

CI/CD Jenkins

Detail Materi



Konsep Dasar Jenkins.

Indikator :
Dapat menjalankan konsep CI/CD
menggunakan Jenkins pada server testing dan
Server Production.



Instalasi Jenkins pada Host
dan juga Container



CI/CD Menggunakan Jenkins



Management Node Untuk
Membagi Beban.



Deployment Aplikasi pada Node

Modul Sekolah DevOps Cilsy

Hak Cipta © 2020 **PT. Cilsy Fiolution Indonesia**

Hak Cipta dilindungi Undang-Undang. Dilarang memperbanyak atau memindahkan sebagian atau seluruh isi buku ini dalam bentuk apapun, baik secara elektronik maupun mekanis, termasuk mecopy, merekam atau dengan sistem penyimpanan lainnya, tanpa izin tertulis dari Penulis dan Penerbit.

Penulis : Adi Saputra, Irfan Herfiandana, Tresna Widiyaman & Estu Fardani

Editor: Muhammad Fakhri Abdillah, Iqbal Ilman Firdaus

Revisi Batch 4

Penerbit : **PT. Cilsy Fiolution Indonesia**

Web Site : <https://cilsyfiolution.com> , <https://devops.cilsy.id>

Sanksi Pelanggaran Pasal 113 Undang-undang Nomor 28 Tahun 2014 tentang Hak Cipta

1. Setiap orang yang dengan tanpa hak melakukan pelanggaran hak ekonomi sebagaimana dimaksud dalam pasal 9 ayat (1) huruf i untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 1 (satu) tahun dan atau pidana denda paling banyak Rp100.000.000,00 (seratus juta rupiah).
2. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak cipta melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf c, huruf d, huruf f, dan atau huruf h, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 3 (tiga) tahun dan atau pidana denda paling banyak Rp500.000.000,00 (lima ratus juta rupiah)
3. Setiap orang yang dengan tanpa hak dan atau tanpa izin pencipta atau pemegang hak melakukan pelanggaran hak ekonomi pencipta sebagaimana dimaksud dalam pasal 9 ayat (1) huruf a, huruf b, huruf e, dan atau huruf g, untuk penggunaan secara komersial dipidana dengan pidana penjara paling lama 4 (empat) tahun dan atau pidana denda paling banyak Rp1.000.000.000,00 (satu miliar rupiah)
4. Setiap orang yang memenuhi unsur sebagaimana dimaksud pada ayat (3) yang dilakukan dalam bentuk pembajakan, dipidana dengan pidana penjara paling lama 10 (sepuluh) tahun dan atau pidana denda paling banyak Rp4.000.000.000,00 (empat miliar rupiah)



Daftar Isi

Cover.....	1
9. CI/CD Jenkins.....	4
Learning Outcomes.....	4
Outline Materi.....	4
9.1. WebHooks.....	5
9.1.1. Apa itu Webhooks ?.....	5
9.2. <i>Setup</i> Webhooks Jenkins.....	5
9.2.1. <i>Setup</i> Github Webhooks.....	5
9.2.2. <i>Setup</i> Webhook Jenkins.....	7
9.2.3. <i>Setup</i> POL SCM dengan aplikasi Golang (Iris).....	12
9.2.4. Exercise.....	15
9.2.5. Exercise.....	15

9.

CI/CD Jenkins

Learning Outcomes

Setelah selesai mempelajari bab ini, peserta mampu :

1. Memahami Konsep Dasar Jenkins.
2. Menginstall Jenkins pada Host.
3. Melakukan CI/CD Menggunakan Jenkins.
4. Melakukan Management Node Untuk Membagi Beban.
5. Melakukan Deployment Aplikasi pada Node.

Outline Materi

1. Webhooks
2. *Setup* Webhooks pada Jenkins

9.1. WebHooks

9.1.1. Apa itu Webhooks ?

Webhooks adalah layanan yang memungkinkan kita membuat atau mengatur aplikasi GitHub yang meng-*subscribe* ke *event* tertentu di GitHub.com. Ketika salah satu dari *event* tersebut di *trigger*, maka akan mengirim payload HTTP POST ke URL konfigurasi webhook. Webhook dapat digunakan untuk memicu pembuatan CI, memperbarui *backup mirror*, atau bahkan men-*deploy* ke *server production*.

Webhooks dapat dipasang pada repositori tertentu. Setelah dipasang, webhook akan men-*trigger* setiap kali satu atau beberapa *event* dari *subscriber* terjadi. Kita dapat membuat hingga 20 webhook untuk setiap *event* pada setiap target pemasangan (organisasi spesifik atau repositori khusus).

Dengan menggunakan layanan webhooks ini kita bisa menghubungkan GitHub dengan Jenkins sehingga ketika kita melakukan *push program* ke GitHub, maka pada saat itu juga Jenkins akan ter-*trigger* dan melakukan *deploy* pada *server* yang kita sudah set sebelumnya.

