçekleştirebilirsiniz. «commit» yaptıktan sonra «merge request» ile onaylanması gereklidir.



# GITLAB CI/CD

# Bu bölümde yapılacaklar

- Bundan sonraki çalışmalarımızı, <https://gitlab.com/> üzerinde gerçekleştireceğiz. Şu aşamaya kadar lokalde, Gitlab kurulumunu ve nasıl kullanılacağını gördük.
- Lokalde Gitlab kurulumunu kullanabilmek ve Gitlab'ı şirketinizde kullanmak için site için geçerli bir adres ve oluşturacağınız/alacağınız bir SSL sertifikası gerekiyor.
- Bu nedenle, lokaldeki Gitlab kullanımına ara veriyoruz.
- Gitlab.com'a üyelik yapacağız ve CI işlemlerini bu sitede devam edeceğiz.
- Lokal makinenizde «**git init**» ile yeni bir repo oluşturulacak ve **gps.tar** ve **gps\_data.tar** dosyalarını repo içinde açarak, «**git add**» ile ekleyeceğiz ve ardından «**git commit**» yapacağız.
- «**mkdir build**» ile build isimli bir dizin oluşturacağız ve **.gitignore** dosyasıyla buradaki dosyaları repoya atmayacağız.
- Gitlabç.com'da yeni bir grup ve ardından proje oluşturacağız. Bu projenin git adresini loakl repoda «**git remote add origin**» ile lokal repoya tanımlayıp «git push» komutunu vereceğiz.
- Gitlab.com'daki projede «**merge request**»i onaylayıp lokalden gelen dosyaları projeye kabul edeceğiz
- CI süreci için «**gitlab-ci.yml**» dosyasını C++ şablonuyla projeye ekleyerek ve «commit» edeceğiz.
- Lokalde **gitlab-runner**'ı kuracağız ve gitlab.com'la bu lokal runner'ı kaydedeceğiz ve pipeline'ı lokal runner'da çalıştıracağız.

# Lokal Git Reposu Oluşturmak

- Size verilen «gps . tar» ve «gps \_ data . tar» dosyalarını ana ev dizinimize yükleyin.
- Gitlab’la çalışabilmek için, yeni bir dizin (örneğin **gps \_ api**) açalım ve dizinin içinde lokal bir repo’yu «git init» ile oluşturalım.
- Tar dosyalarını **gps \_ api** dizininin içinde «tar xvf» ile açalım. İki dizinde ~/gps \_ api/gps ve ~/gps \_ api/gps \_ data çeşitli dosyalar oluşturulacaktır.
- Derleme sırasında oluşacak EXE ve object dosyaları için ~/gps \_ api/build dizini oluşturalım.

```

adminpc@ubuntu:~$ ls
Desktop  Downloads  gps.tar  Pictures  Templates
Documents  gps_data.tar  Music  Public  Videos
adminpc@ubuntu:~$ mkdir gps_api ←
adminpc@ubuntu:~$ cd gps_api
adminpc@ubuntu:~/gps_api$ git init
Initialized empty Git repository in /home/adminpc/gps_api/.git/
adminpc@ubuntu:~/gps_api$ tar xvf ../*.tar
gps/
gps/util.h
gps/util.c
gps/gps.h
gps/gps.c
gps/config.h
gps/main.c
gps/setup.h
adminpc@ubuntu:~/gps_api$ tar xvf ../*_data.tar
gps_data/
gps_data/gps_data.txt
adminpc@ubuntu:~/gps_api$ git add gps/*
adminpc@ubuntu:~/gps_api$ git add gps_data/*
adminpc@ubuntu:~/gps_api$ mkdir build
adminpc@ubuntu:~/gps_api$ cat > .gitignore
build/*
adminpc@ubuntu:~/gps_api$ git add .gitignore
adminpc@ubuntu:~/gps_api$ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:  .gitignore
  new file:  gps/config.h
  new file:  gps/gps.c
  new file:  gps/gps.h
  new file:  gps/main.c
  new file:  gps/setup.h
  new file:  gps/util.c
  new file:  gps/util.h
  new file:  gps_data/gps_data.txt

```

Repo için **gps\_api** dizini oluşturarak içine girin.

«**git init**» ile repoyu oluşturun

Ev dizinize yüklediğiniz tar dosyalarını açınız.  
**gps** ve **gps\_data** dizinleri açılacak

**gps** ve **gps\_data** dizinlerindeki dosyaları repoya ekleyin.

**build** dizinini oluşturun ve bunu repodan **.gitignore** dosyası sayesinde çıkartın

# Lokal git reposu

- git reposunda teslim (commit) işlemini hangi kullanıcının yaptığı önemli bir bilgidir. Bu nedenle öncelikli olarak aşağıdaki komutlarla kullanıcı ismimizi ve e-posta adresimizi «**git config --global**» komutuyla git'e kaydetmeliyiz.

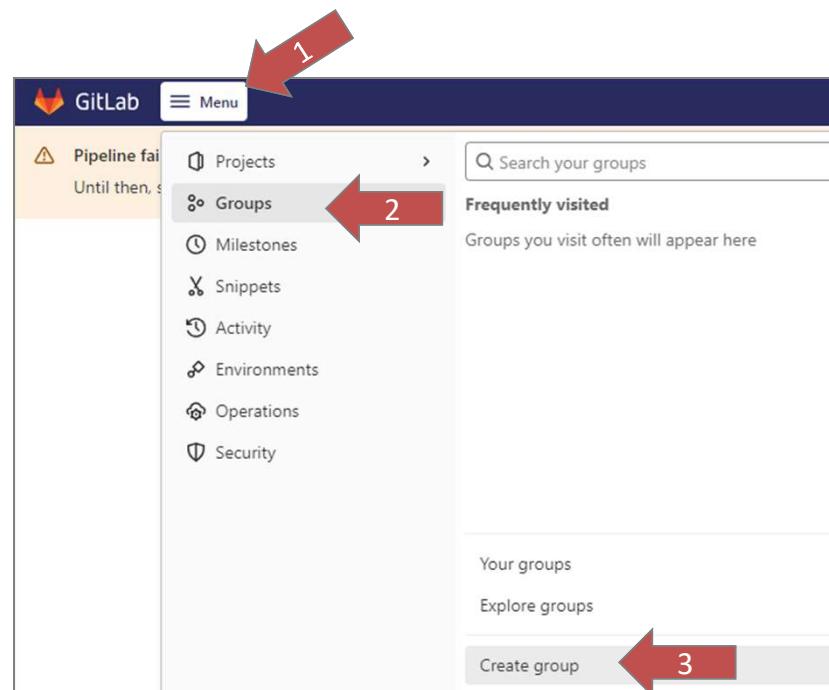
```
$ git config --global user.name "Hakan Senel"  
$ git config --global user.email "hgsenel@gmail.com"
```

- Reporomuz Gitlab'a yüklenmeden önce, «**git commit**» komutuyla hazırlık (staging) alanına aktarılması gereklidir.

```
adminpc@ubuntu:~/gps_api$ git config --global user.name "Hakan Senel"  
adminpc@ubuntu:~/gps_api$ git config --global user.email "hgsenel@gmail.com"  
adminpc@ubuntu:~/gps_api$ git commit -m "ilk commit denememiz"  
[master (root-commit) b977252] ilk commit denememiz  
 9 files changed, 890 insertions(+)  
  create mode 100644 .gitignore  
  create mode 100755 gps/config.h  
  create mode 100755 gps/gps.c  
  create mode 100755 gps/gps.h  
  create mode 100644 gps/main.c  
  create mode 100755 gps/setup.h  
  create mode 100755 gps/util.c  
  create mode 100755 gps/util.h  
  create mode 100755 gps_data/gps_data.txt
```

# Gitlab.com'a kayıt

- Gitlab.com'a kayıt olun ve kendi kullanıcı adınız ve şifrenizle giriş yapın.
- «**Menu**» kısmından, «**Groups**» ardından «**Create Group**» seçeneklerine tıklayın ve yeni bir grup oluşturun.



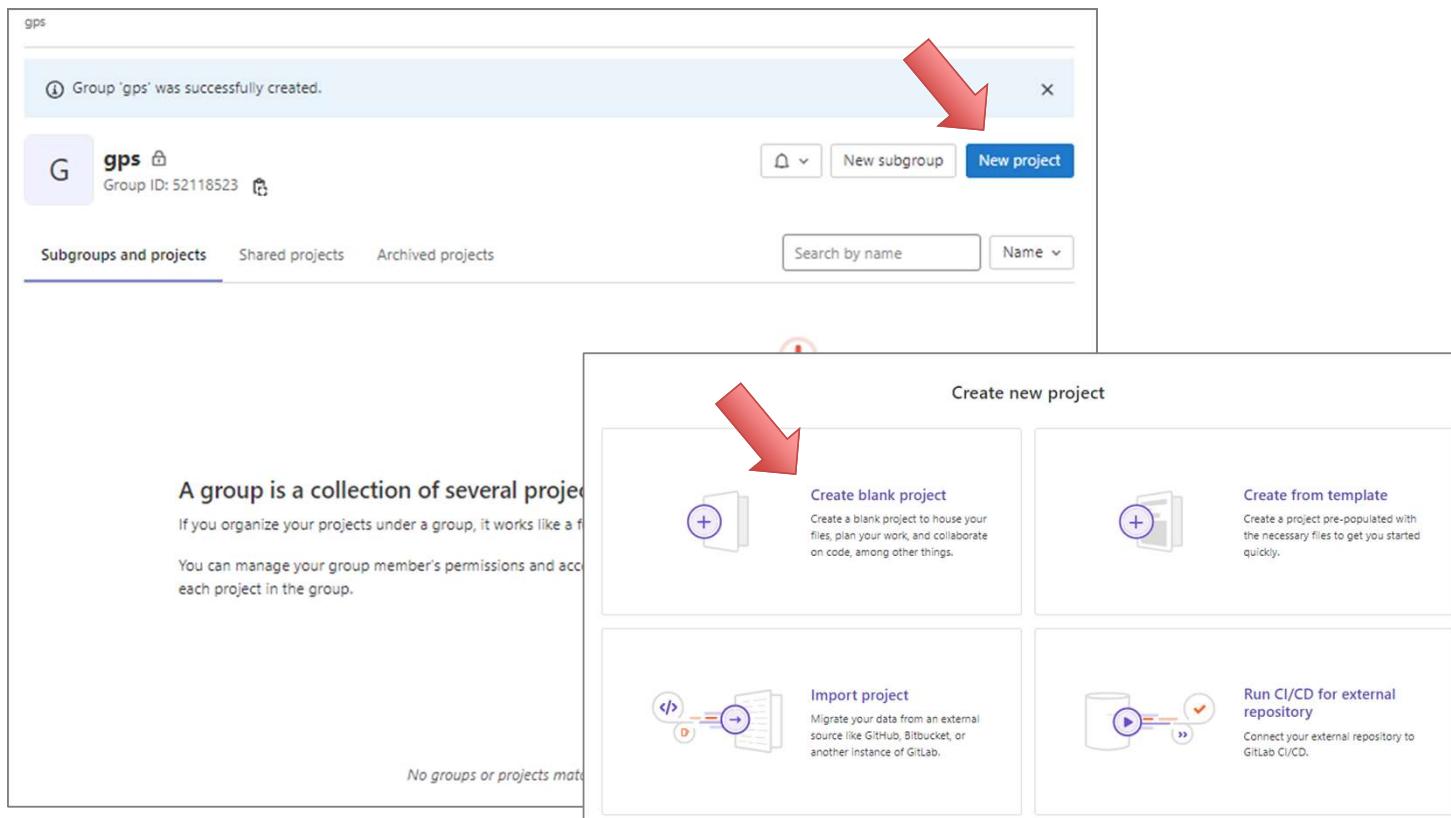
# Grup oluşturma

- Grup ismi olarak herhangi bir isim verebilirsiniz.
- Grubumuz, «**private**» veya «**public**» olabilir.
- Rol olarak «**Devops Engineer**» seçebilirsiniz.
- Takımınızda diğer elemanlar yer alıyorsa, bunların e-postalarını ekleyerek gruba davet edebilirsiniz.

The screenshot shows the 'Create group' page on GitLab. The 'Group name' field contains 'gps'. The 'Group URL' field contains 'https://gitlab.com/g6443'. The 'Visibility level' section has 'Private' selected. The 'Role' dropdown is set to 'Devops Engineer'. The 'Who will be using this group?' section has 'My company or team' selected. The 'What will you use this group for?' dropdown is set to 'I want to store my code'. The 'Invite Members (optional)' section shows an email input 'member1@company.com' and a 'Create group' button at the bottom.

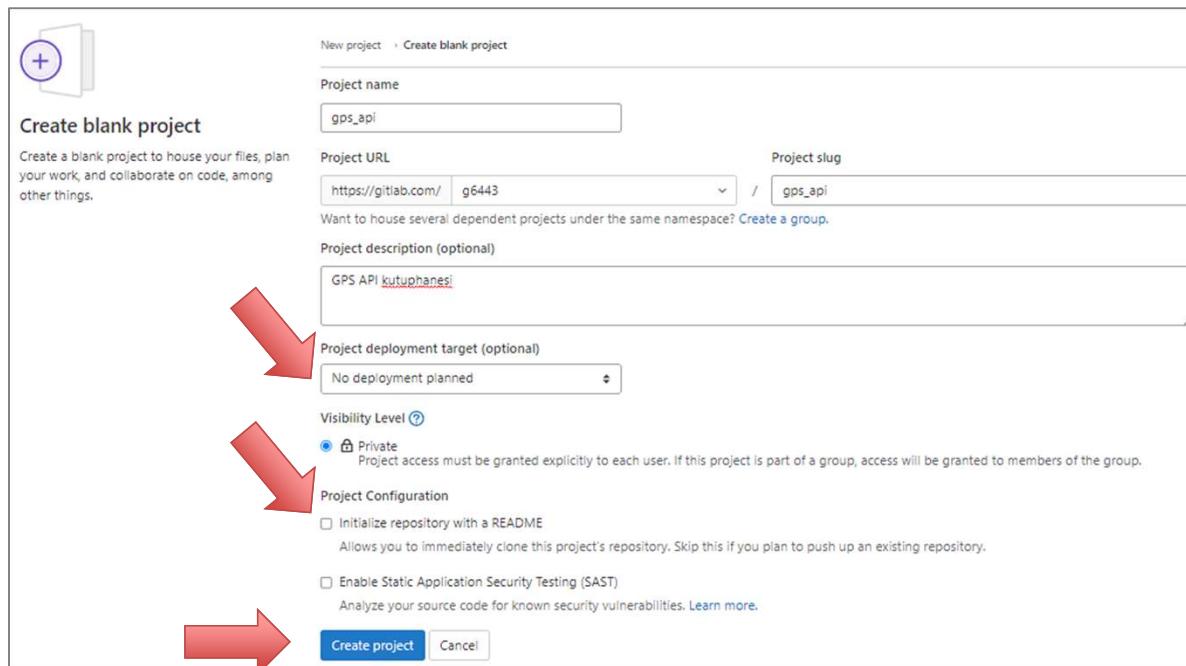
# Proje oluşturma

- Grubun altında proje oluşturulmalıdır. Bir sonraki ekranın «**New Project**» butonuna basarak, yeni bir proje oluşturabiliriz.



# Proje oluşturma

- Projemize «gps\_api» ismini verebiliriz.
- «Project deployment target» kısmında «Not deployment planned» opsyonu seçilebilir.
- «Initialize repository with a README» kısmındaki «check»i kaldırılm çünkü lokaldeki dosyaların hepsi buraya yüklenecek. Lokalde **README.md** dosyamız zaten var.



