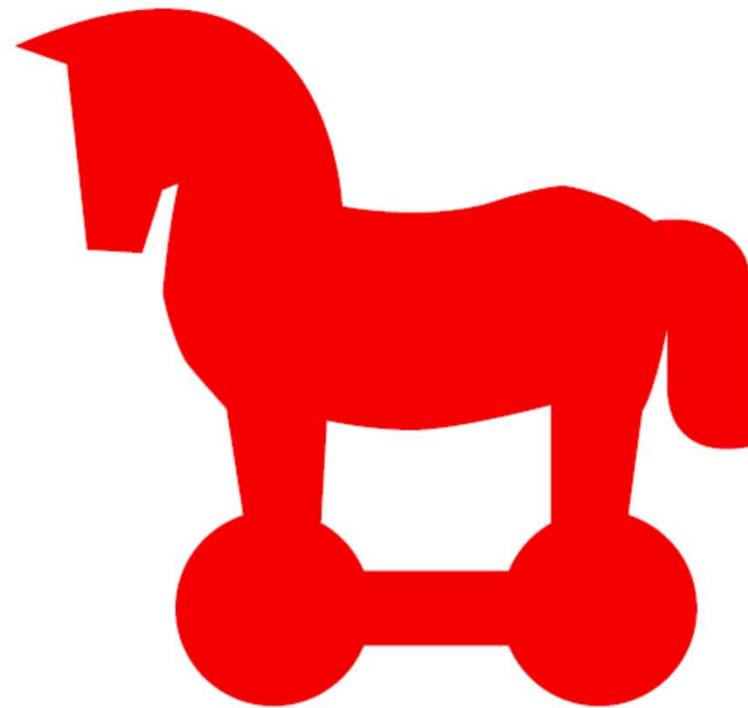


**GANTEK**  
INTEGRATING FUTURE

**GANTEK**  
INTEGRATING FUTURE

**GANTEK**  
INTEGRATING FUTURE



# **GANTEK ACADEMY**

**40 YILLIK TECRÜBE İLE**



**JENKINS**



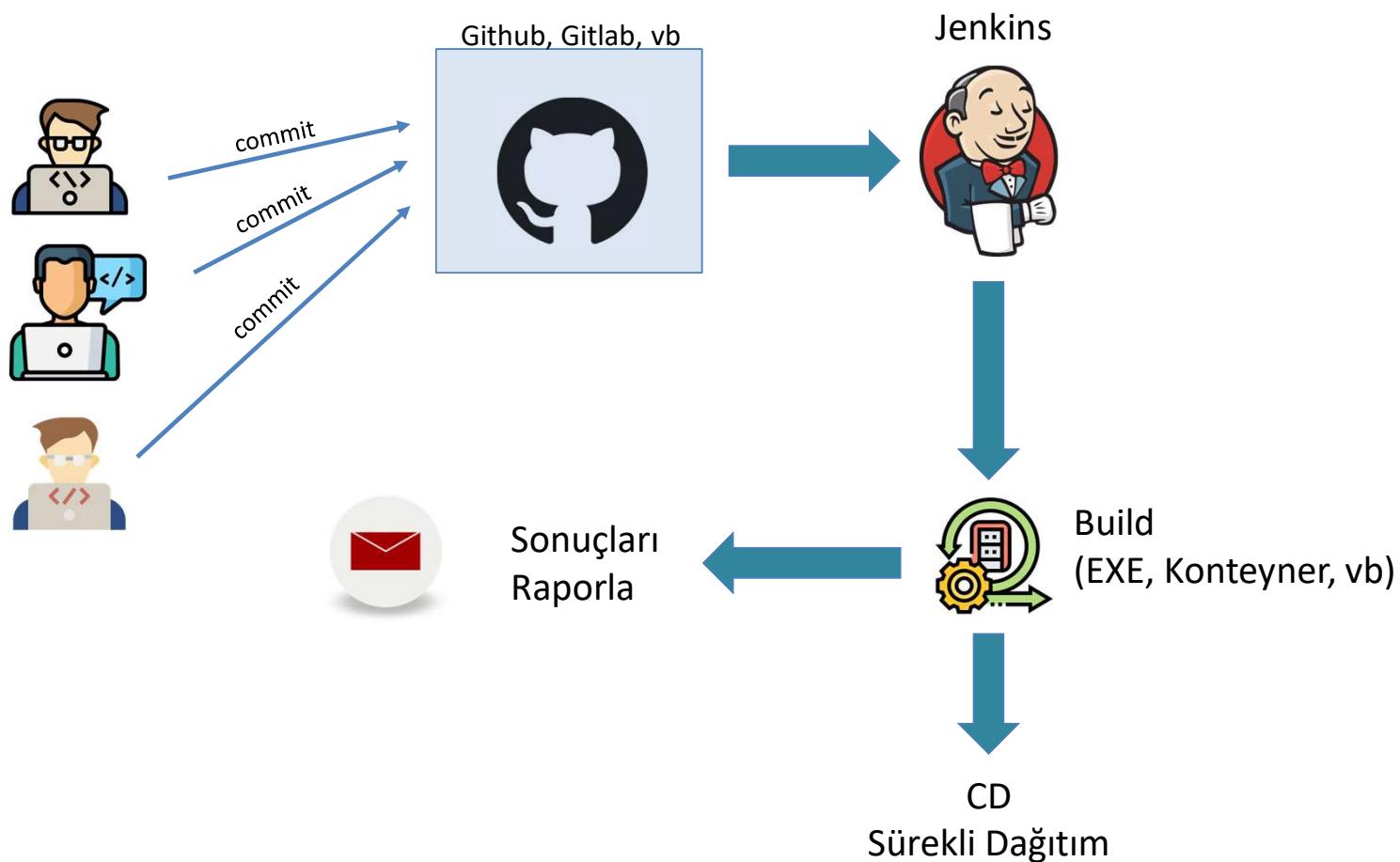
# Jenkins nedir?

- Jenkins açık kaynak kodlu bir otomasyon sunucusudur.
- İlk sürümü 2011'de çıkmıştır.
- Sağladığı yüzlerce eklenti (plugin) sayesinde yazılım geliştirme süreçlerinin, kodlama, sınama, dağıtım ve diğer aşamalarında otomasyon amacıyla kullanılabilmektedir.
- Plugin mekanizması sayesinde yeni özellikler eklenebilir ve önceki özellikler geliştirilebilir.
- Web ara yüzü üzerinden yapılandırılabilir.
- CI/CD süreçlerinde yer alabilir. İstenirse CI tarafında çalışırken, istenirse CD tarafı için de yapılandırılabilir.
- Linux, Windows, MacOS işletim sistemlerinde çalışabilir.
- Docker konteyner olarak Docker Hub'da imajları mevcuttur.

# Jenkins ne yapar?

- Jenkins, projelerin entegrasyonu için bir otomasyon sistemi olarak, kodları veya gerekli diğer dosyaları indirebileceği bir kaynak kodu kontrol sistemi (SCM - Source Control System - örneğin git) ile birlikte çalışması önerilir.
- Kaynak kodlarını SCM'den (veya başka bir alandan) indirerek, hazırlanan betikleri ve Makefile'ları kullanarak «**build**» işlemini gerçekleştirir.
- «**build**» işlemi, aşağıdaki mekanizmalarla tetiklenerek başlatılabilir:
  - Sürüm Kontrol sistemine yapılan «**commit**» işleminden sonra Github/GitLab tarafından çalıştırılan «**webhook**» sayesinde
  - Belirlenen aralıklarla ve zamanda
  - Diğer «**build**» işlemlerinden sonra veya «**build**» işlemlerinin talebiyle
- Hata durumunda, oluşan loglar sistem yöneticisine e-posta üzerinden ilettilir ve geri besleme sağlanır.

# Jenkins nasıl çalışır?



[https://www.flaticon.com/free-icon/web-development\\_1688400](https://www.flaticon.com/free-icon/web-development_1688400)  
[https://www.flaticon.com/premium-icon/coding\\_3242313](https://www.flaticon.com/premium-icon/coding_3242313)  
[https://www.flaticon.com/free-icon/software-development\\_1875674](https://www.flaticon.com/free-icon/software-development_1875674)

# Eklentiler (plugin)

- Jenkins, işlevlerini genişletmek için eklenti (plugin) mekanizmasını etkin bir şekilde kullanır.
- 1000'den fazla eklenti bulunmaktadır.
- Eklentiler sayesinde, Jenkins farklı sürüm kontrol sistemlerine (Git, Subversion, AccuRev, ClearCase, vb) bağlanarak kodları «build» için çekebilir.
- JUnit gibi birim testleri için kullanılan bir sisteme yine eklentilerle ulaşabilir.
- Docker'a API bağlantısı yaparak konteyner oluşturabilir.
- Kubernetes pod'larını oluşturmak için Kubernetes eklentisi kullanılabilir.
- Bulut yapılarında «**build**» düğümleri oluşturmak için çeşitli eklentiler bulunmaktadır.
- Proje yönetimi araçlarıyla (örneğin Jira) yönetilebilir.

# Jenkins'in Alternatifleri

- Jenkins sektördeki en iyi CI araçlarından biridir.
- Alternatifleri şunlardır:
  - **Buddy** (ücretli)
    - <https://buddy.works/compare/jenkins-alternative>
  - **Final builder** (ücretli)
    - <https://www.finalbuilder.com/>
  - **CruiseControl** (açık kaynak kodlu ve ücretsiz)
    - <http://cruisecontrol.sourceforge.net/>
  - **Integrity** (açık kaynak kodlu ve ücretsiz)
    - <http://cruisecontrol.sourceforge.net/>
  - **GoCD** (açık kaynak kodlu ve ücretsiz)
    - <https://www.gocd.org/>
  - **Urbancode** (En iyi Jenkins alternatifi – IBM ürünü)
    - <https://www.urbancode.com/>
  - **Autorabit** (ücretli)
    - <https://www.autorabit.com>
  - **CircleCI**
    - <https://circleci.com/>

<https://www.guru99.com/jenkins-alternative.html>

# Jenkins Kurulumu

- Jenkins, üç şekilde çalıştırılabilir:
  - WAR dosyası Java üzerinde yalnız bir şekilde çalıştırılabilir.
  - Ubuntu, Red Hat, vb sistemlerdeki **apt**, **yum** gibi programlarla kurulabilir. Bu kurulumda, gerekli bileşenler ilgili dizinlere kurulur ve çalıştırılır.
  - ★ Docker üzerinde konteyner olarak çalıştırılabilir.

# Jenkins Kurulumu – War dosyası

- Java üzerinde WAR dosyası çalıştırılacaksa Java programının indirilmesi ve çalıştırılması gereklidir:

```
$ sudo apt install openjdk-11-jdk
```

```
$ wget https://mirrors.jenkins.io/war-stable/latest/jenkins.war
```

```
$ sudo java -jar jenkins.war --httpPort=8080 --prefix=/girisekrani
```

- Jenkins kurulumu için WAR dosyasının LTS sürümünün kullanılması önerilmektedir.
- WAR dosyası çalıştırılırken konsolda, uzun bir **admin** şifresi görülecektir. Sisteme giriş yapıldığında giriş ekranı olarak bu girilmelidir.

**<http://localhost:8080/girisekrani>**

# Jenkins Kurulumu - Ubuntu

- Öncesinde, sistem kurulumu için aşağıdaki komutları çalıştırmak gerekmektedir:

```
$ wget -q -O - https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo apt-key add -
$ sudo sh -c 'echo deb https://pkg.jenkins.io/debian-stable binary/ > /etc/apt/sources.list.d/jenkins.list'
```

- Java ve Jenkins kurulumları yapılmalıdır:

```
$ sudo apt update
$ sudo apt install fontconfig openjdk-11-jdk git
$ sudo apt install jenkins
```

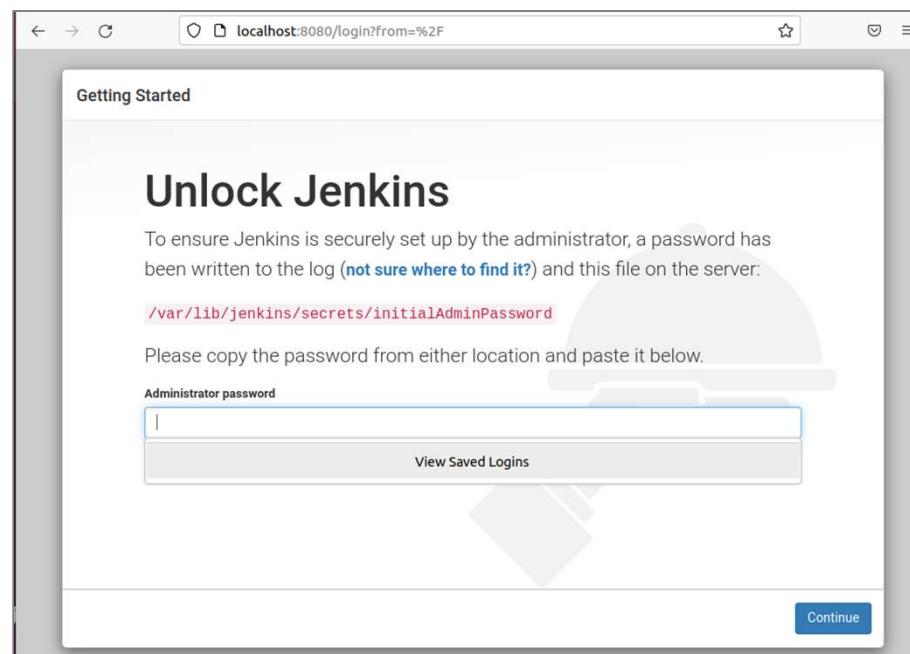
- Jenkins servisinin yüklenildiğinden emin olmak için şu komut kullanılabilir:  
`$ sudo systemctl status jenkins`
- Jenkins servisinin her sistem açılışında yeniden yüklenildiğinden emin olmak için şu komut kullanılmalıdır:  
`$ sudo systemctl enable jenkins`

<https://pkg.jenkins.io/debian-stable/>

# Giriş

- Firefox'ta `http://localhost:8080/` girilerek Jenkins'e erişilebilir. Eğer Jenkins `war` dosyasından çalıştırılsa ilk girişte bir `admin` şifresi isteyecektir. Bu da şu komutla görülebilir:

```
$ journalctl -u jenkins.service
```



# Jenkins Eklentileri (plugin)

- Genel amaçlı bir yazılımı her şeyi öngörerek hazırlamak imkansızdır.
- Jenkins, sonucta bir otomasyon yazılımıdır ve hangi sistemlerde kullanılacağı önceden bilinmeden hazırlanmıştır.
- Jenkins'i hazırlayanlar, farklı projelere özgü olan süreçleri, eklentilerle çözmeyi planlamıştır.
- Jenkins, başlatıldığında eklentilerin yüklenmesi için iki seçenek sunar:

**Install suggested  
plugins**

Install plugins the  
Jenkins community finds  
most useful.

**Select plugins to  
install**

Select and install plugins  
most suitable for your  
needs.

# Ekenglilerin Yüklenmesi

Getting Started

## Getting Started

The screenshot shows the Jenkins 'Getting Started' page. At the top, there's a 'Getting Started' section with a progress bar. Below it is a table of available Jenkins plugins. The table has five columns: Folders, OWASP Markup Formatter, Build Timeout, Credentials Binding, and various others. Plugins like Folders and OWASP Markup Formatter have green checkmarks, while others like Timestamper and Workspace Cleanup have blue circular icons. A tooltip for 'OWASP Markup Formatter' indicates it's a required dependency. At the bottom of the table, there's a note about required dependencies. The footer of the page shows 'Jenkins 2.332.1'.

✓ Folders	✓ OWASP Markup Formatter	↻ Build Timeout	↻ Credentials Binding	** JavaBeans Activation Framework (JAF) API ** JavaMail API ** SSH server Folders OWASP Markup Formatter ** Structs ** Token Macro
↻ Timestamper	↻ Workspace Cleanup	↻ Ant	↻ Gradle	
↻ Pipeline	↻ GitHub Branch Source	↻ Pipeline: GitHub Groovy Libraries	↻ Pipeline: Stage View	
↻ Git	↻ SSH Build Agents	↻ Matrix Authorization Strategy	↻ PAM Authentication	
↻ LDAP	↻ Email Extension	↻ Mailer		

\*\* - required dependency

Jenkins 2.332.1

# Kullanıcı oluşturma

- **admin** kullanıcısı olarak yetkilere sahip bir kullanıcının oluşturulması istenir:

## Create First Admin User

Username:

Password:

Confirm password:

Full name:

E-mail address:

.332.1 Skip and continue as admin

# Jenkins Kullanımı

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

32.1 **Jenkins is ready!**  
Your Jenkins setup is complete.  
[Start using Jenkins](#)

Not now [Save and Finish](#)

Giriş için kullanılacak URL'in girilmesi beklenmektedir.

# Örnek Uygulama

- Jenkins'te «**Freestyle project**» oluşturarak,
  - basit bir Java programını github.com'dan indirerek  
<https://github.com/LuisJoseSanchez/hello-world-java.git>
  - «**javac HelloWorld.java**» ile derleyerek
  - «**java HelloWorld**» programını çalıştıracak projeyi gerçekleştireceğiz.
- Derleme ve çalışma (test) sonucunun görülebileceği seçenek «**Freestyle project**» opsiyonudur.
- Hata çıkarsa «**Console Output**»a bakacağız ve hatayı giderecek işlemleri yapacağız ve tekrar çalıştıracağız.

# Örnek Uygulama

The screenshot shows the Jenkins dashboard. On the left sidebar, there is a list of navigation items: Dashboard, New Item (which is highlighted with a red box and has a red arrow pointing to it), People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. Below the sidebar, there are two sections: 'Build Queue' (No builds in the queue) and 'Build Executor Status' (1 Idle, 2 Idle). The main content area features a 'Welcome to Jenkins!' message, a 'Start building your software project' section with a 'Create a job' button, and a 'Set up a distributed build' section with 'Set up an agent', 'Configure a cloud', and 'Learn more about distributed builds' options.

Yeni bir proje eklemek  
için «**New Item**»  
opsiyonuna  
tıklanmalıdır.

# Örnek Uygulama

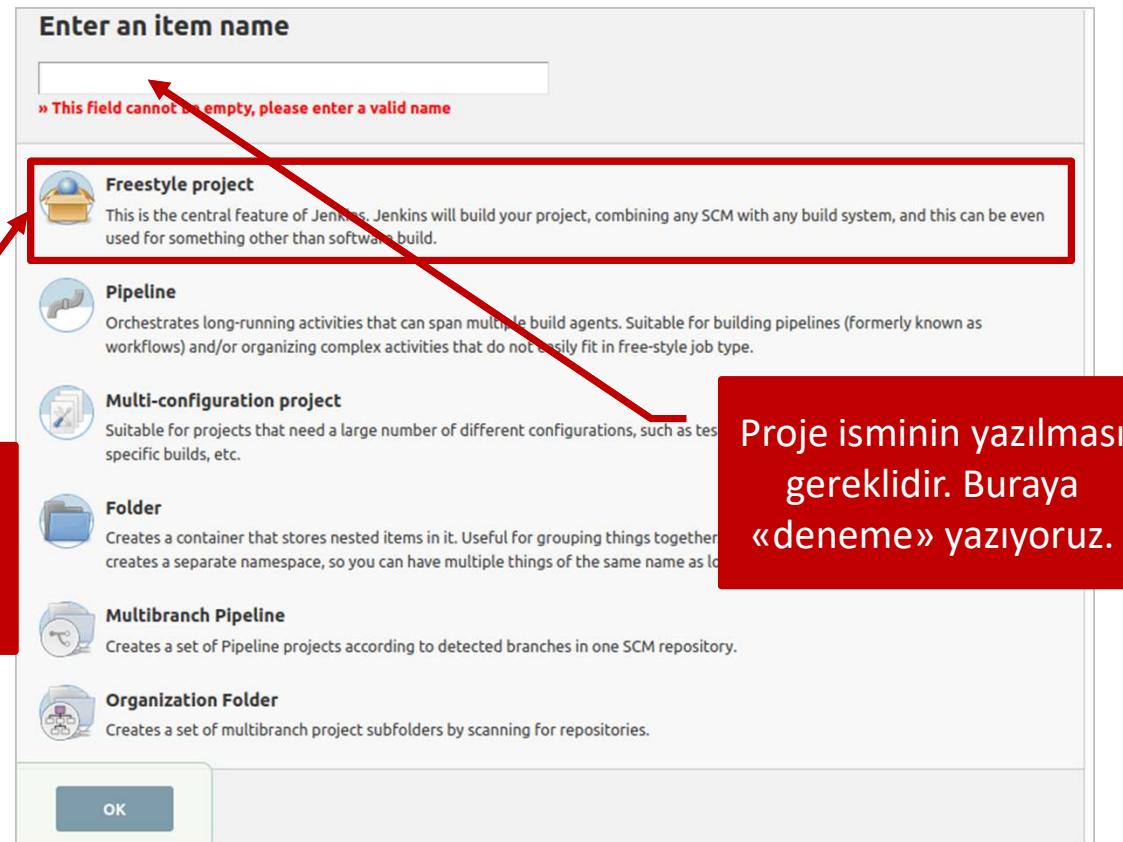
«Freestyle project» esnek ve kullanılışlı bir opsiyondur.  
Herhangi bir proje için kullanılabilir.

Proje ismi yaz  
(Deneme Program)

«Freestyle project»  
opsiyonunu seçiyoruz.

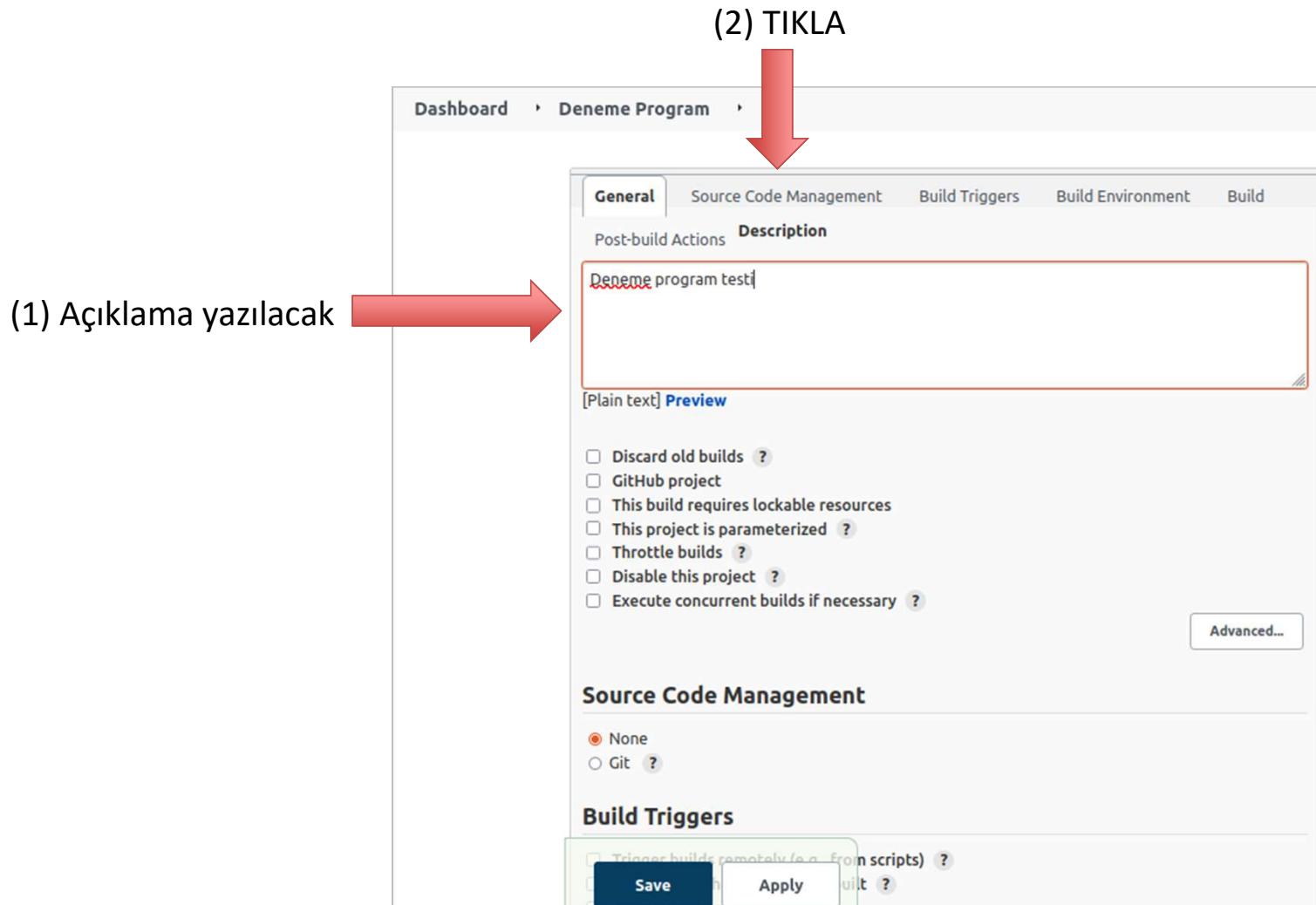
En son tıkla

Proje isminin yazılması  
gereklidir. Buraya  
«deneme» yazıyoruz.



<https://www.guru99.com/create-builds-jenkins-freestyle-project.html>

# Örnek Uygulama

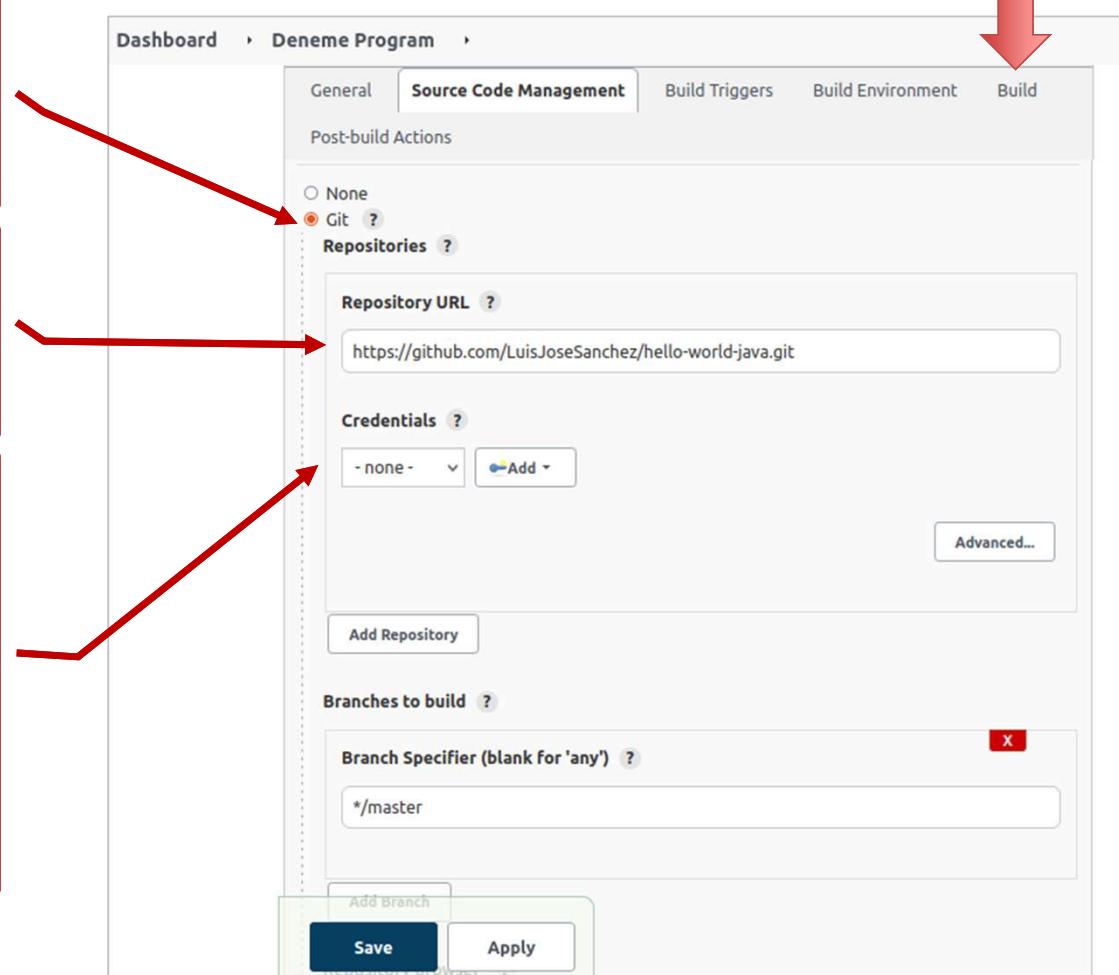


# Örnek Uygulama

«Git» opsiyonunu seçiyoruz.

Kodların indirileceği GitHub adresini yazıyoruz.

Eğer kod deposu özel Olsaydı (kullanıcı adı ve şifreyle girilebilseydi) «Add» butonuna basarak Github için kullanıcı adı ve şifre yazacaktık.