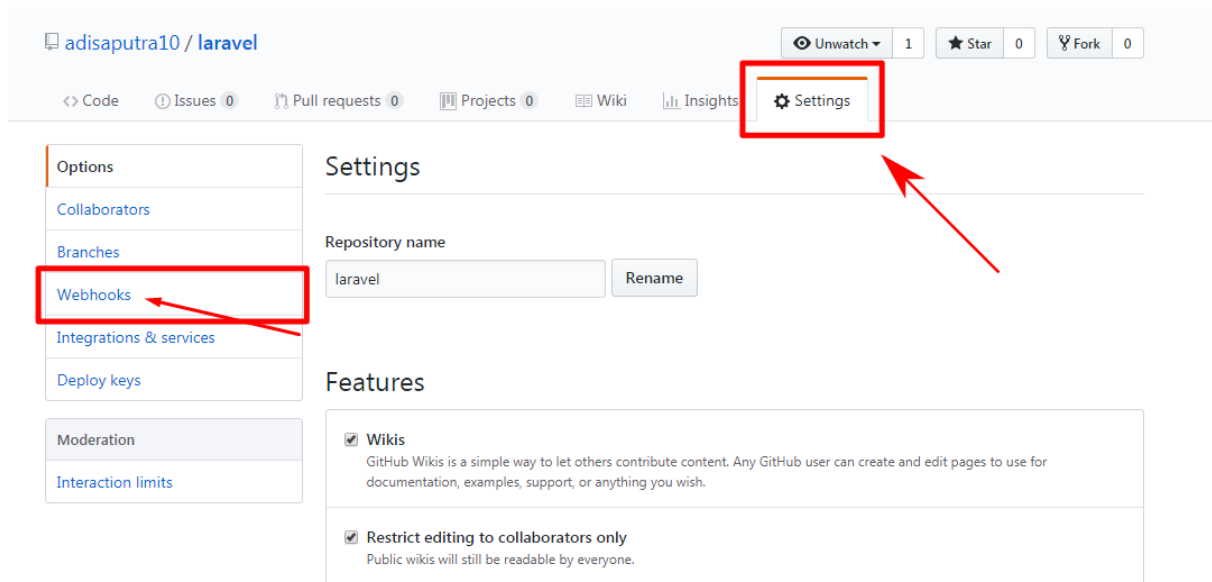
9.2. Setup Webhooks Jenkins

9.2.1. Setup Github Webhooks

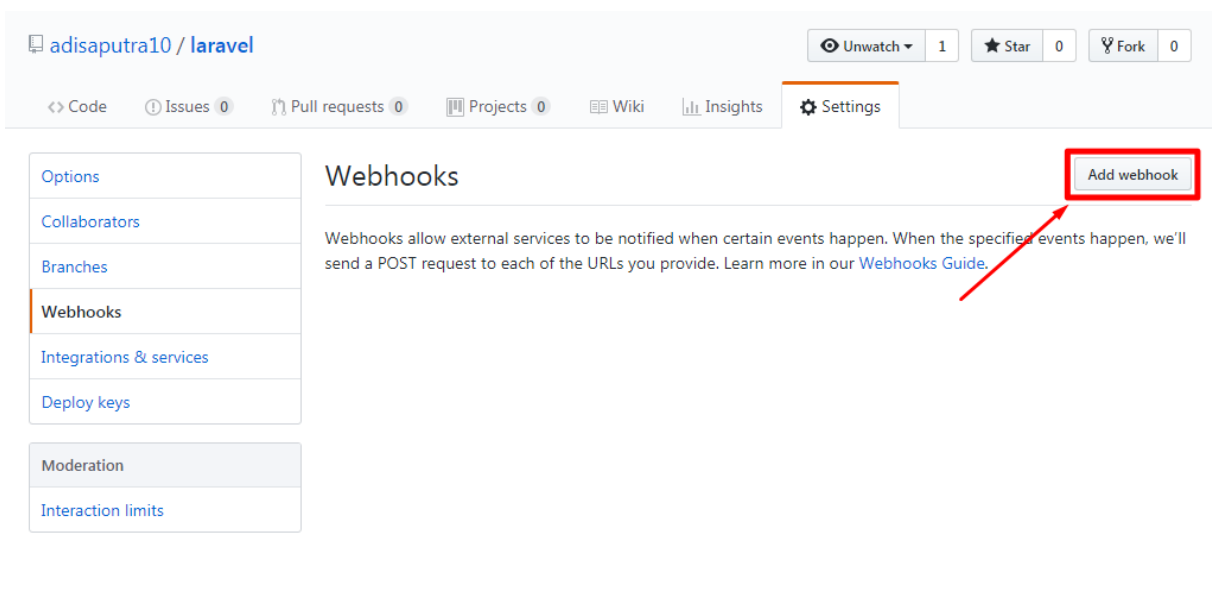
Pada bagian ini kita akan coba melakukan **setup webhooks** pada **github** yang akan kita hubungkan pada *server* Jenkins yang kita miliki. Kita akan coba melakukan *trigger*, dimana ketika kita melakukan push pada Github maka pada saat itu Jenkins tersebut akan otomatis melakukan *deploy* pada *server*.

Aplikasi yang akan kita coba deploy adalah aplikasi laravel yang sudah kita *push* ke *repository* github. Kalian dapat men-*clone* isi dari *repository* ini ke *repository* kalian yang baru sebagai sampel aplikasi laravel <https://github.com/adisaputra10/laravel.git>.

Pertama buka Repository kita yang ada di github. Kemudian setelah itu klik tombol **Setting** yang ada disamping atas selanjutnya masuk kedalam menu **Webhooks** yang ada disamping kiri menu.

