Katkıda Bulunanlar: Gökhan Sevinç, Mehmet Durmaz, Naime Damataşı,Ömer Selim Gürel

# SSH Kaydı

TMM-DEV-T08 SSH Oluşturma ve Kullanma talimatına göre SSH anahtarınızı oluşturup, reponuzun olduğu hesaplara oluşturduğunuz ve Webhostin hesaplarına girişini yapınız.

# ORTAMINIZI AYARLAYINIZ

## GIT ayarlarını listele

$ git config --list

## Uzaktaki VCS'ni repository'i alarak "git global setup" yapalım

$ git config --global user.name "xxxxxxxx"

$ git config --global user.email xxxxxxxxxxxx@temax.us

$ git config --global http.sslverify false

## Remote ayarları yapılır

### Origin (Kendi forkunuz)

(Kendi VCS hesabınıza göre uyarlayınız)

$ git remote add origin https://gitlab.temax.us:1590/<username>/hotcoldshop.git

ör: git remote add origin <https://gitlab.temax.us:1590/mehmetdurmaz/hotcoldshop.git>

Origin’e projeyi kendi hesabımızda forkladığımız ssh

### Upstream (Ana repo)

$ git remote add upstream https://gitlab.temax.us:1590/krautz-temax/hotcoldshop.git

Upstream’e ana proje ssh

### Auto.git için (Deploy için)

### git remote add hcs [hotcoldshopcom@ssh052.webhosting.be:auto.git](mailto:hotcoldshopcom@ssh052.webhosting.be:auto.git)

## Kullanıcı adı ve şifre girmeden işlem yapabilmek için

$ git remote add xxxxxx [https://<username>:<password>@gitlab.temax.us:1590/<username>/hotcoldshop.git](https://%3cusername%3e:%3cpassword%3e@gitlab.temax.us:1590/%3cusername%3e/hotcoldshop.git)

Xxxxx: origin veya upstream olabilir

# SİTE KLONUNUN ÇEKİLMESİ VE WORKFLOW İÇİN AYARLANMASI

## Forkladığınız site test kopyası klonlanır.

VCS kullanıcı adı ve şifrenizi isteyecek hazırlıklı olunuz

git clone <https://github.com/mehmetdurmaztemax/hotcoldshop.git>

cd ~/hotcoldshop

Remote ve Upstream yolu gösterilir (madde 1.3)

## Klonlama sırasında bir kesinti olursa ve yeniden denemeniz gerekirse aşağıdaki komutu giriniz

rm -rf ~/hotcoldshop

# BRANCH OLUŞTURMAK

## En önce Forkumuzun güncelliğini garantiye alalım

$ git checkout develop

$ git pull upstream develop

$ git push origin develop

## Conflictleri önlemek için habersiz olabileceğimiz değişiklikleri kontrol ederim

  $ git status

## Eğer değişiklikler varsa gönderelim ya da aşağıdaki seçeneklerden birini değerlendirelim

$ git stash

$ git rm <dosya ismi>

$ git restore <dosya ismi>

## Branch kontrolü yapalım developda oladuğumuzdan emin olalım

 $ git branch

## Eğer developta değilsek

  $ git checkout develop

## Orijinal repoyu indirelim

$ git pull upstream develop

## Kendi forkumuza pushlayalım

$ git push origin develop

## Yeni issue için branch açalım

$ git checkout -b HCSV2-5

# DEĞİŞİKLİKLERİ PUSHLAMAK

## Yapılan değişiklikleri ekle ve commitle

$ git add .

$ git commit -m "HCSV2-5 current HCS theme added"

## Merge request için pushla

 $ git push origin HCSV2-5

## Merge request linki gelecek, İmzacı olarak Merge yetkisi olanı yetkiliyi işaretleyelim

 Merge REQUEST ONAYI için bekle, eğer beklemeden başka branch açarsan conflict yaşarsın (konu composer.json dosyası ise)

## Onay gelince develop branchına geri dön ve KENDİ FORKUNU GÜNCELLE

 $ git checkout develop

$ git pull upstream develop

$ git push origin develop

# DEPLOYING

## STAGING alt sitesine deploy için

  Gitlab'e gir, developtan stagine merge request oluştur

$ git checkout staging

$ git pull upstream staging

$ git push hcs staging

## MASTER DEPLOY İÇİN

Gitlab'e gir, staginden mastera merge request oluştur

git checkout master

git pull upstream develop master

git push hcs master

# Lokalinizde ve forkunuzda kullanmadığınız branchleri silin

## Lokaldeki branchleri listeler

$ git branch

## Lokal branch silmek için

$ git branch -D <branch ismi>

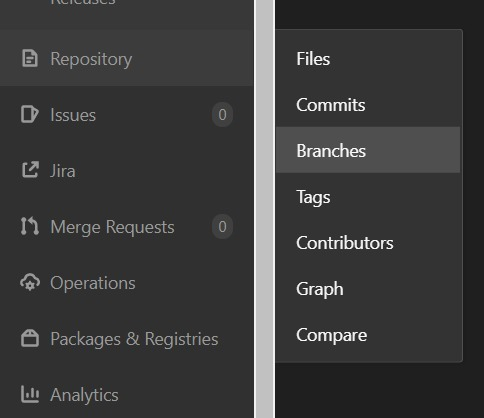
## Remote branchleri listeler

$ git branch -a

## Remote’deki atıl branchleri budamak için;

$ git remote prune origin

Eğer budanmadıysa gitlabdeki forkunuza uğrayın



Buraya girdiğinizde dallar çıkacak, dalların yanında kırmızı çöp kutusu da olacak, bu şekilde istemediğinizi silin, ardından yine

$ git remote prune origin

## Commitlerinizi tagleyebilirsiniz

$ git commit -m 'Composer package test' u

$ git tag v1.0.0

$ git git push origin master

$ git push origin v1.0.0

## Kazara eklediğin remote kaldırmak:

$ git remote remove origin

## Kazara git add işlemini geri almak

Kazanın vuku bulduğu branch’de olduğunuzdan emin olunuz

$ git reset

# En son yapılan commiti geri çekmek

**Dikkat bu işlemi workflowda yapmak ölümcül hatalara sebebiyet verebilir. En güzeli yaptığınız düzeltmeyi bir daha düzeltip güncelleme yapmaktır;**

Öncelikle işin buraya gelmemiş olması gerekir. Ama eğer gerekiyorsa;

1. Commit geçmişini tespit etme,
2. Gerekli commite geri çekme

İşlemleri yapılır.

## Commit geçmişini almak

git log

yada

$ git log --pretty=format:"%h %s" --graph

## Lokalde geriye çekmek

Burada commit id yerine en son iyi durumda oldunu bildiğiniz commit’e ait id’yi giriniz. Dikkat geri alma işlemiizden sonra geri dönemezsiniz…

git reset --hard <commidId> && git clean -f

Repoyu güncellemek

git push --force

# GIT CHEAT SHEET

Aşağıdaki adresten kaynak alarak çeşitli fikirler edinebilirsiniz:

<https://www.bojankomazec.com/2016/08/git-cheat-sheet.html>