(3) Tıkla

<https://github.com/LuisJoseSanchez/hello-world-java.git>

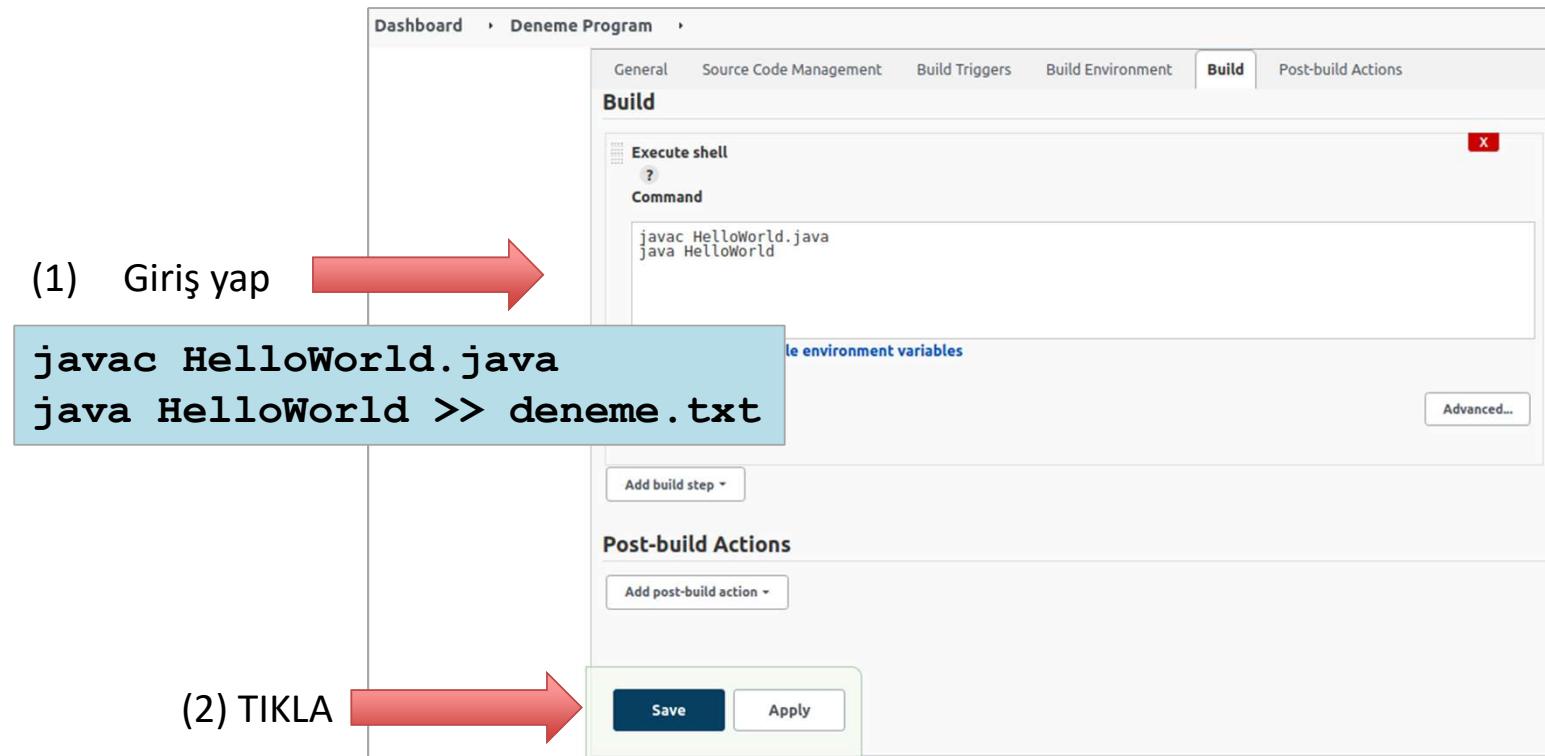
# Örnek Uygulama

Derleme ve test süreçlerini Bash kabuk programıyla gerçekleştireceğiz.  
Bu nedenle «**Execute Shell**» opsiyonunu seçeceğiz.

The screenshot shows a Jenkins build configuration for a project named "Deneme Program". The "Build" tab is active. In the "Build Environment" section, there are several checkboxes for post-build actions like "Build periodically" and "GitHub hook trigger for GITScm polling". The "Build" section contains a dropdown menu titled "Add build step". The "Execute shell" option is highlighted with a red arrow. Other options in the dropdown include "Execute Windows batch command", "Invoke Ant", "Invoke Gradle script", "Invoke top-level Maven targets", "Run with timeout", and "Set build status to "pending" on GitHub commit". At the bottom of the "Build" section are "Save" and "Apply" buttons.

<https://github.com/LuisJoseSanchez/hello-world-java.git>

# Örnek Uygulama

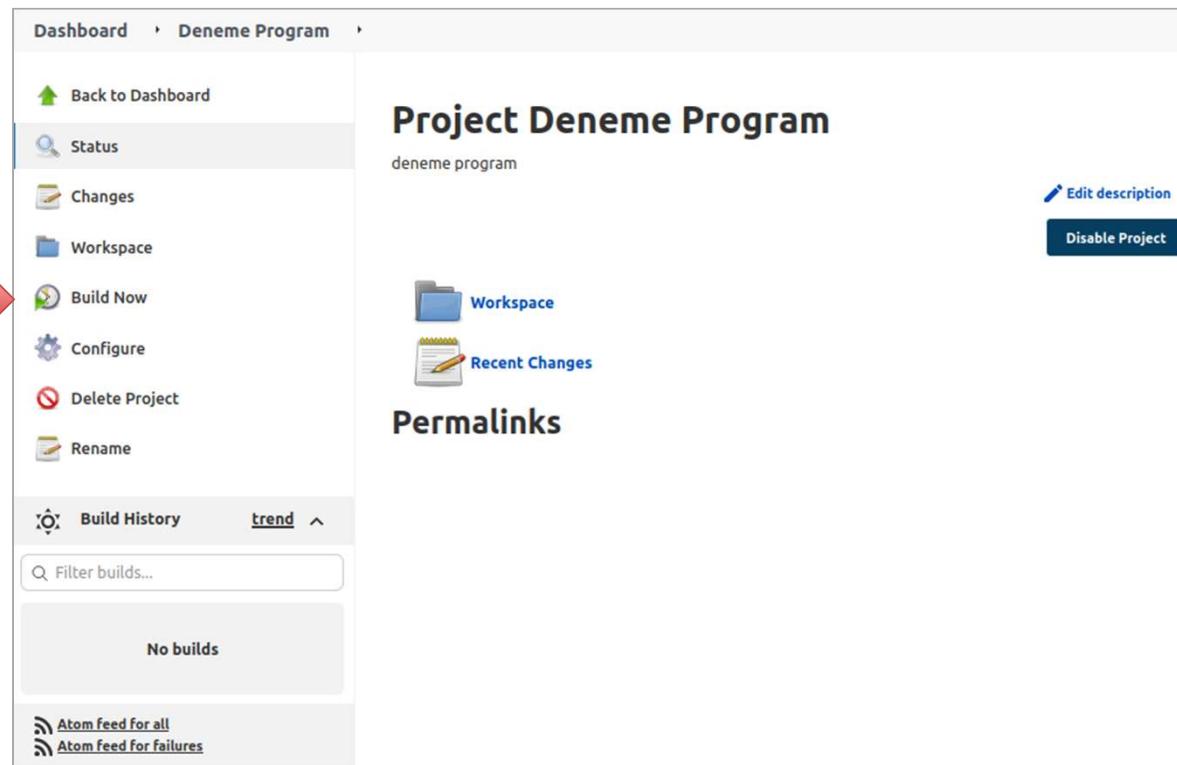


<https://github.com/LuisJoseSanchez/hello-world-java.git>

# Örnek Uygulama

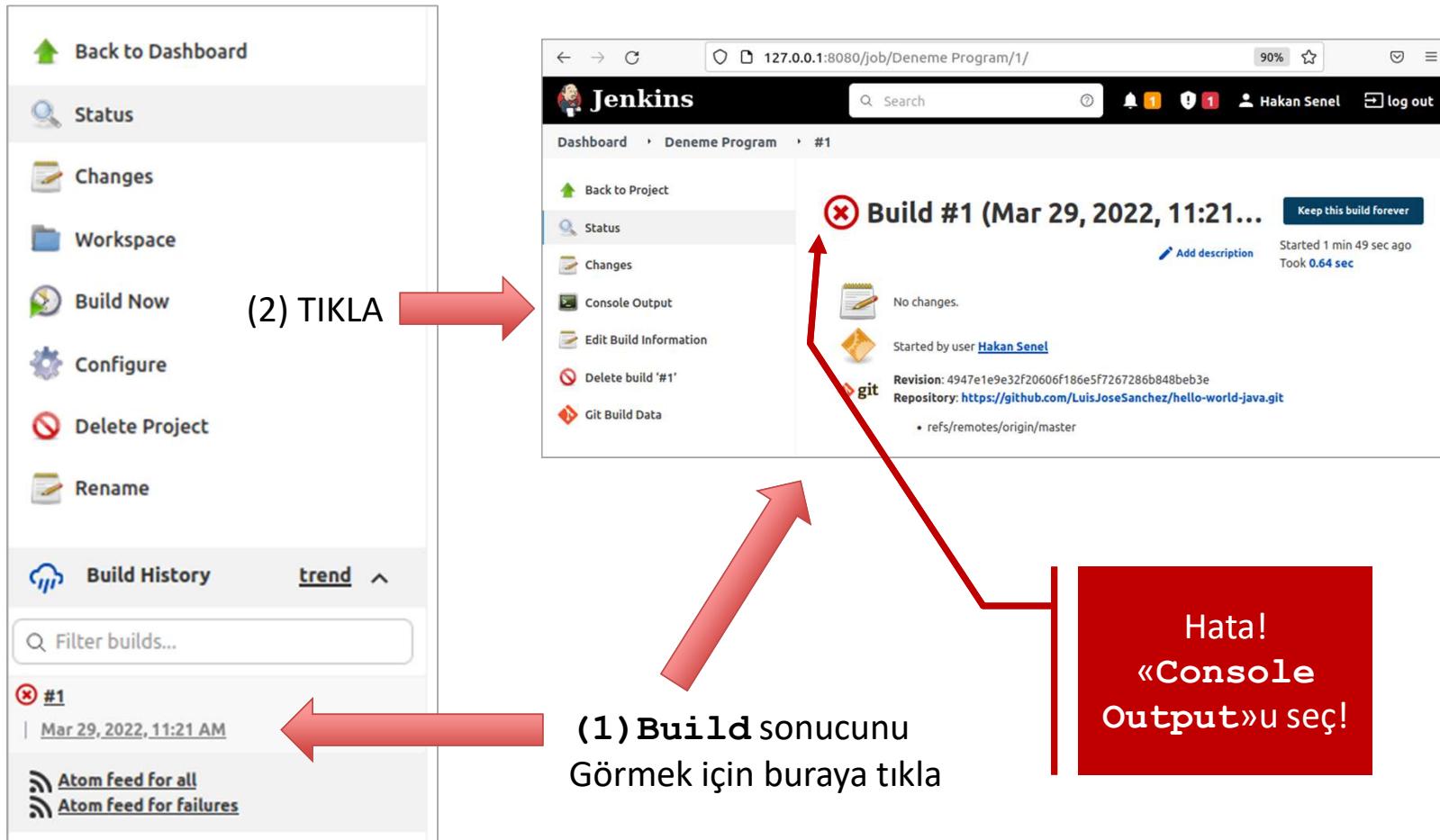
- Soldaki menüden «Build Now» seçilerek «Build» prosesi başlatılabilir.

(1) TIKLA



The screenshot shows a software interface for managing a project named "Deneme Program". The left side features a sidebar with various options: Back to Dashboard, Status, Changes, Workspace, Build Now, Configure, Delete Project, and Rename. Below this is a "Build History" section with a "trend" dropdown and a "Filter builds..." search bar. At the bottom of the sidebar are links for Atom feeds. The main content area is titled "Project Deneme Program" and displays "deneme program". It includes sections for "Workspace" (with a blue folder icon) and "Recent Changes" (with a notepad icon). There are also "Edit description" and "Disable Project" buttons on the right. The overall layout is clean and modern, typical of a web-based development tool.

# Örnek Uygulama



The image shows two screenshots of the Jenkins interface. The left screenshot is a sidebar menu with options like 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build Now' (which has a red arrow pointing to it labeled '(2) TIKLA'), 'Configure', 'Delete Project', and 'Rename'. Below this is a 'Build History' section with a filter input and a list of builds. The first build in the list is '#1 Mar 29, 2022, 11:21 AM' with a red arrow pointing to it labeled '(1) Build sonucunu Görmek için buraya tıkla'. At the bottom are links for 'Atom feed for all' and 'Atom feed for failures'. The right screenshot shows the Jenkins dashboard for project 'Deneme Program' with build '#1'. The build status is marked with a red circle and an 'X', indicating failure. The message says 'Build #1 (Mar 29, 2022, 11:21...)' and 'No changes.' It also shows the user 'Hakan Senel' started the build, the revision '4947e1e9e32f20606f186e5f7267286b848beb3e', the repository 'https://github.com/LuisJoseSanchez/hello-world-java.git', and the ref 'refs/remotes/origin/master'. A red callout box on the right says 'Hata! «Console Output»u seç!'.

(2) TIKLA

(1) Build sonucunu  
Görmek için buraya tıkla

Hata!  
«Console  
Output»u seç!

# Örnek Uygulama

- Hata veren bir «Build», üzerine tıklanarak görüntülenebilir ve «Console Output» opsyonuyla neden hata verdiği anlaşılabilir.

**javac programı Sisteme yüklenmediğinden Hata veriyor.**

The screenshot shows a Jenkins console output for a build named '#2'. The output indicates that the build was started by user 'Hakan Senel' and is running as SYSTEM. It shows the git configuration and fetching of changes from a GitHub repository. The build step 'Execute shell' failed because 'javac' was not found. A red box highlights the error message: '+ javac HelloWorld.java /tmp/jenkins78650702361048795.sh: 2: javac: not found'. Below the screenshot, a red arrow points to the error message with the text 'JDK yüklenmeli: sudo apt install openjdk-11-jdk-headless'.

```

Dashboard > Deneme Program > #2
Back to Project
Status
Changes
Console Output
View as plain text
Edit Build Information
Delete build '#2'
Git Build Data
Previous Build

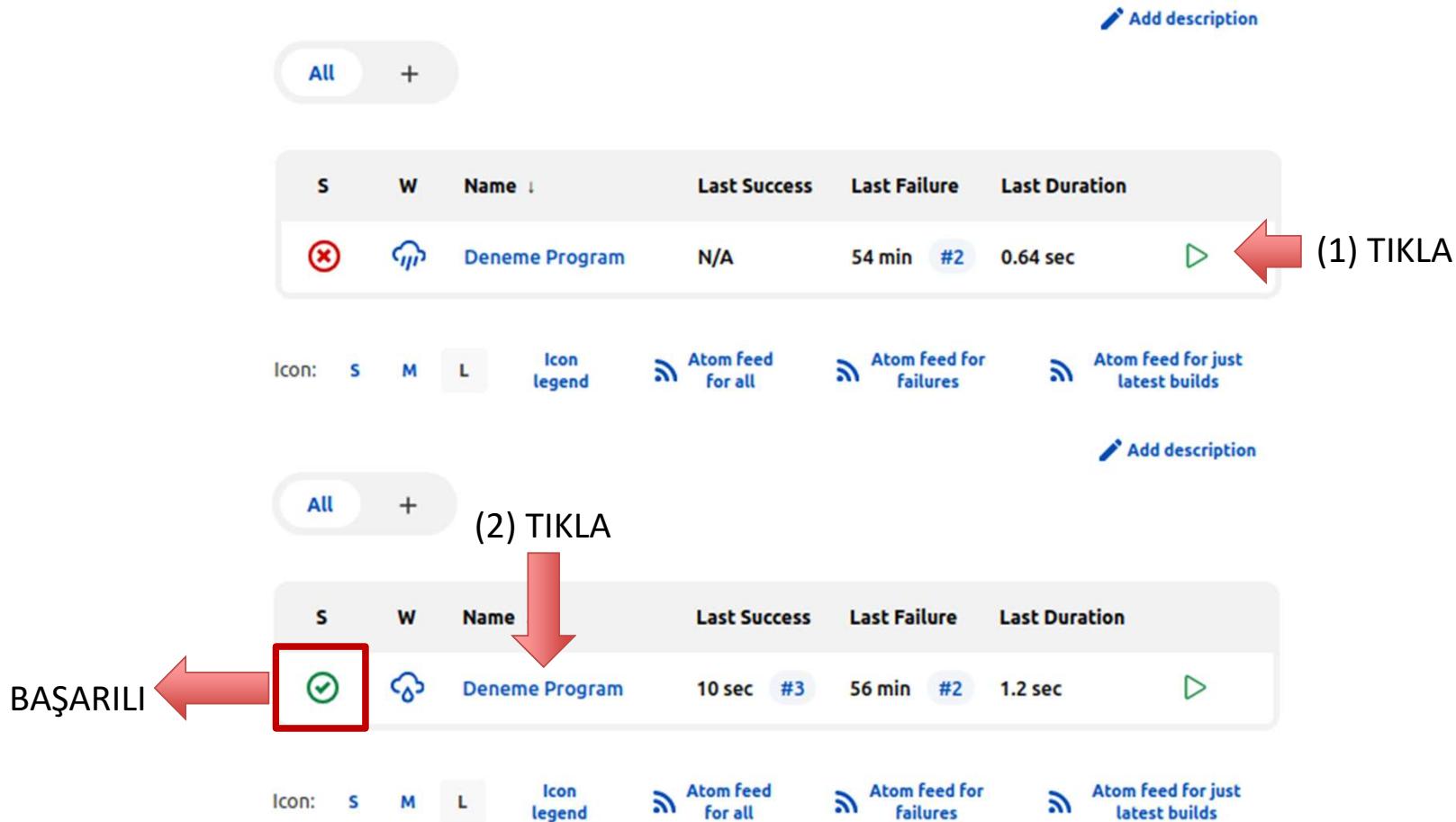
⑧ Console Output
Started by user Hakan Senel
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Deneme Program
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Deneme Program/.git #
timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/LuisJoseSanchez/hello-world-java.git
# timeout=10
Fetching upstream changes from https://github.com/LuisJoseSanchez/hello-world-java.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/LuisJoseSanchez/hello-world-java.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 4947ele9e32f20606f186e5f7267286b848beb3e (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 4947ele9e32f20606f186e5f7267286b848beb3e # timeout=10
Commit message: "Updating README.md"
> git rev-list --no-walk 4947ele9e32f20606f186e5f7267286b848beb3e # timeout=10
+ Jenkins Program $ /bin/sh -xe /tmp/jenkins78650702361048795.sh
+ javac HelloWorld.java
/tmp/jenkins78650702361048795.sh: 2: javac: not found
Build step 'Execute shell' marked build as failure
Finished: FAILURE

```

JDK yüklenmeli:  
**sudo apt install openjdk-11-jdk-headless**

# Örnek Uygulama

- Sol üst köşeden «Dashboard» seçilerek, görevler görüntülenebilir.



Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✖	☁️	Deneme Program	N/A	54 min #2	0.64 sec

Icon: S M L    Icon legend    Atom feed for all    Atom feed for failures    Atom feed for just latest builds

Add description

All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
✔	☁️	Deneme Program	10 sec #3	56 min #2	1.2 sec

Icon: S M L    Icon legend    Atom feed for all    Atom feed for failures    Atom feed for just latest builds

# Örnek Uygulama

- Proje üzerine tıklanırsa, «**Workspace**» ve «**Recent Changes**» bölümleri görülebilir.

## Project Deneme Program

deneme program

 Edit description

 Disable Project

 **Workspace** (1) TIKLA

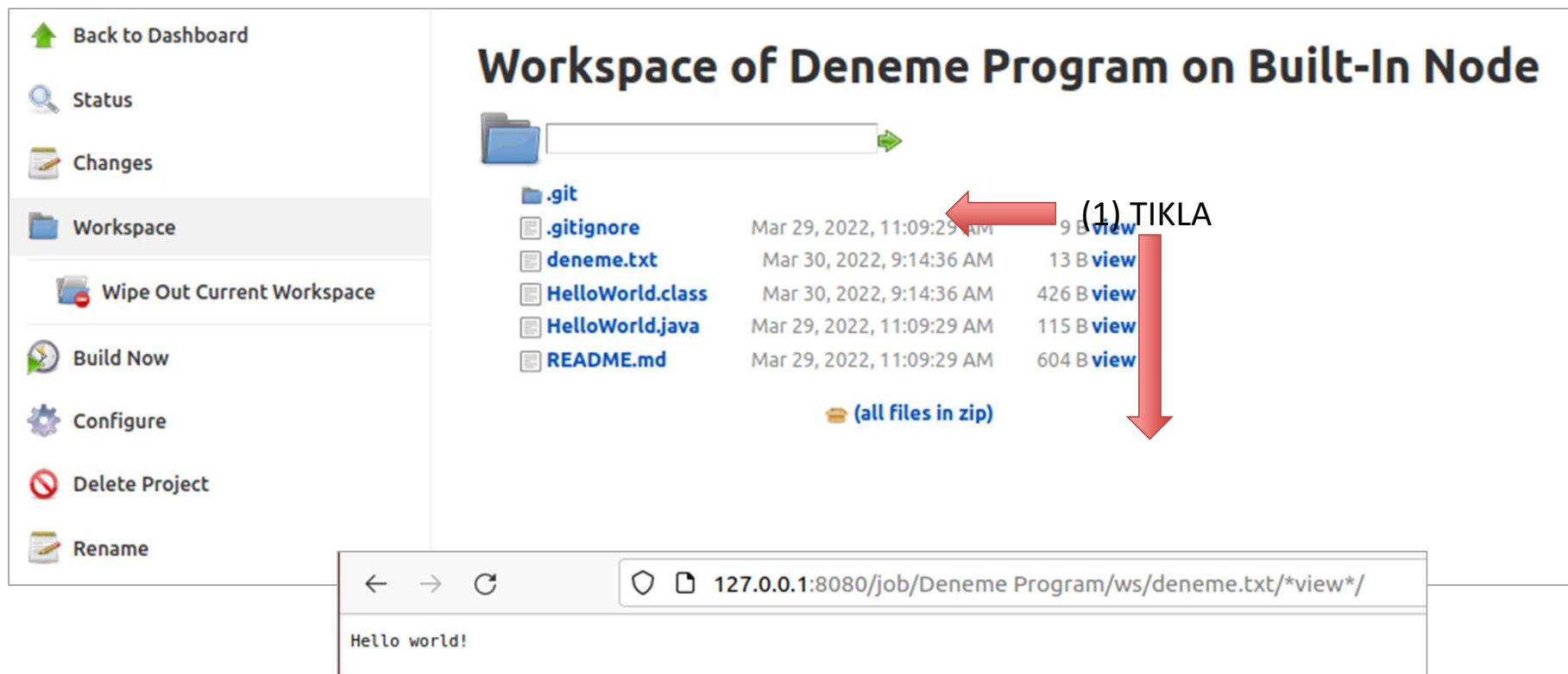
 **Recent Changes**

### Permalinks

- [Last build \(#3\), 20 hr ago](#)
- [Last stable build \(#3\), 20 hr ago](#)
- [Last successful build \(#3\), 20 hr ago](#)
- [Last failed build \(#2\), 21 hr ago](#)
- [Last unsuccessful build \(#2\), 21 hr ago](#)
- [Last completed build \(#3\), 20 hr ago](#)

# Örnek Uygulama

- Projenin «**build**» sonrası çalışma alanındaki dosyalar görülebilir ve içeriğine bakılabilir.



# Örnek Uygulama

- Proje bilgileri görüntülenirken, sol taraftaki menüdeki «Console Output» opsyonuyla, derlemenin başarılı olup olmadığı ve başarılı değilse neden hata verdiği anlaşılabilir.

Dashboard > Deneme Program > #3

[Back to Project](#)  
[Status](#)  
[Changes](#)  
**Console Output** View as plain text  
[Edit Build Information](#)  
[Delete build '#3'](#)  
[Git Build Data](#)  
[Previous Build](#)

### Console Output

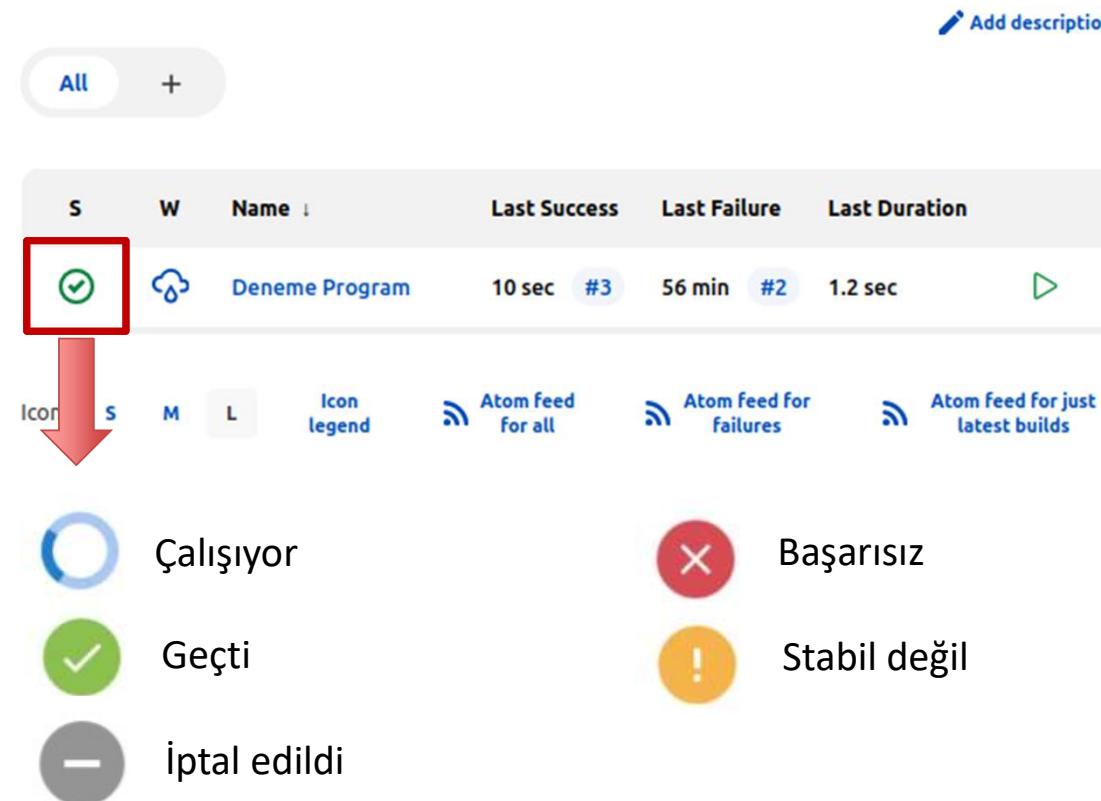
Started by user [Hakan Senel](#)  
 Running as SYSTEM  
 Building in workspace /var/lib/jenkins/workspace/Deneme Program  
 The recommended git tool is: NONE  
 No credentials specified  

```
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Deneme Program/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/LuisJoseSanchez/hello-world-java.git
# timeout=10
Fetching upstream changes from https://github.com/LuisJoseSanchez/hello-world-java.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/LuisJoseSanchez/hello-world-java.git +refs/heads/*:refs/remotes/origin/*
# timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 4947e1e9e32f20606f186e5f7267286b848beb3e (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git rev-list --no-walk 4947e1e9e32f20606f186e5f7267286b848beb3e # timeout=10
[Deneme Program] $ /bin/sh -xe /tmp/jenkins18423789557168625077.sh
+ javac HelloWorld.java
+ java HelloWorld
Hello world!
Finished: SUCCESS
```

←
BAŞARILI

# Durum İkonları

- Jenkins, uzun dönemli «build» işlemlerinin sonuçlarını çeşitli ikonlarla göstermektedir.



<https://www.jenkins.io/doc/book/blueocean/dashboard/>

# Sağlık İkonları

- Jenkins, yapılan «**build**» işlemlerinin kaçının başarısız olduğu bilgisini tutmakta ve bunu «hava durumu» ikonuyla ifade etmektedir.