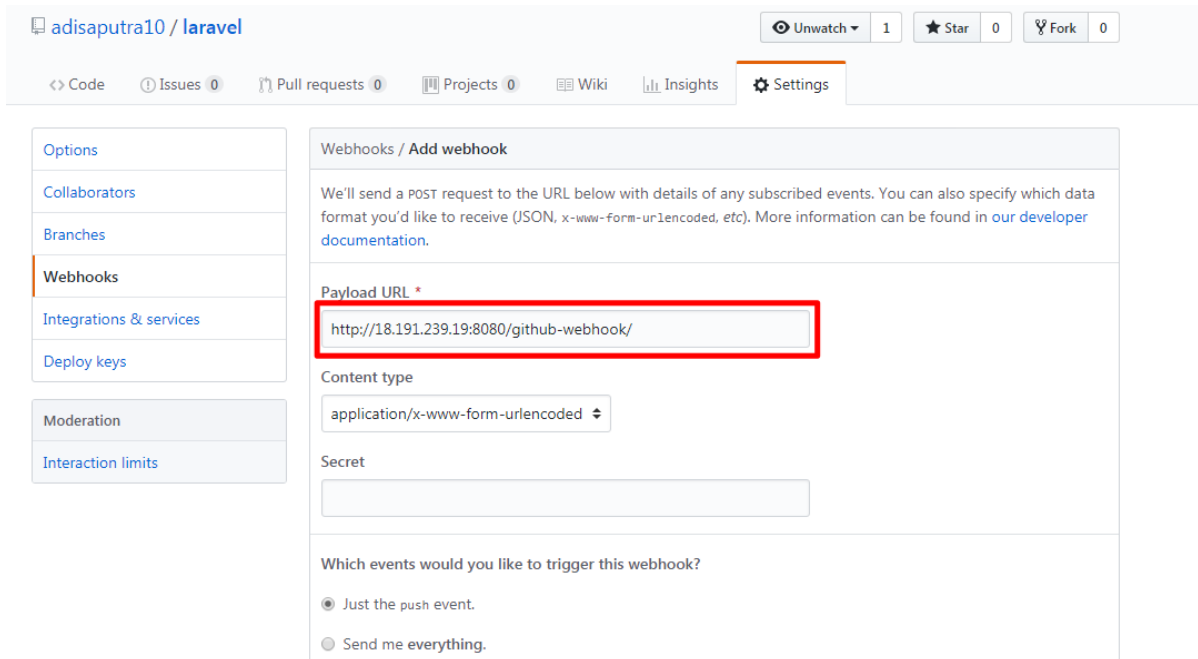


Selanjutnya klik tombol **Add webhook** yang ada disamping kanan atas untuk menambahkan alamat jenkins untuk menghubungkannya ke GitHub.



Kemudian tambahkan alamat URL dari jenkins yang kita miliki dengan menambahkan **/github-webhooks/** diujung alamatnya seperti dibawah ini.

Biarkan bagian **content type** dan **event trigger** secara *defaults*. Setelah itu simpan.



adisa Putra10 / laravel

Unwatch 1 Star 0 Fork 0

Code Issues 0 Pull requests 0 Projects 0 Wiki Insights Settings

Options
Collaborators
Branches
Webhooks
Integrations & services
Deploy keys

Moderation
Interaction limits

Webhooks / Add webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://18.191.239.19:8080/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

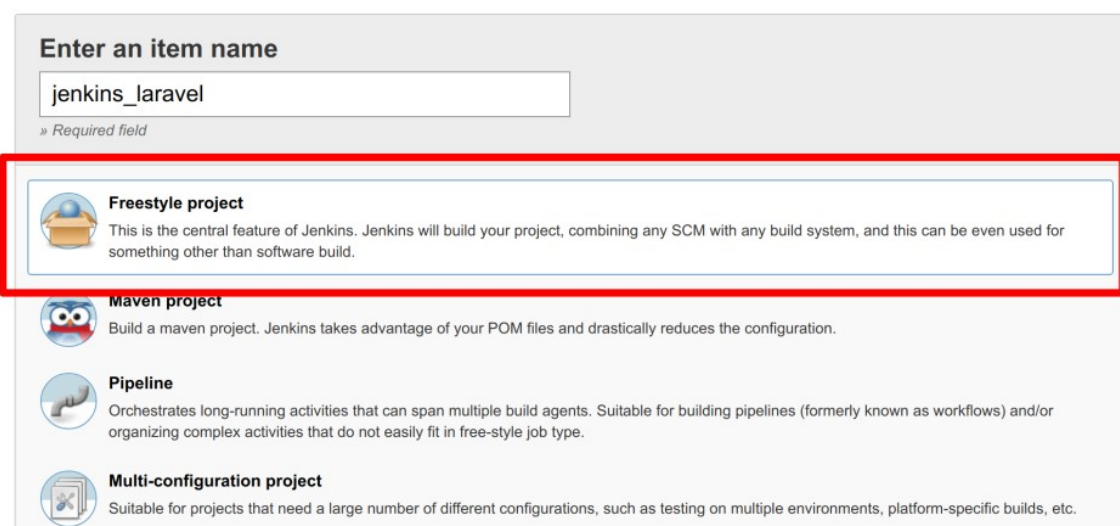
☐ Send me everything.