# SSH Anahtarının oluşturulması

- Eğer daha önce oluşturmadıysanız, lokal git reponuzun olduğu yerde «**ssh-keygen**» komutuyla açık ve gizli anahtarlarımıza oluşturalım ve açık anahtarımızı bulunduran «**~/.ssh/id\_rsa.pub**» dosyasının içeriğini panoya (Ctrl-C) kopyalayalım.

```
adminpc@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adminpc/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/adminpc/.ssh/id_rsa
Your public key has been saved in /home/adminpc/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1v+7ymxjM6NyZnMB87EmdOywSkdKlgPuze8j08V5g adminpc@ubuntu
The key's randomart image is:
+---[RSA 3072]---+
| . .
| ..o
| *oo
| .-
| .. .+.
| . XooE .
| ..X..@..
| o .^B=0
| o+*o.+o
+---[SHA256]---+
adminpc@ubuntu:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDc+yvBRglZdiyvRAUnDRuqjmezXl8RiTURqWapQ4vcn5++NKtm1D/35a9YPPJvLTy6p24ULTwPGUM6PygjEOvxH0JYbPEnsbt
2EXrTxng1j7GhCs2PuNV9sSHRBkP7
GgPS3nZ2G5dy3gzNxxvAhdbATgkNd4
NFQaJ8cLM5VK5j9dcIM6yGYxvvvKhU
05u2lz0= adminpc@ubuntu
adminpc@ubuntu:~$
```

Copy

Copy as HTML

Paste

Read-Only

Preferences

New Window

New Tab

Show Menubar

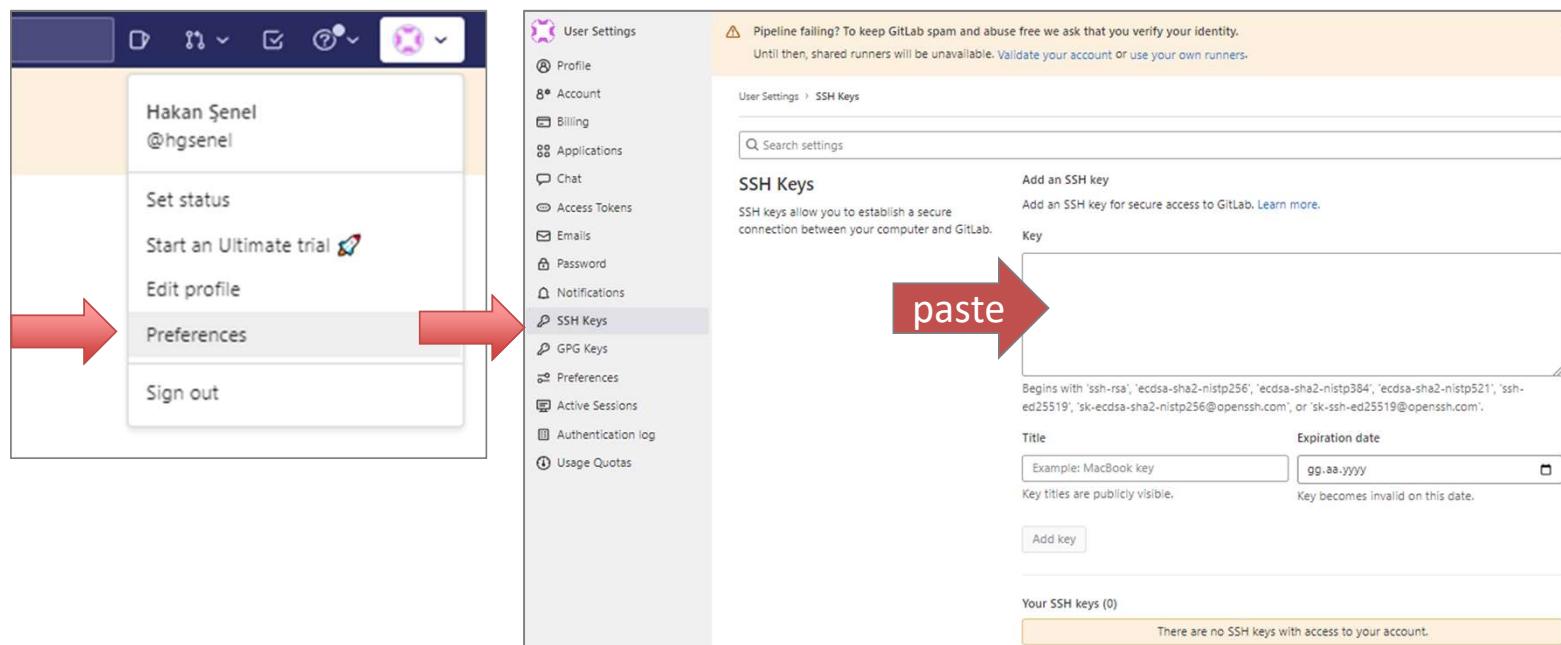
Sign in with

Improvements to GitHub, but also removes some deprecated features. Visit the [deprecations page](#) to see what is scheduled for removal in 15.0, and check for any **breaking changes** that could impact your workflow.

Twitter

# SSH Anahtarının girilmesi

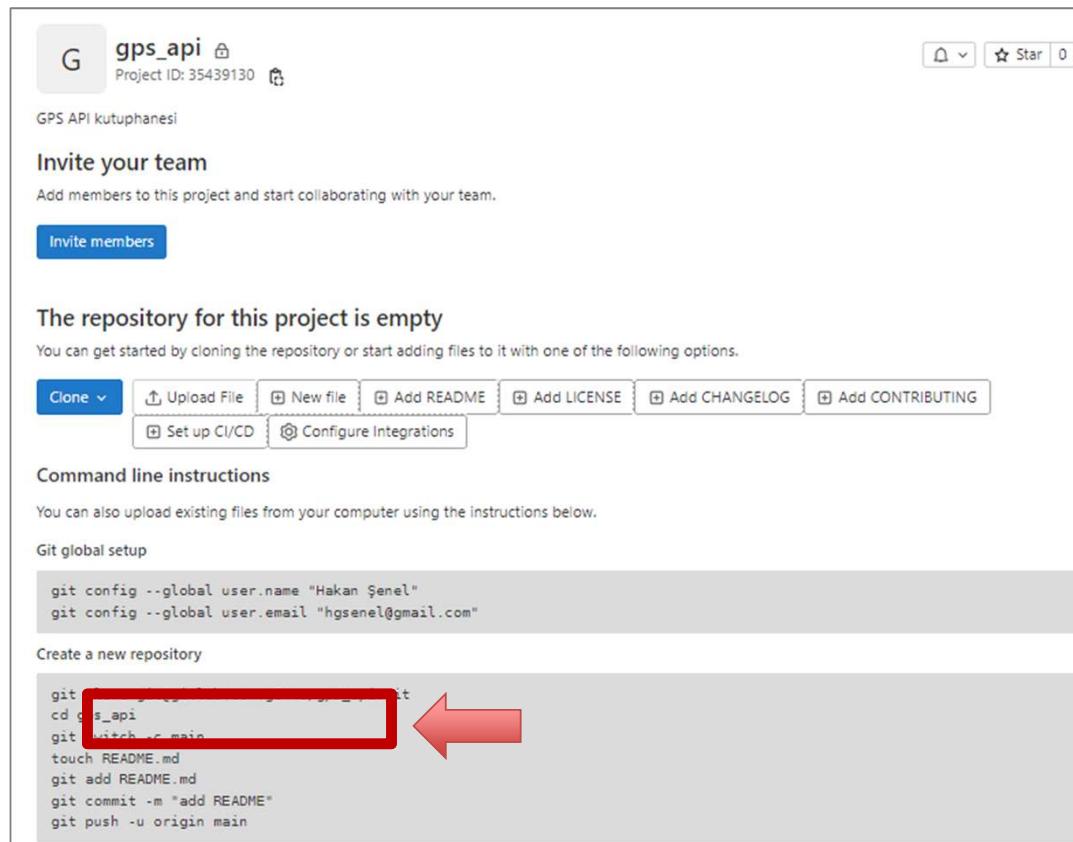
- SSH anahtarı henüz girmedigimizden, projemize lokal git repomuzdan «**push**» işlemi yapamayız. Bu nedenle, **gitlab.com**'da sağ üst köşeden «**Preferences**» opsiyonunu seçmemiz ve ardından sol taraftaki menüden «**SSH keys**» opsiyonunu seçmemiz gereklidir.



# Proje git adresinin bulunması

- **gitlab.com**da oluşturulan projenin bilgi sayfasının aşağısında, projeye nasıl erişilebileceği ile ilgili bilgi bulunmaktadır.

FORMAT: git@gitlab.com:grup/projeadi.git  
git@gitlab.com:g6443/gps\_api.git



The screenshot shows a GitLab project page for 'gps\_api'. The top navigation bar includes a 'G' icon, the project name 'gps\_api', a star count of '0', and a 'Star' button. Below the header, there's a section titled 'GPS API kutuphanesi' with a 'Invite your team' button. A message states 'The repository for this project is empty' and provides options like 'Clone', 'Upload File', and 'New file'. A red arrow points to the command-line instructions section at the bottom, which contains the following text:

```
git config --global user.name "Hakan Şenel"
git config --global user.email "hgsenel@gmail.com"

Create a new repository

git clone https://gitlab.com/g6443/gps_api.git
cd gps_api
git branch -r main
touch README.md
git add README.md
git commit -m "add README"
git push -u origin main
```

# Lokal reponun Gitlab'a yüklenmesi

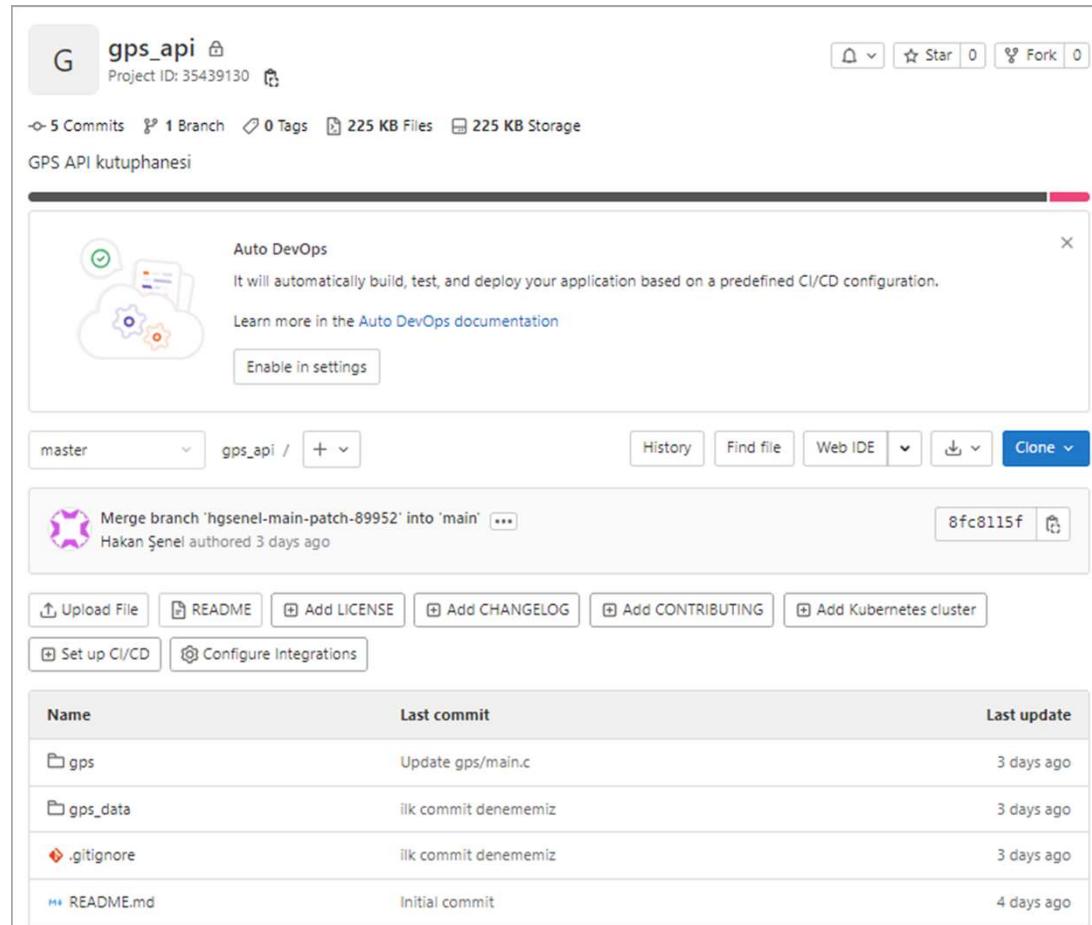
- Komut satırında, projenin Gitlab adresinin ne olduğunu «**git remote add origin**» komutuyla belirtmemiz ve ardından «**git push**» komutuyla, «**master**» dala Gitlab'a yüklememiz gereklidir.

```
git remote add origin git@gitlab.com:g6443/gps_api.git
git push --set-upstream origin master
```

```
adminpc@ubuntu:~/gps_api$ git remote add origin git@gitlab.com:g6443/gps_api.git
adminpc@ubuntu:~/gps_api$ git push --set-upstream origin master
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ECDSA key fingerprint is SHA256:HbW3g8zUjNSksFbqTiUWPWg2Bq1x8xdGUrliXFzSnUw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com,172.65.251.78' (ECDSA) to the list of known hosts.
Enumerating objects: 23, done.
Counting objects: 100% (23/23), done.
Delta compression using up to 2 threads
Compressing objects: 100% (20/20), done.
Writing objects: 100% (23/23), 11.78 KiB | 804.00 KiB/s, done.
Total 23 (delta 5), reused 0 (delta 0)
To gitlab.com:g6443/gps_api.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

# Proje bilgi sayfasi

- Eğer proje sayfası yenilenirse, lokal git reposunun projeye aktarıldığını görebilirsiniz.



The screenshot shows a GitHub project page for 'gps\_api'. At the top, there's a modal window titled 'Auto DevOps' with the subtext: 'It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.' It includes a 'Learn more in the Auto DevOps documentation' link and a 'Enable in settings' button. Below the modal, the main project interface is visible. It shows a commit history with one merge commit from 'hgsenel-main-patch-89952' into 'main' by Hakan Şenel, dated 3 days ago. There are buttons for 'Upload File', 'README', 'Add LICENSE', 'Add CHANGELOG', 'Add CONTRIBUTING', 'Add Kubernetes cluster', 'Set up CI/CD', and 'Configure Integrations'. A table at the bottom lists repository files with their names, last commit messages, and last update times. The table has columns for 'Name', 'Last commit', and 'Last update'. The data is as follows:

Name	Last commit	Last update
gps	Update gps/main.c	3 days ago
gps_data	ilk commit denememiz	3 days ago
.gitignore	ilk commit denememiz	3 days ago
README.md	Initial commit	4 days ago

## Önemli Not: «merge request»

- Eğer projede daha önce bulunan dosyalar varsa, «**README .md**» gibi, «**Merge Request**» yapılması gereklidir.
- Hatırlayın: Bu nedenle proje oluşturulurken, «**README .md**» dosyasını otomatik olarak oluşturmamasını istedik. Eğer bu dosya olsayıdı, «**Merge Request**» başlatacak ve onay mekanizmasının işletilmesi gerekecekti.

# Gitlab CI/CD – Terminoloji

- **Pipeline** (sureç): CI/CD süreçlerinin tanımlanması için kullanılan bileşendir.
  - «Pipeline» içinde «aşamalar» ve «işler» (stages, jobs) tanımlanır. Örneğin: «build», «test», «deploy» gibi aşama isimleri bulunur.
- **Stage** (aşama): Yapılacak işlerin kronolojik sırasını belirler. İlk önce «build» yapılır, ardından «test» yapılır, vb.
- **Job** (iş): Her aşama içinde gerçekleştirilen adımlardır.
  - Her aşamada birden fazla iş olabilir.
  - Betik programlar gibi komut satırından verilen komutlar da iş olarak çalıştırılabilir.
- **Runner** (çalıştırıcı): Aşamalar içinde tanımlı işler içinde gömülü komutları çalıştmak için kullanılan açık kaynak kodlu bir uygulamadır.

# Gitlab.com'da «runner» çalışma

- **gitlab.com**, küçük projeler için paylaşılan «runner» imkanı tanımaktadır.
- Her kullanıcıyla ücretsiz dakikalar, paylaşılan «runner»ların kullanılabilmesini sağlamaktadır.
- Diğer yandan, kendi makinenizde «**gitlab-runner**» kurarak, **gitlab.com**'a bunu kaydedebilir ve CI/CD süreçlerinizde kullanabilirsiniz.
- Not: Gitlab'daki runner kavramı Jenkins'te «Build Agent» olarak bilinmektedir.

# gitlab-runner nedir?

- gitlab-runner Gitlab içinde, işleri (job) çalıştırın ek bir yazılımdır. Ayrıca yüklenmelidir.
- Docker (sadece «engine» olabilir) ve Gitlab sisteme yüklenikten sonra, runner'ın da kurulması ve Gitlab'a kaydedilmesi (register) gerekmektedir.
- Ubuntu ve Debian sürümlerinde aşağıdaki komut çalıştırılmalı ve kod deposu bilgileri işletim sisteminde kaydedilmelidir:

```
$ sudo curl -L --output /usr/local/bin/gitlab-runner  
"https://gitlab-runner-  
downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-  
amd64"  
$ sudo chmod +x /usr/local/bin/gitlab-runner  
$ sudo useradd --comment 'GitLab Runner' --create-home gitlab-  
runner --shell /bin/bash  
$ sudo gitlab-runner install --user=gitlab-runner --working-  
directory=/home/gitlab-runner  
$ sudo gitlab-runner start
```

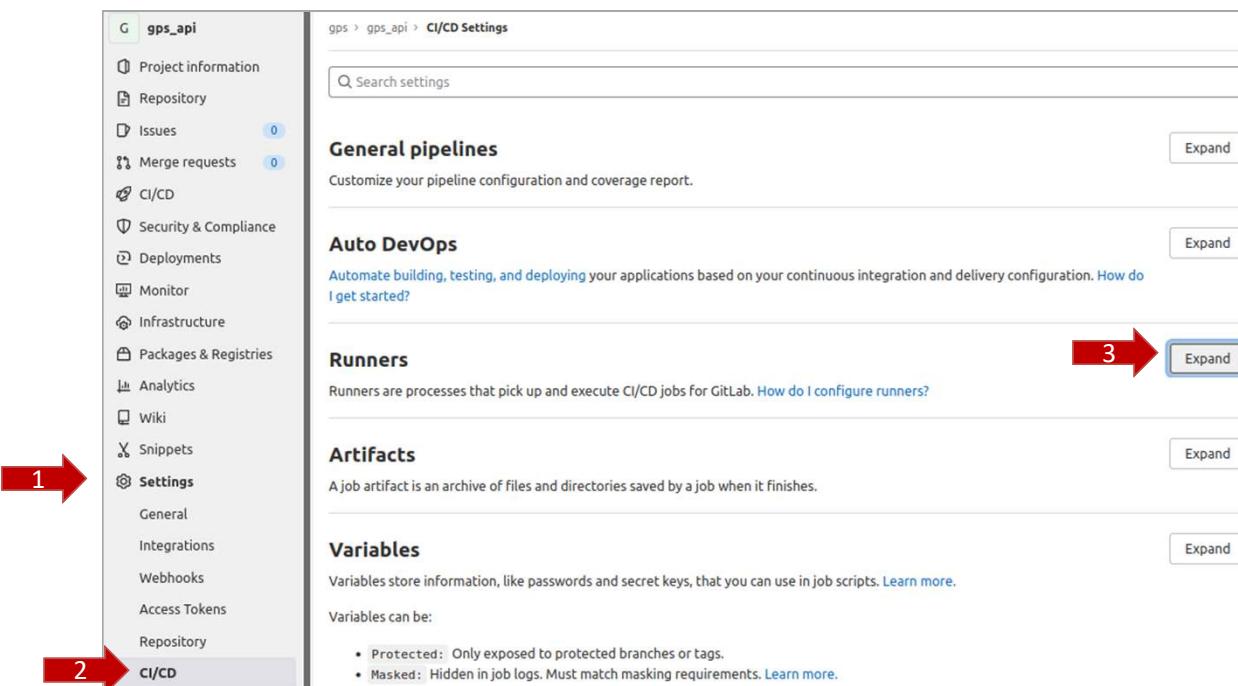
<https://docs.gitlab.com/runner/install/linux-manually.html>

# «gitlab-runner» kurulumu

- Kurulum sırasında, «**gitlab-runner**» paketi **amazonaws.com** adresinden **/usr/local/bin/gitlab-runner** dizinine aktarıldı.
- Bu dizine çalışma yetkisi verdik.
- «**gitlab-runner**» isimli bir kullanıcı oluşturduk ve ev adresi olarak **/home/gitlab-runner** verildi. Ayrıca **Bash** kabuk programını kullanması istendi.
- Yapılacak bütün CI (Continuous Integration) işleri, **/home/gitlab-runner** dizini içinde **gitlab-runner** kullanıcısı tarafından gerçekleştirilecek.
- En son olarak **gitlab-runner** programını kurduktan sonra çalıştık.

# «runner» kaydı

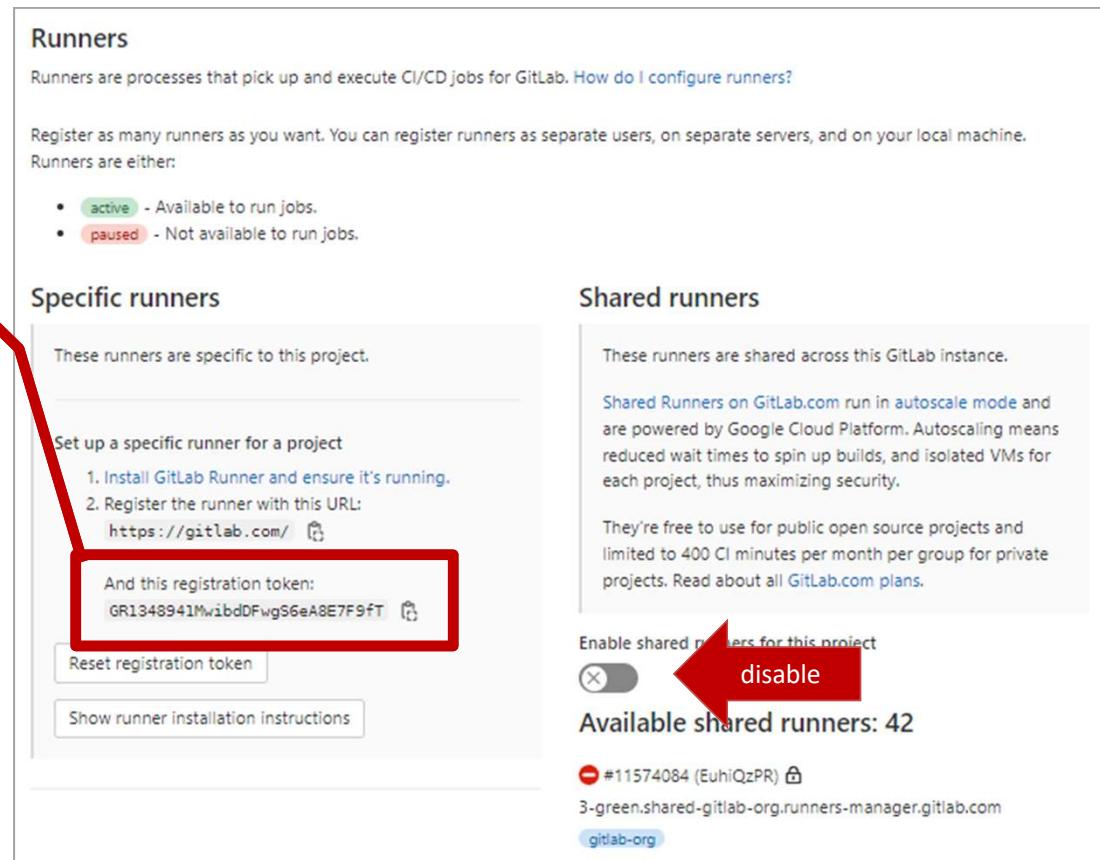
- «git-runner» paketini kurmamıza rağmen, herhangi bir «runner» örneğini [gitlab.com](https://gitlab.com)'a henüz tanıtmadık.
- Proje menüsünde, «Settings» bölümünde «CI/CD» opsiyonunu seçmelisiniz. «runner»ların kaydedilmesi için «Runners» bölümünün yanındaki «Expand» butonuna basarak gerekli bilgiyi alabiliriz.



# «runner» kaydı

- «Runners» bölümü altında, «Shared runners» olarak listelenenler, bütün `gitlab.com`'da ortak kullanılan «runner»ları belirtmektedir.
- Kendi «runner» uygulamamızı kurduğumuz için bu proje için ortak olanları kullanmayacağız.

Lokal «runner»ı tanıtambilmek için gerekli şifreyi kopyala.  
«Ctrl-C»



The screenshot shows the 'Runners' configuration page on GitLab. On the left, under 'Specific runners', there's a section for setting up a runner for the current project. It includes instructions to install the GitLab Runner and register it with a URL (https://gitlab.com/) and a registration token (GR1348941MwibdDFwgS6eA8E7F9fT). A red box highlights the registration token field. On the right, under 'Shared runners', it says 'These runners are shared across this GitLab instance.' Below this, it describes shared runners running in autoscale mode on Google Cloud Platform. A red arrow points to a toggle switch labeled 'enable' with the word 'disable' written above it. It also shows 'Available shared runners: 42'. At the bottom, there's a link to a merge request (#11574084) and a note about the runner being part of the 'gitlab-org' group.

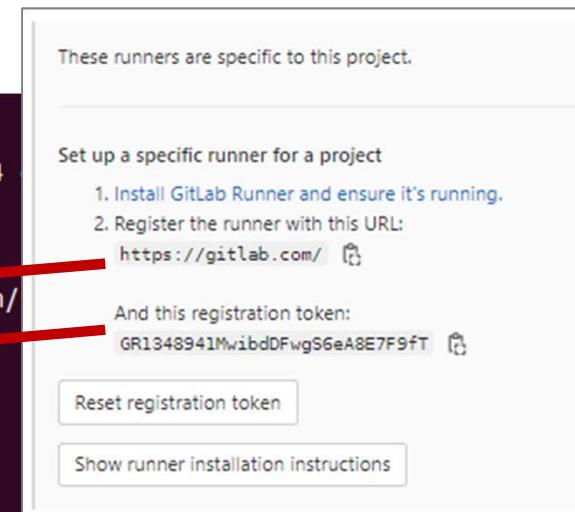
# «runner» kaydı

- **gitlab.com'a lokaldeki «gitlab-runner»'ı tanıtabilmek için «sudo gitlab-runner register» komutunu kullanmamız gereklidir.**

```
adminpc@ubuntu:~$ sudo gitlab-runner register
[sudo] password for adminpc:
Runtime platform                                arch=amd64
version=bd40e3da version=14.9.1
Running in system-mode.

Enter the GitLab instance URL (for example, https://gitlab.com)
https://gitlab.com
Enter the registration token:
GR1348941MwibdDFwgS6eA8E7F9fT
Enter a description for the runner:
[ubuntu]: VMware ubuntu
Enter tags for the runner (comma-separated):
Ubuntu, VMware, 20.04
Enter optional maintenance note for the runner:

Registering runner... succeeded                  runner=GR134894
Enter an executor: shell, docker-ssh+machine, kubernetes, custom, docker, docker-ssh,
parallels, ssh, virtualbox, docker+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the
config should be automatically reloaded!
```



# «executor»

- «runner» kaydı yapılırken, «runner»nın ne tür bir çalışma ortamını sağlayacağı belirtilmelidir.
- Burada, «shell» diyerek, Bash kabuk programının kullanılacağını belirteceğiz.
- Eğer istenirse, Docker, Kubernetes gibi ortamlar da «executor» olarak verilebilir.

```
Enter an executor: shell, docker-ssh+machine, kubernetes, custom, docker, docker-ssh,
parallels, ssh, virtualbox, docker+machine:
shell
Runner registered successfully. Feel free to start it, but if it's running already the
config should be automatically reloaded!
adminpc@ubuntu:~$
```

- **ÖNEMLİ NOT:** Entegrasyon için gerekli yazılımların hepsinin (örneğin gcc, g++, Java JDK, vb) önceden «runner»ının bulunduğu sisteme kurulması gereklidir.

# CI/CD «pipeline» oluşturma



- Jenkins'te CI/CD çözümlerini kendiniz üretmeniz gerekmektedir. Github'dan veya kendi git reponuzdan çektiğiniz kodları, betikler sayesinde derlemeniz/integre etmelisiniz.
- Gitlab'ın en önemli özelliklerinden biri, CI/CD çözümlerinin de sistemin içinde bulunmasıdır.
- CI/CD özelliklerini kullanabilmek için projenize, «`.gitlab-ci.yml`» isimli bir dosyayı eklemeniz gerekmektedir.
- Bu dosya, lokal repoda oluşturularak «`git push`»la Gitlab'a gönderilebilir veya Web IDE arayüzü ile de oluşturulabilirsiniz.
- Web IDE üzerinden dosyanın oluşturulması daha avantajlıdır.

[https://docs.gitlab.com/ee/ci/quick\\_start/](https://docs.gitlab.com/ee/ci/quick_start/)

# Not: YAML ne demek?

- YAML'ın açılımı «**Y**et **A**nother **M**arkup **LY**AML **A**in't **M**arkup **L- İnsanlar tarafından okunabilen ve anlaşılabilen türden yapılandırma dosyalarının oluşturulması amacıyla kullanılmaktadır.**

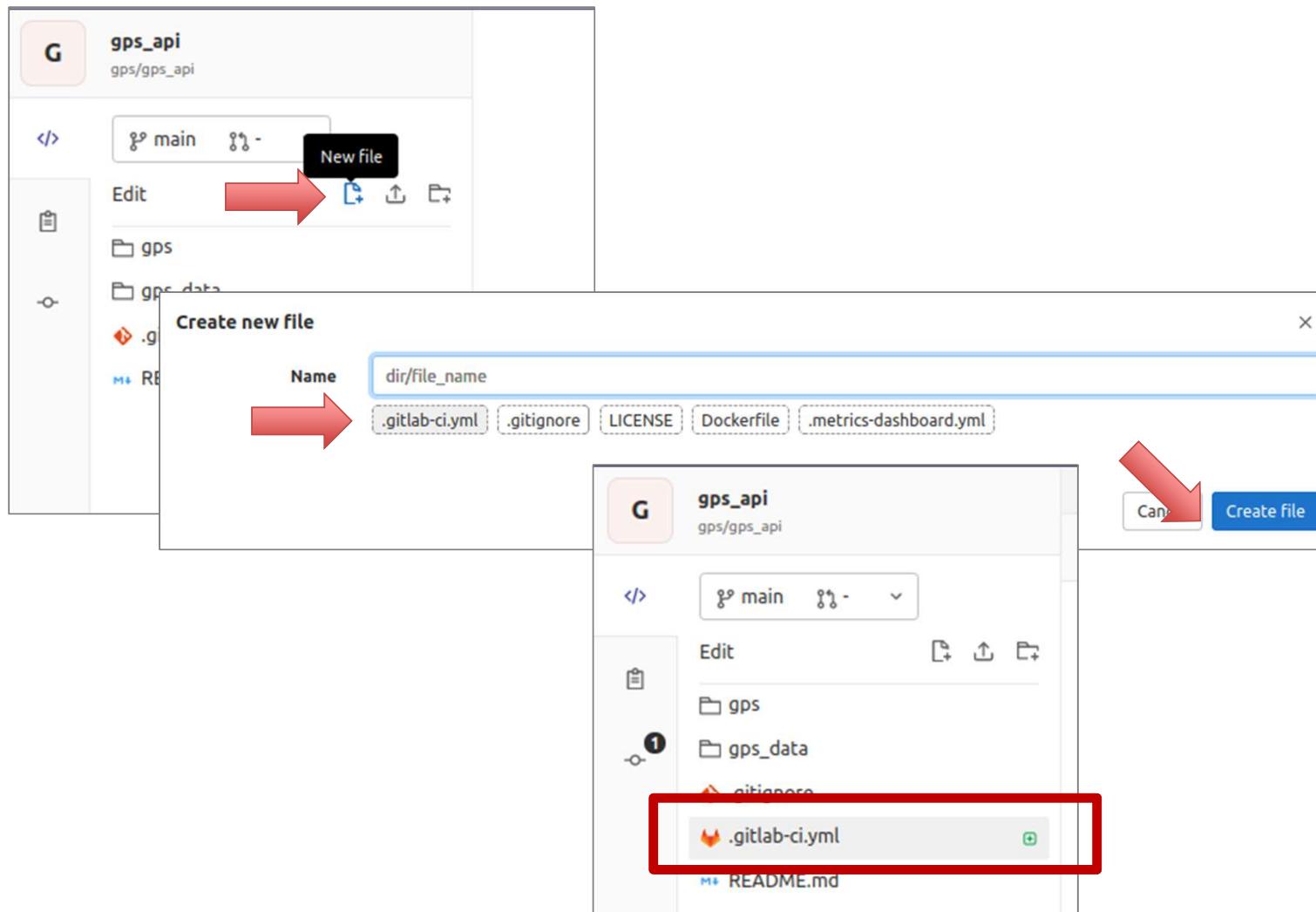
<https://yaml.org/>

<https://www.cloudbees.com/blog/yaml-tutorial-everything-you-need-get-started>