Tıklayarak konsol  
çıktısına erişilebilir

The screenshot shows a Jenkins Blue Ocean dashboard. At the top, there's a search bar with 'All' and a '+' button. Below it is a table row with columns: S, W, Name, Last Success, Last Failure, and Last Duration. The 'Name' column contains 'Deneme Program'. The 'Last Success' column shows '10 sec #3'. The 'Last Failure' column shows '56 min #2'. The 'Last Duration' column shows '1.2 sec'. A red box highlights the 'W' icon in the first column. An arrow points from this icon to the explanatory text above. Another arrow points from the 'W' icon down to the weather icon legend below. The legend includes icons for sun, cloud, and lightning, each associated with a success rate range: '%80'den fazlası başarılı', '%41 - %60 başarılı', and '%20'den azı başarılı' respectively. There are also links for 'Icon legend', 'Atom feed for all', 'Atom feed for failures', and 'Atom feed for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration
(green checkmark)	(red box)	Deneme Program	10 sec #3	56 min #2	1.2 sec

Icon: S L Icon legend Atom feed for all Atom feed for failures Atom feed for just latest builds

%80'den fazlası başarılı      %61 - %80 başarılı

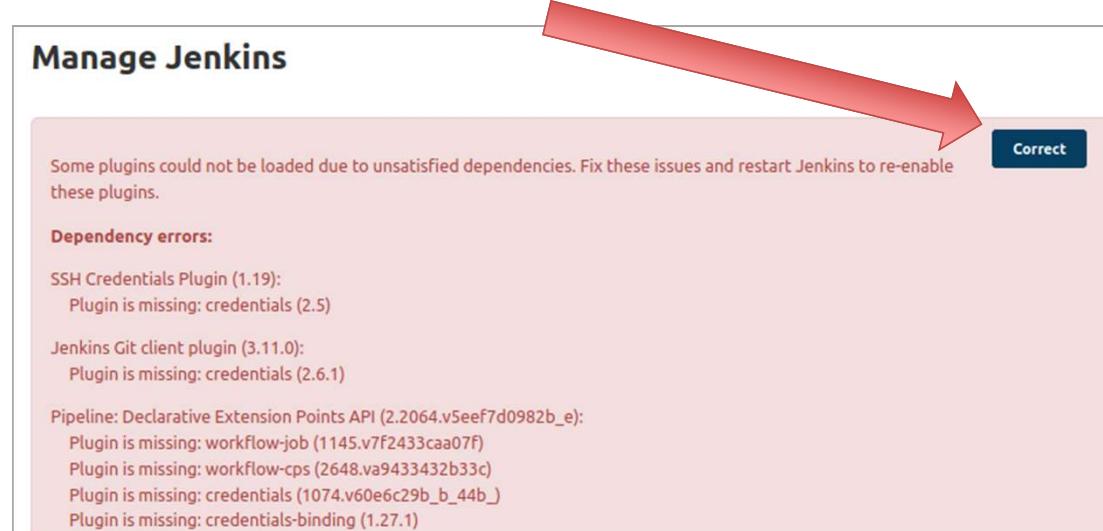
%41 - %60 başarılı      %21 - %40 başarılı

%20'den azı başarılı

<https://www.jenkins.io/doc/book/blueocean/dashboard/>

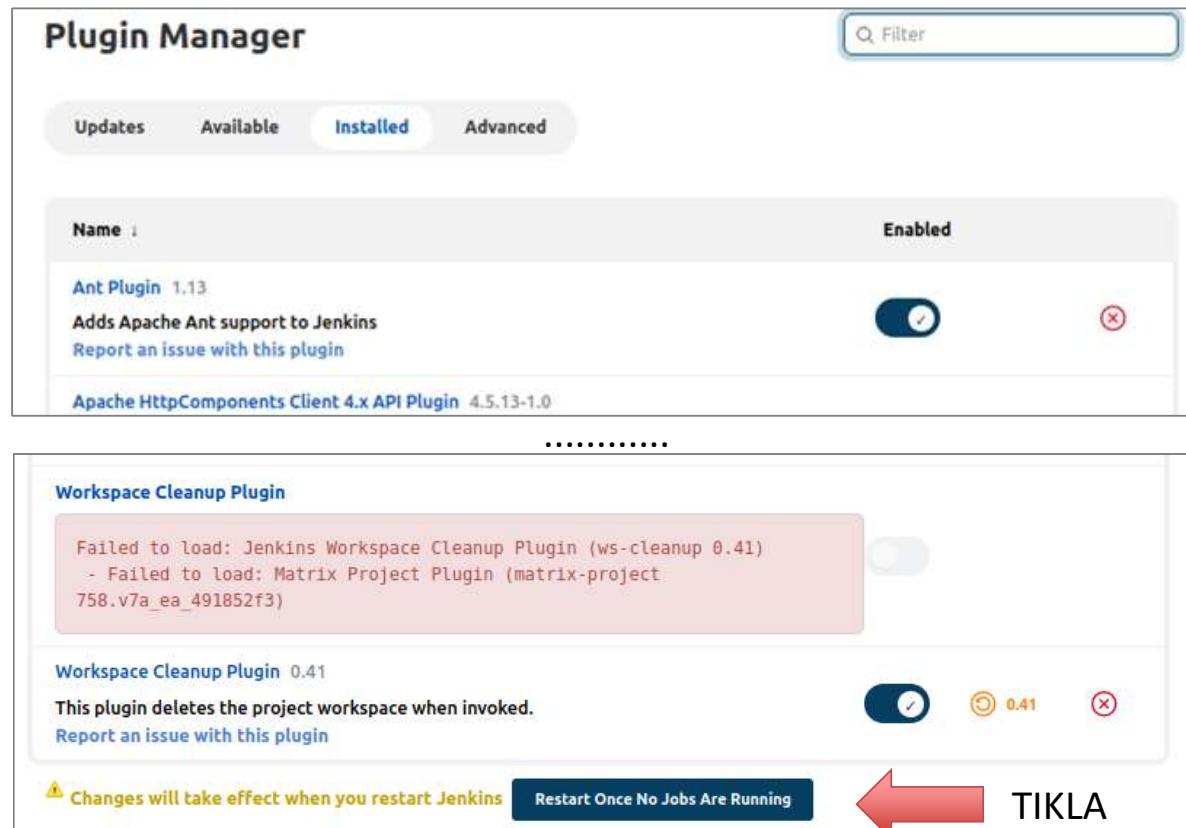
# Jenkins'in Yönetimi

- Sol taraftaki ana menüdeki «**Manage Jenkins**» seçeneği, Jenkins'te yapılandırma işlemlerinin yapılmasını sağlar.
- İlk girildiğinde, eklentilerin yüklenemediği görüldüğünde, «**Correct**» butonuna basılmalıdır.



# «Plugin Manager»

- Eklenti yöneticisinde «**Installed**» bölümüne tıklandığında bazı eklentilerin yüklenemediği görüldüğünde, en aşağıya inilerek «**Restart Once No Jobs Running**»e tıklanarak Jenkins yeniden başlatılmalıdır.



**Plugin Manager**

Updates Available **Installed** Advanced

Name Enabled

[Ant Plugin](#) 1.13  [Report an issue with this plugin](#)

[Apache HttpComponents Client 4.x API Plugin](#) 4.5.13-1.0

.....

**Workspace Cleanup Plugin**

Failed to load: Jenkins Workspace Cleanup Plugin (ws-cleanup 0.41)  
- Failed to load: Matrix Project Plugin (matrix-project  
758.v7a\_ea\_491852f3)

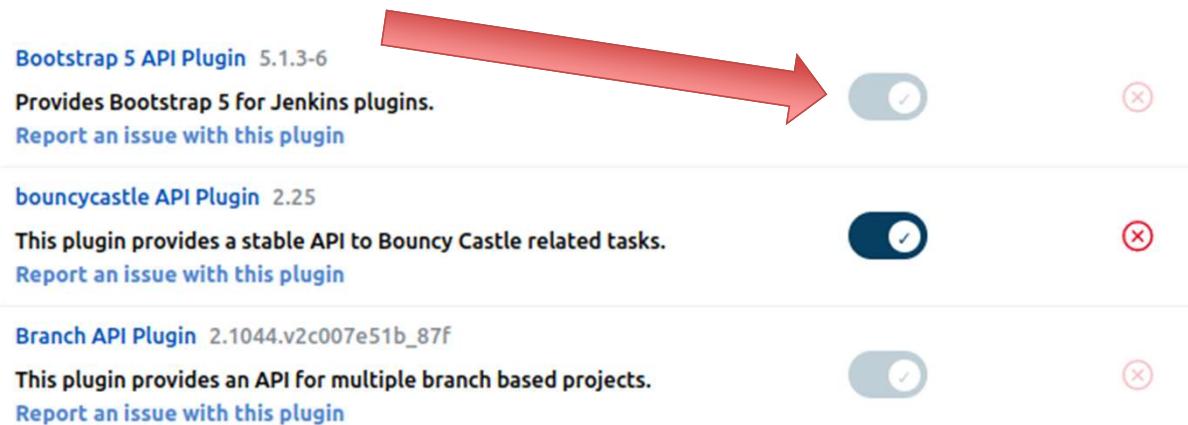
[Workspace Cleanup Plugin](#) 0.41  0.41 [Report an issue with this plugin](#)

**TIKLA**

⚠ Changes will take effect when you restart Jenkins **Restart Once No Jobs Are Running**

# Plugin Manager

- «Plugin manager» bölümüne girildiğinde, güncellenebilecek eklentiler ekranда gösterilmektedir.
- Bazı eklentiler «Enabled» bazıları da «disabled» durumdadır.
- Bazları da sistem için önemli oldukları veya başka eklentiler için kullanıldıklarından «disable» edilemezler.



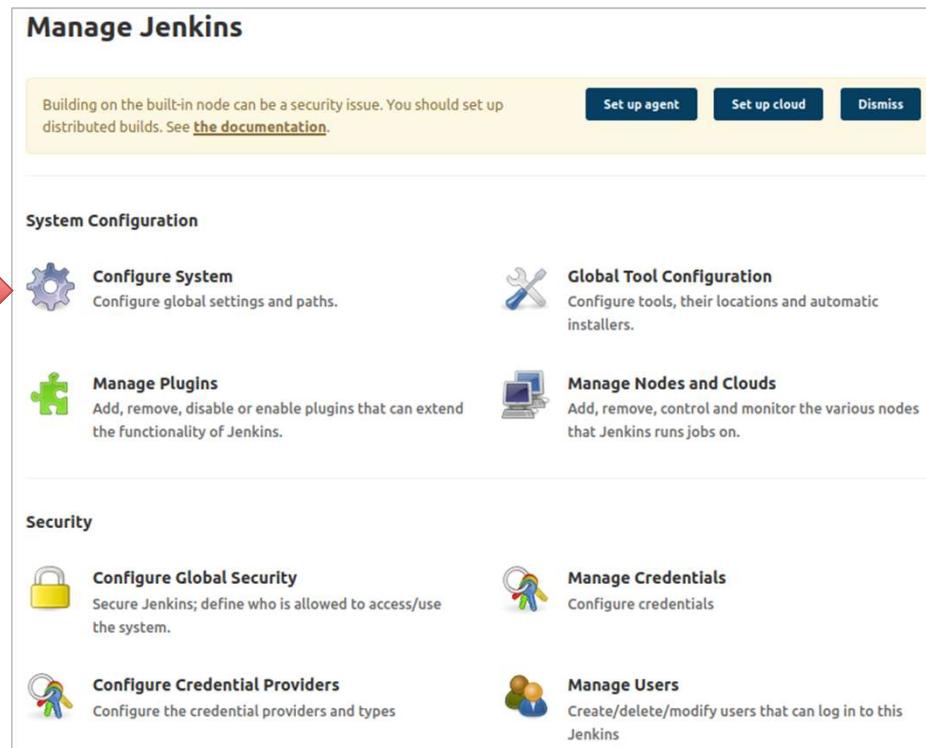
# Jenkins Dosya Sistemi

- Jenkins, verileri depolamak için **/var/lib/jenkins** dizinini kullanmaktadır.
  - Plugin'ler: **/var/lib/jenkins/plugins**
  - Build workspace: **/var/lib/jenkins/plugins**
  - loglar: **/var/lib/jenkins/logs**
  - Kullanıcılar: **/var/lib/jenkins/users**
  - Projeler: **/var/lib/jenkins/jobs**

# JENKINS - YAPILANDIRMA

# Jenkins'in Yönetimi

- «Manage Jenkins» bölümü, Jenkins'in yaptığı işlerin yapılış şeklinin değiştirilmesini sağlar.



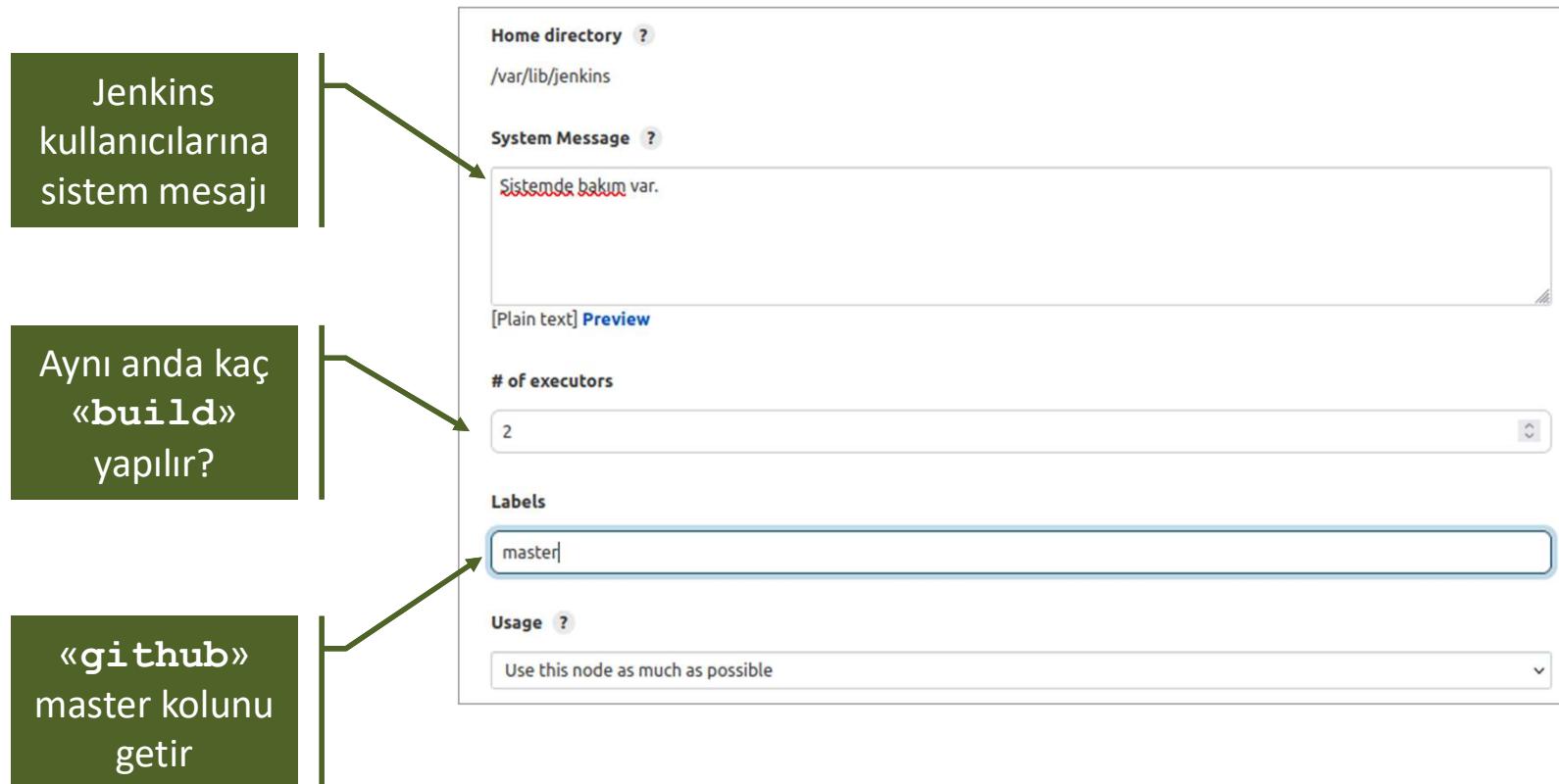
The screenshot shows the Jenkins 'Manage Jenkins' dashboard. At the top, there is a yellow banner with the text: "Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#)". Below the banner are three buttons: "Set up agent", "Set up cloud", and "Dismiss".

The main content area is divided into sections:

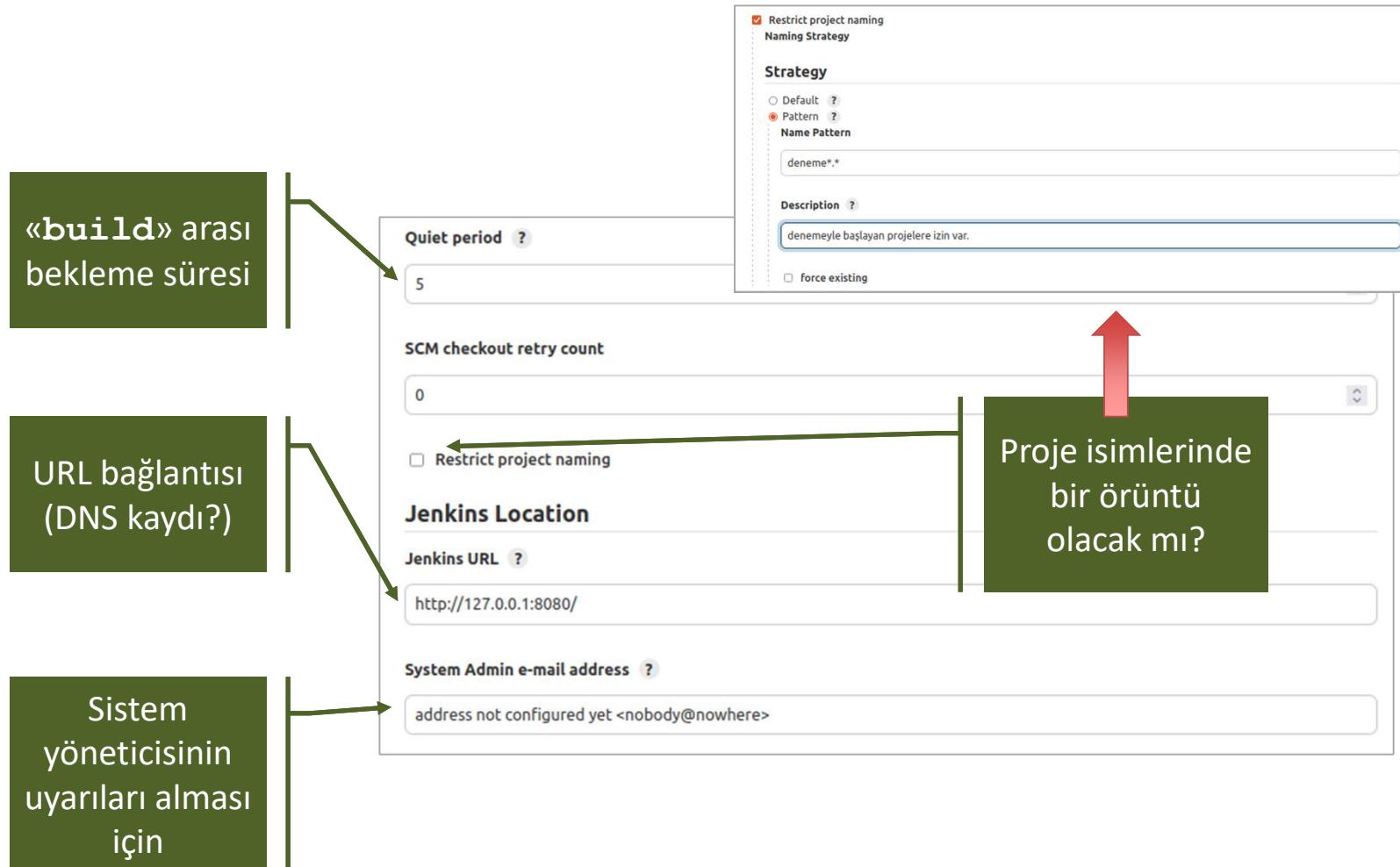
- System Configuration**:
  - Configure System**: Configure global settings and paths.
  - Global Tool Configuration**: Configure tools, their locations and automatic installers.
  - Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
  - Manage Nodes and Clouds**: Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Security**:
  - Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
  - Manage Credentials**: Configure credentials.
  - Configure Credential Providers**: Configure the credential providers and types.
  - Manage Users**: Create/delete/modify users that can log in to this Jenkins.

TIKLA

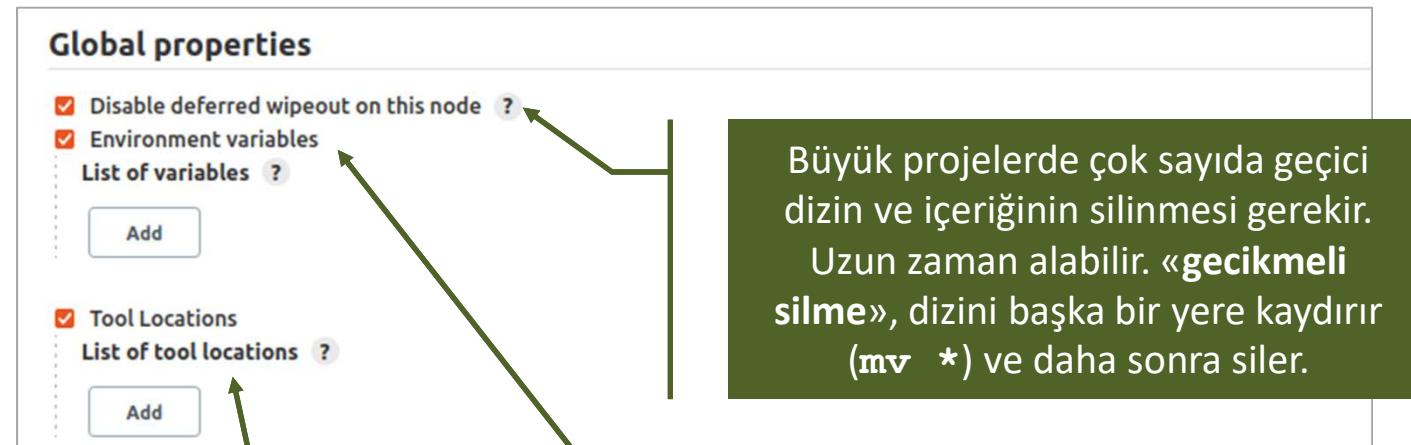
# «Configure System»



# «Configure System»



# «Configure System»



**Global properties**

- Disable deferred wipeout on this node ?
- Environment variables  
[List of variables](#) ?
- Tool Locations  
[List of tool locations](#) ?

**Add**

**Büyük projelerde çok sayıda geçici dizin ve içeriğinin silinmesi gereklidir. Uzun zaman alabilir. «geçikmeli silme», dizini başka bir yere kaydırır (`mv *`) ve daha sonra siler.**

**Betiklerde kullanmak üzere çevre değişkeni tanımlanabilir.**

**Kullanılacak araçların yerlerini göstermek için**

# «Configure System»

«pipeline» işinin  
nasıl yapılacağı

Sağlık seviyesini  
nasıl  
belirleyeceğiz?

Sistem saatini  
nasıl  
göstereceğiz?

**Pipeline Speed/Durability Settings**

**Pipeline Default Speed/Durability Level** ?

None: use pipeline default (MAX\_SURVIVABILITY)

None: use pipeline default (MAX\_SURVIVABILITY)

Performance-optimized: much faster (requires clean shutdown to save running pipelines)

Less durability, a bit faster (specialty use only)

Maximum durability but slowest (previously the only option)

Help make Jenkins better by sending anonymous usage statistics and crash reports to the Jenkins project. ?

**Folder**

**Health Metrics**

Add +

Child item with worst health

Health of the primary branch of a repository

**Timestamper**

**System clock time format** ?

'**<b>HH:mm:ss</b>**'

**Elapsed time format** ?

'**<b>HH:mm:ss.S</b>**'

Enabled for all Pipeline builds ?

# «Configure System»

Alt projelerde,  
birbirlerinin ürettiği  
dosyaları  
kullanıyorlarsa?

## Fingerprints

Disable Fingerprint Cleanup

«**build**» sırasında ne  
tür durum bilgilerini  
tutacağını belirtir. Ne  
kadar çok bilgi verirse  
o kadar iyi!

## Administrative monitors configuration

Administrative monitors...

Eski logların  
silinmesi nasıl  
yapılacak?

## Global Build Discarders

### Project Build Discarder

Build discarders configured for a job are only run after a build finishes. This option runs jobs' configured build discarders periodically, applying configuration changes even when no new builds are run. This option has no effect if there is no build discarding configured for a job.

Delete

Add ▾

### Access Control for Builds Security

If access control for builds is not set up, this shows a warning, explaining the problem.

### Ambiguous Permission Assignments Security

Shows a warning when the built-in node has executors despite agents (static or clouds) being configured.

### Built-In Node Name and Label Migration

When updating Jenkins from before 2.307, this allows administrators to migrate the built-in node name and label.

### Built-In Node With Agents Executors Configured Monitor Security

Shows a warning when the built-in node has executors despite agents (static or clouds) being configured.

### Built-In Node Without Agents Executors Configured Monitor Security

Shows a warning when the built-in node has executors and recommends that agents (static or clouds) are set up.

### CSRF Protection Monitor Security

Warns about disabled **CSRF protection**.

### Check URI Encoding

A diagnostic feature warns administrators about incorrect encoding of URLs, typically with improperly set up reverse proxies or containers.

### Deprecated Plugin Monitor

Inform administrators about the deprecation of one or more currently installed plugins. This is metadata provided by the configured plugin documentation will usually explain why the plugin is deprecated.

### Disabled Security Security

Authentication (security realm) and authorization (authorization strategy) should be set up.

### Disk Usage Monitor

This warning shows up when the available disk space for the Jenkins home directory falls below a certain threshold.

### Enforce TCP Agent Port

Informs administrators that the TCP agent port is different than its enforced value and will be reset on restart.

### GitHub Hooks Problems

### Invalid Plugin Configuration

Inform administrators about a required plugin update. This is unrelated to plugin updates being available.

### JVM Crash Reports

Informs about the presence of JRE crash dumps after an abnormal Jenkins termination, and allows viewing them.

### Jenkins Initialization Monitor

Warns about an incomplete initialization, indicating a bug in Jenkins.

# «Configure System»

Kullanıcı adı/sifre vererek Github sunucusu eklemek için (Github plugin)

Github API kullanımının sınırlını aşmamak için

Şirket içinde çalıştırılan git deposu

«pipeline» kodlarını ortak havuzdan kullanma

**GitHub**

**GitHub API usage**

**GitHub Enterprise Servers**

**Global Pipeline Libraries**

```

Jenkinsfile (Declarative Pipeline)
pipeline {
    agent { docker { image 'python:3.10.1-alpine' } }
    stages {
        stage('build') {
            steps {
                sh 'python --version'
            }
        }
    }
}

```

<https://www.jenkins.io/doc/pipeline/tour/hello-world/>

# «Configure System»

Github'dan kaynak kodu indirildiğinde, kodu kim «**commit**» etmişse bu isim yerine kullanılacak kişiler.

Kaynak kodunu «**commit**» eden kullanıcılar Jenkins'te yoksa, istenirse oluşturulur.

Konsol çıktısında depodan indirmek için kullanılan kullanıcının ismi gösterilmez.

## Git plugin

Global Config user.name Value ?

Global Config user.email Value ?

Create new accounts based on author/committer's email ?

Use existing account with same email if found ?

Show the entire commit summary in changes ?