Url webhooks github

9.2.2. Setup Webhook Jenkins

Langkah selanjutnya kita akan melakukan konfigurasi pada **Jenkins**, maka kita akses dan *login* ke jenkins yang kita miliki. Setelah itu kita buat *item* baru.

Masukan nama *task* yang akan kita buat, misalkan disini kita buat dengan nama **jenkins_laravel**. Kemudian pilih **freestyle** sebagai itemnya, setelah itu klik **OK**.



Enter an item name

jenkins_laravel

» 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.

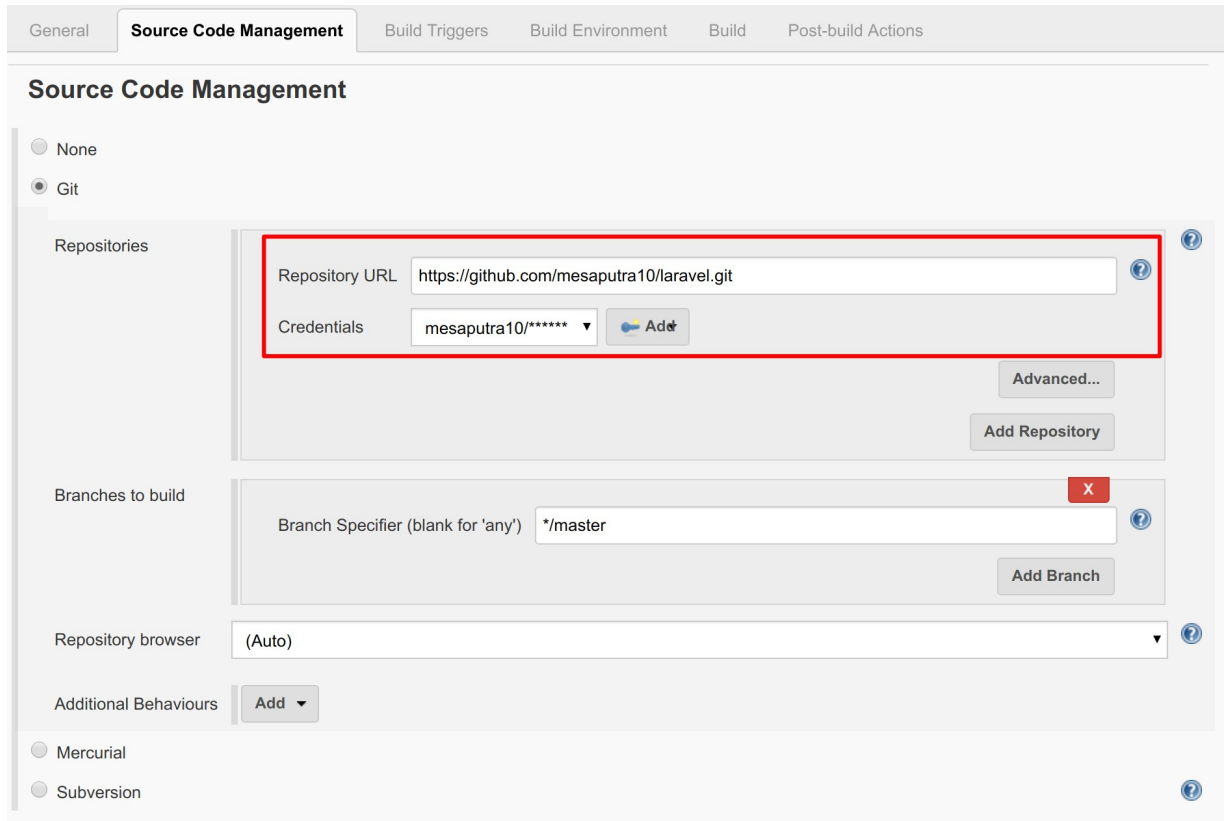
Maven project
Build a maven project. Jenkins takes advantage of your POM files and drastically reduces the configuration.

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.

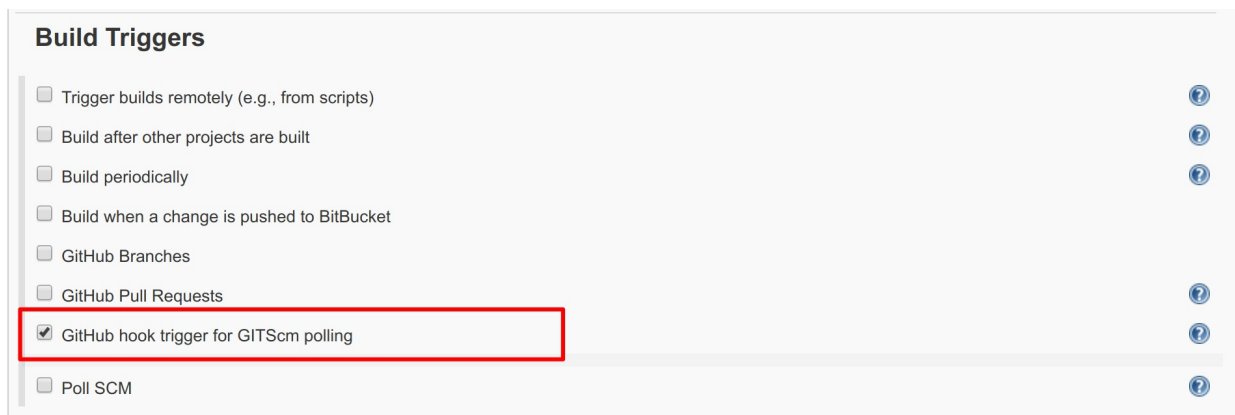


Pada bagian **Source Code Management**, pilih Git Kemudian copy URL dari repository aplikasi laravel yang kita miliki yaitu <https://github.com/adisaputra10/laravel.git>. Pilih juga **Credentials** yang akan kita gunakan.