```
---  
# comment  
doe: "a deer, a female deer"  
ray: "a drop of golden sun"  
pi: 3.14159  
xmas: true  
french-hens: 3  
calling-birds:  
  - huey  
  - dewey  
  - louie  
  - fred  
xmas-fifth-day:  
  calling-birds: four  
  french-hens: 3  
  golden-rings: 5  
  partridges:  
    count: 1  
    location: "a pear tree"  
  turtle-doves: two
```

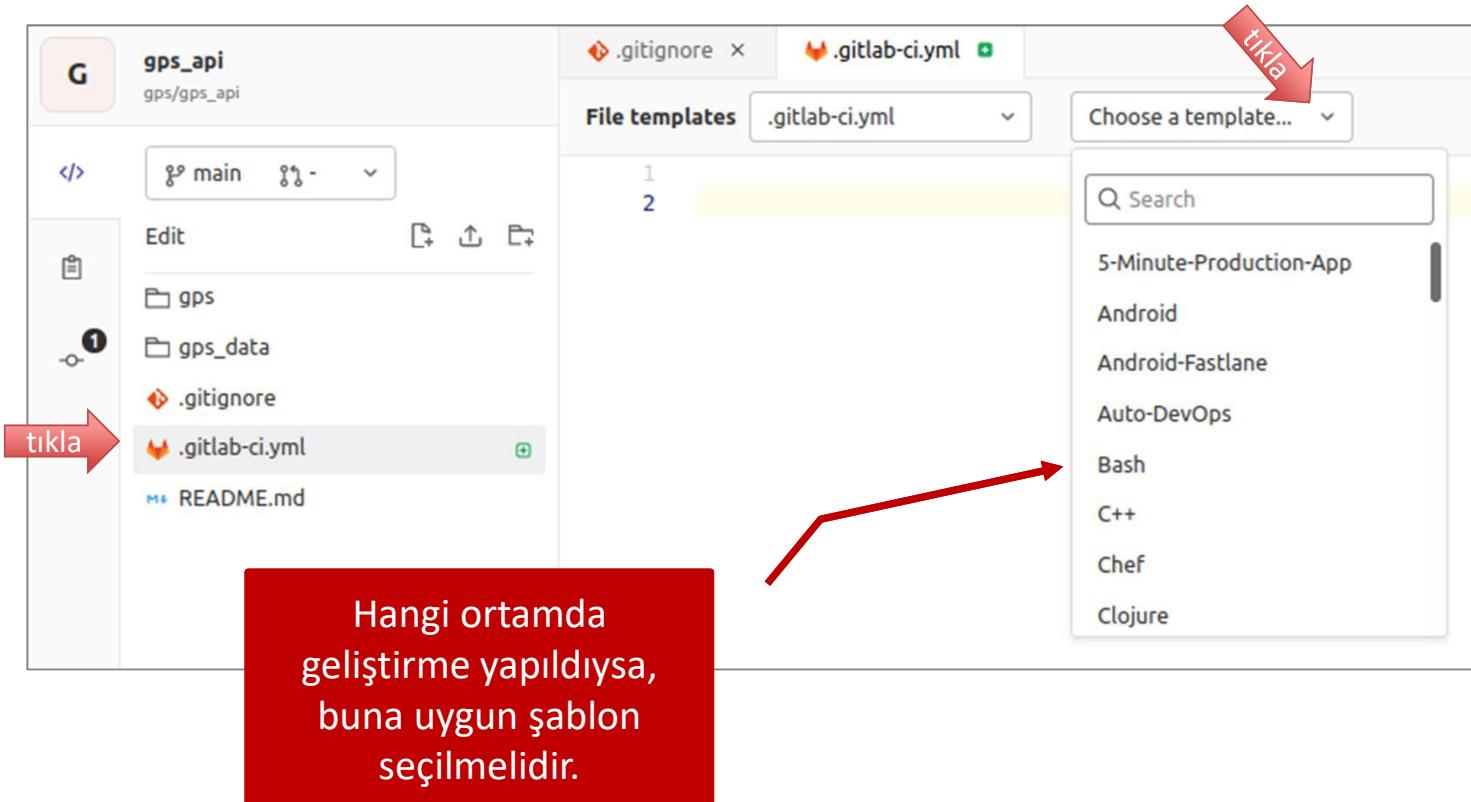
# WEB IDE ile .gitlab-ci.yml oluşturma

**GANTEK**  
INTEGRATING FUTURE



# WEB IDE ile .gitlab-ci.yml oluşturma

- Gitlab, `.gitlab-ci.yml` dosyası için farklı geliştirme ortamları ve dilleri için çeşitli şablonlar sunmaktadır. `gps_api` projesi C/C++ ile yazıldığından C++ opsyonu seçilmelidir. (Ancak burada **Bash** opsyonunu ilk örnek olarak seçeceğiz)



# .gitlab-ci.yml (C++ şablon)



```
# This file is a template, and might need editing before it works on your project.
# To contribute improvements to CI/CD templates, please follow the Development guide at:
# https://docs.gitlab.com/ee/development/cicd/templates.html
# This specific template is located at:
# https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/C++.gitlab-ci.yml

# use the official gcc image, based on debian
# can use verions as well, like gcc:5.2
# see https://hub.docker.com/_/gcc/

image: gcc

build:
  stage: build
  # instead of calling g++ directly you can also use some build toolkit like make
  # install the necessary build tools when needed
  # before_script:
  #   - apt update && apt -y install make autoconf
  script:
    - g++ helloworld.cpp -o mybinary
  artifacts:
    paths:
      - mybinary
      # depending on your build setup it's a good idea to cache outputs to reduce build time
      # cache:
      #   paths:
      #     - "*.o"

  # run tests using the binary built before
test:
  stage: test
  script:
    - ./runmytests.sh
```

# .gitlab-ci.yml (Bash şablon)

```
# See https://docs.gitlab.com/ee/ci/yaml/index.html for all available options
```

```
# you can delete this line if you're not using Docker
image: busybox:latest
```



```
before_script:
  - echo "Before script section"
  - echo "For example you might run an update here or install a build dependency"
  - echo "Or perhaps you might print out some debugging details"

after_script:
  - echo "After script section"
  - echo "For example you might do some cleanup here"

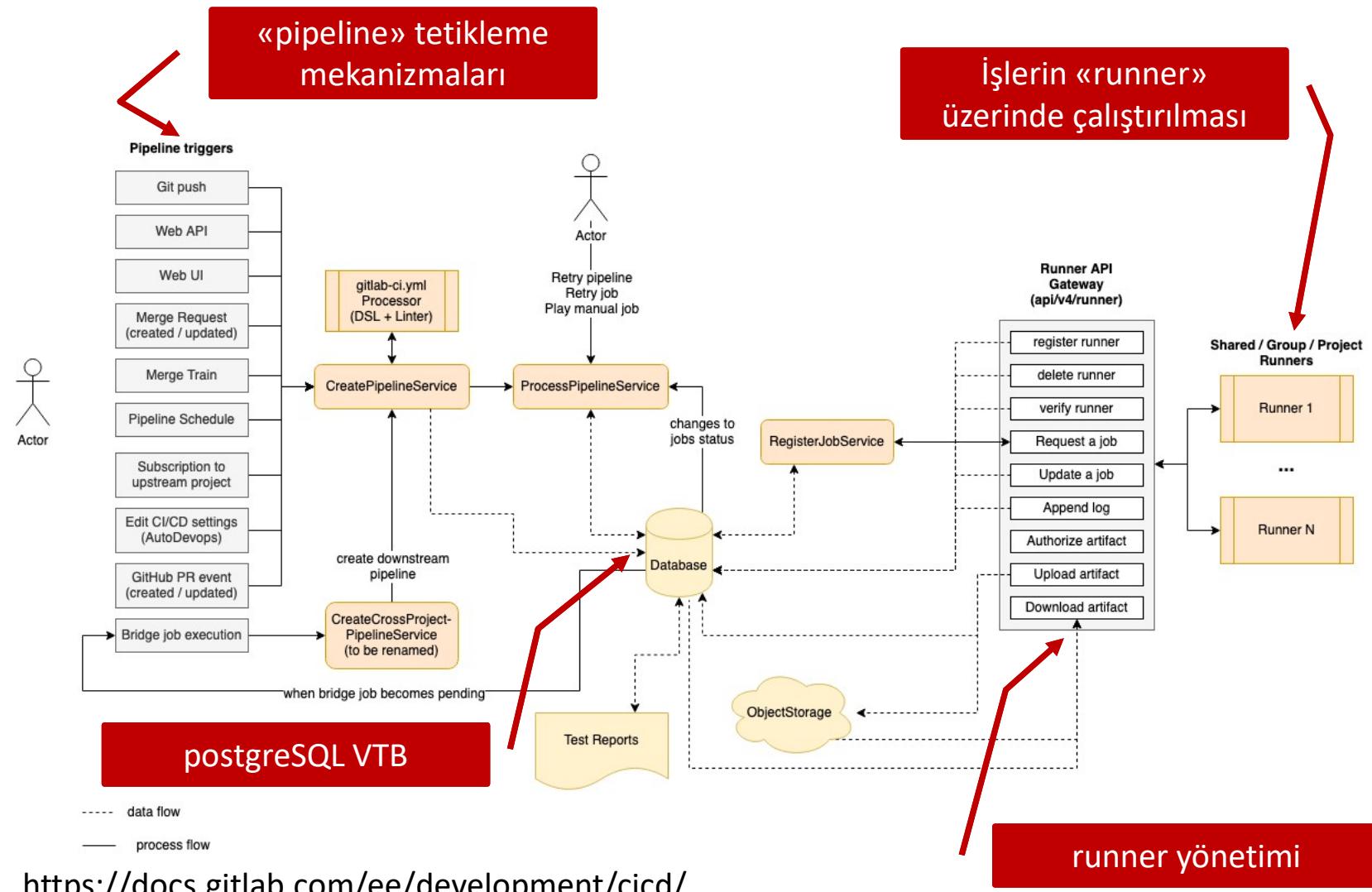
build1:
  stage: build
  script:
    - echo "Do your build here"

test1:
  stage: test
  script:
    - echo "Do a test here"
    - echo "For example run a test suite"

test2:
  stage: test
  script:
    - echo "Do another parallel test here"
    - echo "For example run a lint test"

deploy1:
  stage: deploy
  script:
    - echo "Do your deploy here"
```

# Gitlab CI Mimarisi



<https://docs.gitlab.com/ee/development/cicd/>

# Gitlab CI YML dosyasının yapısı

- **.gitlab-ci .yml** dosyası, YML dosyası genelinde geçerli anahtar kelimelerle iş (job) için kullanılan anahtar kelimelerden oluşur.
- YML dosyası genelinde kullanılan anahtar kelimeler şunlardır:
  - **default**: iş (job) anahtar kelimeleri için varsayılan değerleri belirtir.
  - **include**: diğer YAML dosyalarındaki tanımları alır
  - **stages**: «pipeline» aşamalarının isimlerini ve çalışma sırasını belirtir.
  - **variables**: İşlerin (job) bütününde kullanılacak CI/CD değişkenlerini tanımlar.
  - **workflow**: ne tür «pipeline» tiplerinin çalışacağını kontrol eder.

# default

- **default** anahtar kelimesi, bütün YML dosyası içinde geçerli olacak varsayılan işleri tanımlamak için kullanılabilir.

```
default:  
  image: ruby  
  include:  
    - local: '.sirket_tanimlar.yml'  
    - remote: 'https://gitlab.com/gps/gps_get/deneme.yml'  
  after-script:  
    - echo "asama bitti"  
  before-script:  
    - echo "asamaya girildi"
```

«runner» bütün aşamaları gcc Docker imajında çalıştıracak.

Her iş (job) bittiğinde «asama bitti» yazacak.

«runner» lokal repodaki ve uzaktaki makinedeki YML dosyalarını yükleyecek.

Her iş (job) başlamadan önce «asamaya girildi» yazacak.

# stages

- **stages** anahtar kelimesi, bütün YML dosyası içinde kronolojik olarak yürütülecek aşamaları belirtmektedir.

```
stages:  
  - build  
  - test  
  - deploy
```

- «**build**» içinde tanımlı bütün işler (jobs) paralel yürütülür.
- «**build**» içindeki bütün işler tamamlanınca, «**test**» aşamasındaki işler paralel yürütülür.
- «**test**» içindeki bütün işler tamamlanınca, «**deploy**» işleri paralel olarak başlatılır.
- «**deploy**» içindeki bütün işler bitince, «**pipeline**» «**passed**» olarak işaretlenir.

# stages

- Eğer bir iş yani «**job**» için hangi aşamada çalışacağı belirtilmemişse, «**test**» aşamasında çalıştırılacağı kabul edilir.
- Eğer bir «**stage**» tanımlanmış ancak hiçbir «**job**» bunu kullanmıyorsa, bu «**stage**» «**pipeline**» içinde görülebilir nitelikte yani kullanımda değildir.

# job

- Herhangi bir aşama, «**stage**» içinde yapılacak işleri tanımlamak için kullanılır.

```
job1:  
  stage: test  
  script:  
    - script_1.sh
```

Bu iş «**test**» aşamasında yürütülecek ve «**script\_1**» betiği çalıştırılacak.

```
job2:  
  stage: test  
  script:  
    - script_2.sh  
  allow_failure: true
```

Bu iş başarısız olabilir, olursa «**pipeline**» devam etsin.

```
job3:  
  stage: deploy  
  script:  
    - deploy_to_staging.sh
```

Diger örnek:  
**allow\_failure:**  
**exit\_codes:**  
 - 137  
 -255

# job

```
job3:  
  stage: deploy  
  before-script:  
    - echo "job3 basladi"  
    - deploy_job3_clean.sh  
  after-script:  
    - echo "job3 bitti"  
  script:  
    - script3.sh
```

«job» başlamadan önce  
çalıştırılacak komutlar

«job» bittikten sonra  
çalıştırılacak komut

Bu iş «deploy» aşamasında  
yürüttülecek ve  
«script3.sh» betiği  
çalıştırılacak.

# workflow

- Belirli kurallarla (rules) «**pipeline**»ın çalıştırılmasını denetlemek amacıyla kullanılan ve 2020'de sunulan bir özelliktir.
- Örneğin, projedeki son «**commit**» mesajının ilk satırı (title) spesifik bir kelime bulunuyorsa (örneğin «**-draft**») «**pipeline**» çalıştırılmaması isteniyorsa aşağıdaki bölüm YML dosyasına eklenebilir.

```
workflow:  
  rules:  
    - if: $CI_COMMIT_TITLE =~ /-draft$/  
      when: never
```

# workflow – Örnek

```

variables:
  DEPLOY_VARIABLE: "default-deploy"

workflow:
  rules:
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
      variables:
        DEPLOY_VARIABLE: "deploy-production" # Override globally-defined
        DEPLOY_VARIABLE
    - if: $CI_COMMIT_REF_NAME =~ /feature/
      variables:
        IS_A_FEATURE: "true"                  # Define a new variable.
    - when: always                         # Run the pipeline in other cases

job1:
  variables:
    DEPLOY_VARIABLE: "job1-default-deploy"
  rules:
    - if: $CI_COMMIT_REF_NAME == $CI_DEFAULT_BRANCH
      variables:                           # Override DEPLOY_VARIABLE defined
      DEPLOY_VARIABLE: "job1-deploy-production" # at the job level.
    - when: on_success                     # Run the job in other cases
  script:
    - echo "Run script with $DEPLOY_VARIABLE as an argument"
    - echo "Run another script if $IS_A_FEATURE exists"

job2:
  script:
    - echo "Run script with $DEPLOY_VARIABLE as an argument"
    - echo "Run another script if $IS_A_FEATURE exists"

```

<https://docs.gitlab.com/ee/ci/yaml/index.html#workflow>

# Docker ve Gitlab

- Gitlab'da oluşturulan bir proje için, eğer spesifik bir teknoloji kullanılıyorsa (örneğin, Ruby dili) «**pipeline**» öğelerinin çalıştırılması için gerekli programların önceden kurulmuş olması şarttır.
- Gitlab, «**runner**»lar için Docker imajının tarif edilmesini istemektedir.
- Örneğin, proje Ruby programlama dilindeyse, Ruby programlarının işlenebilmesi için Ruby Docker imajının tarif edilmesini ister. Bu amaçla, «**default**» bölümünde «**image: ruby:3.0**» denilerek, aksi belirtilmediği sürece her işin Ruby imajında çalıştırılması gerekiği ifade edilir.

```
default:  
  image: ruby:3.0  
  
job1:  
  script: bundle exec rspec
```

<https://docs.gitlab.com/ee/ci/yaml/index.html#image>

# Gitlab ve Docker

- Eğer istenirse, «**pipeline**» içinde kendi Docker imajınızı da kullanabilirsiniz.
- «**hub.docker.com**»da oluşturacağınız hesapta bulunan ve kendi oluşturduğunuz Docker imajı YML dosyasında «**image**» bölümünde tanımlanabilir.
- Örneğin, **https://hub.docker.com/hgsenel/benimimaj** adresindeki «**hgsenel/benimimaj**» adlı imaj aşağıdaki gibi tanımlanabilir.