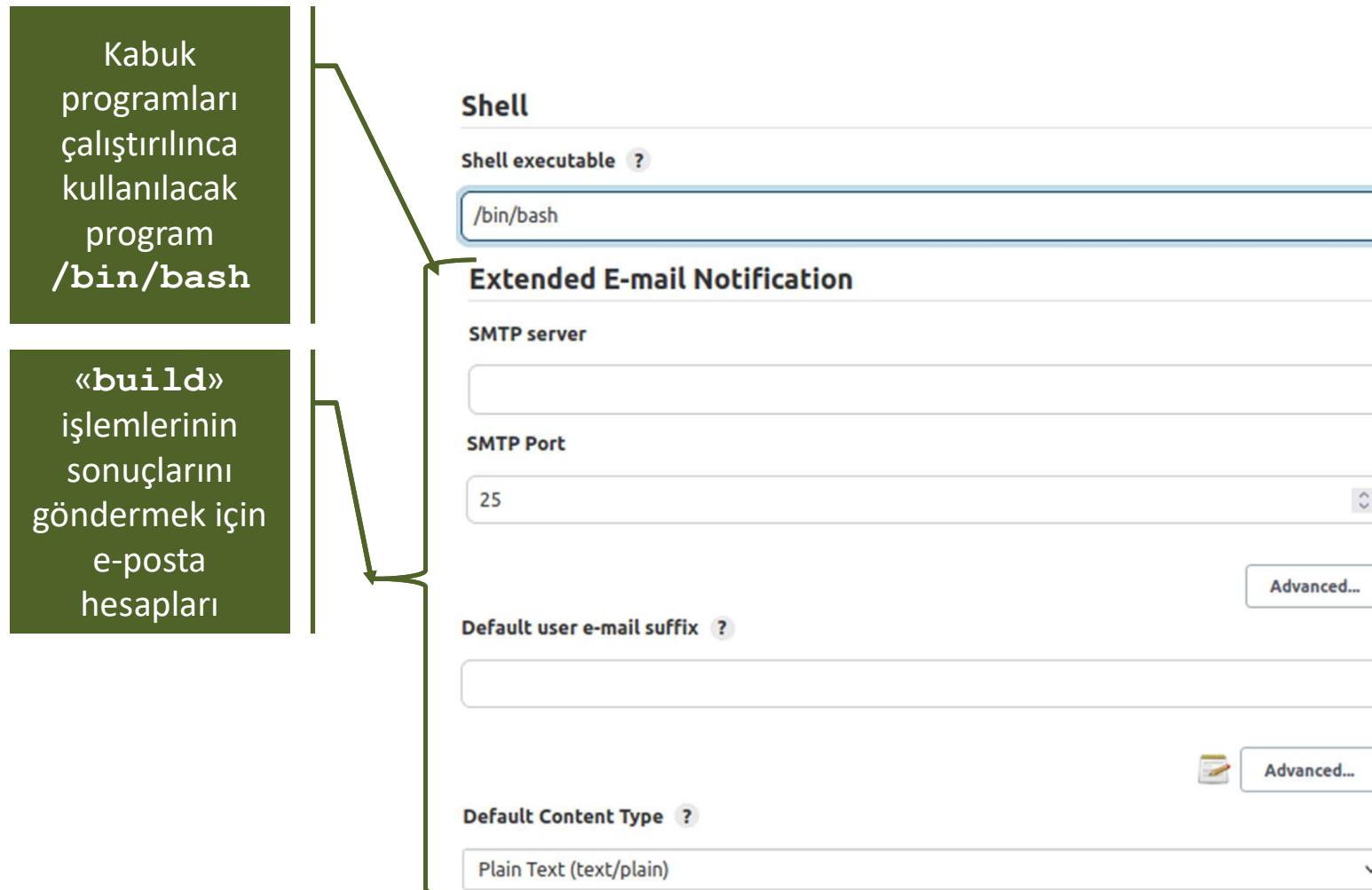
Hide credential usage in job output ?

Disable performance enhancements ?

Preserve second fetch during checkout ?

Add git tag action to jobs ?

# «Configure System»



# Güvenlik

- «Configure Global Security» bölümü, Jenkins'in güvenlik politikalarının yapılandırılmasını sağlar.

**Manage Jenkins**

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

**System Configuration**

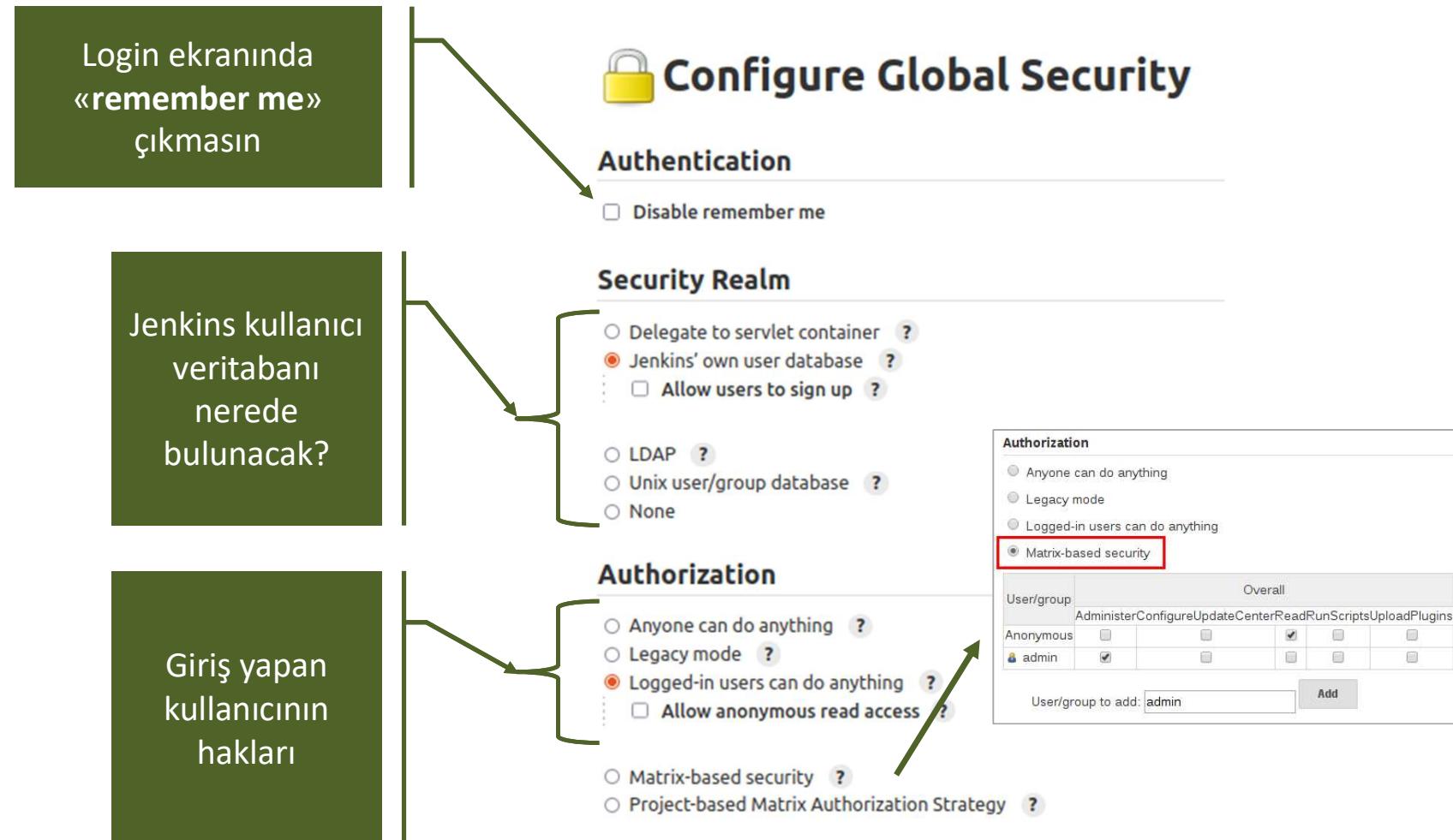
-  **Configure System**  
Configure global settings and paths.
-  **Global Tool Configuration**  
Configure tools, their locations and automatic installers.
-  **Manage Plugins**  
Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
-  **Manage Nodes and Clouds**  
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.

**Security**

-  **Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.
-  **Configure Credential Providers**  
Configure the credential providers and types
-  **Manage Credentials**  
Configure credentials
-  **Manage Users**  
Create/delete/modify users that can log in to this Jenkins

TIKLA 

# Genel Güvenlik

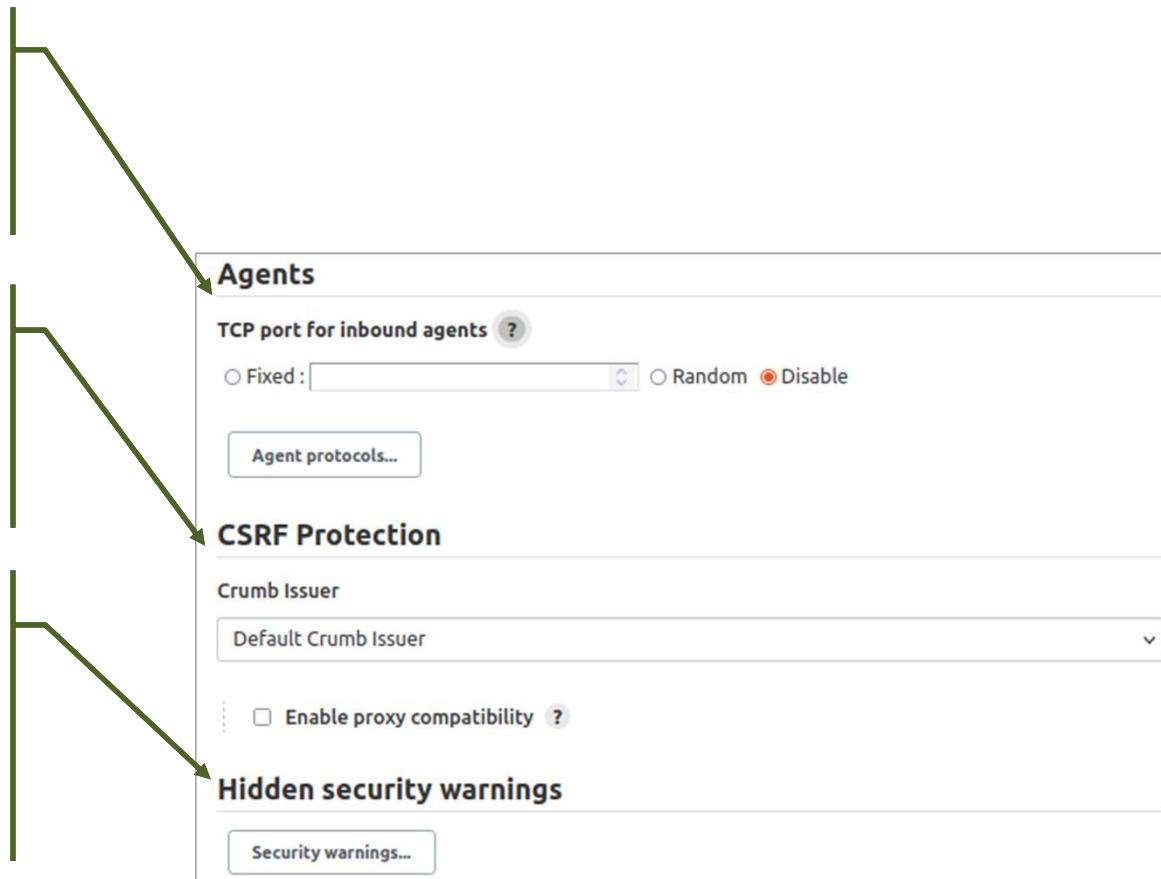


# Genel Güvenlik

JNLP protokolüyle  
Jenkins'e TCP ile  
uzaktan bağlanma

CSRF sahtekarlığına  
karşı «**token**»  
 önlemi alınacak mı?

Kullanılmayan  
bileşenlerle ilgili  
genel güvenlik  
uyarıları varsa  
bunları gizle



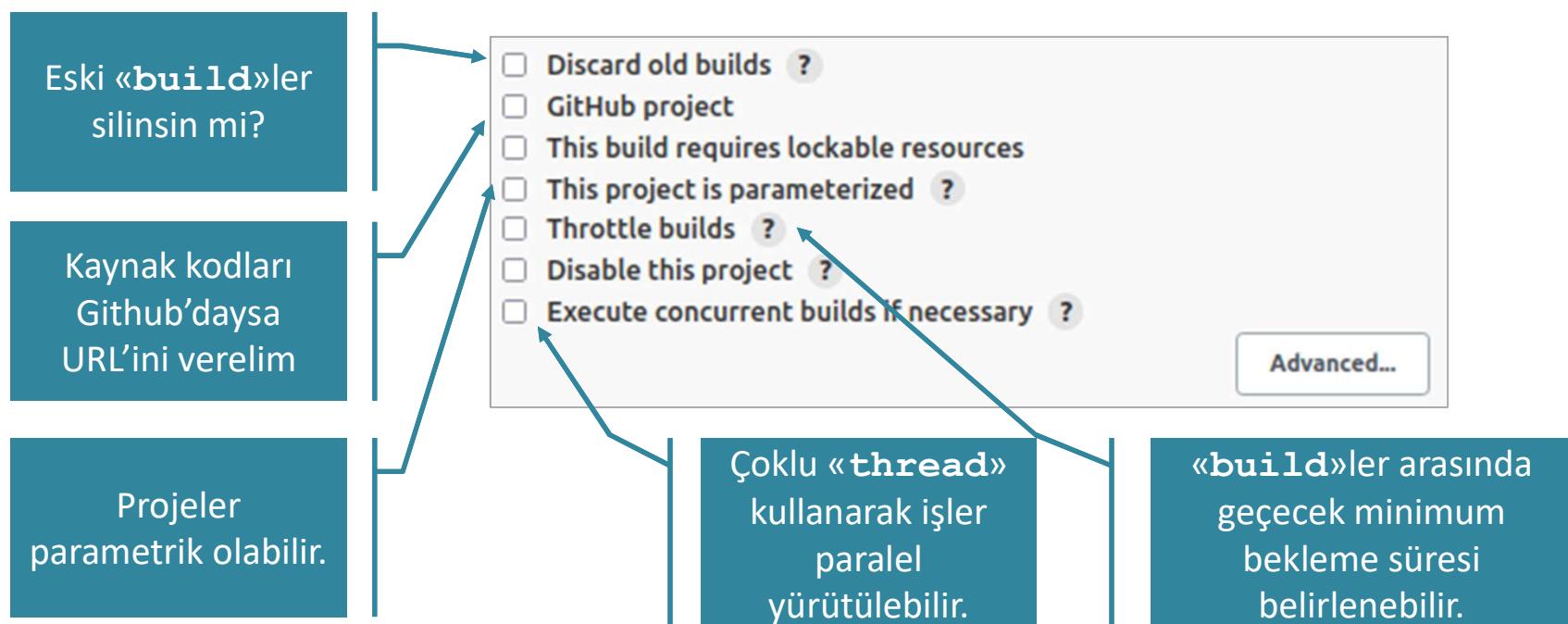
<https://www.jenkins.io/doc/book/security/>

<https://www.jenkins.io/doc/book/security/csrf-protection/>

# «FREESTYLE PROJECT»

# «Freestyle Project»

- «build» işlemi için kullanılabilecek bir proje türüdür.
- Proje oluşturulurken, çok sayıda parametre sayesinde yapılacak iş özelleştirilebilir.



# «Freestyle Project»

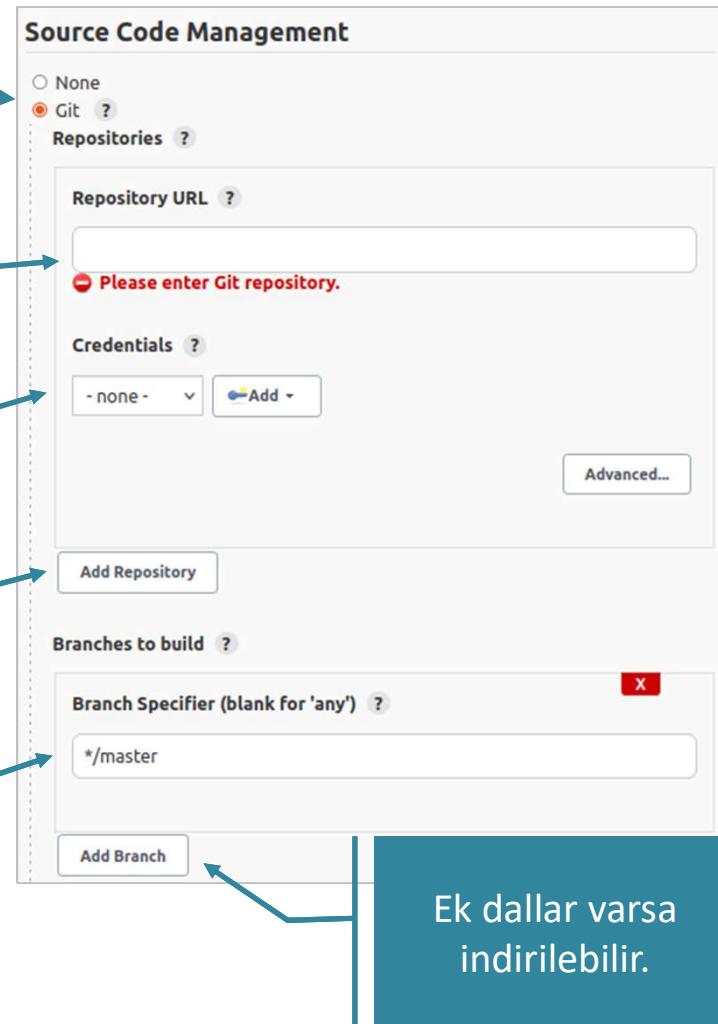
Kaynak kodları  
nereden  
indirilecek?

Git sunucudan  
hangi URL ile  
alacak?

Kullanıcı adı/şifre  
(proje gizliyse)

Birden fazla git  
deposu eklenebilir.

Git'teki hangi  
**«branch»**ten  
indirilecek?



# «Freestyle Project»

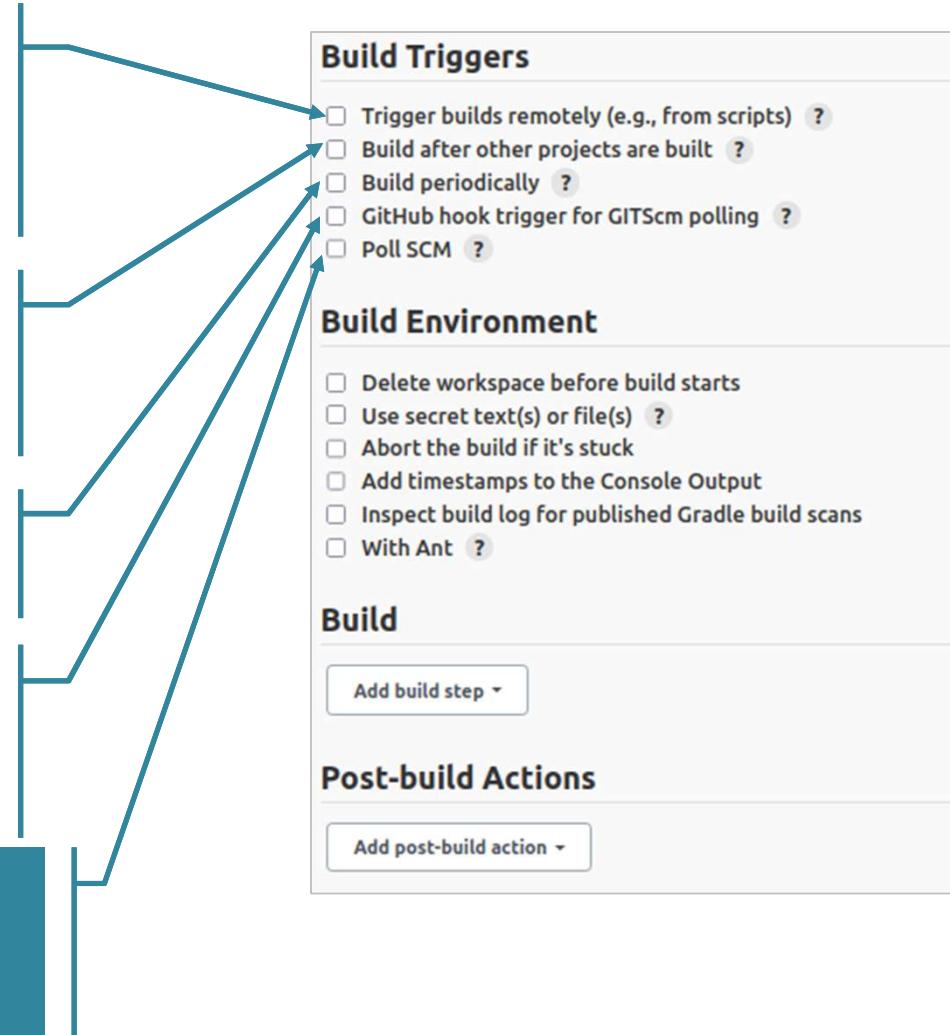
«build» emri dışarıdan verilebilir («webhook» , yeni «commit», pluginler)

Farklı bir projedeki «build», bu projenin oluşturulmasını tetikleyebilir.

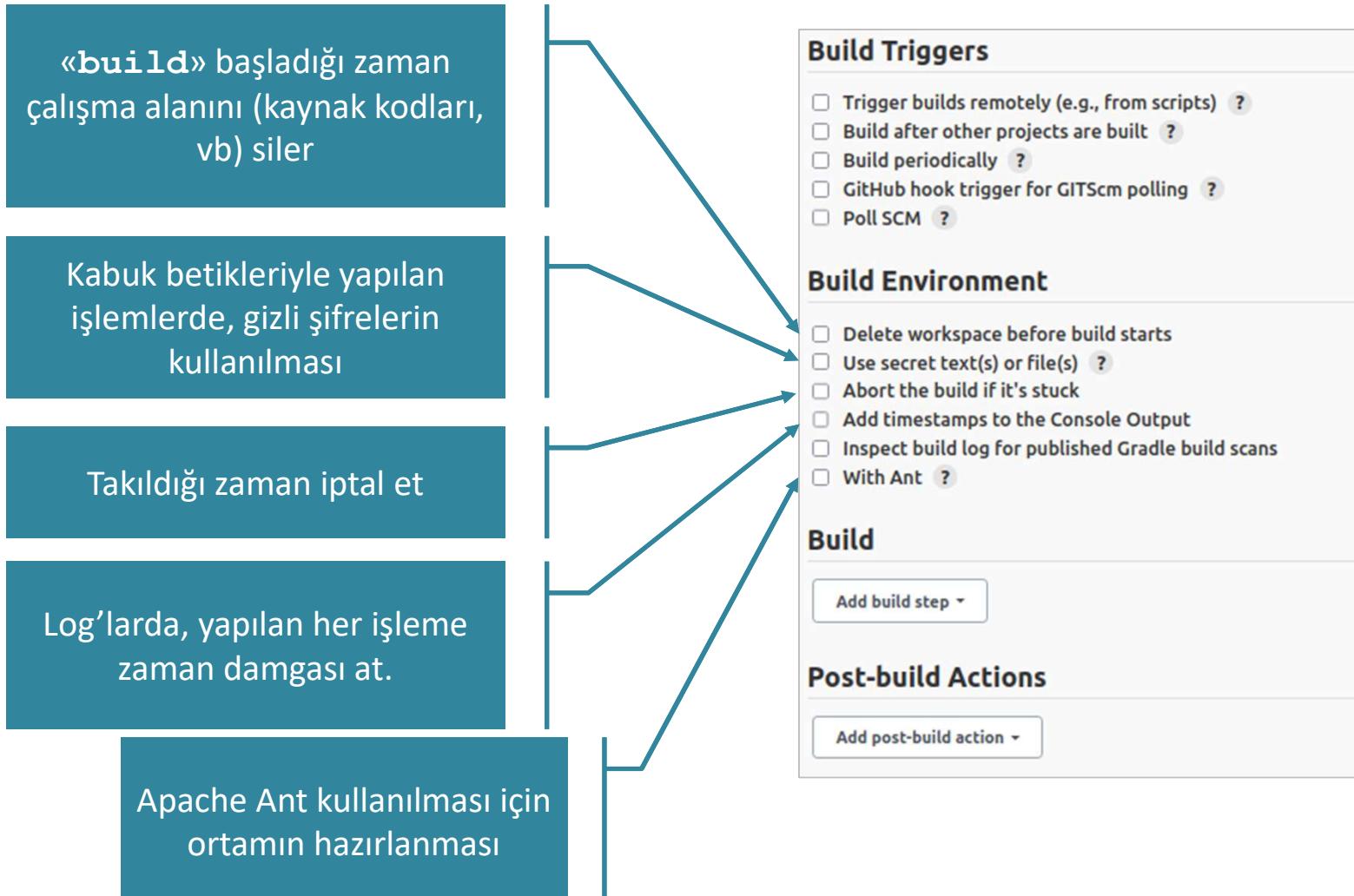
Periyodik olarak oluşturulabilir

Github «push hook»

Kaynak kodu deposu aralarla yoklanabilir.



# «Freestyle Project»



# «Freestyle Project»

«**build**» başladığı zaman  
yapılacak adımı tanımla

1. Windows betiği
2. Kabuk betiği
3. Ant
4. Gradle betiği
5. Maven
6. Zaman aşımında çalışma
7. Github'lara mesaj ekleme

«**build**» tamamlandıktan sonra  
alınacak aksiyonlar

## Build Triggers

- Trigger builds remotely (e.g., from scripts) ?
- Build after other projects are built ?
- Build periodically ?
- GitHub hook trigger for GITScm polling ?
- Poll SCM ?

## Build Environment

- Delete workspace before build starts
- Use secret text(s) or file(s) ?
- Abort the build if it's stuck
- Add timestamps to the Console Output
- Inspect build log for published Gradle build scans
- With Ant ?