The screenshot shows the Jenkins 'Source Code Management' configuration page. The 'Git' option is selected. The 'Repository URL' is set to 'https://github.com/mesaputra10/laravel.git'. The 'Credentials' dropdown is set to 'mesaputra10/*****'. The 'Branches to build' section has a 'Branch Specifier' of '*/master'. The 'Repository browser' is set to '(Auto)'.

Pada bagian **Build Triggers**, centang option **GitHub hook trigger for GITScm polling** seperti gambar dibawah ini.



The screenshot shows the Jenkins 'Build Triggers' configuration page. The 'GitHub hook trigger for GITScm polling' checkbox is checked and highlighted with a red box.

Scroll kebagian bawah, pada bagian Build klik **Add build step** lalu tambahkan **Execute shells** baru.

Selanjutnya masukan *script* untuk *build* aplikasi/docker seperti dibawah ini.

```
docker build . -t adisaputra10/laravel:$BUILD_NUMBER
docker push adisaputra10/laravel:$BUILD_NUMBER
```



Beralih ke server Jenkins, Pastikan docker sudah ter-*install* pada server jenkins dan jangan lupa login pada docker dengan menggunakan perintah berikut.

```
docker login
```

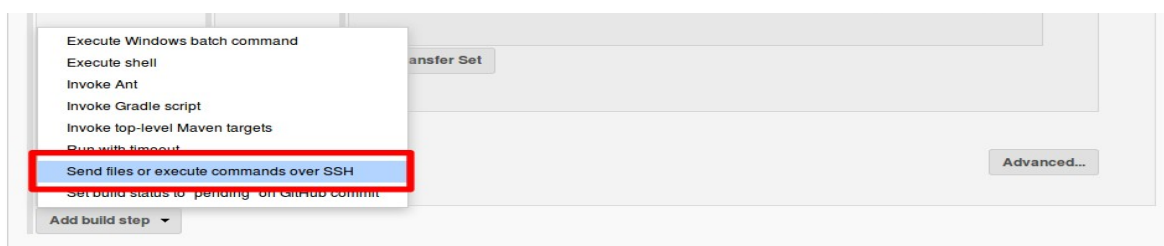
Masukan *user* dan *password* dari akun docker hub yang kita miliki, apabila kita belum memiliki akun tersebut kita bisa buat akun terlebih dahulu di <https://hub.docker.com>.

```
root@ip-172-31-0-15:/home/ubuntu# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: adisaputra10
Password:
Login Succeeded
root@ip-172-31-0-15:/home/ubuntu#
```

Setelah itu masukan juga *user* ubuntu dan jenkins kedalam *group* docker agar kita bisa mengakses tanpa perintah sudo.

```
usermod -aG docker ubuntu
usermod -aG docker Jenkins
```

Selanjutnya klik kembali tombol **Add build step**, setelah itu pilih **Send files or execute commands over SSH**.



Kita akan menjalankan docker tersebut pada *server* dengan menggunakan perintah dibawah

```
docker pull adisaputra10/laravel:$BUILD_NUMBER
docker rm -f laravel
docker run -dit --name laravel -p 80:8000 adisaputra10/laravel:$BUILD_NUMBER
```

Sesuaikan pengisian form pada jenkins seperti dibawah ini.

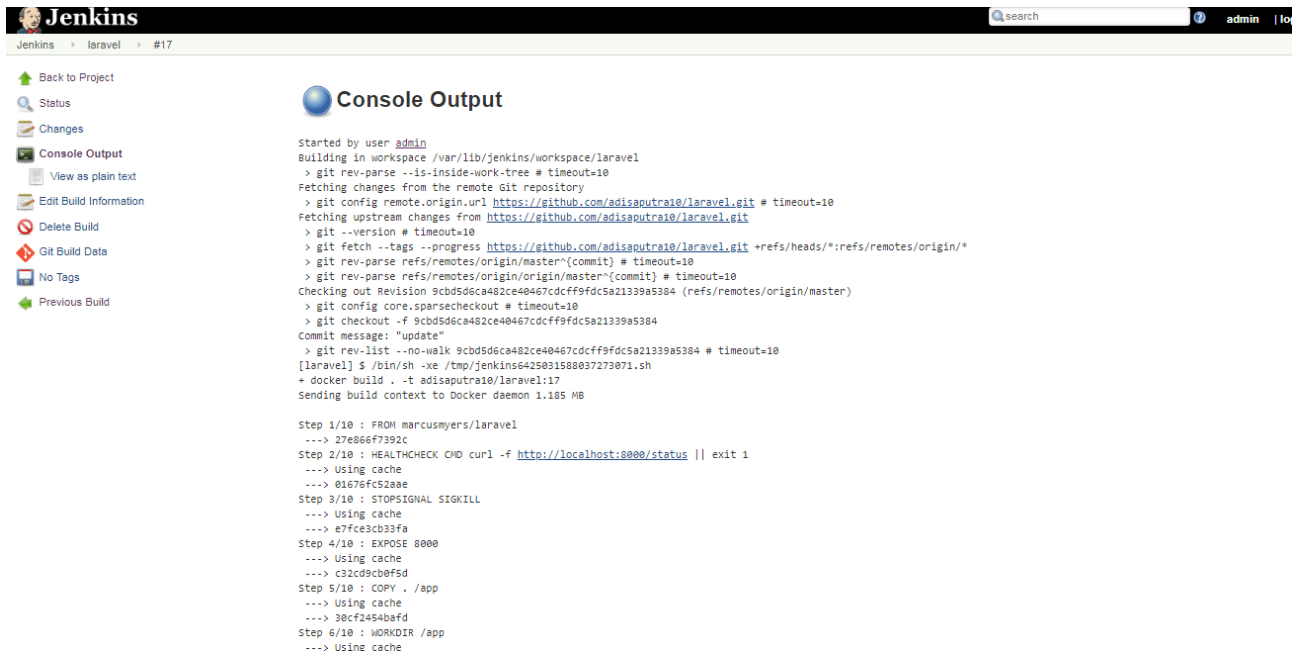
Setelah itu simpan *setting-an* yang sudah kita buat tersebut, maka akan muncul seperti dibawah ini hasilnya.