```
1 # This file is a template, and might need editing before it works on your project.
2 # To contribute improvements to CI/CD templates, please follow the Development guide at:
3 # https://docs.gitlab.com/ee/development/cicd/templates.html
4 # This specific template is located at:
5 # https://gitlab.com/gitlab-org/gitlab/-/blob/master/lib/gitlab/ci/templates/Bash.gitlab-ci.yml
6
7 # See https://docs.gitlab.com/ee/ci/yaml/index.html for all available options
8
9 # you can delete this line if you're not using Docker
10 image: hgsenel/benimimaj
11
12 before_script:
13   - echo "Before script section"
14   - echo "For example you might run an update here or install a build dependency"
15   - echo "Or perhaps you might print out some debugging details"
16
17 after_script:
18   - echo "After script section"
19   - echo "For example you might do some cleanup here"
```

# **artifact** (eser)



- «**artifact**» CI sürecinin sonunda oluşturulan ürün anlamına gelmektedir.
- CI pipeline içinde, hangi işin (job) hangi eserleri ürettiği ayrıca «**artifact**» anahtar kelimesiyle ifade edilmelidir. Bu şekilde ara ürünlerin (mesela C++’da \*.o gibi dosyalar) dikkate alınmaması sağlanabilir.
- «**artifact:özellik**» yapısıyla, eserler arasında farklılaşma yaratılabilir veya isimlendirilebilir veya ne kadar süre geçerli olacağı belirtilebilir.

<https://docs.gitlab.com/ee/ci/yaml/index.html#artifacts>

# artifact (eser)

```
job1:
  artifacts:
    paths:
      - binaries/
      - .config
    exclude:
      - binaries/**/*.o
  expire_in: 1 week
  name: "job1-artifacts-file"
  public: false
```

«**job1**» içinde ortaya çıkacak eserlerle ilgili bilgiler

Eserler (artifact) **binaries/** adlı dizinde bulunacak ve **.config** isminde diğer bir eser de oluşturulacak.

**binaries/** adlı dizinin altında yer alan bütün alt dizinlerdeki \*.o dosyaları «**artifact**» olarak kabul edilmeyecek.

Eserin bir hafta içinde süresi bitecek.

Esere proje dışından açık olarak erişilebilecek mi?  
Hayır!

Eserin ismi. Eğer istenirse, farklı çevre değişkenleriyle çeşitlendirilebilir.

## Örnek: Bash Şablonu kullanarak derleme

- Bu örneğimizde, «**.gitlab-ci.yml**» dosyası için Bash şablonu kullanacağız.
- Şablonda, Docker kullanmayacağız ve bu nedenle «**image: busybox:latest**» satırını sileceğiz.
- Sadece «**pipeline**»nın çalıştığını görebilmek YAML dosyasında bir değişiklik yapmadan, «**.gitlab-ci.yml**» dosyasını «**master**» dalına «**commit**» yapacağız.
- Web IDE'den çıkararak sol üst köşedeki proje ismine (gps\_api) üzerine tıkladığımızda, «**CI/CD**» menüsünden, «**Pipeline**» opsiyonuna basıldığında, «**pipeline**»ın çalıştığı (veya çalışmadığı) görülebilir

# .gitlab-ci.yml



```
# See https://docs.gitlab.com/ee/ci/yaml/index.html for all available options
before_script:
  - echo "Before script section"
  - echo "hello"

after_script:
  - echo "After script section"
  - echo "For example you might do some cleanup here"

build1:
  stage: build
  script:
    - echo "Do your build here"

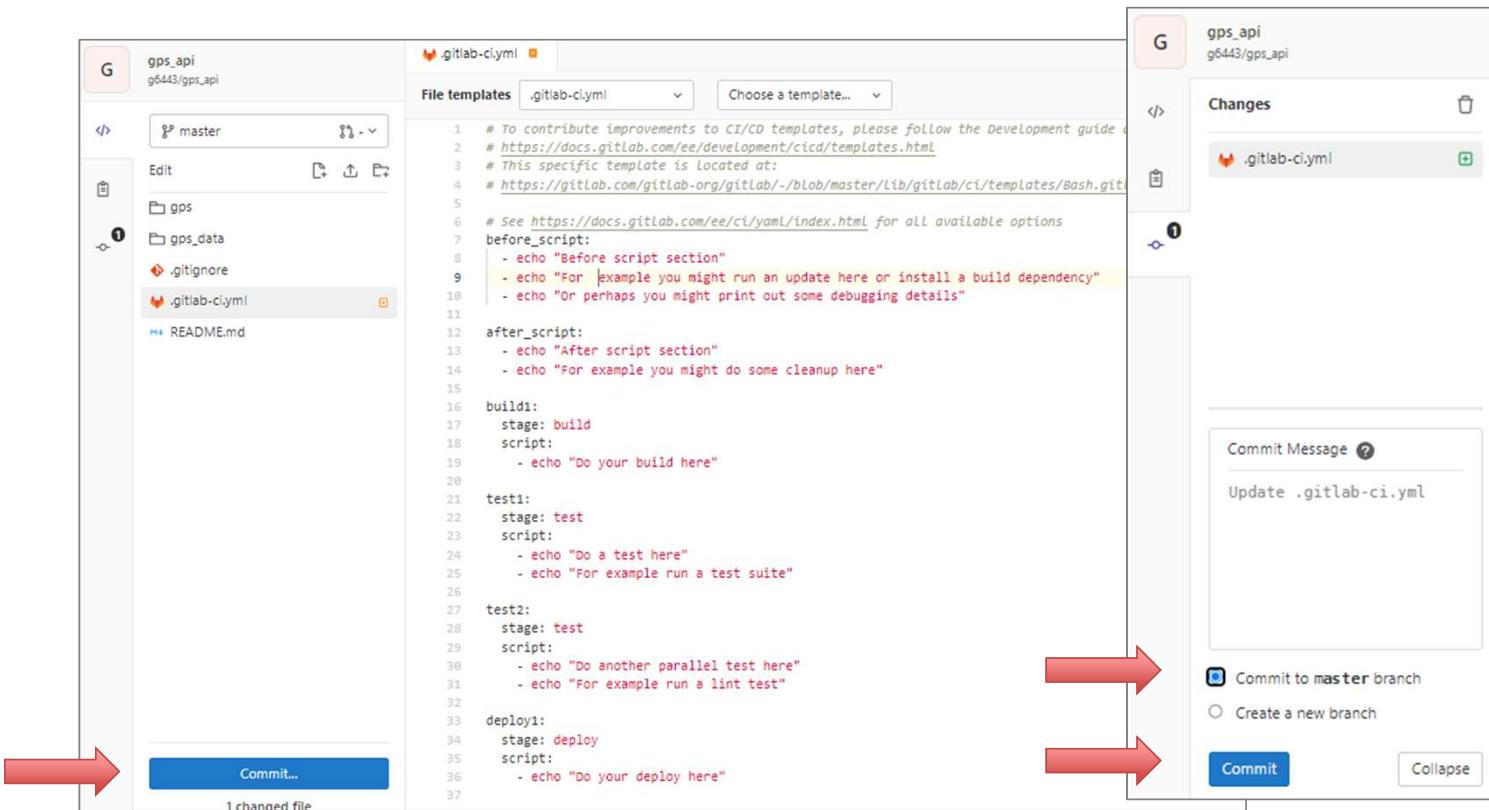
test1:
  stage: test
  script:
    - echo "Do a test here"
    - echo "For example run a test suite"

test2:
  stage: test
  script:
    - echo "Do another parallel test here"
    - echo "For example run a lint test"

deploy1:
  stage: deploy
  script:
    - echo "Do your deploy here"
```

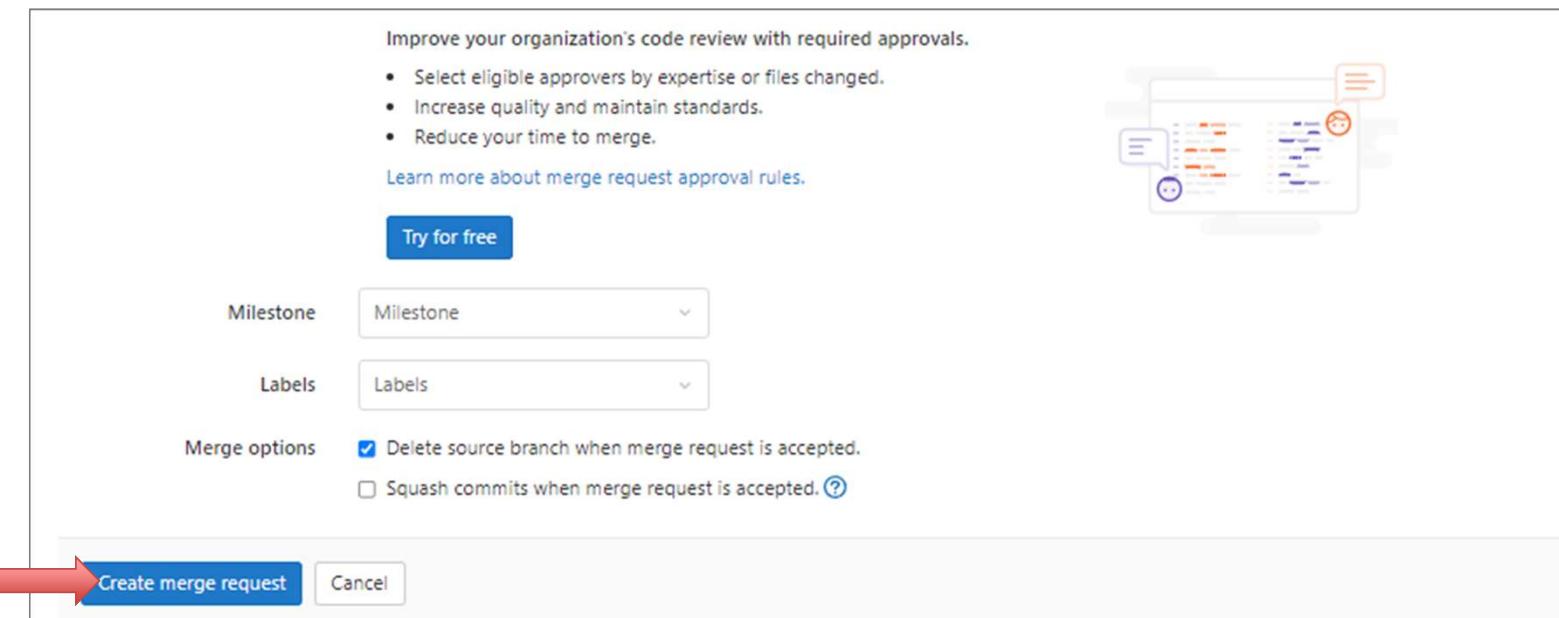
# «commit»

- «pipeline»nın çalışabilmesi için, «`.gitlab-ci.yml`» dosyasının «master» dala «commit» edilmesi gereklidir.



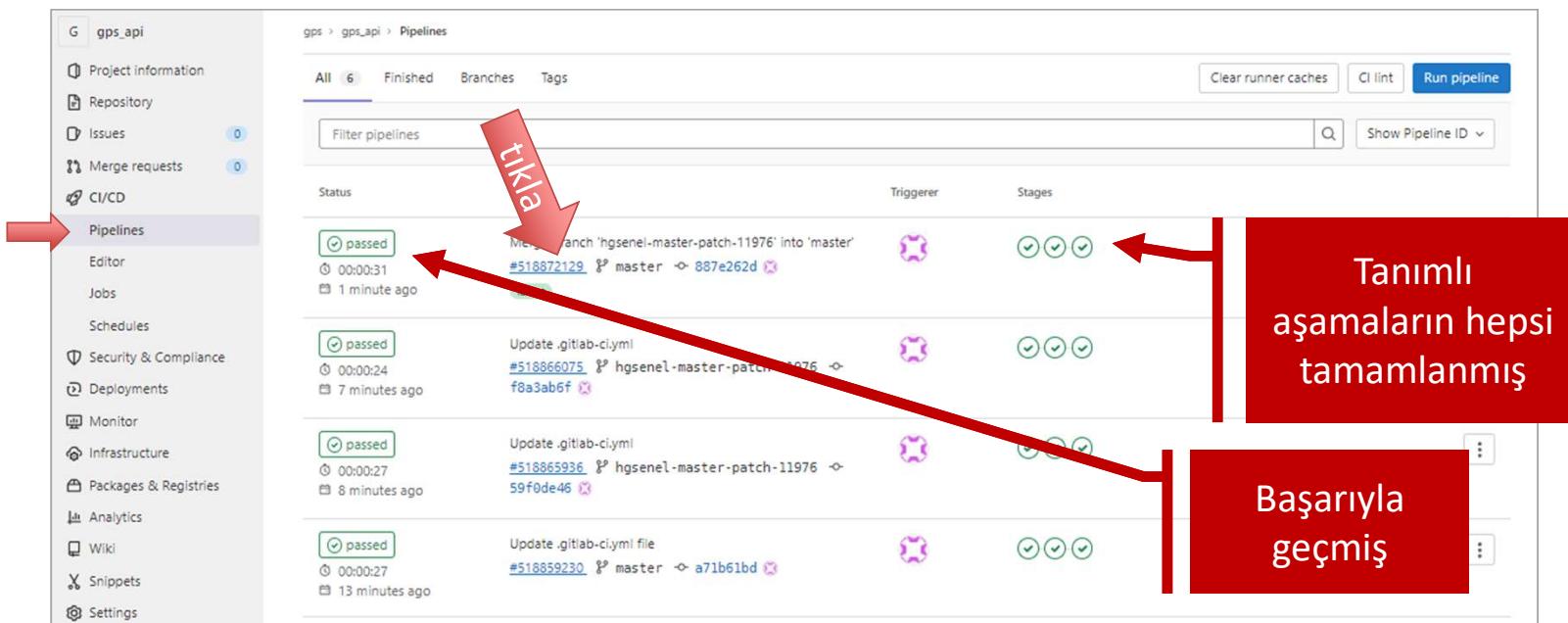
# «merge request»

- «`.gitlab-ci.yml`» dosyasında değişiklik yapıldığında «`commit`»in ardından «`merge request`» yapılmalı ve ardından «`merge`» yapılarak ana dalla birleştirilmelidir.
- Her «`merge`» işlemi «`pipeline`» prosesini tetiklemekte ve «`runner`» içinde «`.gitlab-ci.yml`» dosyasında tarif edilen süreçler işletilmektedir.



# «pipeline»

- Her «merge request» ve ardından yapılan «merge» işlemi «pipeline» işlemini tetikler.
- Proje menüsünde, sol tarafta, CI/CD bölümünün altındaki «Pipelines» opsyonuna tıklayarak, çalışan «pipeline»lar görülebilir.



**Pipelines**

**tıkla**

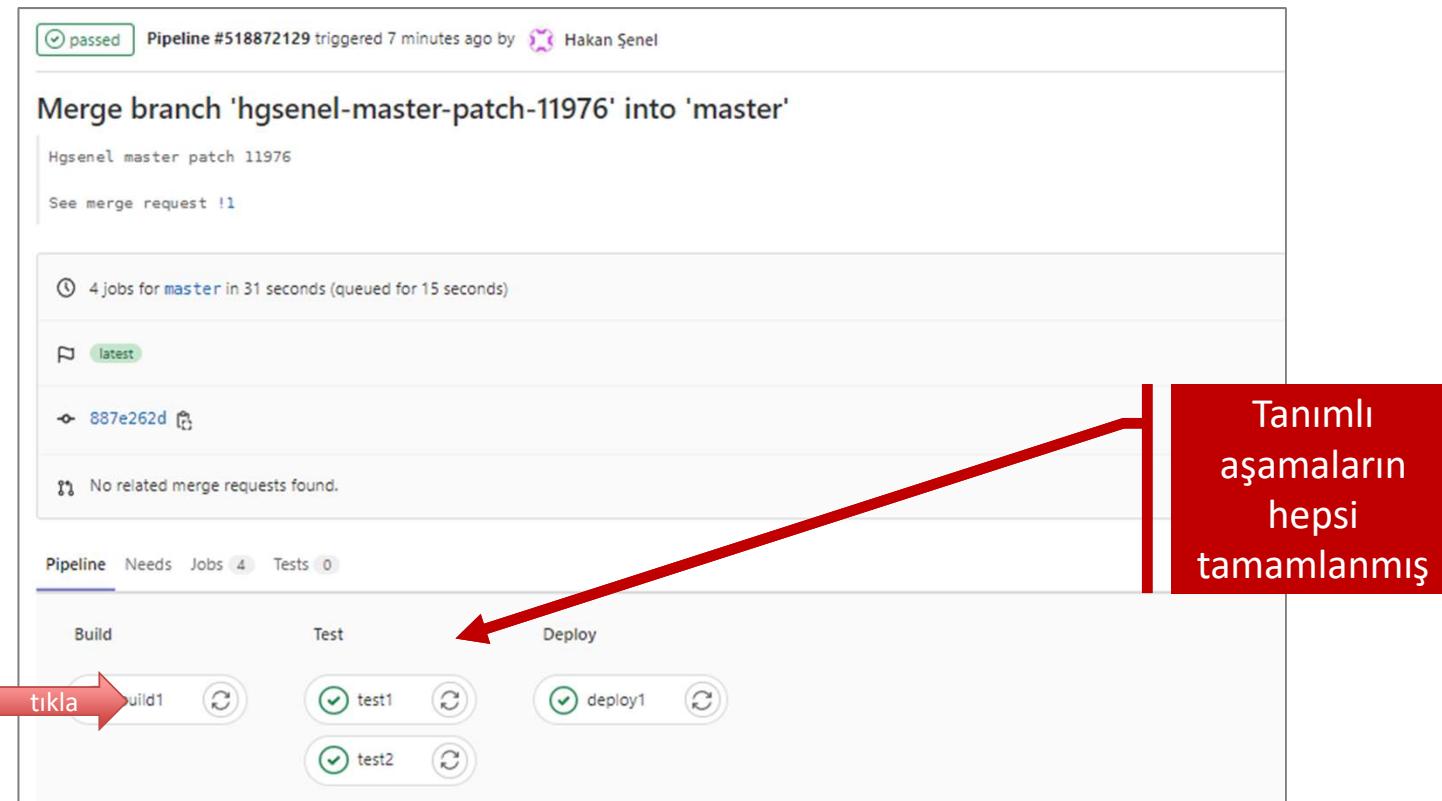
Status	Triggerer	Stages
passed 0 00:00:31 1 minute ago	#518872129 → master -> 887e262d	✓ ✓ ✓
passed 0 00:00:24 7 minutes ago	Update .gitlab-ci.yml #518866075 → hgsenel-master-patch-11976 -> f8a3abef	✓ ✓ ✓
passed 0 00:00:27 8 minutes ago	Update .gitlab-ci.yml #518865936 → hgsenel-master-patch-11976 -> 59f0de46	✓ ✓ ✓
passed 0 00:00:27 13 minutes ago	Update .gitlab-ci.yml file #518859230 → master -> a71b61bd	✓ ✓ ✓

Tanımlı aşamaların hepsi tamamlanmış

Başarıyla geçmiş

# «pipeline»

- Hangi aşamaların çalıştığı ve hangi aşamada başarısız olduğu bilgisi görülebilir.



# «pipeline» logu

- Aşama üzerine tıklamayla, log bilgisi açılmaktadır.
- Başarısız olan aşamanın nedeni konusunda loglar sayesinde bilgi alınabilir.

passed Job build1 triggered 17 minutes ago by Hakan Şenel