## Build

Add build step ▾

Execute Windows batch command

Execute shell

Invoke Ant

Invoke Gradle script

Invoke top-level Maven targets

Run with timeout

Set build status to "pending" on GitHub commit

## Post-build Actions

Add post-build action ▾

# Apache Ant nedir?

- Java projelerinin «derlenmesinde» kullanılan ve «**make**» komutuna benzer bir araçtır.
- «**ant**» XML veri yapısını kullanır (**build.xml**)

```
<project>

    <target name="clean">
        <delete dir="build"/>
    </target>

    <target name="compile">
        <mkdir dir="build/classes"/>
        <javac srcdir="src" destdir="build/classes"/>
    </target>

    <target name="jar">
        <mkdir dir="build/jar"/>
        <jar destfile="build/jar/HelloWorld.jar" basedir="build/classes">
            <manifest>
                <attribute name="Main-Class" value="oata.HelloWorld"/>
            </manifest>
        </jar>
    </target>

    <target name="run">
        <java jar="build/jar/HelloWorld.jar" fork="true"/>
    </target>

</project>
```

```
package oata;

public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

ant compile  
ant jar  
ant run

# Apache Groovy nedir?

- Groovy Java'ya benzer nesneye dayalı bir programlama dilidir.
- Hem statik (bytecode'a dönüştürülebilir) hem de dinamik (betik) özellikleri vardır.
- 2003'te geliştirilmeye başlandı ve ilk sürümü 2007'de çıkarıldı. İlk önceleri betik programlama diliyken 2012'den itibaren bytecode'a dönüştürülebilmektedir.
- Betik program olarak çalıştırılabilmektedir. Jenkins'teki kullanım amacı budur.

# Apache Groovy

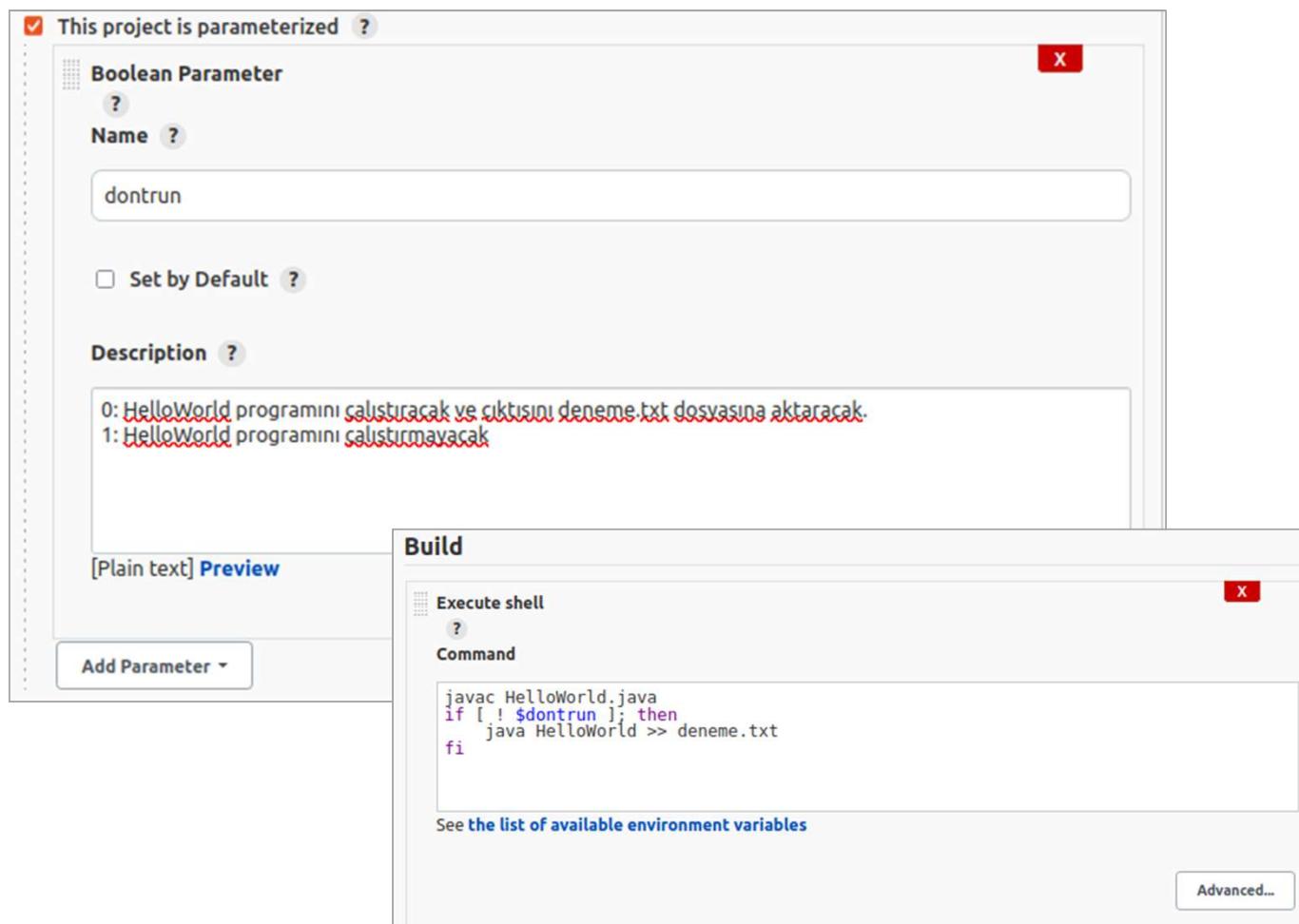
- Groovy yüklemek için  
`sudo apt install groovy`
- `JAVA_HOME` çevre değişikene JAVA JDK dizin adresinin atanması gereklidir.  
`export JAVA_HOME=/usr/lib/jvm/java-1.11.0-openjdk-amd64`
- `PATH` değişkenine eklenmeli  
`export PATH=$JAVA_HOME/bin:$PATH`
- Basit bir program yazılarak  
`println "Hello World"`
- Groovy'de çalıştırılabilir  
`groovy deneme.groovy`

# ÖRNEK – PARAMETRELİ PROJE

# Parametreli Proje Örneği

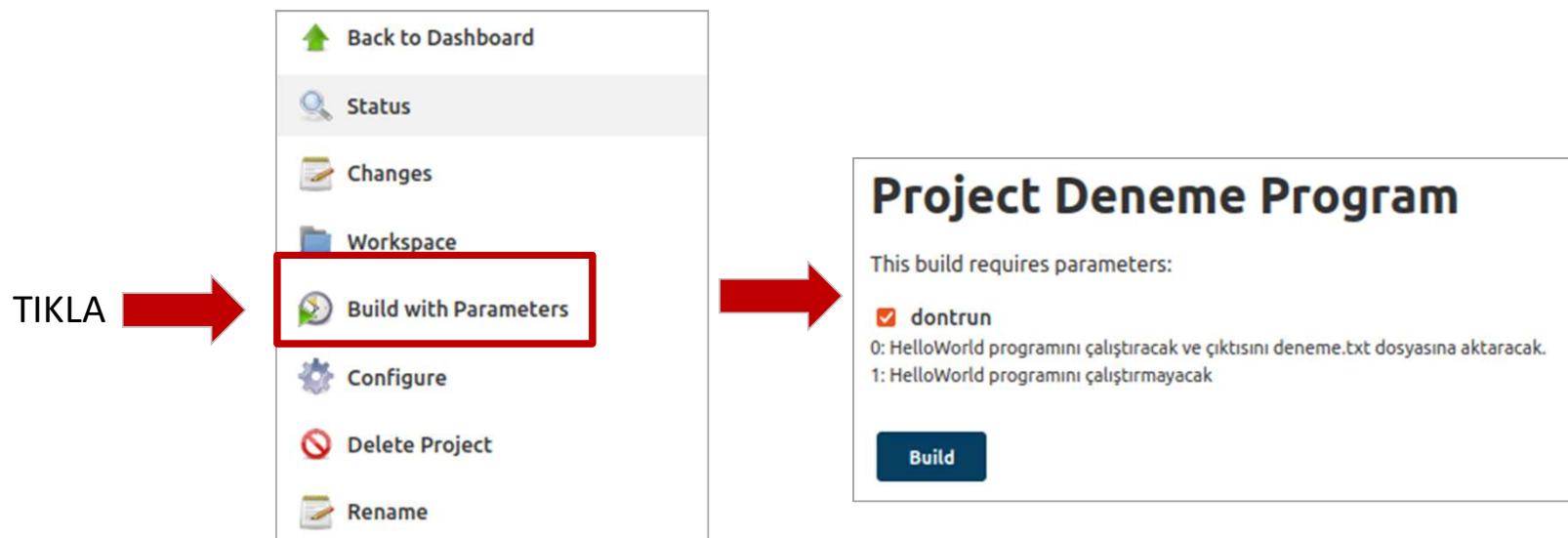
- Bazı projelerde, parametrelere göre «build» işlemini yapmak gereklili olabilir.
- Projeye parametre eklendiğinde, bu parametre çevre değişkeni olarak betik programlarda kullanılabilir.
- «**build**» işlemini başlatınca, Jenkins parametre değerlerini kullanıcıya sormaktadır.
- Bu örnekte, Bash Kabuk betiği kullanacağız ve parametre olarak Boolean türünde «**dontrun**» adlı parametreyi projeye ekleyeceğiz.

# Parametre Ekleme

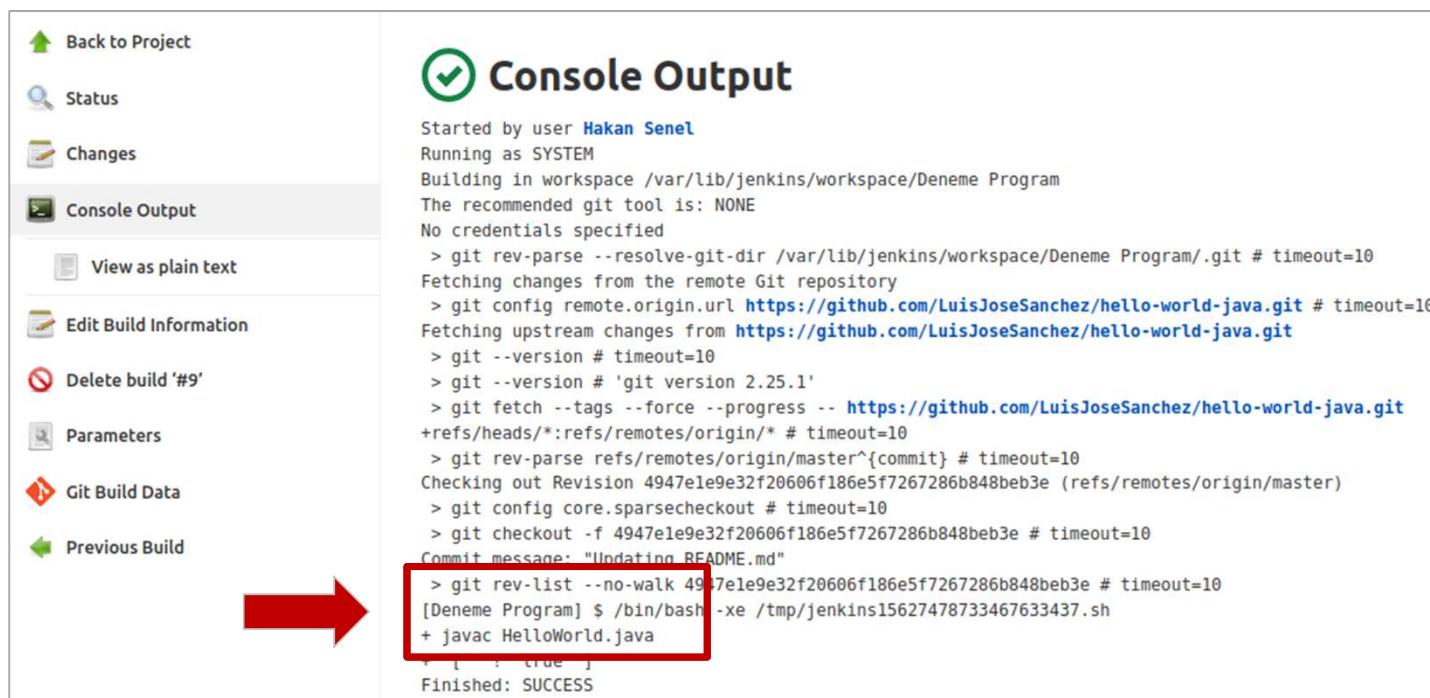


# «Build with Parameters»

- Soldaki menüden «Build with Parameters» opsiyonu seçilerek parametrenin girilmesinin istediği «build» ekranı görülebilir.



# Konsol Çıktısı

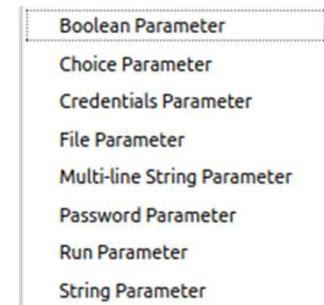


The screenshot shows the Jenkins 'Console Output' page for a build named 'Deneme Program'. The left sidebar lists various build actions: Back to Project, Status, Changes, **Console Output** (selected), View as plain text, Edit Build Information, Delete build '#9', Parameters, Git Build Data, and Previous Build. A large red arrow points from the sidebar towards the bottom of the main content area. The main content area displays the console output log, which includes the following text:

```
Started by user Hakan Senel
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/Deneme Program
The recommended git tool is: NONE
No credentials specified
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/Deneme Program/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/LuisJoseSanchez/hello-world-java.git # timeout=10
Fetching upstream changes from https://github.com/LuisJoseSanchez/hello-world-java.git
> git --version # timeout=10
> git --version # 'git version 2.25.1'
> git fetch --tags --force --progress -- https://github.com/LuisJoseSanchez/hello-world-java.git
+refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git config core.sparsecheckout # timeout=10
> git checkout -f 4947ele9e32f20606f186e5f7267286b848beb3e # timeout=10
Commit message: "Updating README.md"
> git rev-list --no-walk 4947ele9e32f20606f186e5f7267286b848beb3e # timeout=10
[Deneme Program] $ /bin/bash -xe /tmp/jenkins15627478733467633437.sh
+ javac HelloWorld.java
+ [output]
Finished: SUCCESS
```

# Parametre Türleri

- Jenkins'te parametreli «**build**» için farklı tür parametreler sağlanabilir:
  - **Boolean**: Az önce kullandığımız 0 veya 1 olan parametrelerdir.
  - **Choice**: parametre girilirken her satır bir tane olmak üzere seçenekler de girilir ve «**build with parameters**» denilince bu seçeneklerden birini seçmeniz istenir.
  - **File parameter**: parametre olarak bir dosya ismi girilir ve «**build**» edilmeden önce konağın (host) dosya sisteminden bir dosya girilmesi istenir. Bu dosya parametrede belirtilen isimde proje «**workspace**»sine yüklenir.
  - **Multi-line string**: Çok satırlı string
  - **String**: metin parametresi



# BİLDİRİM

# Bildirim

- Süreklilik arz eden CI süreçlerinin en önemli işlerinden biri, her «**build**» işleminden sonra, sonuçları e-mail’le veya başka bir kanal üzerinden göndermektir.
- Jenkins, her «**build**» sonucunda logların gönderilmesini sağlayan e-posta bildirimi, «**Manage Jenkins : Configure System**» bölümünde belirtilebilmektedir.
- «**Use SMTP Authentication**» seçilirse, SMTP sunucusu için kullanıcı adı ve şifre girilebilir.

**E-mail Notification**

SMTP server  
smtp.gmail.com

Default user e-mail suffix ?

Use SMTP Authentication ?  
 Use SSL ?

Use TLS

SMTP Port ?

Reply-To Address

Charset  
UTF-8

# Bildirim

- Jenkins, farklı bildirim yöntemlerini plugin'ler üzerinden desteklemektedir.
- «**Manage Plugins**» bölümünden, «**Search**» kısmından «**notification**» yazarak bildirim imkanı sağlayan eklentiler görüntülenebilir.

The screenshot shows the Jenkins Plugin Manager interface. At the top, there is a search bar containing the text "notification". Below the search bar, there are tabs: "Updates", "Available" (which is selected), "Installed", and "Advanced". The main area displays a table with two rows of plugin information. The columns are "Install", "Name", and "Released".

Install	Name	Released
<input type="checkbox"/>	<b>Notification</b> 1.15 <small>Build Notifiers</small> This plugin from <a href="#">Tikal Knowledge</a> allows sending running Jobs status notifications.	8 mo 18 days ago
<input type="checkbox"/>	<b>Slack Notification</b> 608.v19e3b_44b_b_9ff <small>slack Build Notifiers</small> Integrates Jenkins with Slack, allows publishing build statuses, messages and files to Slack channels.	14 days ago

# Faydalı Bildirim eklentileri

- **SMS Notification:** SMS üzerinden bildirim imkanı sağlamaktadır.
- **instant-messaging:** Jabber üzerinden bildirim
- **Google Chat Notification:** Google Chat üzerinden bildirim göndermek
- **Notify.Events:** Telegram, MS Teams, Jabber üzerinden bildirim yapmak
- **Zoom:** Zoom'da bir kanala bildirim yapılması



# BUILD AGENT

## «Build Agent» oluşturma

- Şu ana kadar yaptığımız örneklerde, «**build**» işlerini Jenkins'in bulunduğu konakta gerçekleştirdik.
- Jenkins, otomasyon sistemi olarak «**build**» işlemlerini farklı makinelerde «**node**» oluşturarak yapabilmektedir.
- Burada yapacağımız örnekte, bilgisayarımız üzerinde iki farklı sanal makineyi farklı ortamlarda çalıştıracağız:
  - **VMware üzerinde** (veya Virtual Box) Ubuntu 20.04 (Jenkins sunucu burada çalışacak). **Master Jenkins** düğümüdür.
  - **WSL2 üzerinde** Ubuntu (**node** olarak çalışacak ve Jenkins sunucusu için «**build**» işini yapacak. (**Jenkins agent**)

# WSL2'de SSH kurulumu

- Derleme düğümü olarak çalışacak WSL2'de OpenSSH'ın kurulması gereklidir. Bazı durumlarda WSL2'de OpenSSH kurulu halde gelmektedir. Ancak bazı hataları engellemek için aşağıdaki komutu kullanarak, kurulu olan OpenSSH sunucusunu sistemden kaldırmak gerekmektedir:  
`$ sudo apt remove --purge openssh-server`
- Ardından OpenSSH sunucusunu kuralım (kurulumda Host-Key'ler de `/etc/ssh` dizini içinde oluşturulacaktır)  
`$ sudo apt install openssh-server`
- `/etc/ssh/sshd_config` dosyası, SSH daemon'un yapılandırma dosyasıdır.
- Bu dosyaya bazı değişikliklerin yapılması gereklidir.
- Şu satırı ekleyin veya varsa «**no**» ifadesi yerine «**yes**» yazın.  
`PasswordAuthentication yes`
- «**sudo service ssh restart**» diyerek yapılan yapılandırmanın uygulanmasını sağlayın.

# WSL2'deki **jenkins** kullanıcısı

- WSL2 üzerinde (**agent**) Jenkins sunucusunun bağlanarak «**build**» işlemi yapacağı bir dizin oluşturmak için, **sudo** komutuyla **/var/lib/jenkins** dizini oluşturma gereklidir:  
**sudo mkdir /var/lib/jenkins**
- Jenkins kullanıcısını sisteme ekleyerek **/var/lib/jenkins** dizinin ev dizini olmasını sağlayacağız  
**\$ sudo useradd -d /var/lib/jenkins -s /bin/bash jenkins**  
**\$ sudo chown jenkins:jenkins /var/lib/jenkins**
- Jenkins için kullanıcı şifresini unutmayın.

# WSL2'deki jenkins kullanıcısı

- WSL2 üzerinde (agent) «**sudo -i -u jenkins**» komutuyla **jenkins** kullanıcıı olarak login olacağız.
- «**mkdir .ssh**» komutuyla dizin oluşturacağız ve «**nano .ssh/authorized\_keys**» yazarak, bu dosyanın içine VMware'deki yani Jenkins sunucusunun çalıştığı sunucudaki açık (**~/ .ssh/id\_rsa.pub**) anahtarını WSL2'de az önce nano ile açtığımız dosya olan **~jenkins/.ssh/authorized\_keys** dosyasına kopyalayacağız.
- Ancak VMware'deki makinede (master) henüz **id\_rsa.pub** dosyası oluşturulmamış olabilir. Bunu oluşturmak için **ssh-keygen** programını kullanacağız.

# VMware (master node)

- VMware'de (Jenkins sunucusunun çalıştığı), «**sudo -i -u jenkins**» diyerek, **jenkins** kullanıcısı olarak login olacağız (Bu kullanıcı Jenkins kurulumunda zaten oluşturulmuştur)
- Ardından, «**ssh-keygen**» komutuyla **.ssh** dizini içinde açık/gizli anahtarları oluşturacağız.

Jenkins'in kurulu  
olduğu konaktaki  
**jenkins**  
kullanıcısının kabuğu  
(VMware)

NOT: Jenkins'in kurulu  
olduğu konaktaki **jenkins**  
kullanıcısının ev dizini de  
**/var/lib/jenkins**

```
adminpc@ubuntu:~$ sudo -i -u jenkins
[sudo] password for adminpc:
jenkins@ubuntu:~$ ls .ssh
ls: cannot access '.ssh': No such file or directory
jenkins@ubuntu:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
Created directory '/var/lib/jenkins/.ssh'.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:x0ZKsTy7lfZ1MPXkoD2+BxN/vvXCqlM70Reu/j1R4Zg jenkins@ubuntu
The key's randomart image is:
+---[RSA 3072]---+
|   .   o|
| . o   o * .|
| = . . X + |
| . * . E 0 .|
| S B   . * * |
| * .o..0o|
| . ..+o.=|
| . o..o=|
|       .oo++..+|
+---[SHA256]---+
```

**id\_rsa.pub**: açık anahtar (agent'taki  
jenkins kullanıcısının  
\$HOME/.ssh/authorized\_keys  
dosyasına kopyalanacak.

# VMware (master)

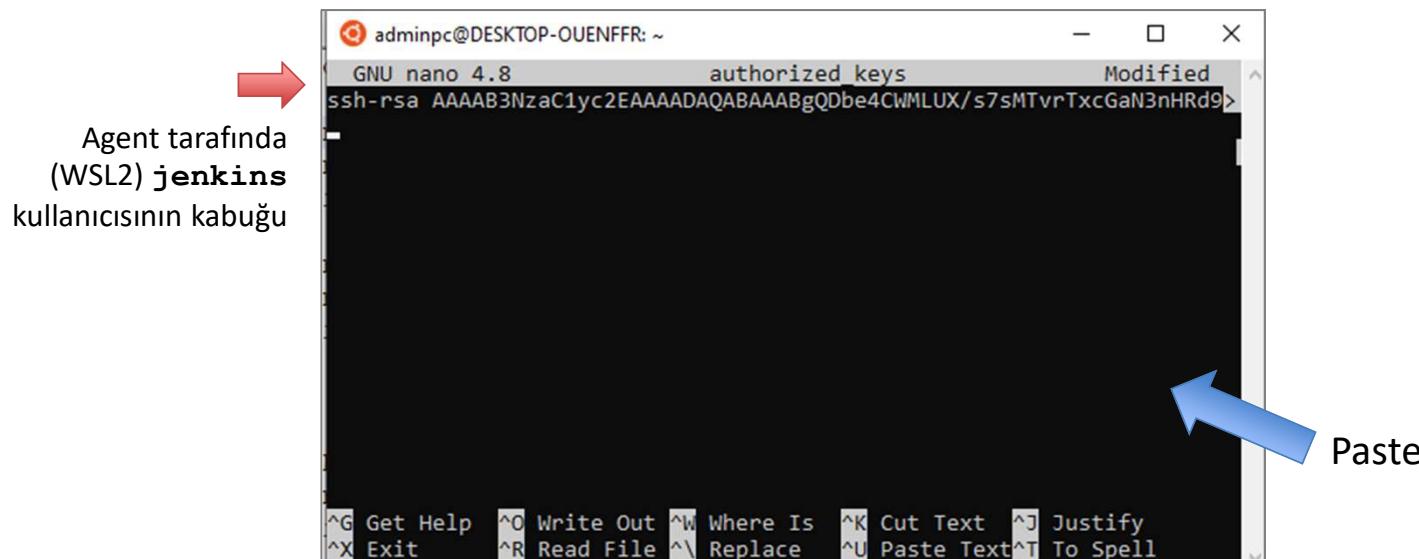
- VMware tarafında (Jenkins'in kurulu olduğu konakta) «**cat .ssh/id\_rsa.pub**» ile açık anahtarı konsolda gösterip tümünü seçtikten sonra, sağ klikle «**Copy**» yapacağız.
  - Çünkü bunu **agent** tarafında (WSL2) aşağıdaki dosyaya «**Paste**» etmemiz gereklili:
- /var/lib/jenkins/.ssh/authorized\_keys**

```
jenkins@ubuntu:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAQABAAQgQDbe4CWMLUX/s7sMTvrTxcGaN3nHRd9cmkPbU1M95bRQ07Kb0Hmz9ITDlkN0/5UAWrTz766
eJlsSsm3rvb0SmBTLnGxLtjl/INnoKBe65pDDctdGmkbQR4U7+yiAHesZGYDfZ4TI/4QXU7U3uaQypjIzs+QdxERpvVpszfpbIlxsAYiur8
v/OC9AVkkruVgIQgF79K/y07e7YP0IW4s40oLv0JBBoZKpxKwNdQateszylQ80woDEuQmufA5lOMEUwcjRC72+x0u09vPgEVT8VCz9750L1+c
0WBsctv4UHGHrQV9Z7FhCcbKB0eoreQXOW6+5MRFXwikr+GhR2ho6Sat+epdUJGDRSu58u1TGgN03xPp/TSFFREOBfjNLa00HUzKpovYx32E
1ba9y0FuNyifFNQEieebdKyp2dX75cqZjs6fe91ccA6E5lFpn4Mdj7q9Syx5odB0egdiS02PJ77x624973NWu5ru/24qSMhvWv/A0NX3Vgeu
d0I4IzIdc9c= jenkins@ubuntu
```



# WSL2'deki jenkins kullanıcısı

- WSL2 üzerinde «**nano .ssh/authorized\_keys**» yazdıktan sonra «**Paste**» yapacağız.

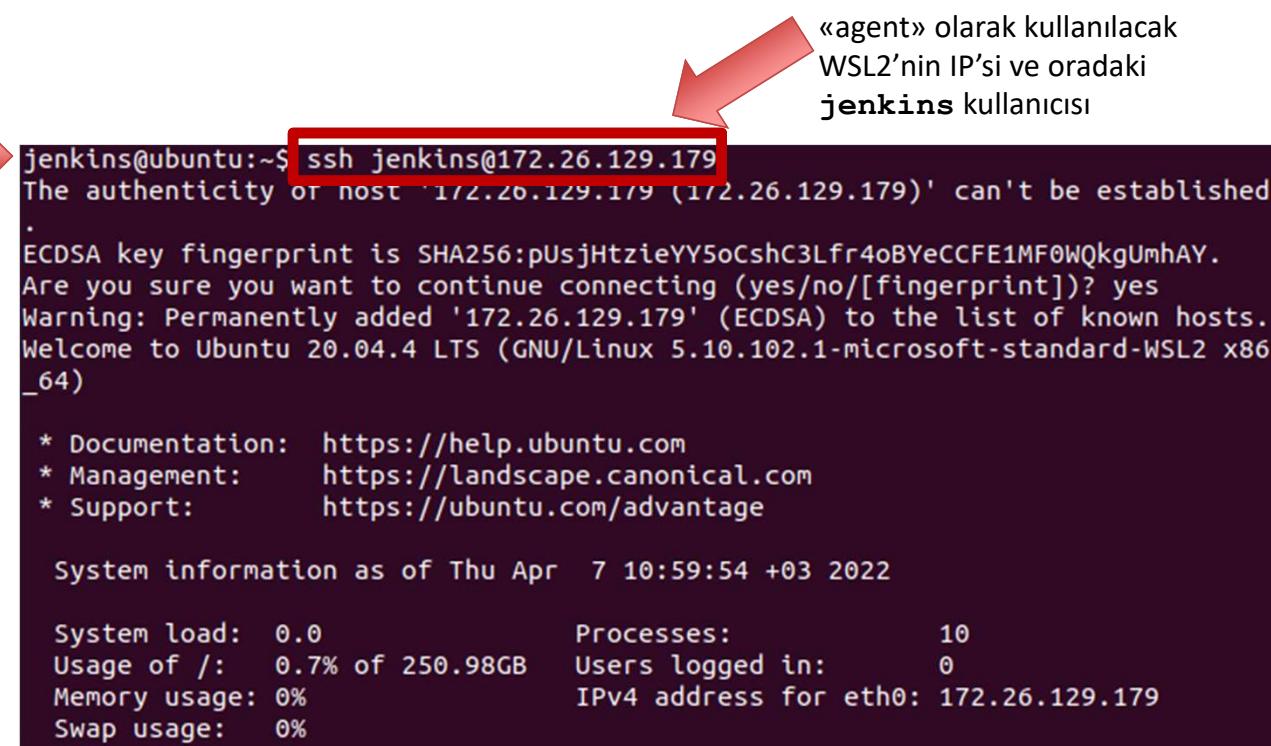


- **Neyi yaptık?** Master'daki jenkins kullanıcısının **.ssh/id\_rsa.pub** dosyasının içindekileri agent'taki **/var/lib/jenkins/.ssh/authorized\_keys** dosyasına aktardık.

# VMware Ubuntu

- VMware Ubuntu'da (master), «**sudo -i -u jenkins**» komutuyla jenkins kullanıcıı olarak login olduktan sonra «**ssh jenkins@agent-IP-no**» yazarak, giriş yaptığımızı göreceğiz.

Jenkins'in kurulu olduğu konaktaki  
jenkins  
kullanıcısının kabuğu  
(VMware)



The screenshot shows a terminal window on a host system (Ubuntu) connecting via SSH to a guest system (Ubuntu 20.04.4 LTS running in VMware). A red arrow points from the left towards the terminal window, and another red arrow points from the right towards the IP address in the command line.