All						
S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
		jenkins_laravel	10 min - #9	23 min - #6	1.4 sec	

Selanjutnya kita coba **mengubah code di repository** dan melakukan **push** pada **GitHub**. Maka dengan otomatis github akan *men-trigger* ke jenkins, sehingga jenkins akan melakukan *build* dengan sedirinya seperti dibawah ini.

Build History		trend
	#9	Jul 15, 2018 8:13 AM
	#8	Jul 15, 2018 8:12 AM
	#7	Jul 15, 2018 8:09 AM

Jika kita ingin melihat log secara detail, kita hanya perlu klik salah satu nomor di atas, kemudian klik menu **console output** disamping kiri, dan apabila *deployment* berhasil maka hasilnya akan seperti di bawah ini.



The screenshot shows the Jenkins web interface. On the left sidebar, the 'Console Output' option is selected. The main area displays the console output for a build named 'laravel' with ID '#17'. The output shows the build process starting by user 'admin', cloning the repository from GitHub, checking out the latest commit, and building the application using Docker. The build steps include: FROM marcusmyers/laravel, HEALTHCHECK, STOPSIGNAL, EXPOSE, COPY, and WORKDIR. The build is successful and the context is sent to the Docker daemon.

```

Started by user admin
Building in workspace /var/lib/jenkins/workspace/laravel
> git rev-parse --is-inside-work-tree # timeout=10
> git rev-parse --progress https://github.com/adisaputra10/laravel.git # timeout=10
> git config remote.origin.url https://github.com/adisaputra10/laravel.git # timeout=10
> git fetch --tags --progress https://github.com/adisaputra10/laravel.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 9cbd5d6ca482ce40467cdcf9f9dc5a21339a5384 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f 9cbd5d6ca482ce40467cdcf9f9dc5a21339a5384
Commit message: "update"
> git rev-list --no-walk 9cbd5d6ca482ce40467cdcf9f9dc5a21339a5384 # timeout=10
[laravel] $ /bin/sh -xe /tmp/jenkins6425031588037273071.sh
+ docker build . -t adisaputra10/laravel:17
Sending build context to Docker daemon 1.185 MB

Step 1/10 : FROM marcusmyers/laravel
--> 27e866f7392c
Step 2/10 : HEALTHCHECK CMD curl -f http://localhost:8000/status || exit 1
--> Using cache
--> 01676f6c52a0e
Step 3/10 : STOPSIGNAL SIGKILL
--> Using cache
--> e7fce3cb33fa
Step 4/10 : EXPOSE 8000
--> Using cache
--> c32cd9cb0f5d
Step 5/10 : COPY . /app
--> Using cache
--> 30cf2454ba9d
Step 6/10 : WORKDIR /app
--> Using cache
  
```

Setelah berhasil kita lakukan *deploy*, coba kita akses *server* tujuan yang kita *deploy* aplikasi laravel tersebut dengan menggunakan *port* 8000. seperti **IPSERVER:8000**. Jika berhasil, maka hasilnya akan seperti di bawah ini.