```
1 Running with gitlab-runner 14.9.1 (bd40e3da)
2 on ubuntu DUGP9TgW
3 Preparing the "shell" executor
4 Using Shell executor...
5 Preparing environment
6 Running on ubuntu...
7 Getting source from Git repository
8 Fetching changes with git depth set to 20...
9 Initialized empty Git repository in /home/gitlab-runner/builds/DUGP9TgW/0/g6443/gps_api/.git/
10 Created fresh repository.
11 Checking out 887e262d as master...
12 Skipping Git submodules setup
13 Executing "step_script" stage of the job script
14 $ echo "Before script section"
15 Before script section
16 $ echo "For example you might run an update here or install a build dependency"
17 For example you might run an update here or install a build dependency
18 $ echo "Or perhaps you might print out some debugging details"
19 Or perhaps you might print out some debugging details
20 $ echo "Do your build here"
21 Do your build here
22 Running after_script
23 $ echo "After script section"
24 After script section
25 $ echo "For example you might do some cleanup here"
26 For example you might do some cleanup here
27 Cleaning up project directory and file based variables
28 Job succeeded
```

# ÖDEV: **gps\_api** projesinin entegrasyonu

- Bu ödevde, «**.gitlab-ci.yml**» dosyası üzerinde değişiklik yaparak, projenin **gcc** ile derlenmesi gerçekleştirilecektir.
- Yaptığınız değişiklikleri, «**commit**», «**merge request**» ve «**merge**» işlemlerini ekran görüntüsü alarak dokümente ediniz ve ekran görüntüleriyle birlikte açıklamalarınızla birlikte bir pdf oluşturarak, e-posta ile gönderiniz.



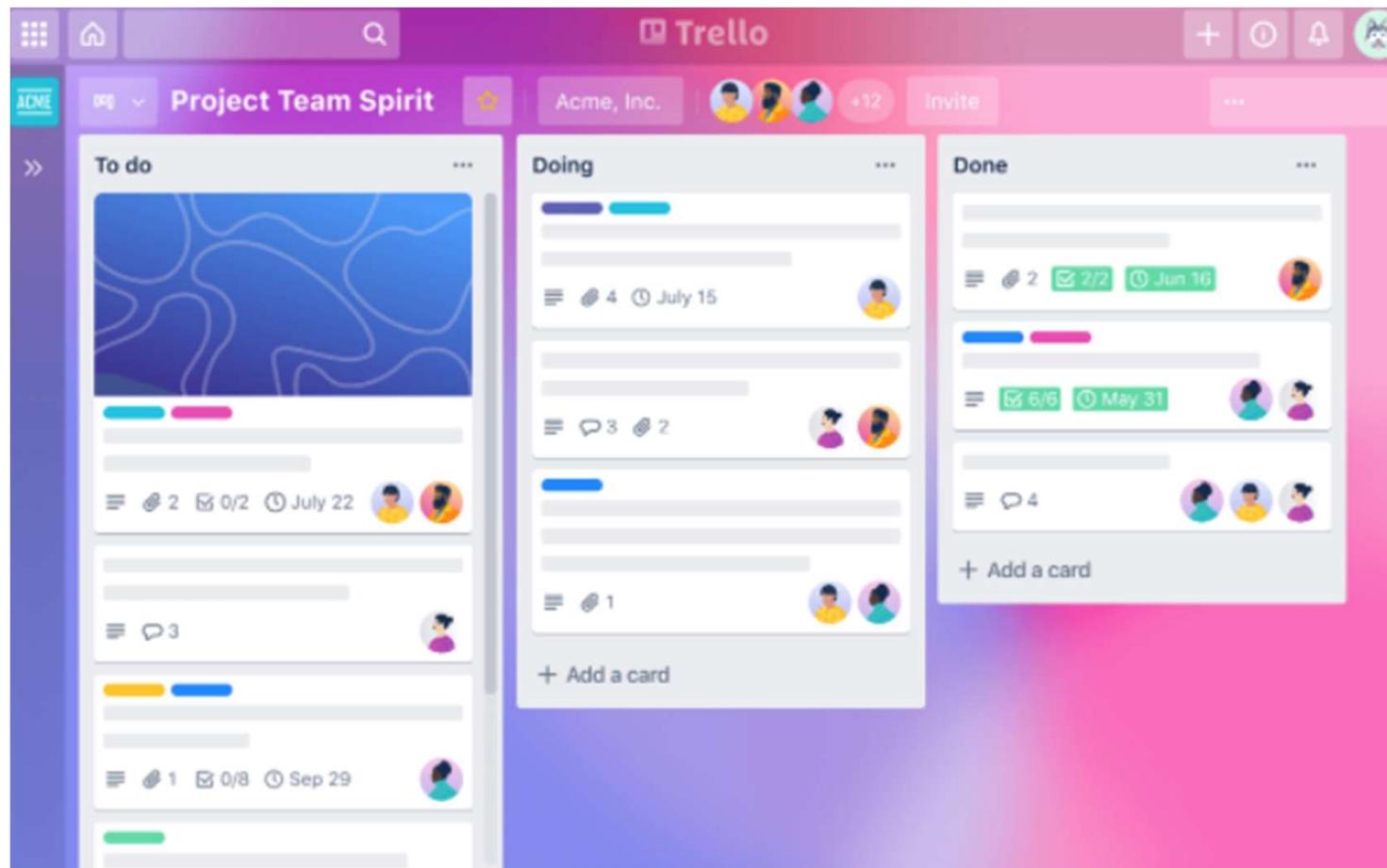
# BITBUCKET

# Bitbucket nedir?

- Yazılım geliştirme ekipleri için kod depolama, yönetme ve takip etme gibi özellikleri olan bulut temelli bir çözümdür.
- Git sistemi dışında, bazı önemli özellikleri bulunmaktadır:
  - «**pull request**»lerinde kodun gözden geçirilmesi için bir çözüm sunmaktadır.
  - JIRA entegrasyonu bulunmaktadır. Kod içindeki hataların tespiti ve takibi için Bitbucket'a entegre edilebilmektedir.
  - Her kod parçası için yorum eklenebilmektedir.
  - Github üzerinde kimlik doğrulama yapılmaktadır.
  - 5 kişilik takımlara kadar ücretsizdir.
  - Farklı tür repoları (SVN, CVS, vb) sisteme yükleyebilmektedir.
  - Trello ile entegre edilebilmektedir.

# Trello

## «To-do list software»



# Jira

- Jira, yazılım endüstrisinde «bug takibi», «sorun takibi» ve «proje yönetimi» amaçlarıyla kullanılan, çevik yönetim felsefesine uygun çalışan bir yazılımdır.
- Genellikle yazılım geliştirme süreçlerinde hataların ve sorunların giderilmesiyle ilgili çalışmalarda yaygın olarak kullanılıyor. Proje yönetimi için de çeşitli özellikleri var.
- Jira, genellikle aşağıdaki işlerde de kullanılır:
  - Gereksinim ve test durumlarının yönetimi
  - Proje yönetimi
  - Çevik yazılım geliştirme metodolojilerinin uygulanması
  - Yazılım geliştirme
  - Ürün yönetimi
  - Görev yönetimi
  - Hata takibi

<https://www.atlassian.com/software/jira/guides/use-cases/what-is-jira-used-for>

## Jira'nın kullanımı

- Jira üzerinde, şablonlar arasından biri seçilerek bir proje yaratılır ve takım üyeleri davet edilir.
- Açık bir «Issue» oluşturulur ve takım elemanlarıyla paylaşılır.
- Takım üyeleri «issue» üzerinde çalışıkları zaman durum olarak «IN PROGRESS» gösterilir.
- «Issue» üzerinde yapılan çalışmalar sonucunda, «Resolved», «Reopened», «Closed» gibi durum kodları atanarak, «bug» veya «defect»lerin takibi yapılır.
- Çözüm bulunan hatalar raporlanır.

# Bitbucket ve Jira Entegrasyonu

- Bitbucket, kod deposunun yönetim yazılımı olarak, kod üzerindeki değişikliklerin takibini kolaylaştıran bir yazılım.
- Jira ise, «bug», «defect» takibini yapan bir yazılım.
- Her ikisi de aslında benzer amaçlarla ancak farklı alanlarda çalışıyor. Jira daha sözel bilgilere dayanırken, Bitbucket daha çok kod üzerindeki çalışmaların yapılmasını sağlıyor.
- Jira ve Bitbucket bir arada çalıştırıldığında, Jira'da takip edilen bir «bug» üzerinde ne tür kod değişikliklerinin olduğunu, hangi «commit»lerin gerçekleştirildiğini gösterebilir.
- Jira'ya sadece Bitbucket Pipeline değil farklı CI/CD yazılımları da (CircleCI, Travis, Jenkins, Octopus, vb) entegre edilebilmektedir.

<https://www.atlassian.com/software/jira/guides/developers/ci-cd>

# Bitbucket girişи

 Bitbucket

## Stop coding interruptions - automate instead

Teams that integrate with Jira release 14% more often

- ✓ Automatic development updates in Jira
- ✓ Interact with Jira issues in Bitbucket and your IDE
- ✓ Free for up to 5 Bitbucket and 10 Jira users

YOU SELECTED

 Bitbucket  
More than code management

ADD JIRA FOR FREE

 Jira Software  
The #1 software used by agile teams [Add](#)

Manage projects the same way you manage your code using automated updates from Git-based actions.

[Next](#)

NO CREDIT CARD REQUIRED



## Automate your way to faster releases

+  Bitbucket  
 Jira Software

- ✓ Automatic development updates in Jira
- ✓ Interact with Jira issues in Bitbucket and your IDE
- ✓ Free for up to 5 Bitbucket and 10 Jira users

## Create your account

Work email

hgsenel@eskisehir.edu.tr

Full name

Hakan Senel

By clicking below, you agree to the Atlassian Cloud Terms of Service and Privacy Policy.

Agree

NO CREDIT CARD REQUIRED

This site is protected by reCAPTCHA and the Google [Privacy Policy](#) and [Terms of Service](#) apply.



**Atlassian hesabınızı oluşturun**  
Atlassian uygulamasına devam edin

hgsenel@eskisehir.edu.tr

.....

Güçlü  
Kaydolarak Atlassian Cloud Hizmeti Kullanım Şartlarını kabul ediyorum ve [Gizlilik Politikasını](#) onaylıyorum.

**Kaydol**

VEYA

[Google ile devam et](#)

[Microsoft ile devam et](#)

[Apple ile devam et](#)

Bu sayfa reCAPTCHA ile Google [Gizlilik Politikası](#) tarafından korunmaktadır  
ve [Hizmet Şartları](#) geçerlidir



## Automate your way to faster releases

+  Bitbucket  
 Jira Software

- ✓ Automatic development updates in Jira
- ✓ Interact with Jira issues in Bitbucket and your IDE
- ✓ Free for up to 5 Bitbucket and 10 Jira users

Welcome back, Hakan

Work email

hgsenel@eskisehir.edu.tr

[Sign in with a different Atlassian account](#)

Your site

hgsenel  .atlassian.net 

By clicking below, you agree to the Atlassian Cloud [Terms of Service](#) and [Privacy Policy](#).

**Agree**

NO CREDIT CARD REQUIRED



One moment, your site is starting up



Thanks for signing up! Our robots are working on your Atlassian Cloud site.  
This won't take more than a minute or two. You'll be taken there once it's ready.

# bitbucket.com'a giriş

The screenshot shows the Bitbucket homepage. At the top, there are navigation tabs: 'Your work' (selected), 'Repositories', 'Projects', 'More', and a 'Create' button. A blue arrow points from the text 'Projeler, repoların organizasyonu için kullanılır' to the 'Projects' tab. Below the tabs, there's a welcome message: 'Welcome to Bitbucket! Let's get started'. It includes a 'Create repository' button, an 'Import repository' button, and a 'Create workspace' button. To the right of this message is a blue box containing the text 'Projeler, repoların organizasyonu için kullanılır'. In the 'Recent repositories' section, there's a 'Create repository' button with a blue arrow pointing to it. To the right of this section is a blue box containing the text 'Bir repo oluşturalım.'.

Recent repositories +

Create repository

Pull requests

You have no open pull requests.

Jira issues

Welcome to Bitbucket! Let's get started

Get started building your personal projects, testing out ideas, and more in your hgsenel workspace.

Create repository Import repository

Plan to collaborate with your team? Create a new workspace to get started.

Create workspace

Projeler, repoların organizasyonu için kullanılır

Bir repo oluşturalım.

# bitbucket.com'a giriş

**Create a new repository**

Import repository

Workspace: Hakan Şenel

Project name\*: gps

Repository name\*: gps\_api

Access level:  Private repository

Uncheck to make this repository public. Public repositories typically contain open-source code and can be viewed by anyone.

Include a README?: No

Default branch name: main

Include .gitignore?: No

Advanced settings

Description:

Forking:

Language: Select language... (C, C#, C++, Go, HTML/CSS)

Create repository Cancel

- Proje ismi olarak «gps» ismini verelim.
- Repo ismi olarak gps\_api kullanalım
- README.md olmasın
- Ana dal olarak «main» yazalım
- .gitignore dosyasını lokal repodan göndereceğiz.
- «Advanced Settings» altındaki Language bölümünde C dilini belirtelim

# SSH anahtarları girişi

The screenshot shows the Bitbucket user interface. On the left, there's a sidebar with a search bar, a help icon, and a user profile icon (HS). The sidebar menu includes: ACCOUNT (Switch account, Manage account), RECENT WORKSPACES (Hakan Senel, All workspaces), and SETTINGS (Personal settings, Labs, Log out). A red arrow labeled 'tıkla' points from the 'Personal settings' link in the sidebar to the 'Personal settings' page on the right. On the right, the 'Personal settings' page has a sidebar with GENERAL, ACCESS MANAGEMENT, SECURITY, and FEATURES sections. The SECURITY section is expanded, showing SSH keys, Two-step verification, Sessions, and Audit log. Under the SSH keys section, there's a sub-section titled 'SSH keys' with the text: 'Use SSH to avoid password prompts when you push code to Bitbucket. Learn how to generate an SSH key.' Below this is a blue button labeled 'Add key'. Another red arrow labeled 'tıkla' points to this 'Add key' button. To the right of the 'Add key' button is a table header for 'Key' with columns for 'Added' and 'Last used'. The table body below the header says 'There are no keys configured'.

SSH Anahtarı girişi için  
Tıkla

tıkla

tıkla

**ÖNEMLİ NOT:** Repo'ya da SSH anahtarları girilebilmektedir. Kullanıcı hesabındaki «Personal settings» bölümünden SSH anahtarları girilmelidir. Aksi taktirde «`git push`» yapıldığında «Unauthorized» hatası alınabilir.

# SSH anahtar girişi

Add SSH key

Label

Key\*

**Don't have a key?**  
Learn how to [generate an SSH key](#).

**Already have a key?**  
Copy and paste your key here .

**Problems adding a key?**  
Read our [troubleshooting page](#) for common issues.

**Add key**

Açık SSH Anahtarını  
«Ctrl-V»

Personal settings

- GENERAL
  - [Account settings](#)
  - [Email aliases](#)
  - [Notifications](#)
- ACCESS MANAGEMENT
  - [App authorizations](#)
  - [App passwords](#)
- SECURITY
  - SSH keys**
  - [Two-step verification](#)
  - [Sessions](#)
  - [Audit log](#)
- FEATURES
  - [Labs](#)

SSH keys

Use SSH to avoid password prompts when you push code to Bitbucket. Learn how to [generate an SSH key](#).

Key	Added	Last used
Lokal VMware	just now	Never

tıkla

# Lokal reponun aktarılması

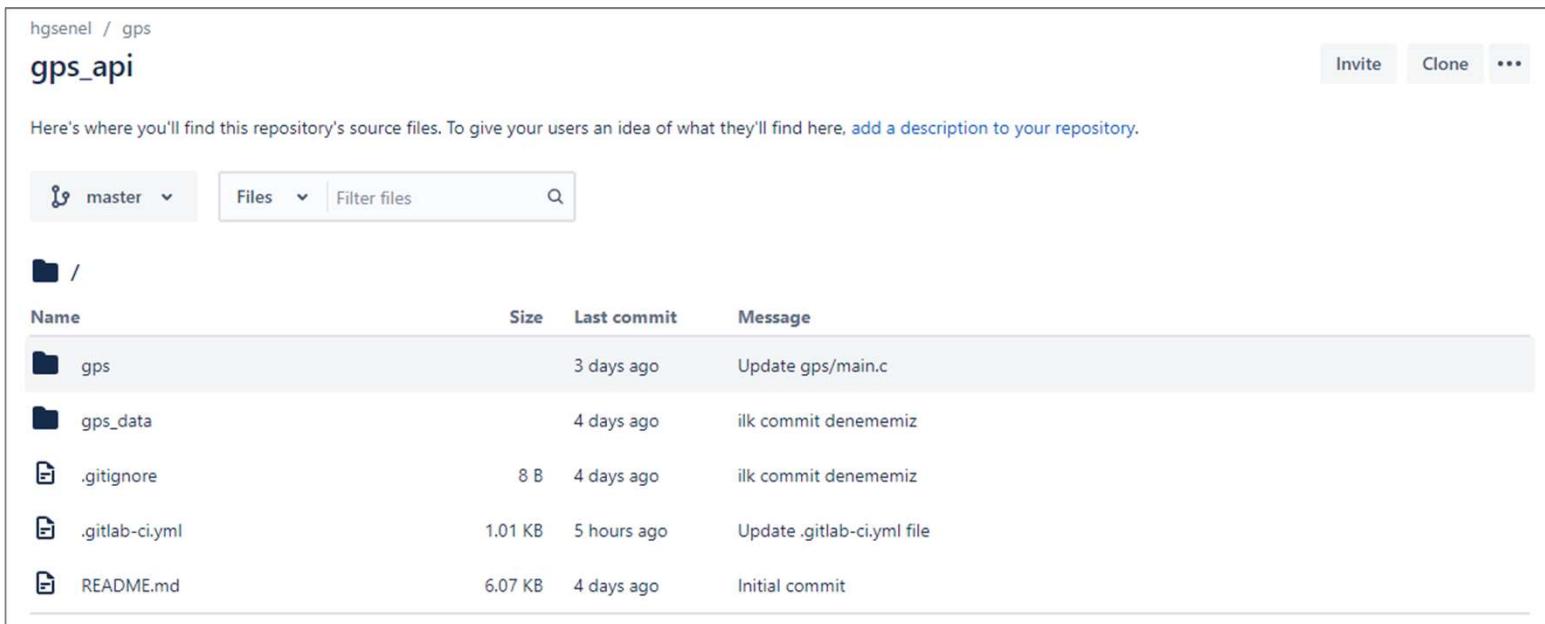
- Lokal git reposu için daha önce farklı bir sisteme «**git remote add origin**» komutu verilmiş ve uzaktaki repoya bağlantı kurulmuş ise, «**git remote remove origin**» komutu kullanılarak, eski bağlantı sonlandırılmalıdır.
- Uzaktaki Bitbucket reposuna aktarım için aşağıdaki komutlar verilebilir:

```
$ git remote add origin  
git@bitbucket.org:hgsenel1/gps_api.git  
$ git push --set-upstream origin master
```

```
adminpc@ubuntu:~/gps_api$ git remote remove origin  
adminpc@ubuntu:~/gps_api$ git remote add origin git@bitbucket.org:hgsenel1/gps_api.git  
adminpc@ubuntu:~/gps_api$ git push --set-upstream origin master  
Enumerating objects: 43, done.  
Counting objects: 100% (43/43), done.  
Delta compression using up to 2 threads  
Compressing objects: 100% (40/40), done.  
Writing objects: 100% (43/43), 14.51 KiB | 594.00 KiB/s, done.  
Total 43 (delta 16), reused 0 (delta 0)  
To bitbucket.org:hgsenel1/gps_api.git  
 * [new branch]      master -> master  
Branch 'master' set up to track remote branch 'master' from 'origin'.  
adminpc@ubuntu:~/gps_api$
```

# Bitbucket'taki Repo

- «git push» komutuyla lokal repo `bitbucket.org`'taki repoya yüklediği görülebilir.

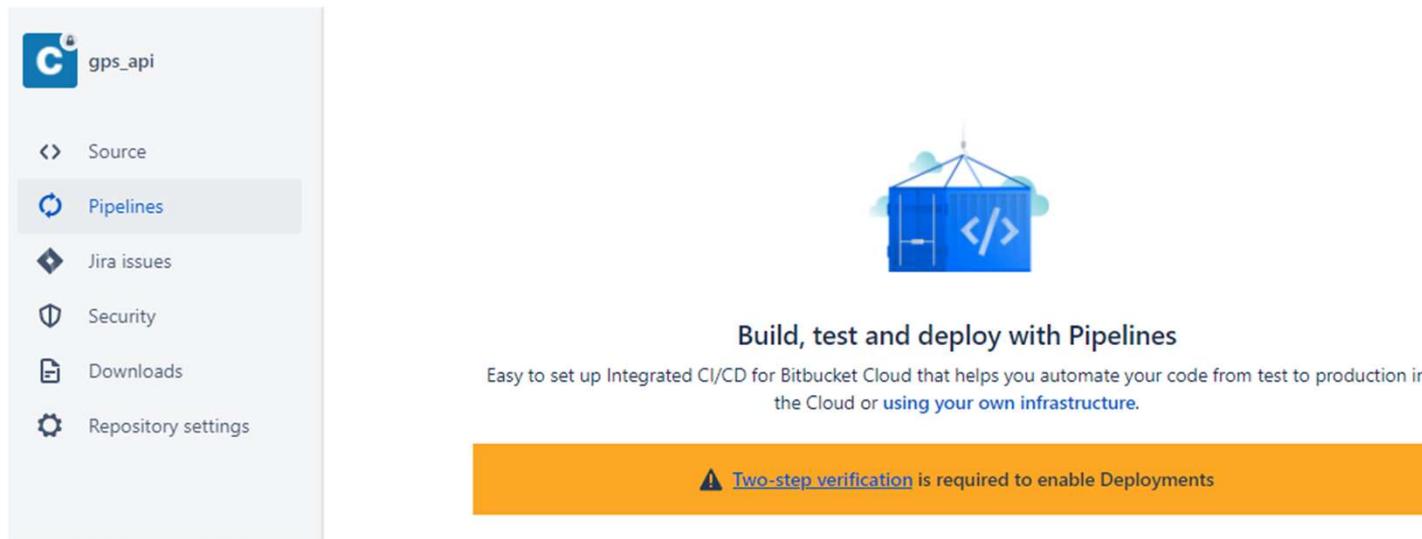


The screenshot shows a Bitbucket repository page for 'gps\_api' owned by 'hgsenel'. The repository has one branch, 'master', which is currently selected. The interface includes a search bar and filters for 'Files' and 'Filter files'. The main area displays a table of files and their details:

Name	Size	Last commit	Message
gps		3 days ago	Update gps/main.c
gps_data		4 days ago	ilk commit denememiz
.gitignore	8 B	4 days ago	ilk commit denememiz
.gitlab-ci.yml	1.01 KB	5 hours ago	Update .gitlab-ci.yml file
README.md	6.07 KB	4 days ago	Initial commit

# Bitbucket'ta CI/CD

- Bitbucket Cloud'un pipeline özelliğini kullanmak için, iki yönlü kimlik doğrulama sisteminin devreye alınması gereklidir.
- Android telefonlarda, Authy uygulaması yüklenerek, bu özellik kullanılabilmektedir.
- Bitbucket HTTPS üzerinden kod aktarımını 2022 Mart ayı itibariyle kısıtlamıştır.



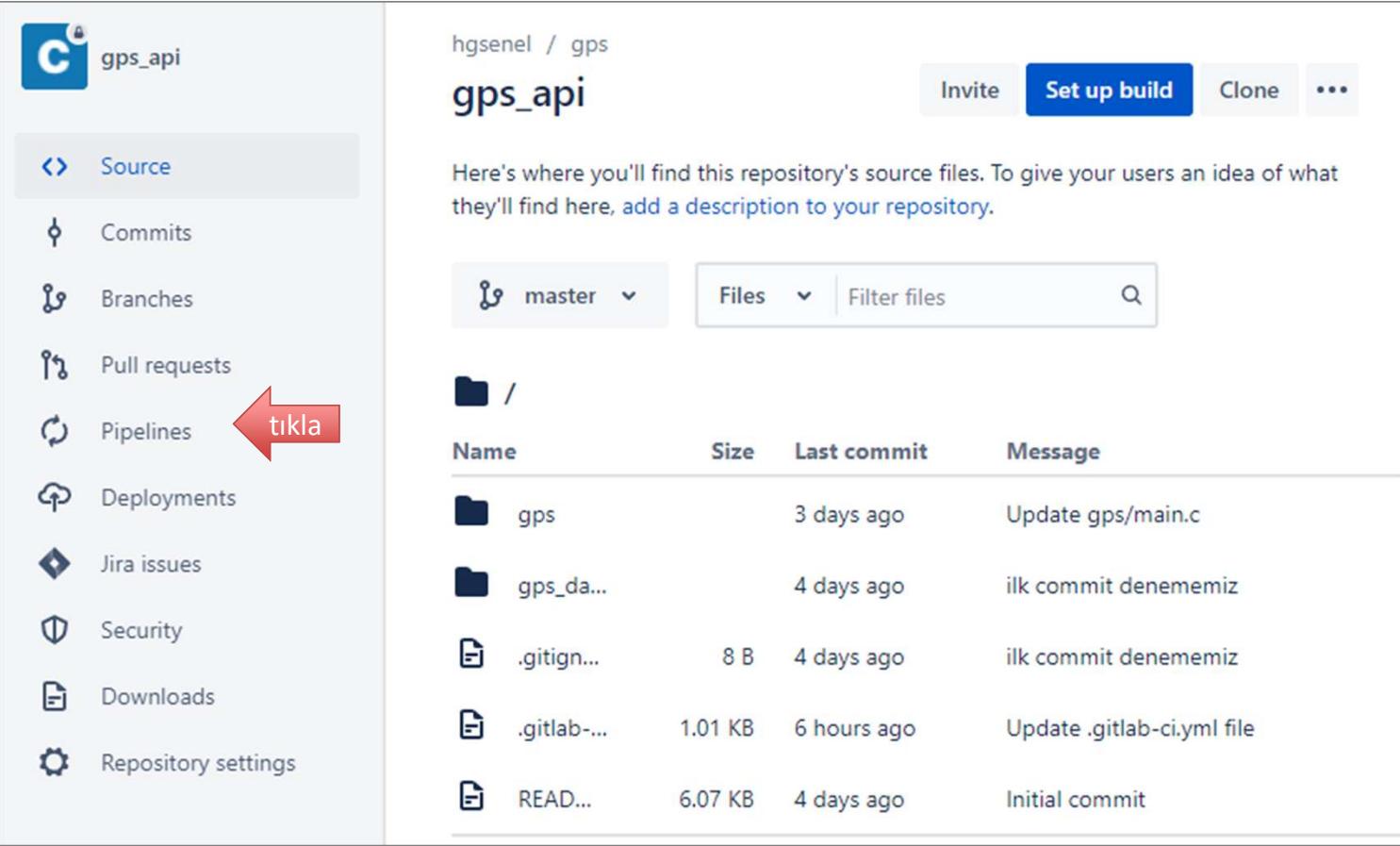
# Bitbucket CI/CD olanakları

- Bitbucket, farklı araçların (CircleCI, Travis, Jenkins, vb) entegre edilmesiyle CI/CD süreçlerinin uygulanmasını sağlamaktadır.
- Bitbucket, «Bitbucket Cloud» isimli bir hizmetle CI/CD çalıştırıcı hizmeti sağlamaktadır.
- Ücretsiz sürümde, 50 dakikalık bir çalışma süresine izin vermekte ve daha kapsamlı geliştirme işleri için ücretli planlara kayıt yapılması gerekmektedir.
- Bu bölümde, Bitbucket Cloud kullanımı hakkında bilgi verilecektir. Dışarıda Linux/Microsoft/MacOS üzerinde «Runner» kurulumu karmaşık bir yapılandırma gerektirdiğinden bu bölümde gösterilmeyecektir. Konuya ilgili bilgi için şu web sayfasına bakılabilir:

<https://support.atlassian.com/bitbucket-cloud/docs/runners/>

# «Bitbucket Pipelines»

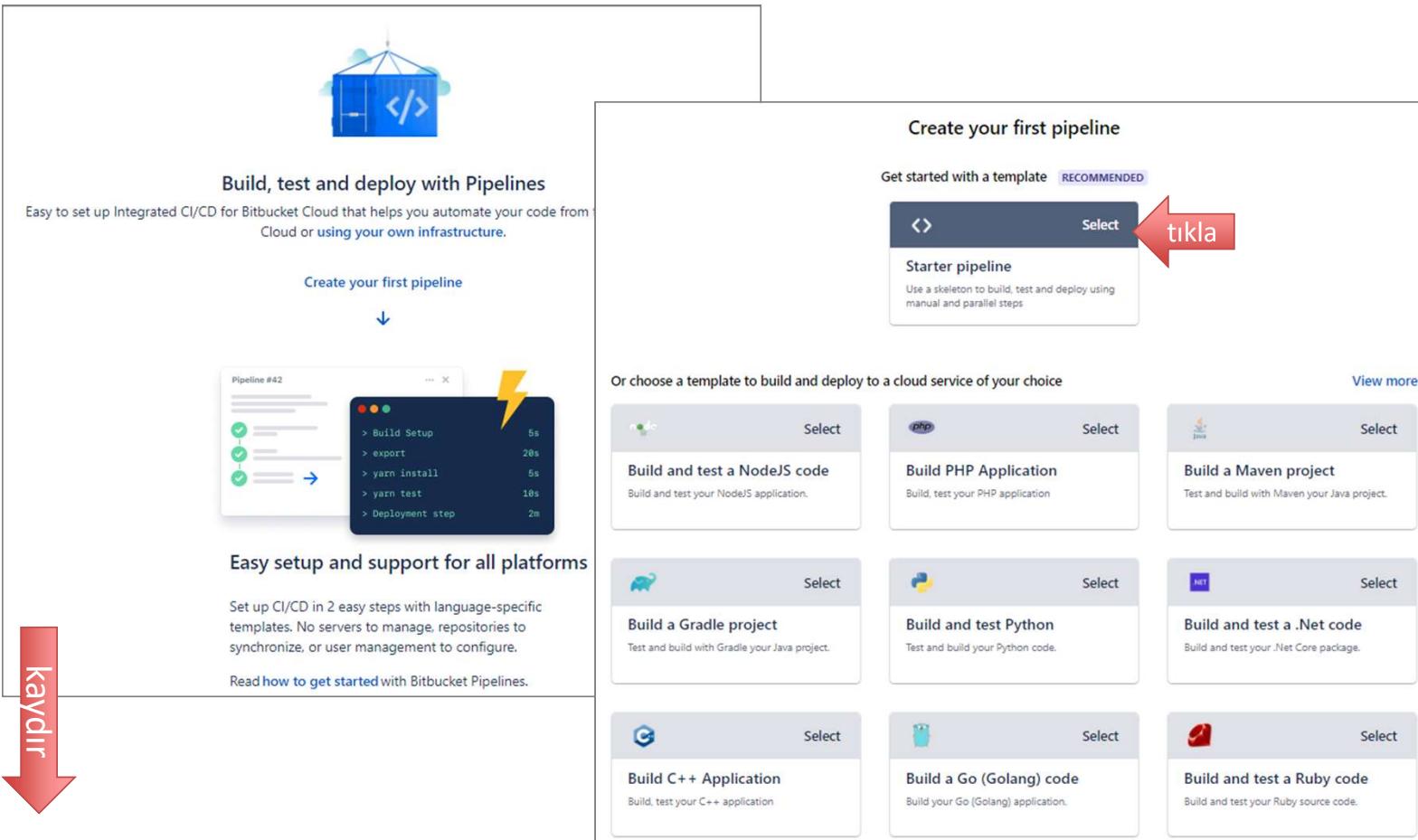
- Bitbucket, ayda 50 dakikalık pipeline kullanma imkanı tanımaktadır.



A screenshot of a Bitbucket repository page for 'gps\_api'. On the left, there's a sidebar with navigation links: Source (highlighted), Commits, Branches, Pull requests, Pipelines (with a red arrow pointing to it), Deployments, Jira issues, Security, Downloads, and Repository settings. The main area shows the repository details: 'hgsenel / gps' and 'gps\_api'. It includes buttons for Invite, Set up build (which is blue), Clone, and more. A description placeholder says: 'Here's where you'll find this repository's source files. To give your users an idea of what they'll find here, add a description to your repository.' Below this is a file browser with 'master' selected. The table lists files and their details:

Name	Size	Last commit	Message
gps		3 days ago	Update gps/main.c
gps_da...		4 days ago	ilk commit denememiz
.gitign...	8 B	4 days ago	ilk commit denememiz
.gitlab-ci.yml	1.01 KB	6 hours ago	Update .gitlab-ci.yml file
README.md	6.07 KB	4 days ago	Initial commit

# Bitbucket Pipeline



The screenshot shows the Bitbucket Pipeline setup interface. It starts with a brief introduction to Pipelines, followed by a 'Create your first pipeline' button. Below this, there's a visual representation of a pipeline with several steps and a lightning bolt icon. A red arrow labeled 'kaydır' points down from this section. To the right, the main 'Create your first pipeline' screen is displayed, featuring a 'Starter pipeline' template selected (indicated by a red arrow labeled 'tıkla'). Below this, a grid of nine language-specific pipeline templates is shown:

- Build and test a NodeJS code
- Build PHP Application
- Build a Maven project
- Build a Gradle project
- Build and test Python
- Build and test a .Net code
- Build C++ Application
- Build a Go (Golang) code
- Build and test a Ruby code

Each template includes a 'Select' button.

# bitbucket-pipelines.yml

bitbucket-pipelines.yml  
Dosyası repoya  
eklenecektir.

Gitlab'daki YML dosyasına  
benzer bir yapıdadır.

Bu aşamada, dosyada  
Hiçbir değişiklik yapmadan  
«Commit» butonuna  
Basın ve «pipeline» otomatik  
olarak çalışmaya  
başlayacaktır.

Create your first pipeline

[Send Feedback](#)

`gps_api/bitbucket-pipelines.yml` [EDITABLE](#)