```

jenkins@ubuntu:~$ ssh jenkins@172.26.129.179
The authenticity of host '172.26.129.179 (172.26.129.179)' can't be established.
ECDSA key fingerprint is SHA256:pUsjHtzieYY5oCshC3Lfr4oBYeCCFE1MF0WQkgUmhAY.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.26.129.179' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.4 LTS (GNU/Linux 5.10.102.1-microsoft-standard-WSL2 x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

System information as of Thu Apr  7 10:59:54 +03 2022

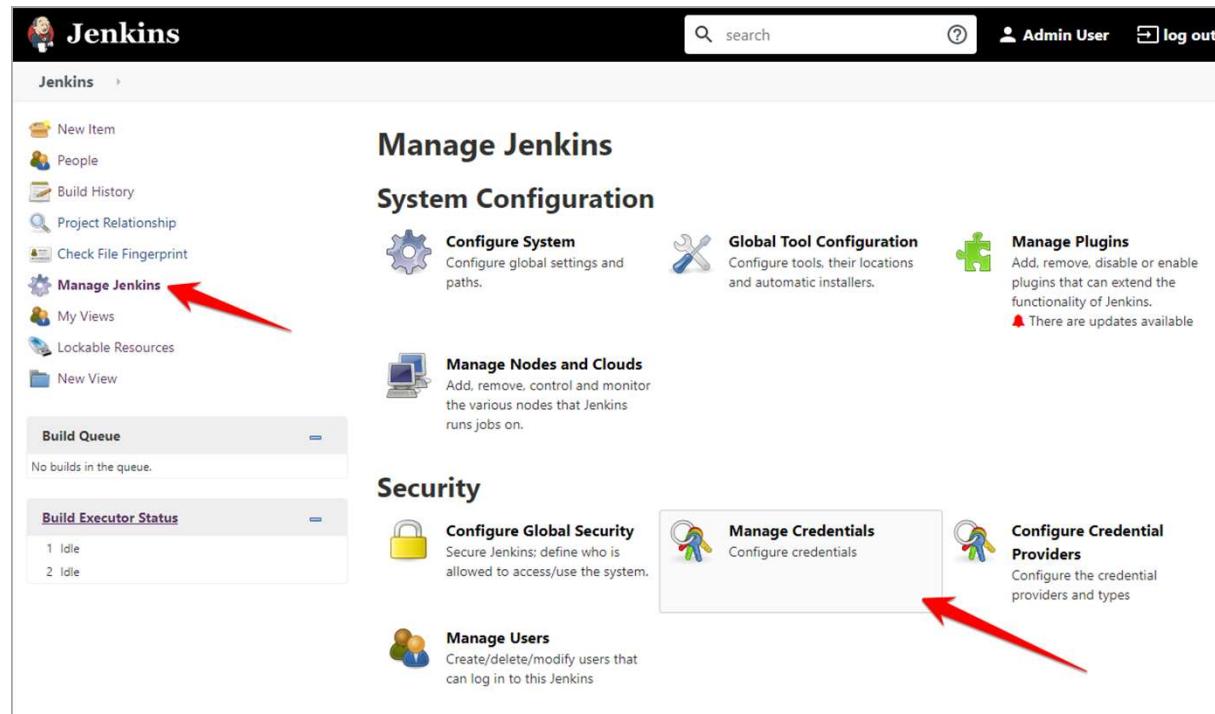
System load:  0.0          Processes:           10
Usage of /:   0.7% of 250.98GB  Users logged in:    0
Memory usage: 0%
Swap usage:   0%          IPv4 address for eth0: 172.26.129.179

```

<https://www.jenkins.io/doc/book/using/using-agents/>

# «manage credential» opsiyonu

- Jenkins dashboard'da «Manage Jenkins» ve ardından «Manage Credential» opsiyonunu seçin.



<https://www.jenkins.io/doc/book/using/using-agents/>

# «manage credential» opsiyonu

- «drop» menüden «Add Credentials» opsiyonu seçmek gereklidir.

The screenshot shows the Jenkins web interface for managing credentials. On the left, there's a sidebar with various links: New Item, People, Build History, Project Relationship, Check File Fingerprint, Manage Jenkins, My Views, Lockable Resources, and New View. The main content area is titled 'Credentials' and displays a table of credential stores. The table has columns for P (Priority), Store (sorted by name), Domains, ID, and Name. It lists one store named 'Jenkins' under the 'Domains' column. At the bottom right of the table, there's a blue button labeled 'Add credentials' with a key icon, which is highlighted by a red arrow. The top right of the screen shows the user is 'Admin User' and includes a 'log out' link.

<https://www.jenkins.io/doc/book/using/using-agents/>

# Jenkins kullanıcıı için SSH bilgileri

- Kind: «SSH Username with private key»
- id: **jenkins**
- description: **jenkins ssh anahtarı**
- username: **jenkins**
- Private Key: «Enter directly» opsiyonu seçilecek ve «Add» butonuna tıklanacaktır. VMware Ubuntu'da (master) `~/.ssh/id_rsa` dosyasından kopyalanan metin «paste» edilecektir.

The screenshot shows the Jenkins Global credentials configuration page. In the 'Private Key' section, there is a red arrow pointing from the 'Enter directly' input field towards a terminal window on the right. The terminal window displays the contents of the copied SSH private key:

```
jenkins@ubuntu:~$ more .ssh/id_rsa
-----BEGIN OPENSSH PRIVATE KEY-----
-----END OPENSSH PRIVATE KEY-----
jenkins@ubuntu:~$
```

A large red arrow points from the 'Paste' button in the terminal window towards the 'Enter New Secret Below' input field at the bottom of the Jenkins page.

# Jenkins «credentials»

- «Global credentials» bölümünde oluşturduğumuz agent'taki (WSL2) jenkins kullanıcısı görülecektir.
- Bundaki sonraki aşamada «Manage Jenkins» bölümünden «Manage Nodes and Cloud» kısmına tıklayacağız ve yeni bir «node» ekleyeceğiz. Oluşturduğumuz «jenkins» bilgilerini kullanacağız.

The screenshot shows the Jenkins Global credentials (unrestricted) page. The navigation bar at the top includes links for Dashboard, Credentials, System, and Global credentials (unrestricted). On the left sidebar, there are links for Back to credential domains and Add Credentials. The main content area has a title 'Global credentials (unrestricted)' with a crown icon. Below it, a subtitle reads 'Credentials that should be available irrespective of domain specification to requirements matching.' A table lists a single credential entry:

ID	Name	Kind	Description
	<b>jenkins</b>	jenkins (jenkins ssh anahtarı)	SSH Username with private key

Below the table, there are icons for S, M, and L sizes, with 'S' being highlighted.

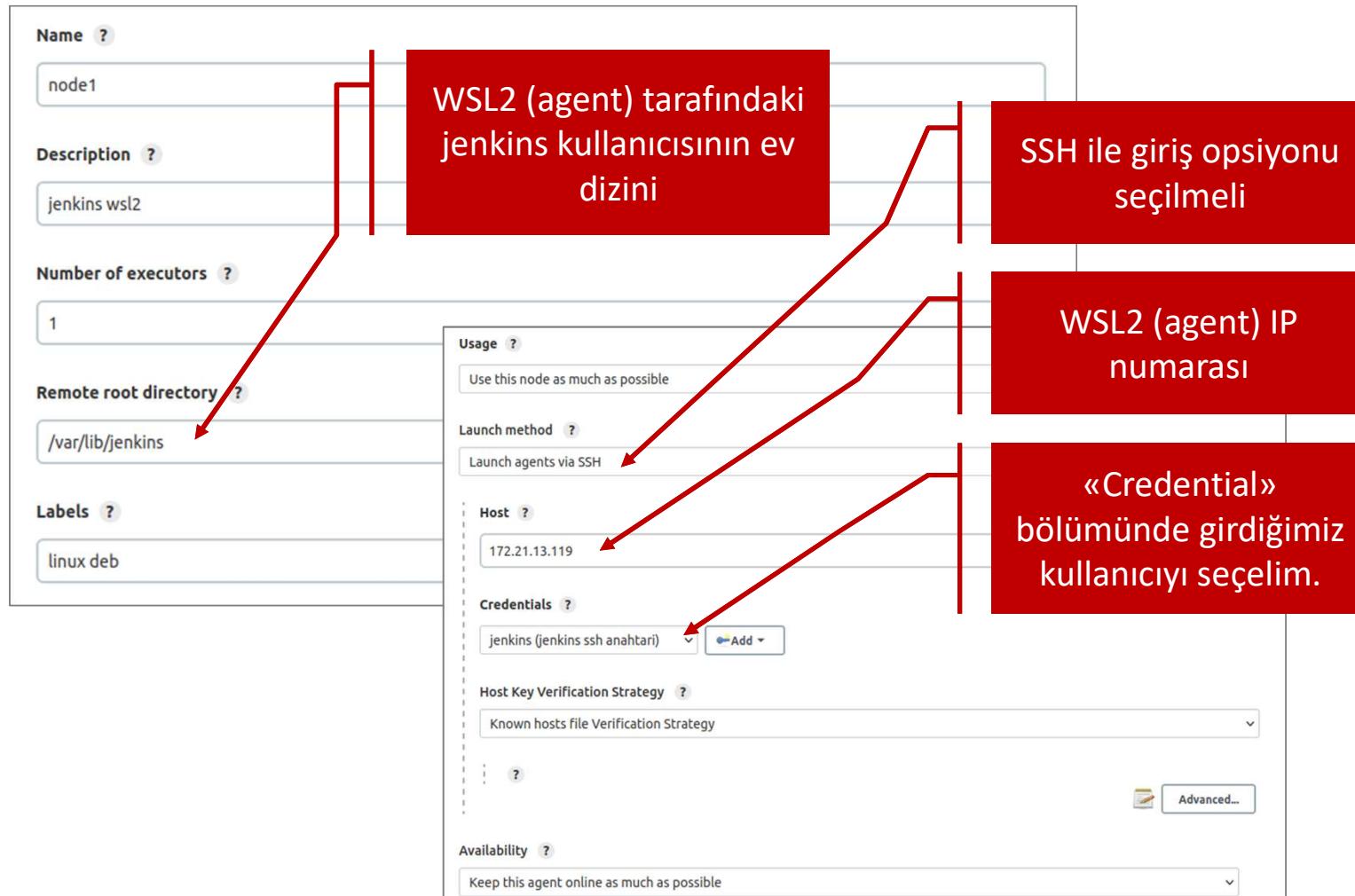
# Düğüm (node) girişi

**WSL2 (agent) tarafından jenkins kullanıcısının ev dizini**

**SSH ile giriş opsyonu seçilmeli**

**WSL2 (agent) IP numarası**

**«Credential» bölümünde girdiğimiz kullanıcıyı seçelim.**

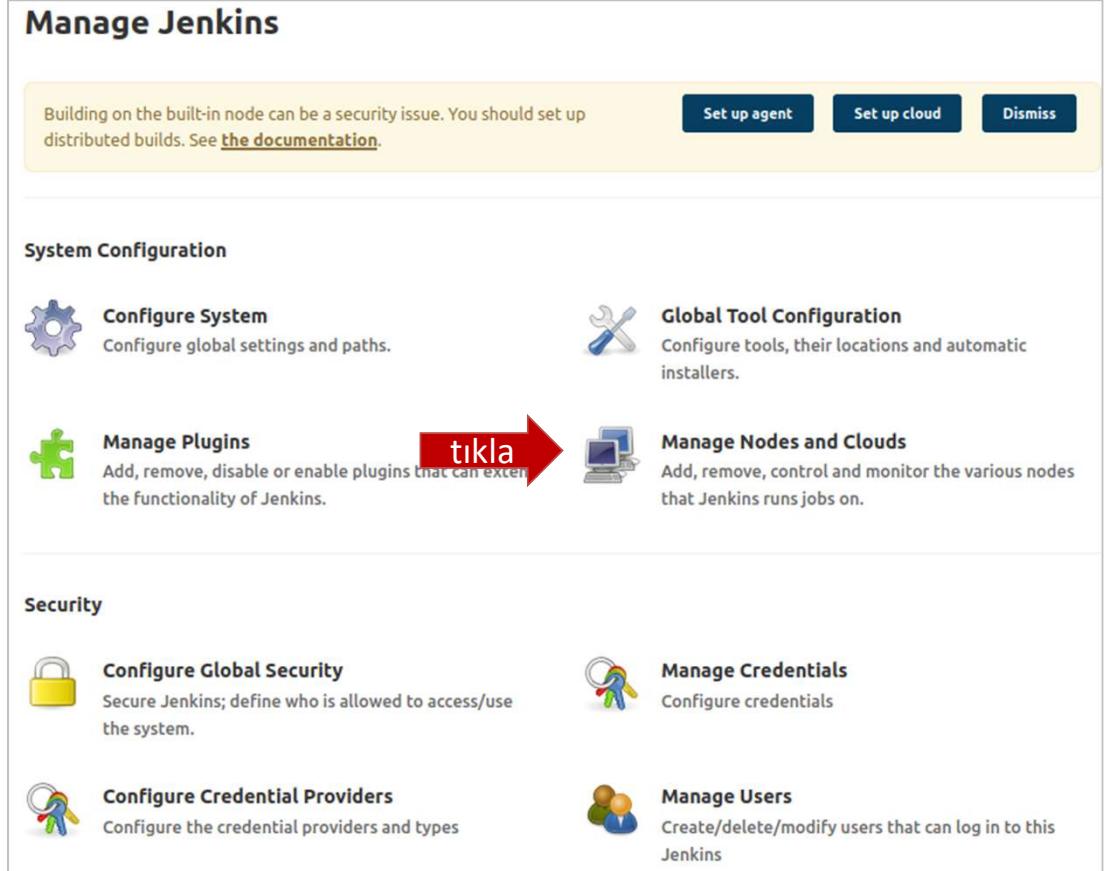


The screenshot shows the Jenkins 'Node Configuration' page for creating a new node named 'node1'. The configuration includes:

- Name:** node1
- Description:** jenkins wsl2
- Number of executors:** 1
- Remote root directory:** /var/lib/jenkins
- Labels:** linux deb
- Usage:** Use this node as much as possible
- Launch method:** Launch agents via SSH
- Host:** 172.21.13.119
- Credentials:** jenkins (jenkins ssh anahtarı) (selected from dropdown)
- Host Key Verification Strategy:** Known hosts file Verification Strategy
- Availability:** Keep this agent online as much as possible

# «Build Agent»

- Şu ana kadar yaptığımız örneklerde, «**build**» işlerini Jenkins'in bulunduğu konakta gerçekleştirdik.
- Jenkins, otomasyon sistemi olarak «**build**» işlemlerini farklı makinelerde «**node**» oluşturarak yaptırabilmektedir.



The screenshot shows the Jenkins 'Manage Jenkins' interface. At the top, there's a message about building on the built-in node being a security issue, with buttons for 'Set up agent', 'Set up cloud', and 'Dismiss'. Below this, the 'System Configuration' section contains links for 'Configure System', 'Global Tool Configuration', 'Manage Plugins', and 'Manage Nodes and Clouds'. A large red arrow points to the 'Manage Plugins' link. The 'Security' section includes 'Configure Global Security', 'Configure Credential Providers', 'Manage Credentials', and 'Manage Users'.

# WSL2 Üzerinde Jenkins Düğümü

- Jenkins, SSH üzerinden buraya bağlanacak.
- SSH için açık ve gizli anahtar çiftini oluşturacağız.  
**\$ ssh-keygen**
- «Enter» butonuna basarak soruları geçebilirsiniz.

```
adminpc@DESKTOP-OUENFFR:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/adminpc/.ssh/id_rsa):
Created directory '/home/adminpc/.ssh'.
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:
SHA256:1Vjq9GXl5WLT2Pu9bOMu/b5XhUgcwCt+NQNnvF8L1e8 adminpc@DESKTOP-OUENFFR
The key's randomart image is:
+---[RSA 3072]---+
| ..+... . |
| o * .. |
| * o... |
| .o. *..o+ |
| .S... ++o* |
| .... o.+E |
| ... o o=o |
| . o o .o+= |
| .o . =B@ |
+---[SHA256]---+
```

# WSL2 üzerinde SSH

- WSL2 Ubuntu üzerinde SSH servisinin başlatılması gereklidir.

```
$ sudo service ssh start
```

```
adminpc@DESKTOP-OUENFFR:~$ sudo service ssh start
 * Starting OpenBSD Secure Shell server sshd [ OK ]
```

- Eğer **sshd** başlatılmadan, Jenkins'te düğüm çalıştırılırsa loglarda «**Connection refused**» mesajı olacaktır.

# SSH Anahtarları

- Jenkins dizininde SSH anahtarlarını saklamak için gizli bir dizin oluşturacağız:

```
$ sudo mkdir /var/lib/jenkins/.ssh
```

- Az önce ev dizininde oluşturduğumuz anahtar çiftini **/var/lib/jenkins/.ssh** dizinine aktaracağız.

```
$ sudo cp ./ssh/id_rsa.pub  
/var/lib/jenkins/.ssh/authorized_keys
```

- Jenkins bir Java programı olduğu için sisteme Java kurulmalıdır. Ana Jenkins sisteminde bulunan Java sürümü olmalıdır :

```
$ sudo apt update
```

```
$ sudo apt install openjdk-11-jdk-headless
```

# Gizli Anahtarın Kopyalanması

- **ssh-keygen** program açık ve gizli iki anahtar üretir.
  - **./.ssh/id\_rsa.pub**: açık anahtar
  - **./.ssh/id\_rsa**: gizli anahtar
- Gizli anahtarı, Jenkins'te düğüm oluşturmak için kullanacağız.
- Bu nedenle, **Jenkins sunucusunun çalıştığı sistemde** gizli anahtarı gösterin ve konsoldan kopyalayın:  
**\$ cat ./ssh/id\_rsa**

# Jenkins : Yeni Düğüm Oluşturma

Düğüm ismi  
(Açıklayıcı bir isim)

Fiziksel sunucuya seçilecek.  
Eğer Docker konteyner ise  
seçilmeyecek

New node

Node name

node1

Type

Permanent Agent

Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.

Create

# Jenkins : Yeni Düğüm

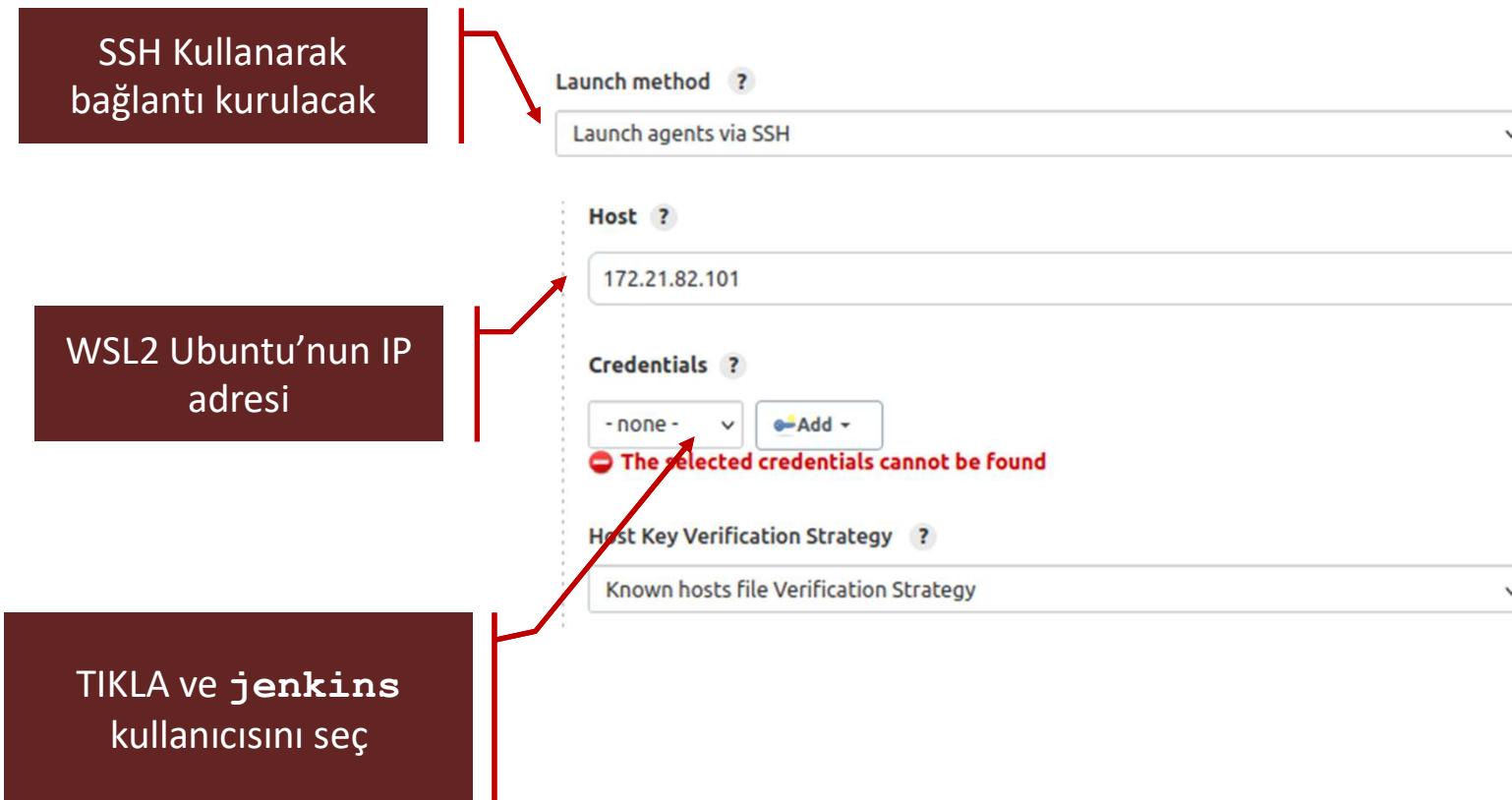
Dizin ismi

Mimariyi gösteren bir etiket



Name ?	<input type="text" value="node1"/>
Description ?	<input type="text" value="WSL2 Ubuntu"/>
Number of executors ?	<input type="text" value="2"/>
Remote root directory ?	<input type="text" value="/var/lib/jenkins"/>
Labels ?	<input type="text" value="linux_deb"/>
Usage ?	<input type="text" value="Use this node as much as possible"/>

# Jenkins : Yeni Düğüm



# Düğümün eklenmesi

Seçin

Böyle bırakın

Host ?  
172.21.82.101

Credentials ?  
jenkins (WSL2 kullanıcısı)

Host Key Verification Strategy ?  
Known hosts file Verification Strategy

Availability ?  
Keep this agent online as much as possible

Node Properties

Disable deferred wipeout on this node ?  
 Environment variables  
 Tool Locations

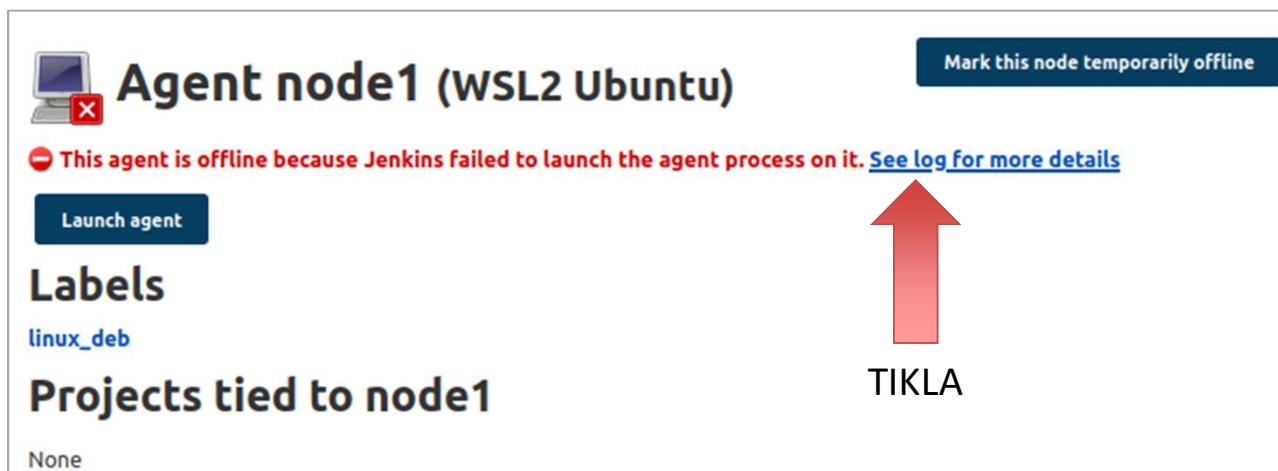
# Yeni Düğüm

- Yeni düğüm eklenene kadar, düğümün ikonu yanıp sönebilir. Bu sistemin kurulduğunu göstermektedir.

Manage nodes and clouds							
S	Name	Architecture	Clock Difference	Free Disk Space	Free Swap Space	Free Temp Space	Response Time
	Built-In Node	Linux (amd64)	In sync	57.88 GB	2.00 GB	57.88 GB	0ms 
	node1		N/A	N/A	N/A	N/A	N/A 
Data obtained		14 sec	14 sec	14 sec	14 sec	14 sec	14 sec

# Jenkins bağlantı sorunları

- Jenkins bağlantı ajanı hata verir durursa veya uzun süre düğüm ikonu yanıp sönerse, ajanın bağlanamadığı anlaşılabılır.
- Hangi hatanın olduğu, düğümün loglarına bakılarak görülmelidir:



# Jenkins – Agent Bağlantısı

Bağlantı kurulduysa karşınıza böyle bir log çıkacaktır:

```
USER=jenkins
_=']'
Checking Java version in the PATH
openjdk version "11.0.14.1" 2022-02-08
OpenJDK Runtime Environment (build 11.0.14.1+1-Ubuntu-0ubuntu1.20.04)
OpenJDK 64-Bit Server VM (build 11.0.14.1+1-Ubuntu-0ubuntu1.20.04, mixed mode, sharing)
[04/07/22 01:06:30] [SSH] Checking java version of /var/lib/jenkins/jdk/bin/java
Couldn't figure out the Java version of /var/lib/jenkins/jdk/bin/java
bash: /var/lib/jenkins/jdk/bin/java: No such file or directory

[04/07/22 01:06:30] [SSH] Checking java version of java
[04/07/22 01:06:30] [SSH] java -version returned 11.0.14.1.
[04/07/22 01:06:31] [SSH] Starting sftp client.
[04/07/22 01:06:31] [SSH] Copying latest remoting.jar...
[04/07/22 01:06:33] [SSH] Copied 1,524,115 bytes.
Expanded the channel window size to 4MB
[04/07/22 01:06:33] [SSH] Starting agent process: cd "/var/lib/jenkins" && java -jar remoting.jar -workDir
/var/lib/jenkins -jar-cache /var/lib/jenkins/remoting/jarCache
Apr 07, 2022 11:06:36 AM org.jenkinsci.remoting.engine.WorkDirManager initializeWorkDir
INFO: Using /var/lib/jenkins/remoting as a remoting work directory
Apr 07, 2022 11:06:36 AM org.jenkinsci.remoting.engine.WorkDirManager setupLogging
INFO: Both error and output logs will be printed to /var/lib/jenkins/remoting
<===[JENKINS REMOTING CAPACITY]==>channel started
Remoting version: 4.13
This is a Unix agent
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by jenkins.slaves.StandardOutputSwapper$ChannelSwapper to constructor
java.io.FileDescriptor(int)
WARNING: Please consider reporting this to the maintainers of jenkins.slaves.StandardOutputSwapper$ChannelSwapper
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
Evacuated stdout
Agent successfully connected and online
```

# **JENKINS'İ DOCKER KONTEYNER OLARAK ÇALIŞTIRMA VE PIPELINE EKLENTİSİ (PIPELINE PLUG-IN)**

# Jenkins'i konteyner olarak çalışma

- Jenkins'in çalıştığı birden fazla Docker imajı bulunmaktadır.
- Tavsiye edilen Jenkins imajı, LTS özelliği olan «**jenkins/jenkins**» imajıdır.
- Fakat bu imajın içinde, Blue Ocean eklentileri bulunmamaktadır ve bu nedenle bu imaj üzerinde değişiklik yapılarak bu eklentilerin yüklenmesi gereklidir.
- Ayrıca, konteyner olarak çalışan Jenkins içinde «**docker**» komutu çalıştırılacaksa, «**jenkins/jenkins**» imajı yanında «**docker:dind**» imajının da konteyner olarak çalıştırılması gereklidir.
- Her iki konteynerin birbirini görebilmesi için «**jenkins**» isimli bir ağın da oluşturulması gereklidir.

# Jenkins'i konteyner olarak çalışma

- İlk olarak «**docker network create jenkins**» diyerek, ağı oluşturuyoruz.
- «**Docker in Docker**» imajını konteyner olarak çalıştırıyoruz.

```
$ docker network create jenkins

$ docker run --name jenkins-docker --rm --detach --privileged --network
jenkins --network-alias docker --env DOCKER_TLS_CERTDIR=/certs --volume
jenkins-docker-certs:/certs/client --volume jenkins-data:/var/jenkins_home
--publish 2376:2376 docker:dind --storage-driver overlay2
```

# Jenkins imajının oluşturulması

- Jenkins imajı üzerine bazı eklemeler yapacağız. Bu nedenle dizinin altında «**Dockerfile**» oluşturalım ve kaydedelim.
- Bu Dockerfile standart bir içeriğe sahip. Bunu değiştirmeliyiz.

```
FROM jenkins/jenkins:2.387.1
USER root
RUN apt-get update && apt-get install -y lsb-release
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
    https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
    signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
    https://download.docker.com/linux/debian \
    $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
```

Bu «Dockerfile» aşağıdaki linkte gösterilen dosya ama biz GNU C Compiler çalıştırmak ve GitHub.com'un Açık anahtarını almak için değiştireceğiz.

- Dockerfile'dan «**docker build -t myjenkins-blueocean:2.387.1-1 .**» komutuyla imaj oluşturalım.

# Jenkins imajının oluşturulması

- GNU C derleyicisi (build-essentials), `curl`, `wget` ve `jq` gibi paketleri kurmamız gerekiyor. Bu nedenle dosya içinde bazı eklemeler yapacağız. Ayrıca, `github.com`'un açık SSH anahtarını (public key) da `.ssh/known_hosts` dosyasına yerlestireceğiz. Bu sahte bir `github.com`'a bağlanmamızı engelleyecek.