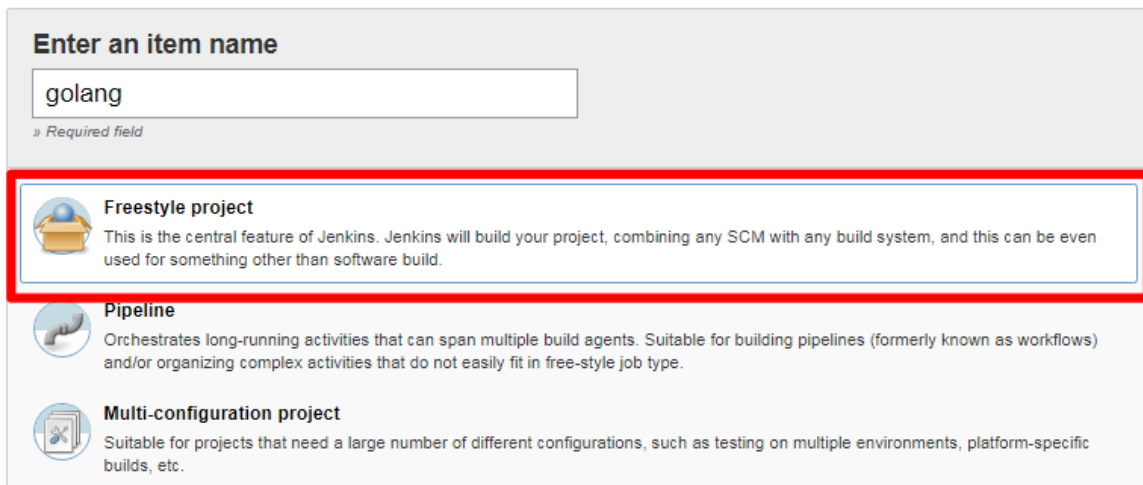
Hasil simple web laravel dengan webhooks



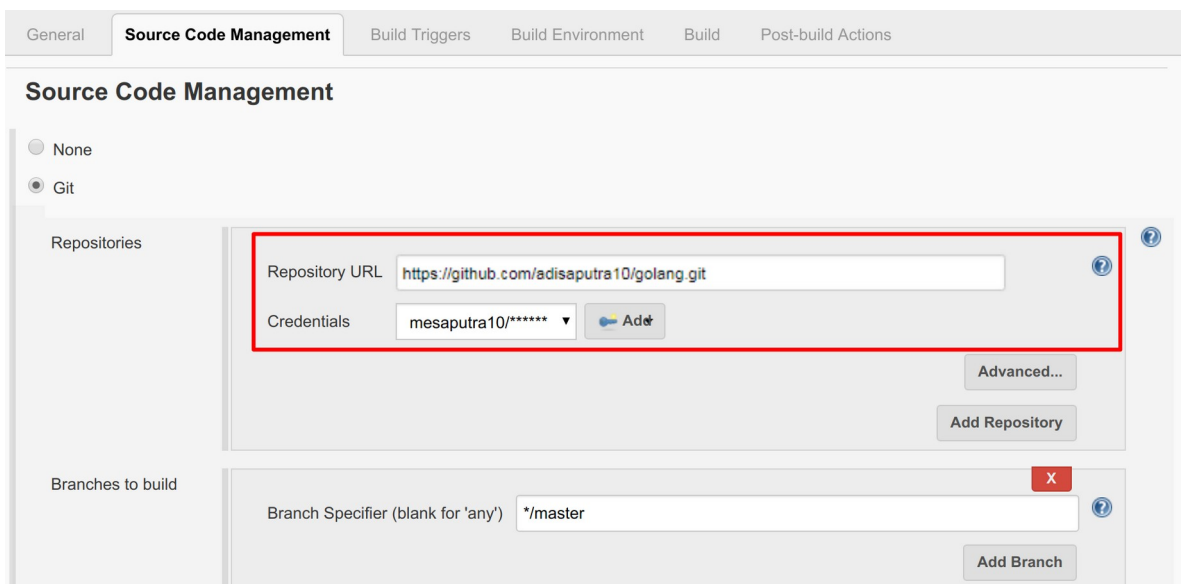
9.2.3. Setup POL SCM dengan aplikasi Golang (Iris)

Selain daripada **webhook**, jenkins sendiri memiliki **POL SCM** yang dapat digunakan sebagai trigger. Perbedaan webhook dan POL SCM adalah sumber *trigger*, jika kita menggunakan POL SCM maka trigger dari jenkins akan diatur dengan waktu tertentu untuk membaca repository jika ada perubahan. Sedangkan webhook melakukan trigger berdasarkan push dari para *programmer*.

Berikut merupakan langkah untuk mengatur **POL SCM** di jenkins, pertama kita buat sebuah item baru pada jenkins dengan **Freestyle Project**.



Selanjutnya pilih Git pada bagian **Source Code Management**, isikan **Repository URL** dengan alamat <https://github.com/adisaputra10/golang.git>. Pilih juga **Credentials** dengan yang sudah kita buat sebelumnya.



Pada bagian *Build Triggers*, centang **Poll SCM** kemudian isi dengan (* * * * *).

Pada bagian *Build*, Tambahkan **Execute Shell** dengan perintah dibawah ini.

```
docker build . -t adisaputra10/golang:$BUILD_NUMBER
docker push adisaputra10/golang:$BUILD_NUMBER
```