```

1 # This is an example Starter pipeline configuration
2 # Use a skeleton to build, test and deploy using manual and parallel steps
3 #
4 # You can specify a custom docker image from Docker Hub as your build
5 # environment.
6
7 image: atlassian/default-image:3
8
9 pipelines:
10   default:
11     - parallel:
12       - step:
13         name: 'Build and Test'
14         script:
15           - echo "Your build and test goes here..."
16       - step:
17         name: 'Lint'
18         script:
19           - echo "Your linting goes here..."
20       - step:
21         name: 'Security scan'
22         script:
23           - echo "Your security scan goes here..."
24
25   # The following deployment steps will be executed for each pipeline run.
26   # To configure your steps and conditionally deploy see
27   # https://support.atlassian.com/bitbucket-cloud/docs/configure-bitbucket-
28   # pipelinesyml/
29     - step:
30       name: 'Deployment to Staging'
31       deployment: staging
32       script:
33         - echo "Your deployment to staging script goes here..."
34     - step:
35       name: 'Deployment to Production'
36       deployment: production
37       trigger: 'manual'
38       script:
39         - echo "Your deployment to production script goes here..."
```

Configure Documentation

[Change the template](#)

The templates allow you to configure your pipeline using your preferred language. The template will override any configuration content.

Starter pipeline

[Add more steps](#)

[Add Pipes \(Integrations\)](#)

[Add variables](#)

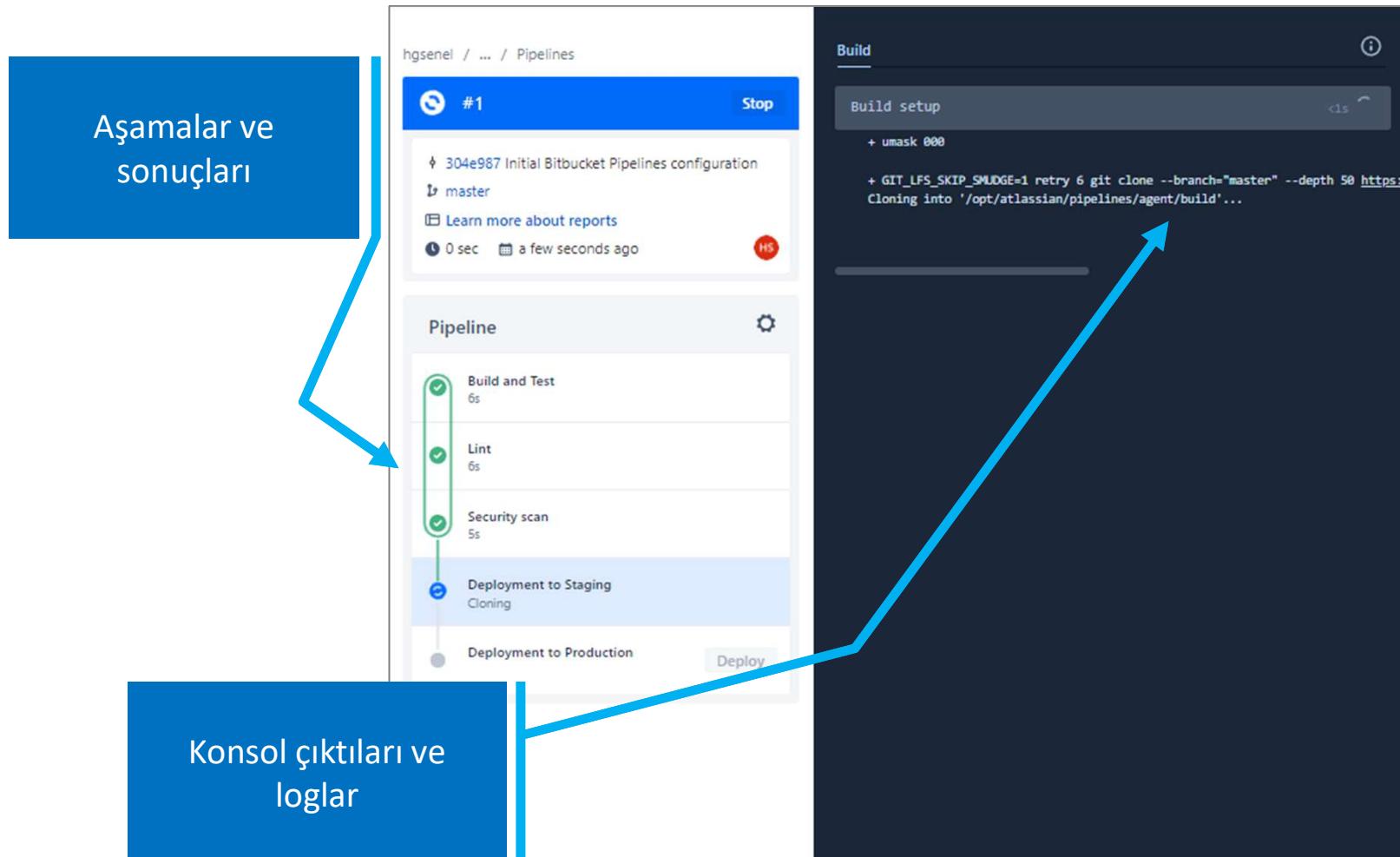
Congrats! You've configured your first pipeline

Now you have to commit the .yml file to get it working. It takes a couple of minutes.

[Commit file](#)

<https://support.atlassian.com/bitbucket-cloud/docs/get-started-with-bitbucket-pipelines/>  
<https://support.atlassian.com/bitbucket-cloud/docs/configure-bitbucket-pipelinesyml/>

# Pipeline'nin çalışması



# bitbucket-pipelines.yml

- Repo içine kaydedilen «**bitbucket-pipelines.yml**» dosyası, CI/CD süreçlerini tanımlamaktadır.
- Bu dosya üzerinde yapılan değişikliklerle, «**CI/CD pipeline**» yapısı değiştirilebilir.



Name	Size	Last commit	Message
gps		3 days ago	Update gps/main.c
gps_data		4 days ago	ilk commit denememiz
.gitignore	8 B	4 days ago	ilk commit denememiz
.gitlab-ci.yml	1.01 KB	6 hours ago	Update .gitlab-ci.yml file
README.md	6.07 KB	4 days ago	Initial commit
bitbucket-pipelin...	1.17 KB	7 minutes ago	Initial Bitbucket Pipelines configurat

# gps\_api derleme

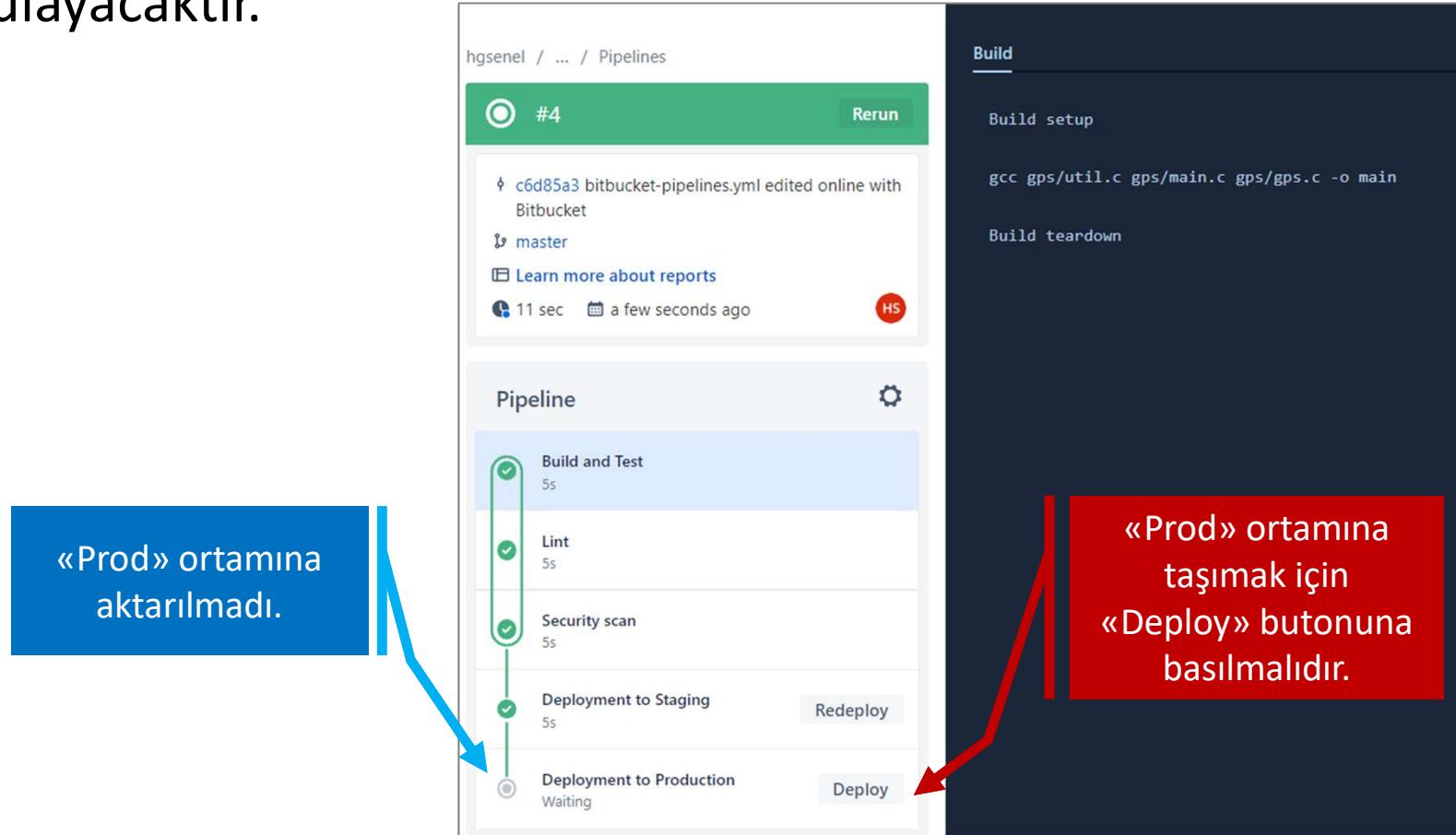
- «Edit» butonuna basacağımız ve «**bitbucket-pipelines.yml**» dosyasında değişiklik yaparak, derleme yapılan kısımda «**gcc gps/util.c gps/main.c gps/gps.c -o main**» yazacağımız.
- «Commit» dedikten sonra «Approve» diyerek onaylayacağız.

```
hgsenel / gps / gps_api
bitbucket-pipelines.yml
Pull requests | Invite | Check out | ...
Source master 304e987 Full commit

gps_api / bitbucket-pipelines.yml Edit ...
1 # This is an example Starter pipeline configuration
2 # Use a skeleton to build, test and deploy using manual and parallel step
3 #
4 # You can specify a custom docker image from Docker Hub as your build env
5
6 image: atlassian/default-image:3
7
8 pipelines:
9   default:
10     - parallel:
11       - step:
12         name: 'Build and Test'
13         script:
14           - echo "Your build and test goes here..."
15       - step:
16         name: 'Lint'
17         script:
18           - echo "Your linting goes here..."
19       - step:
20         name: 'Security scan'
21         script:
22           - echo "Your security scan goes here..."
23
24 # The following deployment steps will be executed for each pipeline run
25     - step:
26       name: 'Deployment to Staging'
27       deployment: staging
28       script:
29         - echo "Your deployment to staging script goes here..."
30     - step:
31       name: 'Deployment to Production'
32       deployment: production
33       trigger: 'manual'
34       script:
35         - echo "Your deployment to production script goes here..."
36
```

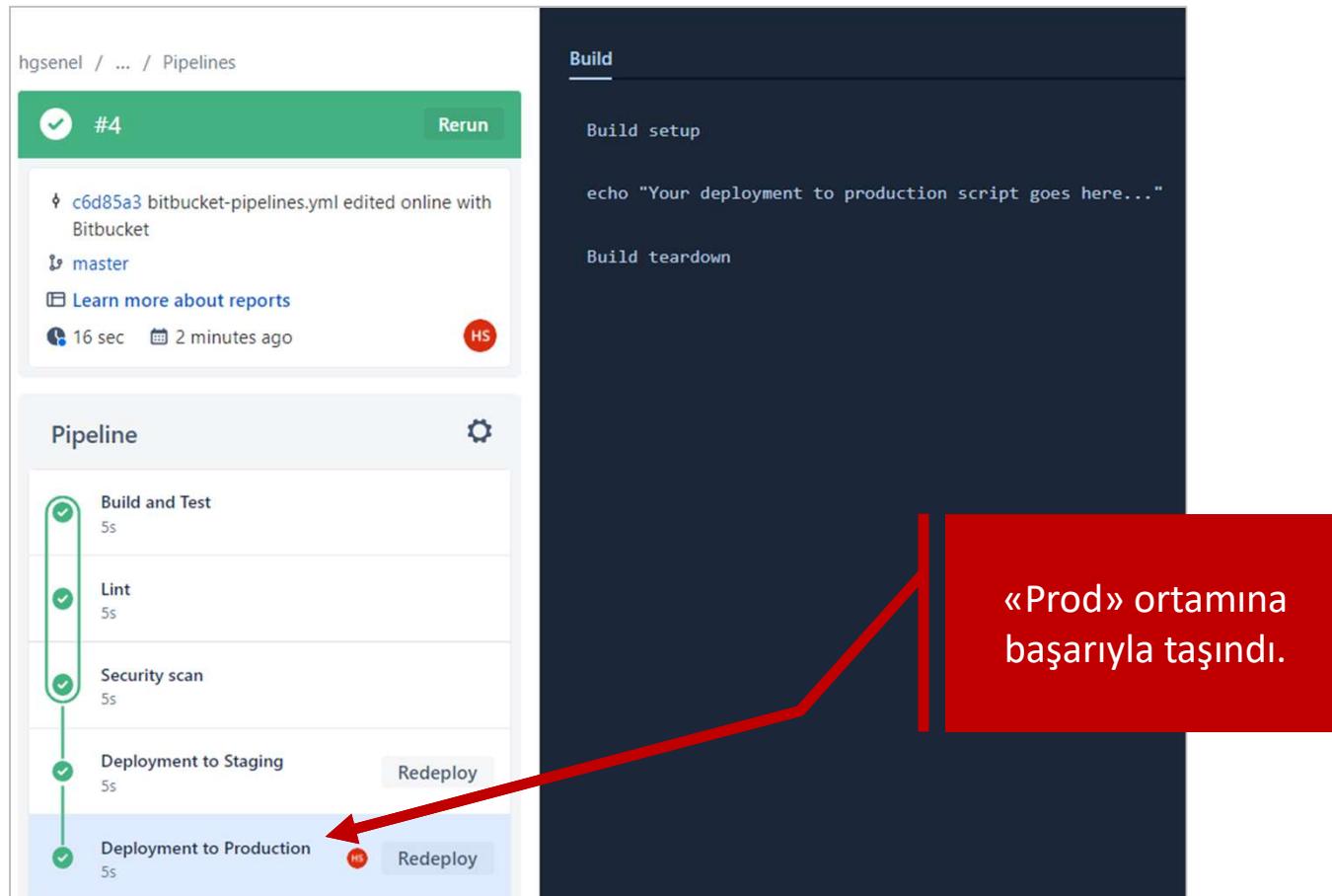
# Pipeline

- Pipeline otomatik olarak çalışacak ve yaptığımız değişikliği uygulayacaktır.



# Pipeline

- Pipeline otomatik olarak çalışacak ve yaptığımız değişikliği uygulayacaktır.



# «deployment»

- «**Staging**» ve «**prod**» ortamına yönelik yapılan CI/CD pipeline sonucu repo menüsünde «**Deployments**» bölümünde görülebilir.