```
FROM jenkins/jenkins:2.387.1
USER root
RUN apt-get update && apt-get install -y lsb-release curl jq wget build-essential
RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \
  https://download.docker.com/linux/debian/gpg
RUN echo "deb [arch=$(dpkg --print-architecture) \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \
  https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN mkdir -p ~/.ssh
RUN ssh-keygen -q -b 4096 -t rsa -f ~/.ssh/id_rsa -N ''
RUN curl -L https://api.github.com/meta | jq -r '.ssh_keys | .[]' | sed -e
's/^/github.com /' >> ~/.ssh/known_hosts
RUN jenkins-plugin-cli --plugins "blueocean docker-workflow"
```

- Dockerfile'dan imajı şu komutla oluşturalım:

```
$ docker build -t myjenkins-blueocean:2.387.1-1 .
```

# Jenkins konteynerinin çalıştırılması

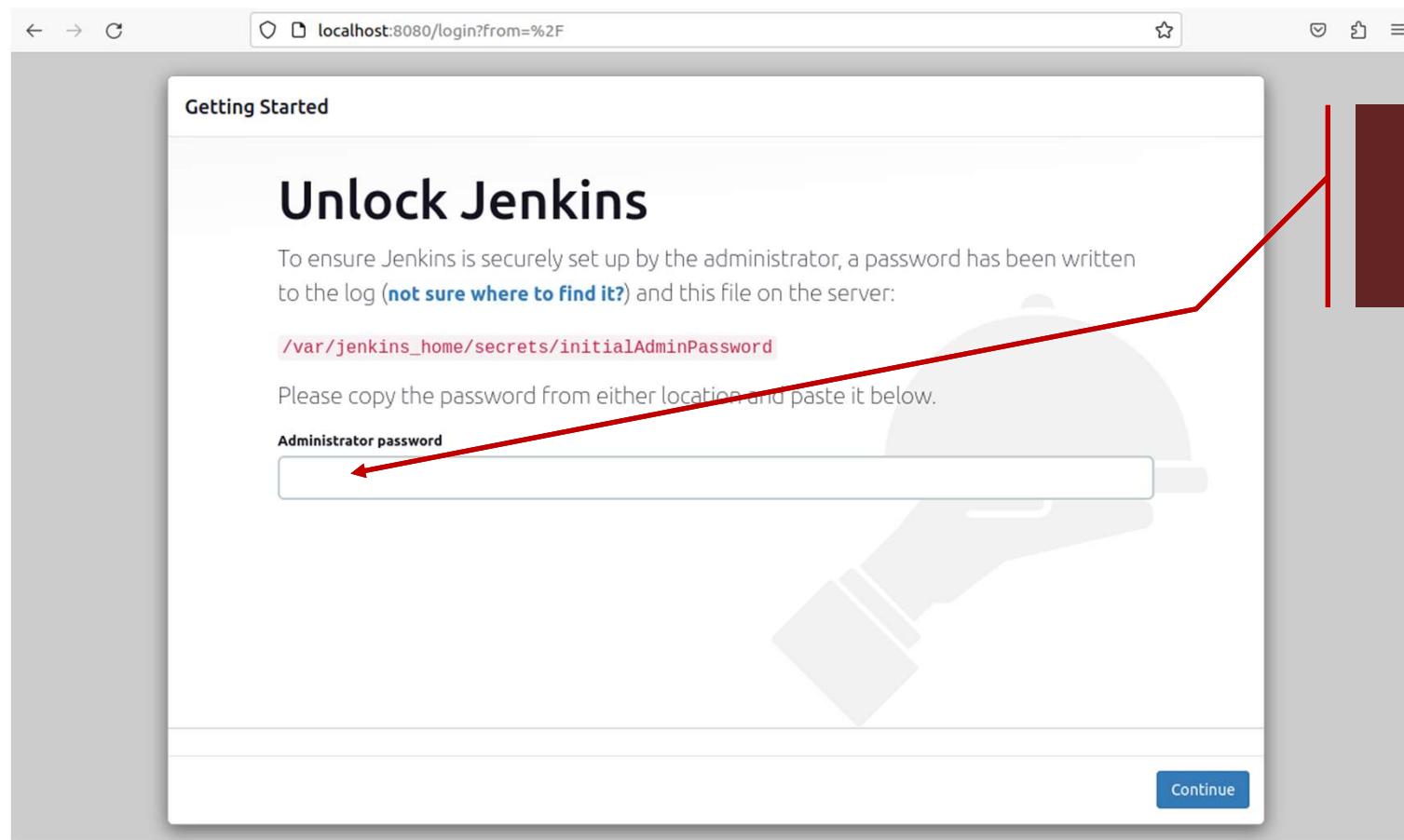
- Az önce oluşturduğumuz imajı konteyner olarak aşağıdaki komutla çalıştıralım:

```
$ docker run --name jenkins-blueocean --restart=on-failure --detach --network jenkins --env DOCKER_HOST=tcp://docker:2376 --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 --publish 8080:8080 --publish 50000:50000 --volume jenkins-data:/var/jenkins_home --volume jenkins-docker-certs:/certs/client:ro myjenkins-blueocean:2.387.1-1

$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              NAMES
PORTS
7b17185de460      myjenkins-blueocean:2.387.1-1   "/usr/bin/tini -- /u..."   6 seconds ago    Up 3 seconds
0.0.0.0:8080->8080/tcp,  ::::8080->8080/tcp, 0.0.0.0:50000->50000/tcp,  ::::50000->50000/tcp   jenkins-blueocean
862b527af2ab      docker:dind           "dockerd-entrypoint..."  32 minutes ago   Up 32 minutes
2375/tcp, 0.0.0.0:2376->2376/tcp,  ::::2376->2376/tcp   jenkins-docker
```

# Jenkins'e erişim

- «jenkins» konteynerine Firefox üzerinden «<http://localhost:8080>» URL'i üzerinden erişilebilir.



«`docker logs konteynerID`»  
yazarak, admin şifresine  
erişilebilir.

# «jenkins» admin şifresi

```
*****  
*****
```

Jenkins initial setup is required. An admin user has been created and a password generated.  
Please use the following password to proceed to installation:

b54ff430cc5f4ab18fc17aa91b0c45c2

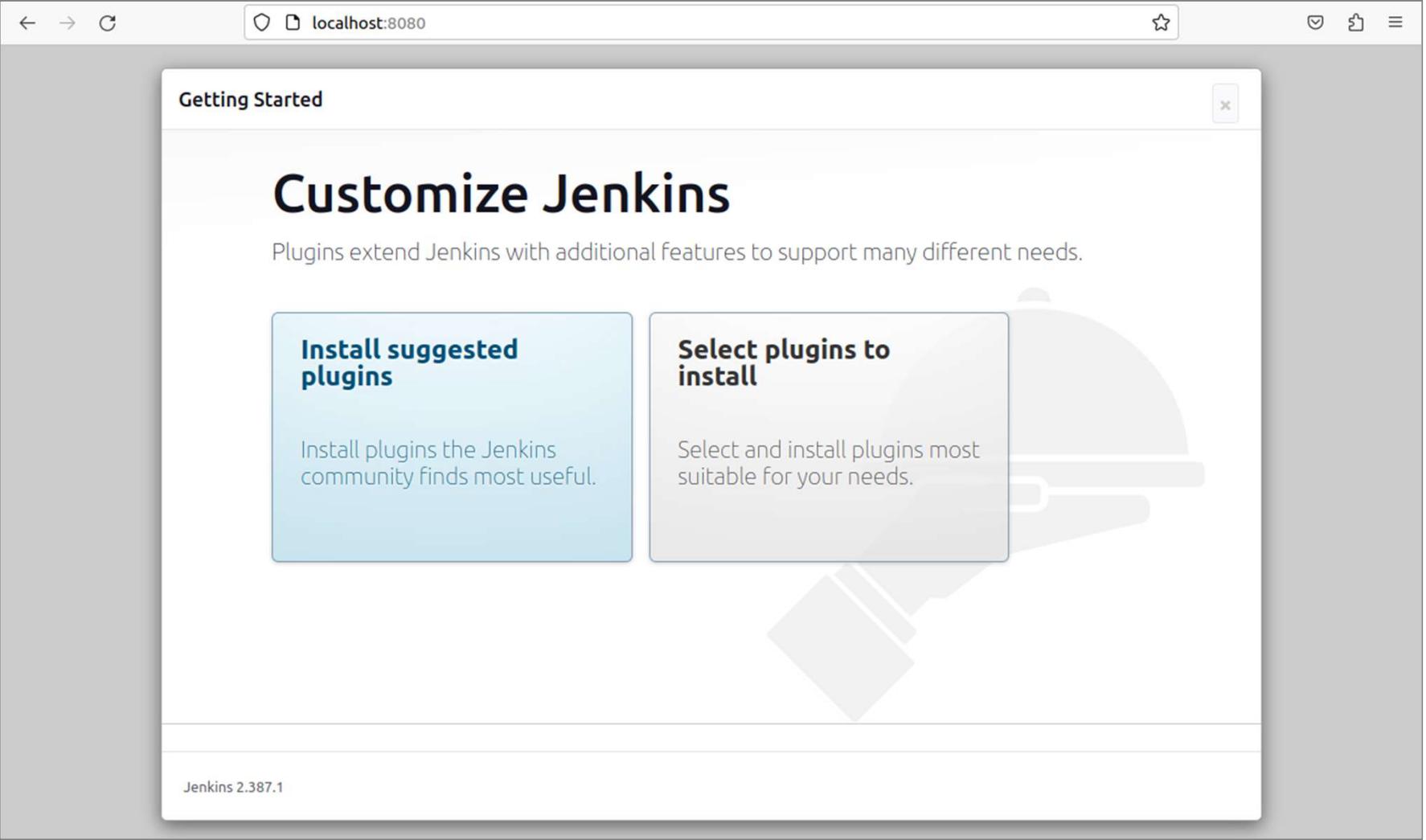
This may also be found at: /var/jenkins\_home/secrets/initialAdminPassword

```
*****  
*****  
*****
```

```
2023-03-27 20:02:29.124+0000 [id=41]    INFO jenkins.InitReactorRunner$1#onAttained: Completed initialization  
2023-03-27 20:02:29.333+0000 [id=24]    INFO hudson.lifecycle.Lifecycle#onReady: Jenkins is fully up and  
running  
2023-03-27 20:02:30.019+0000 [id=60]    INFO h.m.DownloadService$Downloadable#load: Obtained the updated data  
file for hudson.tasks.Maven.MavenInstaller  
2023-03-27 20:02:30.023+0000 [id=60]    INFO hudson.util.Retriger#start: Performed the action check updates  
server successfully at the attempt #1
```

«copy» «paste» edin

# Eklentiler

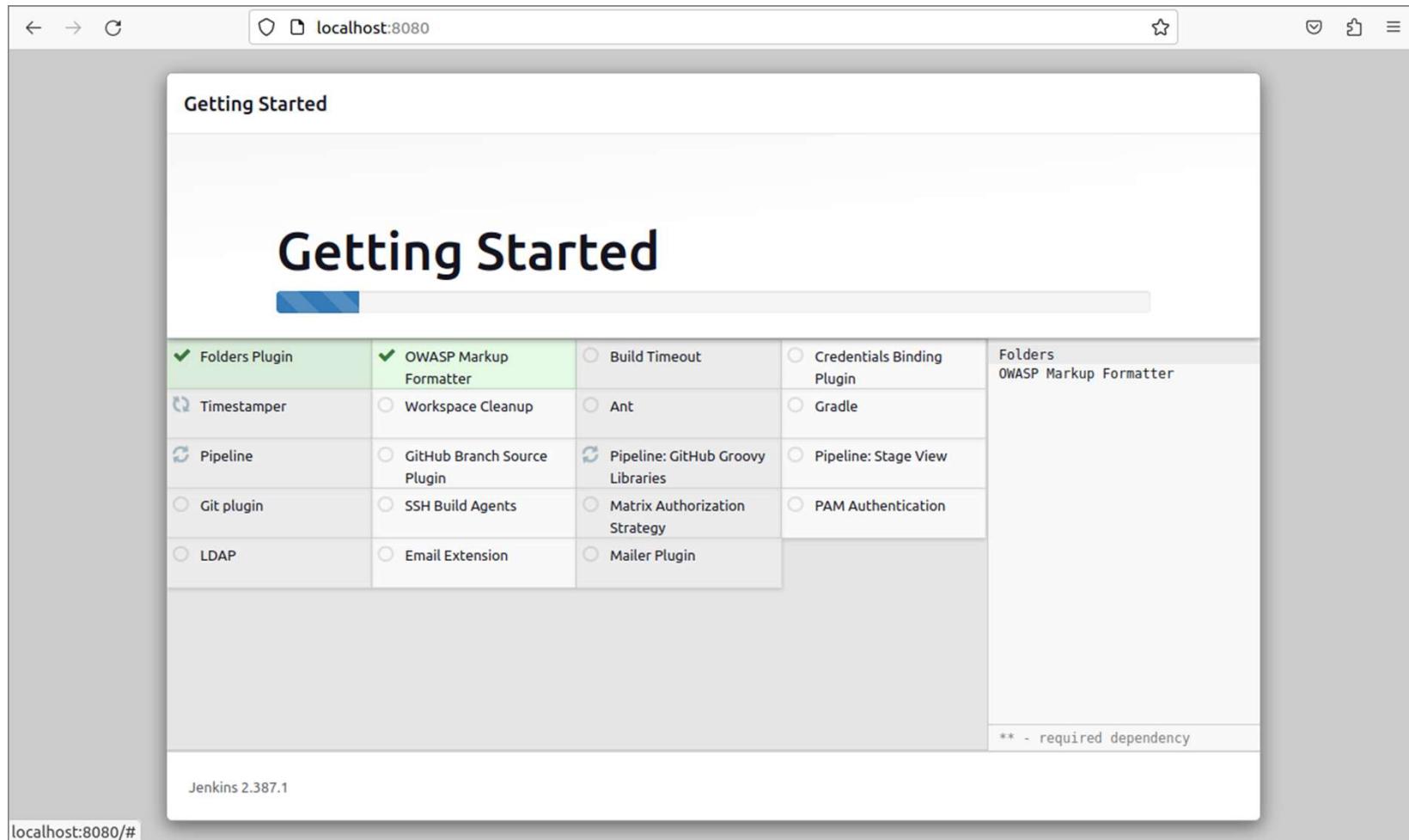


The screenshot shows the Jenkins 'Getting Started' page at `localhost:8080`. The main heading is 'Customize Jenkins' with the subtext: 'Plugins extend Jenkins with additional features to support many different needs.' Below this are two main options:

- Install suggested plugins**: A light blue button with the text 'Install plugins the Jenkins community finds most useful.'
- Select plugins to install**: A light gray button with the text 'Select and install plugins most suitable for your needs.'

A large, semi-transparent icon of a wrench and a screwdriver is positioned in the background of the central area. At the bottom left of the page, it says 'Jenkins 2.387.1'.

# Eklentiler



The screenshot shows the Jenkins "Getting Started" page at [localhost:8080](http://localhost:8080). The page lists various Jenkins plugins categorized into groups:

Available Plugins		Optional Plugins		
<input checked="" type="checkbox"/> Folders Plugin	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input type="radio"/> Build Timeout	<input type="radio"/> Credentials Binding Plugin	Folders
<input type="radio"/> Timestamper	<input type="radio"/> Workspace Cleanup	<input type="radio"/> Ant	<input type="radio"/> Gradle	OWASP Markup Formatter
<input type="radio"/> Pipeline	<input type="radio"/> GitHub Branch Source Plugin	<input checked="" type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="radio"/> Pipeline: Stage View	
<input type="radio"/> Git plugin	<input type="radio"/> SSH Build Agents	<input type="radio"/> Matrix Authorization Strategy	<input type="radio"/> PAM Authentication	
<input type="radio"/> LDAP	<input type="radio"/> Email Extension	<input type="radio"/> Mailer Plugin		

At the bottom of the page, it says "Jenkins 2.387.1". A tooltip for the "Folders" plugin indicates it is a required dependency.

# Kullanıcı adı

Getting Started

## Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Jenkins 2.387.1

Skip and continue as admin

Save and Continue

# Jenkins'e Erişim URL'i

## Getting Started

### Instance Configuration

Jenkins URL:

`http://localhost:8080/`

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

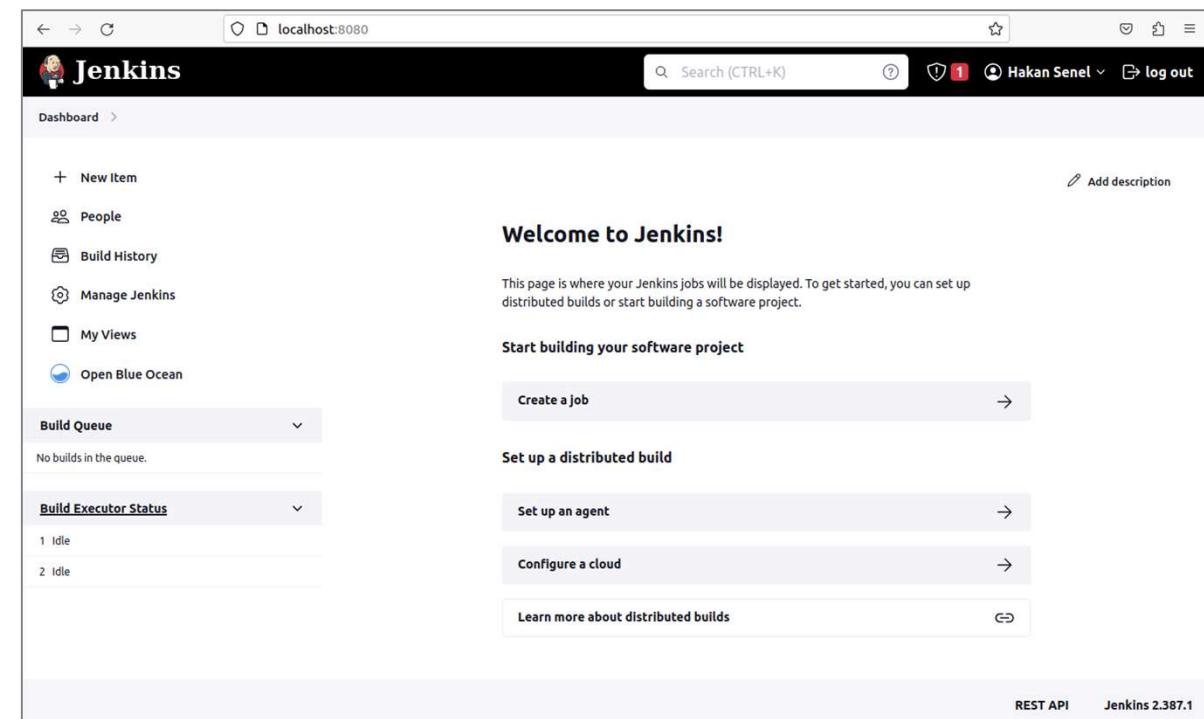
Jenkins 2.387.1

Not now

Save and Finish

# Ana ekran

- **github.com**'dan bir repoyu indirip derleyeceğimiz için, **github.com** ile bu konteyner arasındaki SSH iletişimini sağlamamız gerekiyor.
- Konteynerin **bash** kabuğuna bağlanarak, **Dockerfile** içinde «**ssh-keygen**» programıyla oluşturmuş olduğumuz açık anahtarı **~/.ssh/id\_rsa.pub** dosyasından alacağız ve **github.com**'da kullanıcı ekranımızdaki ana menüde «**SSH and GPG keys**» bölümünde yapıştıracağız.
- Bu şekilde, **github.com** ve Jenkins konteynerimiz, birbirinden veri alışverişi yapabilecek.
- Ayrıca, **Dockerfile** içinde **github.com** sitesinin açık anahtarının **~/.ssh/known\_hosts** dosyasına aktarıldığı bölüm bulunmaktadır.



# **github.com ile bağlantı kurulması**

- «**docker ps**» komutuyla, Jenkins konteynerinin id'sini bulacağız.
- Ardından «**docker exec -it**» komutuyla, konteynerin kabuğuna bağlanacağız ve «**cat ~/.ssh/id\_rsa**» komutuyla açık anahtarı görüntüleyeceğiz. Bu açık anahtarı seçip Ctrl-C ile kopyalayacağız.
- Kopyaladığımız açık anahtarı **github.com** sitesinde ana kullanıcı menüsünde bulunan «**SSH and GPG keys**» bölümünde yapıştıracağız.

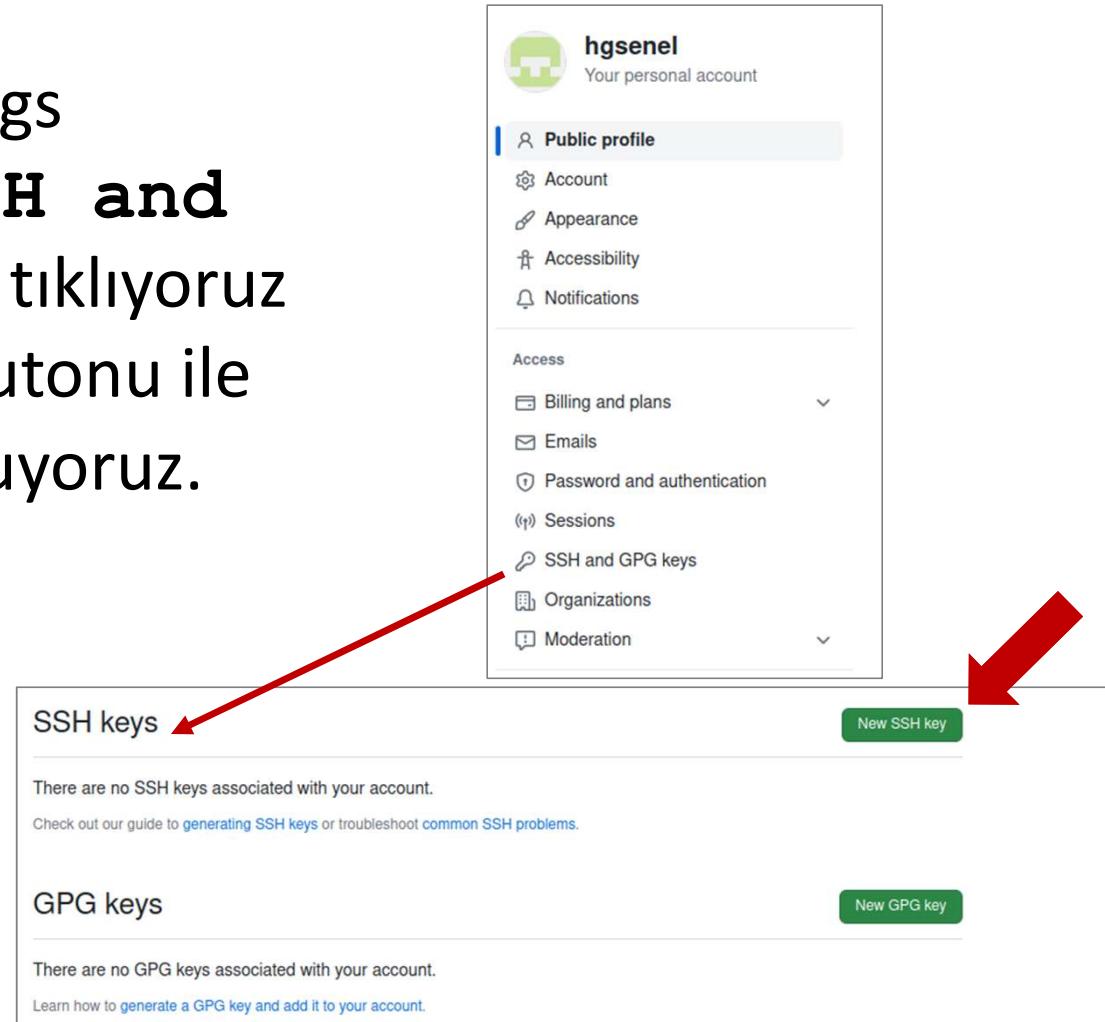
# Konteynere erişim

- Konteynerin ID'sini bulup kabuğuna bağlanacağız.

```
$ sudo docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
NAMES
e6823a01633c      myjenkins-blueocean:2.387.1-1   "/usr/bin/tini -- /u..."   48 minutes ago
                  Up 48 minutes   0.0.0.0:8081->8080/tcp, :::8081->8080/tcp, 0.0.0.0:50001->50000/tcp, :::50001->50000/tcp   jenkins-blueocean2
fc170db33657      docker:dind                 "dockerd-entrypoint..."   49 minutes ago
                  Up 49 minutes   2375/tcp, 0.0.0.0:2376->2376/tcp, :::2376->2376/tcp
jenkins-docker
$ sudo docker exec -it e6823a01633c bash
jenkins@e6823a01633c:/$ cat ~/.ssh/id_rsa.pub
ssh-rsa
AAAAB3NzaC1yc2EAAAQABAAQgQDJp/nHAS7etii+0Y0wVJ+9Bp8nx50TR8oGrRDpDf3zo8/GhJqXcTA
q0rJoN7Sr3rvGd0aGWbM6jWftdOZ8161gpZWjSvo7m8pHDvh70d9Rzm2Hswp5bDpx7EBmeTBSq7FmeV49X1/
z7myb8hbbXvSelZnW0wmEhj814805zmao0STUUEBuNZ4Dm/poxqHI/7wgoiDBsxjfoEYde2OEUVcXT69KAC6
jtBZx1hAP21VjU9UhB4E8bJn1W2x66jniEhSO/kOMfO8cb1LHHJrLmIBPgh4PiB4EMw5TDXHeebu+AmqZYx5
eeaRNyktv7S6pRMEQxTF1laEGJiDewXUNmkUJPkuGFwxv0eM3ml8QT1ULGk7sOHMVyi tq1RC/k0mK+yVVqU8
/U/BaLSUWUToeqjKvb95TWfq2SmL1tdmCj+n9xDJ2CoMXXzE98xF16pwXaDIMo4SXKgd+vDtsPRcOjcqQAdu
Uh5dX1uOz+z8wK6PP01pHSb8S1aZTIowwLkgtsU= jenkins@e6823a01633c
```

# github.com SSH anahtarı girişi

- **github . com**'da Settings menüsünün altında «**SSH and GPG keys**» bölümüne tıklıyoruz ve «**New SSH key**» butonu ile yeni bir anahtar oluşturuyoruz.



# github.com açık SSH anahtarı girişi

- **github.com'a**, konteynerimizdeki SSH anahtar çiftinin açık olanını eklemiş oluyoruz.
- Artık konteynerimiz bu açık anahtar üzerinden ve kendi gizli anahtarıyla bağlantı kurabilecek.

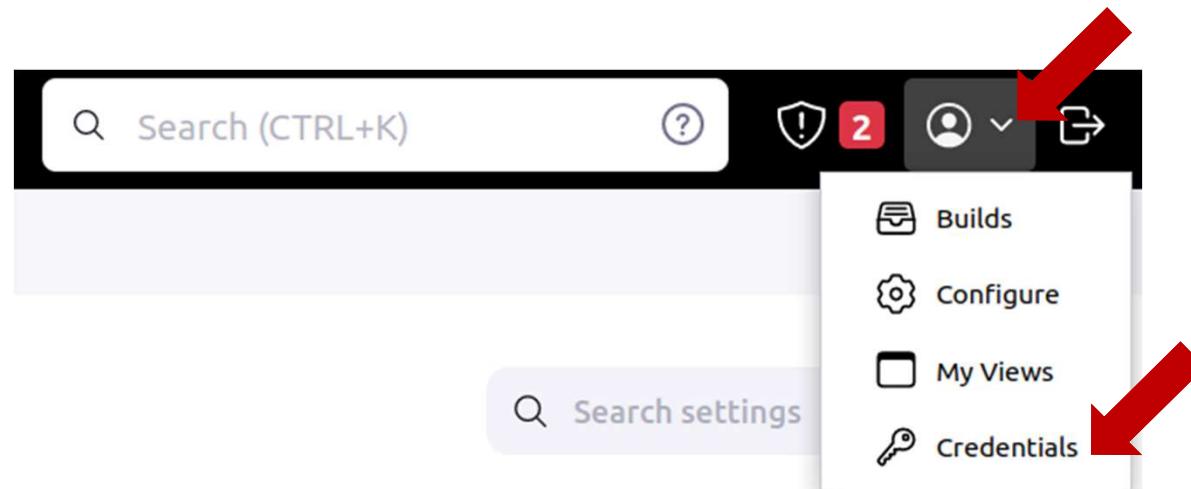
The screenshot shows the GitHub 'SSH keys' page. At the top, there's a header with the title 'SSH keys' and a green button labeled 'New SSH key'. Below the header, a message reads: 'This is a list of SSH keys associated with your account. Remove any keys that you do not recognize.' Underneath, there's a section titled 'Authentication Keys' with a single entry:

SSH	deneme	SHA256:3w90/evFz4f0JUuXkYZvY07H7V3xbq+p+Cotc5wJ52k	Delete
		Added on Mar 30, 2023	
		Never used — Read/write	

At the bottom of the page, there's a note: 'Check out our guide to [generating SSH keys](#) or troubleshoot [common SSH problems](#)'.

# Jenkins konteynerinde gizli anahtar girişi

- Jenkins'in **github.com**'a bağlanması için gizli anahtarın da Jenkins'e «**credential**» olarak tanıtılması gereklidir.
- Bunun için «**Dashboard**» ve sağdaki kullanıcı profil altında «**Credentials**» seçeneği seçilmelidir.



# Credentials

- Jenkins konteynerinde bu sefer `~jenkins/.ssh/id_rsa` dosyasında bulunan gizli anahtarı, Jenkins'te tanımlayarak sistem genelinde çalışacak bir «**credential**» üretmemiz gerekiyor.
- Bu amaçla, «**Stores from parent**» bölümünde, ikinci kolonda bulunan «**(global)**» üzerine fare imlecini getirdiğimizde aşağıya açılacak bir menü belirecek. Açılan menüden, «**Add credentials**»a tıklayacağız.

The screenshot shows the Jenkins 'Credentials' page under 'Stores from parent'. It displays a table with columns: T, P, Store ↓, Domain, ID, and Name. There is one entry: User: Hakan Senel, (global), blueocean-github-domain. Below this table, there is a 'Domains' dropdown menu. A red arrow points to the 'Add credentials' button, which is located at the bottom right of this dropdown menu. The 'Domains' dropdown currently has '(global)' selected.

# Credentials



- «**Kind**» bölümünde «**SSH Username with private key**» opsiyonunu seçmeliyiz. Az önce **github.com**'a Açık anahtarımızı koymuştuk. Şimdi buna karşılık gelen gizli anahtarımızı, konteyner içinde alarak, buraya aktaracağız.
- «**Scope**» olarak «**Global (Jenkins, nodes, items, all child items, etc)**» opsiyonunu seçmeliyiz. Bu şekilde, Jenkins'te her yerde geçerli olmasını sağlayacağız.
- ID olarak, anlaşılır bir isim seçmeliyiz. Çünkü bunu kullanarak **github.com**'a bağlanacağız. Örneğin «**github\_cr**» seçelim.
- **Username** olarak «**jenkins**» seçeceğiz. Çünkü bu kullanıcının açık ve gizli anahtarlarını kullanıyoruz.
- «**Description**» bölümünde, açıklayıcı olsun diye «**github' daki repomuza bağlanmak için kullanacağız**» diye yazalım.
- «**Private Key**» bölümünde «**Enter directly**» opsiyonunu seçeceğiz ve «**Add** diyerek, konteyner içinde kopyaladığımız `~/.ssh/id_rsa` dosyasını «**Key**» bölümüne yapıştıracağız.

# «New credentials»

- «Create» butonuna basarak, yeni «credential»ının tanımlanmasını sağlayalım.

**New credentials**

**Kind**: SSH Username with private key

**Scope**: Global (Jenkins, nodes, items, all child items, etc)

**ID**: github\_cr

**Description**: github'daki repomuza bağlanmak için kullanacağız

**Username**: jenkins

Treat username as secret

**Private Key**:  Enter directly