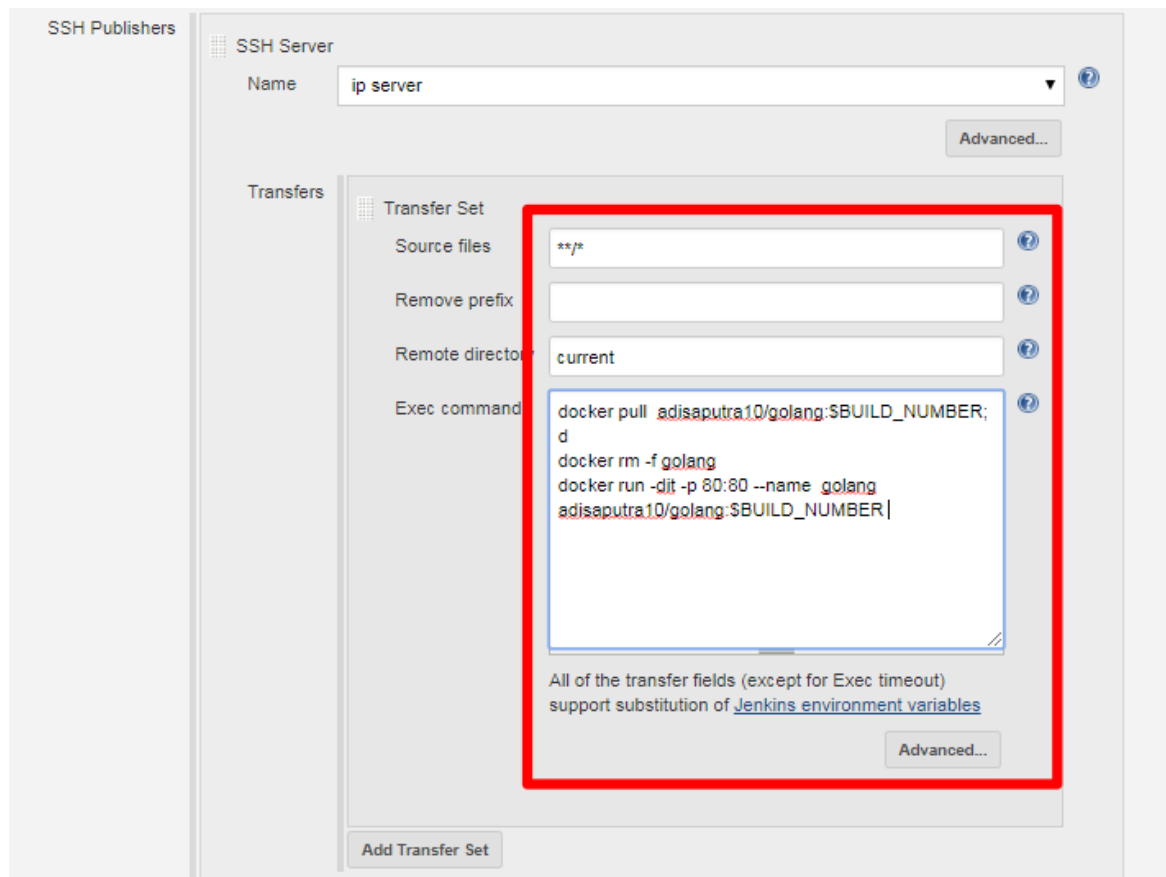
Maka hasilnya akan muncul seperti dibawah ini.

Selanjutnya klik kembali tombol **Add build step**, setelah itu tambahkan **Send files or execute commands over SSH**. Dan pada bagian exec commands, isikan script dibawah ini.

```
docker pull adisaputra10/golang:$BUILD_NUMBER;
docker rm -f golang
docker run -dit -p 80:80 --name golang adisaputra10/golang:$BUILD_NUMBER
```



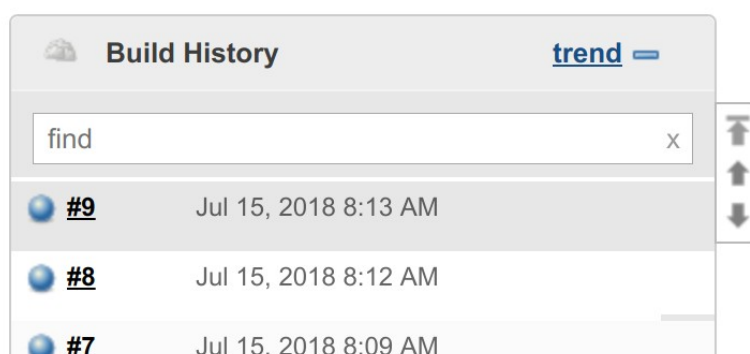
Maka hasilnya bisa kita sesuaikan seperti gambar dibawah ini.



Kemudian simpan konfigurasi yang sudah kita buat. Maka hasilnya akan menjadi seperti dibawah ini.

All						
S	W	Name ↓	Last Success	Last Failure	Last Duration	
		golang	N/A	N/A	N/A	

Selanjutnya kita coba mengubah *code* di *repository* dan melakukan *push* pada GitHub. Maka dengan otomatis github akan *men-trigger* ke jenkins, sehingga jenkins akan melakukan *build* dengan sedirinya seperti dibawah ini.



Jika ingin melihat detail dari *log build history*, kita bisa klik salah satu nomer yang ada diatas.

9.2.4. Exercise

Soal Teori :

1. Dari kedua percobaan yang telah dilakukan diatas, apakah yang membedakan antara Webhooks dan POLL SCM ?

Soal Praktek :

1. Buat Sebuah Automatisation menggunakan Webhook dan POLL SCM dengan menggunakan konten aplikasi pesbuk dan juga landing page.
2. Buat 5 commit baru pada github, sehingga muncul 5 history build dari Continue Delivery Jenkins pada server.

9.2.5. Exercise

Praktek :

1. Buat sebuah item baru pada jenkins untuk webhook dengan menggunakan aplikasi pesbuk dan juga landing page.
2. Build kedua aplikasi tersebut pada cluster yang sama.
3. Pastikan bahwa kedua aplikasi sudah berjalan dengan baik.