```
vug0UWw4Zmnu0cJ37zHmM4pJ3C00r0Xg20V2zLzHnOzH2LzV0YfPfH0w/  
Ager7HxbG+6+sPAAAAFGplbmtpbnNAZTY4MjNHMD2HzNjAQIDBAUG  
-----END RSA PRIVATE KEY-----
```

**Create**

**Global credentials (unrestricted)**

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind	Description
	github_cr	jenkins (github'daki repomuza bağlanmak için kullanacağız)	SSH Username with private key

Icon: S M L

+ Add Credentials

# JENKINSFILE'I JENKINS İÇİNDE OLUŞTURARAK PIPELINE OLUŞTURMA

# Jenkinsfile ve Groovy

- Jenkins'te pipeline oluşturmak için iki yöntem bulunmaktadır.
  - Declarative (Jenkinsfile): içinde aşamaların ve adımların tanımlı olduğu jenerik bir oluşturma yöntemidir. Karmaşık ve koşullu bir süreç gerekmediği sürece kullanılabilecek bir yöntemdir ve pipeline eklentisi yüklenerek gerçekleştirilir.
  - Scripted: Groovy, Java türevi bir programlama dilidir ve pipeline'lar groovy'de oluşturulabilir.

# Jenkinsfile oluşturarak Pipeline yaratma

- Pipeline yani **Jenkinsfile** oluşturma, hem elle hem de Blue Ocean editörü yardımıyla görsel arayüz üzerinde yapılmaktadır.
- Ancak Blue Ocean editöründe geliştirme durmuştur ve **github.com**'a erişim için bazı **plugin**'lerin eski sürümlerinin yüklenmesi gereklidir. Bu nedenle kullanımı tavsiye edilmemektedir.
- Ancak yine de hızlı bir **pipeline** üretebilmek için faydalı bir araçtır.
- Bu bölümde git projemizde bulunan «**Jenkinsfile**»ı kullanarak pipeline'ımızı oluşturacağız. Bu amaçla «**Dashboard**»da «**+ New Item**» butonuna basacağız ve ardından isim yazarak «**Pipeline**» opsiyonunu seçtikten sonra «**OK**» butonuna basacağız.

# «+ New item»

Enter an item name

deneme1 

» Required field

**Freestyle project** 

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

**Pipeline** 

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

**Multi-configuration project**

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

**Folder**

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

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Organization Folder**

Creates a set of multibranch project subfolders by scanning for repositories.

**OK** 

**Configure**

**General** 

**General**

Description: deneme1

[Plain text] [Preview](#)

Discard old builds ?

Do not allow concurrent builds

Do not allow the pipeline to resume if the controller restarts

GitHub project

Pipeline speed/durability override ?

Preserve stashes from completed builds ?

This project is parameterized ?

Throttle builds ?

**Build Triggers**

Build after other projects are built ?

Build periodically ?

GitHub hook trigger for GITScm polling ?

Poll SCM ?

Quiet period ?

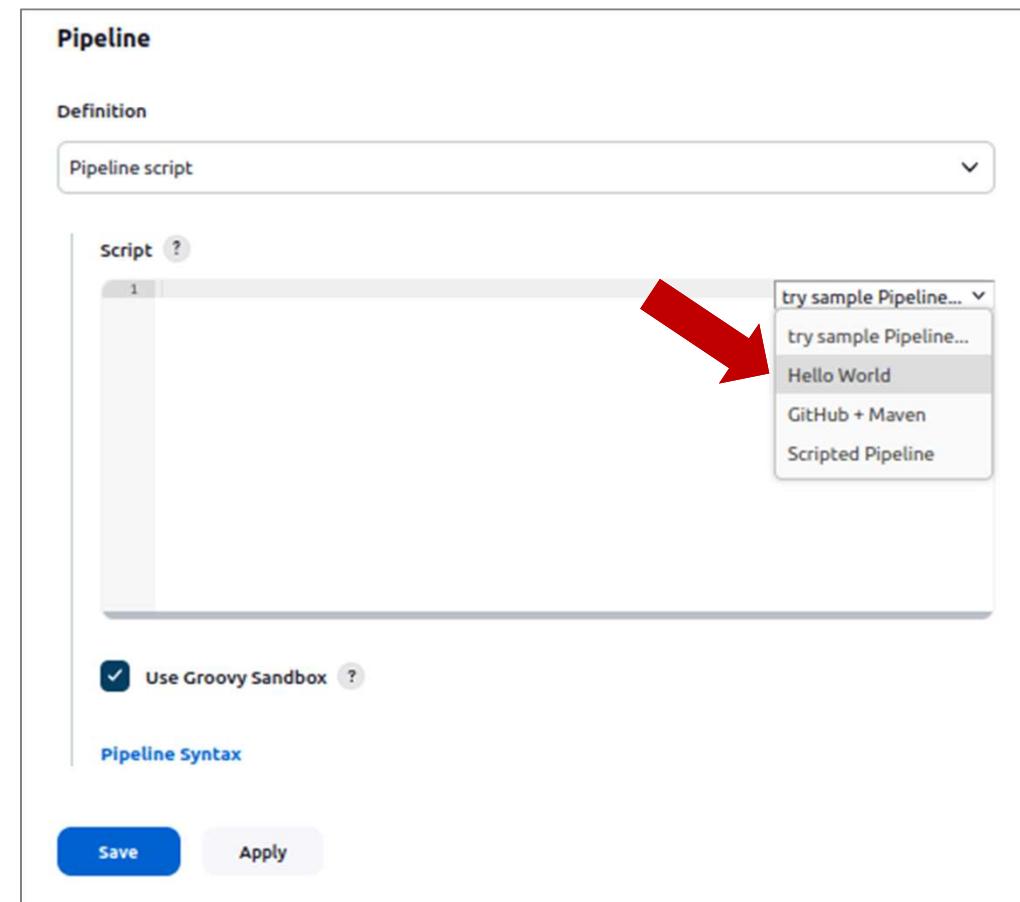
Trigger builds remotely (e.g., from scripts) ?

**Advanced Project Options**

**Save** **Apply**

## «+ New item»

- «Pipeline» bölümünde «Definition» altında, «Pipeline script» opsiyonunu seçeceğiz.
- «Script» bölümünde «pipeline» betığını yazmamız beklenmektedir.
- Sağ tarafta «try sample Pipeline» bölümünde «Hello World» şablonunu seçeceğiz.



# «Hello World» betiği

- «Hello World» betiği, «Pipeline» betiği yazmaya başlamak için yeterlidir. Zira, bütün betik yapısı görülebilmektedir.
- İlk aşamayı değiştireceğiz ve `echo` ile başlayan kısmı sileceğiz.

The screenshot shows a Jenkins Pipeline configuration page. On the left, there's a code editor titled "Pipeline script" containing the following Groovy code:

```
1 pipeline {  
2   agent any  
3  
4   stages {  
5     stage('gitgetir') {  
6       steps {  
7         // This line is highlighted with a red arrow from the first callout.  
8         echo 'Hello World'  
9       }  
10    }  
11  }  
12 }
```

Below the code editor is a checkbox labeled "Use Groovy Sandbox" with a checked status. At the bottom of the editor, there's a blue link labeled "Pipeline Syntax".

The right side of the screen has a sidebar with a dropdown menu set to "Hello World".

Two red callout boxes provide instructions:

- (1) «Pipeline Syntax» linkine tıklayalım ve `github.com`'a «credential» kullanarak bağlanmak ve repoya erişmek için bir kod parçası oluşturacağız.
- (2) Oluşturduğumu kod parçasını «Code Snippet» boş satırda ekleyeceğiz.

# «Pipeline Syntax»

- «Pipeline Syntax», betik parçaları hazırlamak için kullanılan bir arayüzdür. Burada, github'a bağlanarak kodu indirmek için kullanacağımız kod parçasını oluşturacağız ve betiğe kopyalayacağız.

The screenshot shows the Jenkins Pipeline Syntax Snippet Generator interface. On the left, there's a sidebar with links like Snippet Generator, Declarative Directive Generator, etc. The main area has an 'Overview' section explaining the generator's purpose. Below it is a 'Steps' section where a 'Sample Step' is configured. The configuration includes:

- Repository URL:** git@github.com:hgsenel/gpsparser.git
- Branch:** master
- Credentials:** jenkins (github'daki repomuza bağlanmak için kullanacağız)
- Include in polling?**
- Include in changelog?**

At the bottom, a large blue button says 'Generate Pipeline Script'. To its right, the generated script is shown in a code editor:

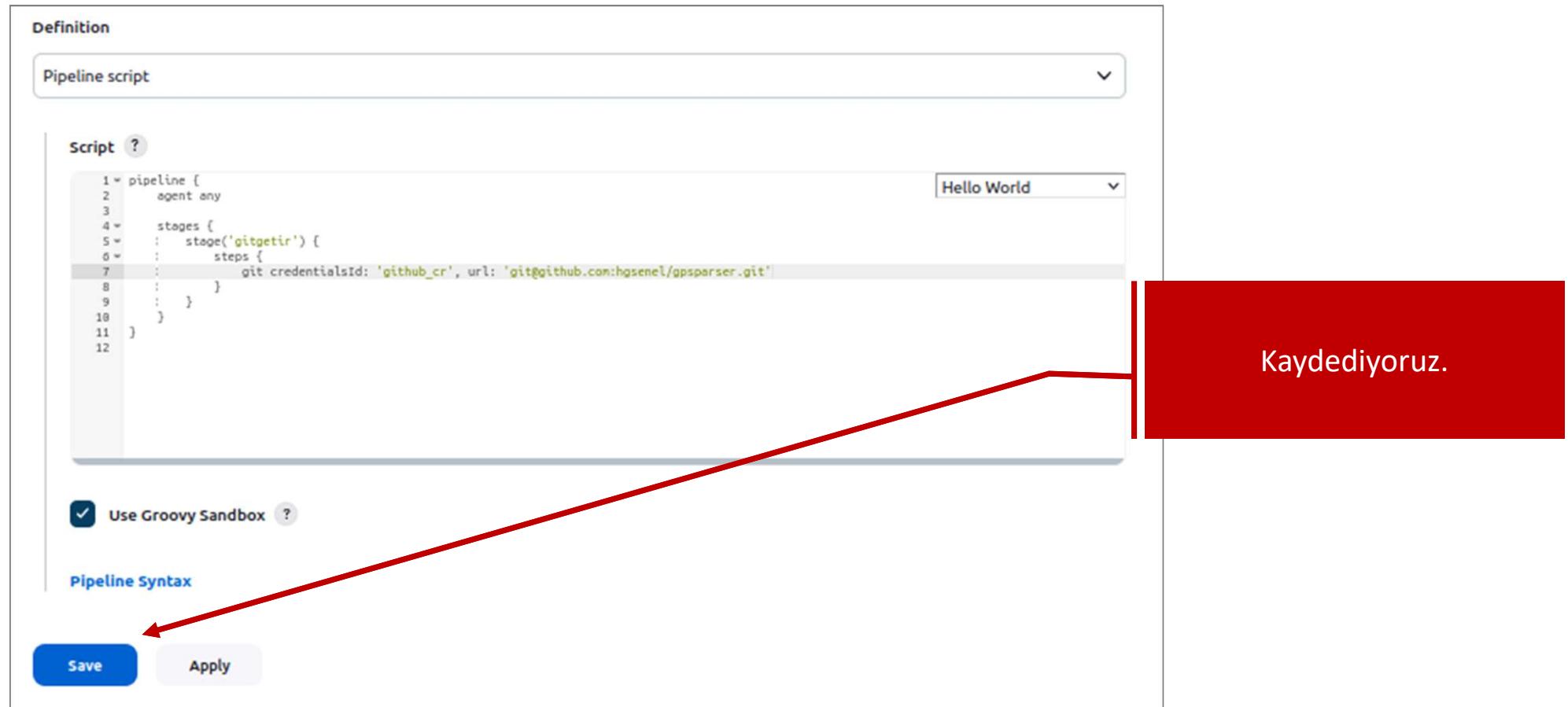
```
git credentialsId:'github_cr', url:'git@github.com:hgsenel/gpsparser.git'
```

Four red arrows point from the right side of the interface to the right, each pointing to a red callout box with explanatory text:

- Repo adresi**
- «Credential» seçmemiz gereklidir.**
- «Generate Script» betik parçasını oluşturur.**
- Buradaki parçayı kopyalayarak önceki tabtaki betik içine yapıştıracağız.**

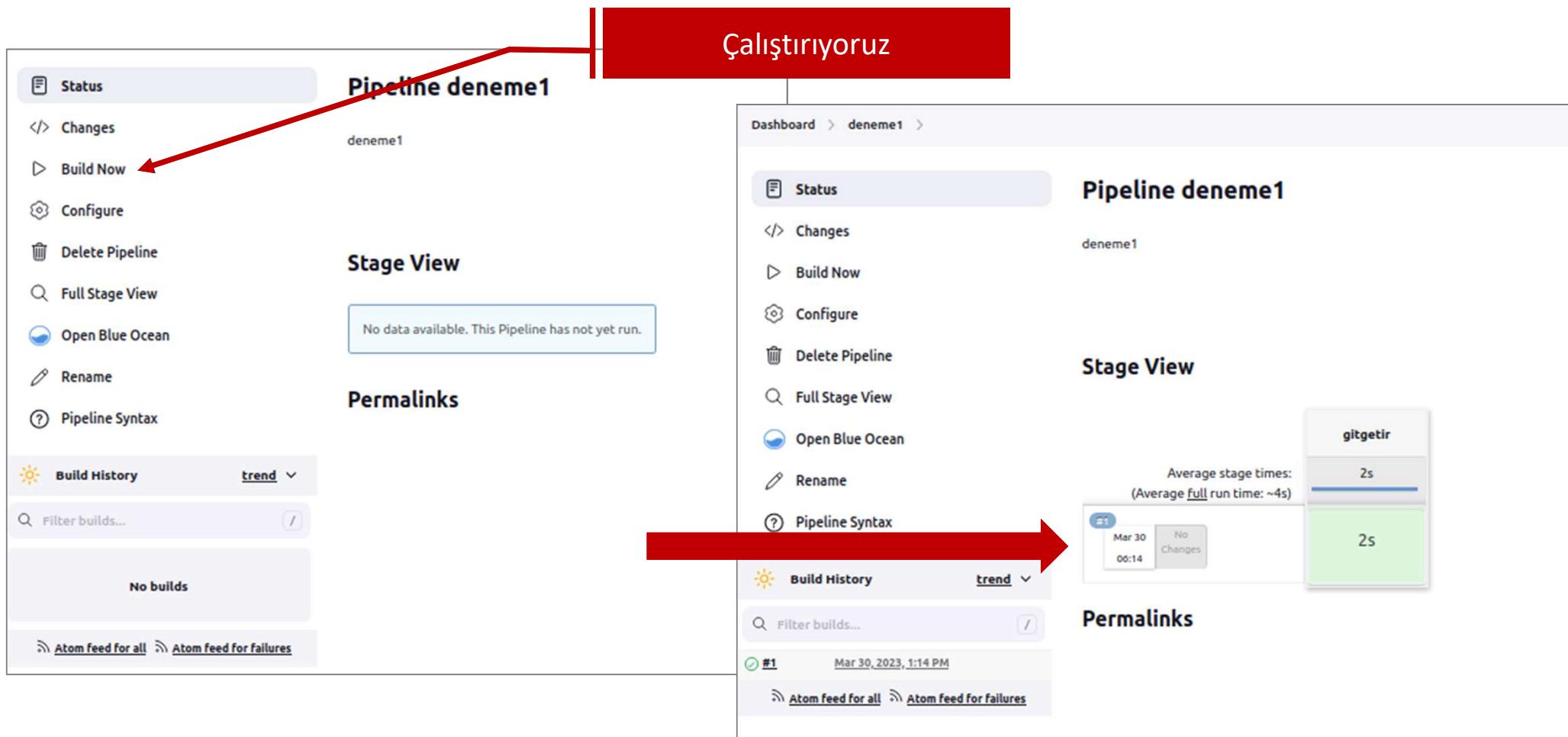
# betik

- Betik içindeki boşalttığımız satırı, kopyaladığımız kod parçasını yapıştırıyoruz ve ardından «Save» butonuna basıyoruz.



# betik

- «Build now» diyerek betiği çalıştırıyoruz.



The image shows two screenshots of a Jenkins Pipeline interface. A red arrow points from the left screenshot to the right one, indicating the progression of the build process.

**Left Screenshot (Initial State):**

- Pipeline Name:** Pipeline deneme1
- Status:** Status (button)
- Actions:** Changes, Build Now (highlighted with a red arrow), Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax.
- Stage View:** Stage View (No data available. This Pipeline has not yet run.)
- Permalinks:** No builds
- Build History:** trend (dropdown), Filter builds... (input field), No builds.
- Links:** Atom feed for all, Atom feed for failures.

**Right Screenshot (After Clicking Build Now):**

- Header:** Çalıştırıyoruz (Working) - A large red banner at the top.
- Pipeline Name:** Pipeline deneme1
- Status:** Status (button)
- Actions:** Changes, Build Now, Configure, Delete Pipeline, Full Stage View, Open Blue Ocean, Rename, Pipeline Syntax.
- Stage View:** Stage View (Average stage times: (Average full run time: ~4s))
- Permalinks:** gitgetir, 2s (highlighted with a red arrow).
- Build History:** trend (dropdown), Filter builds... (input field), #1 Mar 30, 2023, 1:14 PM.
- Links:** Atom feed for all, Atom feed for failures.

# Yeni bir aşama ekleme

- «**pipeline**» içinde, sol taraftaki «**Configure**» butonuna basarak «**pipeline script**» içinde değişiklik yapacağız.
- «**git**» aşamasına ek yenisini ekleyeceğiz.
- Bu aşamada gpssparser projesini GNU C derleyicisi ile derleyeceğiz.
- Bu amaçla «**Pipeline Syntax**»e basarak, yeni bir kabuk betiği ekleyeceğiz.

**Steps**

Sample Step

sh: Shell Script

sh

Shell Script ?

```
mkdir -p build
gcc -o build/gpssparser gps/gps.c gps/main.c gps/util.c
```

Advanced ▾

Generate Pipeline Script

```
sh ""mkdir -p build
gcc -o build/gpssparser gps/gps.c gps/main.c gps/util.c""
```

# Yeni bir aşama ekleme

- «**mkdir -p build**» komutuyla yeni bir dizin oluşturacağız. «**gcc -o build/gpsparser gps/gps.c gps/main.c gps/util.c**» komutuyla projeyi derleyeceğiz ve **build** dizini altında **gpsparser** programını oluşturacağız.
- «**Save**» butonundan sonra «**Build Now**» butonuyla «**pipeline**»ı çalıştıralım.

Definition

Pipeline script

```

Script ?
```

```

1 pipeline {
2   agent any
3
4   stages {
5     stage('gitindir') {
6       steps {
7         git credentialsId: 'github_cr', url: 'git@github.com:hgsenel/gpsparser.git'
8       }
9     }
10    stage('derleme') {
11      steps {
12        sh '''mkdir -p build
13        gcc -o build/gpsparser gps/gps.c gps/main.c gps/util.c'''
14      }
15    }
16  }
17}
18
19

```

Use Groovy Sandbox ?

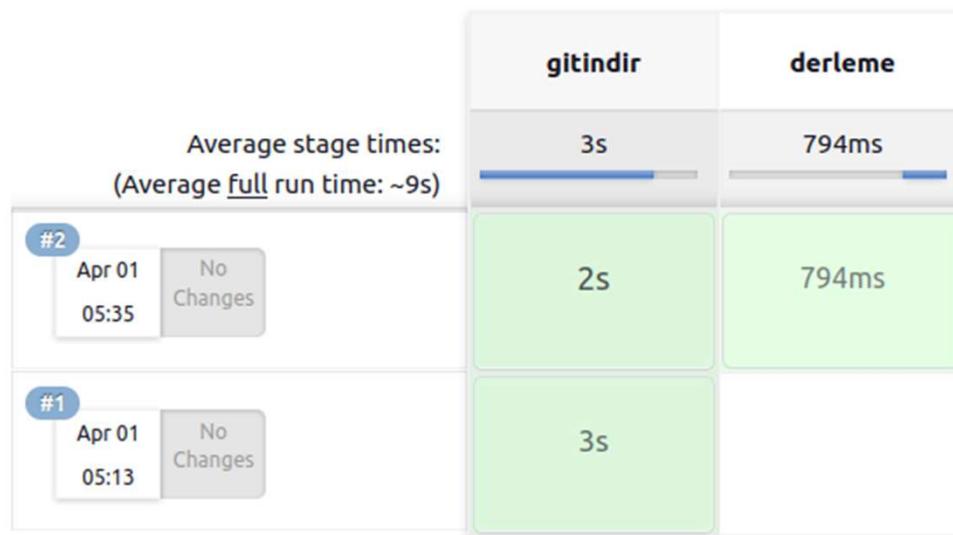
Pipeline Syntax

**Save** **Apply**

# Yeni bir aşama ekleme

- Projeyi iki aşamalı şekilde yazmıştık.
- Derleme sonrasında iki aşamalı derleme sonucu görülecek.

## Stage View



## Permalinks

- [Last build \(#1\), 21 min ago](#)
- [Last stable build \(#1\), 21 min ago](#)
- [Last successful build \(#1\), 21 min ago](#)
- [Last completed build \(#1\), 21 min ago](#)

# Docker hub'a imaj gönderme

- Docker/Kubernetes kullanılan projelerde, Docker Hub'a imaj göndermek gerekebilir.
- Böyle bir durumda Docker Hub'a giriş yapılmalıdır.
- «**docker login -u kullanıcı -p şifre**» yazılarak yapılabıldığı gibi, alternatif «**credential**» yöntemleri de bulunur.
  - D-bus Secret Service
  - Apple MacOS keychain
  - Microsoft Windows Credential Manager
  - Pass: <https://github.com/docker/docker-credential-helpers/releases>
- «**credential**» yöntemlerinden biri kullanılabilir ama bu örnekte **passwd.txt** adında bir dosyaya şifremizi koyup repomuza ekleyeceğiz.
- Ardından «**cat ~/passwd.txt | docker login --username foo --password-stdin**» komutuyla Docker'a **login** olduktan sonra «**docker push**» komutuyla imajımızı Docker Hub'a atacağız..

## «**docker push**»

- Docker Hub'da eğer hesabınız yoksa yeni bir hesap oluşturmamız gerekiyor. Hesap ismi ve şifresi, «**push**» işlemlerinden önce, lokalden Docker Hub'a yapacağımız login işlemi için gerekli.
- «**restapiex**» isimli «**private repository**» oluşturacağız. Lokalde oluşturacağımız imaja «**docker tag**» ile yeni bir etiket atayacağız. Bu etiketin ismi, «**kullanıcıadımız/proje123:sürüm**» şeklinde olacak.
- «**docker push**  
**kullanıcıadımız/restapiex:sürüm**» komutuyla, o repoya imajımızı kaydedeceğiz.

# Özel repo oluşturma

This screenshot shows a user's namespace 'hgsenel' on Docker Hub. The interface includes a search bar, a dropdown for 'All Content', and a prominent blue 'Create repository' button. A red arrow points to this button. Below the button, a message states: 'There are no repositories in this namespace. Tip: Not finding your repository? Try a different namespace.' There is also a small icon of a server or storage unit.

This screenshot shows the 'Create repository' form. It includes fields for 'Namespace' (set to 'hgsenel'), 'Repository Name' (set to 'restapiex'), and a 'Description' field containing 'Devops Teknolojileri eğitimi'. On the right, there is a 'Pro tip' section with CLI instructions for pushing images and a note to change the tagname. The 'Visibility' section offers two options: 'Public' (unchecked) and 'Private' (checked), with the private option being 'Only visible to you'. At the bottom are 'Cancel' and 'Create' buttons, with a red arrow pointing to the 'Create' button.

Repositories > Create >

Using 0 of 1 private repositories. [Get more](#)

Create repository

Namespace: hgsenel Repository Name \*: restapiex

Description: Devops Teknolojileri eğitimi

Pro tip

You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Make sure to change tagname with your desired image repository tag.

Visibility

Using 0 of 1 private repositories. [Get more](#)

Public Appears in Docker Hub search results

Private Only visible to you

[Cancel](#) [Create](#)

# Jenkins içinde imaj oluşturma

- **Jenkinsfile** içinde, yeni bir kabuk programı yazacağız ve sırasıyla
  - GitHub'dan indirdiğimiz repomuzun içinde **containers/** dizinine gireceğiz.
  - «**docker build build -t hgsenel/restapiex:v1.0 .**» komutuyla Dockerfile'da tarif edilen imajı oluşturacağız.
  - «**docker login**» ile Docker Hub'a giriş yapacağız.
  - «**docker push hgsenel/restapiex:1.0**» ile özel repomuza imajı göndereceğiz.
- Burada duracağınız, çünkü bu aşamadan sonrası, projenize özel adımlar olacaktır. Eğer istenirse oluşturulan bu imaj, Kubernetes'de veya OCP'de devreye alınabilir.
- **Burada hgsenel kullanıcı adını, kendi Docker kullanıcı adınızla değiştirmelisiniz.**

# Dockerfile'dan imaj oluşturma

- Aşağıdaki dosyalar Docker imajı oluşturmak için kullanılacak dosyalardır:  
**users.json**  
**Dockerfile**  
**server.js**  
**packages.json**  
**passwd.txt** (Docker Hub'daki şifrenizi tek satır halinde yazınız)
- «**sudo docker build -t hgsenel/restapiex:v1.0 .**» komutuyla imajı oluşturacağız.
- Ardından «**docker push**» komutuyla konteyneri repomuza yükleyeceğiz.

# Dockerfile'dan imaj oluşturma

- Aşağıdaki dosyalar Docker imajı oluşturmak için kullanılacak dosyalardır:  
**users.json**  
**Dockerfile**  
**server.js**  
**packages.json**  
**passwd.txt**
- «**sudo docker build -t hgsenel/restapiex:v1.0 .**» komutuyla imajı oluşturacağız.
- Ardından «**docker push**» komutuyla konteyneri repomuza yükleyeceğiz.
- Bunları kabuk programı olarak yazacağız.

# Dockerfile'dan imaj oluşturma

- Pipeline betiğimiz nihai olarak şu şekilde olacak.

```
pipeline {
    agent any ←
        stages {
            stage('gitindir') {
                steps {
                    git credentialsId: 'github_cr', url: 'git@github.com:hgsenel/gpsparser.git'
                }
            }
            stage('derle') {
                steps {
                    sh '''mkdir -p build
                    gcc -o build/gpsparser gps/util.c gps/main.c gps/gps.c'''
                }
            }
            stage('konteynner') {
                steps {
                    sh '''cd containers
                    cat passwd.txt | docker login -u hgsenel --password-stdin
                    docker build -t hgsenel/restapiex:v1.0 .
                    docker push hgsenel/restapiex:v1.0'''
                }
            }
        }
}
```

Burada, agent olarak istenen bir imaj Docker konteynner olarak da çalıştırılabilir.

# Sonuçlar

	gitindir	derle	konteyner
Average stage times: (Average <u>full</u> run time: ~35s)			
#5 Apr 02 03:58 No Changes	7s	1s	1min 3s
#4 Apr 02 03:57 No Changes	2s	792ms	1min 3s
#3 Apr 02 03:50 No Changes	3s	1s	
#2 Apr 02 03:46 No Changes	10s		
#1 Apr 02 03:46 No Changes	10s		

# Alternatif

- Agent olarak bir Docker imajı da çalıştırılabilir.
  - <https://www.jenkins.io/doc/pipeline/tour/hello-world/#examples>
  - <https://www.jenkins.io/pipeline/getting-started-pipelines/>

Jenkinsfile (Declarative Pipeline)

```
/* Requires the Docker Pipeline plugin */
pipeline {
    agent { docker { image 'python:3.10.7-alpine' } }
    stages {
        stage('build') {
            steps {
                sh 'python --version'
            }
        }
    }
}
```

# Örnekler

- Pipeline oluşturabilmek için Groovy'de script yazabilirsiniz. Groovy, bir programlama ortamı üzerinden size pipeline oluşturma olanağı sağlamaktadır:
- Jenkinsfile yazabilmek için aşağıdaki örnekleri kullanabilirsiniz:
- Örnekler için
  - <https://github.com/jenkinsci/pipeline-examples>