# Mühazirə - 1. Fənnə giriş. Veb texnologiyaların inkişaf mərhələləri. İnternetin tarixi.

Veb texnologiyalar müasir dünyada insanların fəaliyyatində mühüm rol oynayan vəsitəyə çevrilmişdir. Artıq biz hayatımızı internetsiz təsəvvür edə bilmirik. İnternet layihələr insanlarln müxtəlif sahələrdə: tədrisdə, biznesdə, marketinqdə və s. vəcib platforma olmuşdur. Qısa müddət ərzində veb-texnologiyalar böyük sürətlə inkisəf edərək, qlobal informasiya fəzasının bütün istiqamətlərini əhatə edan Ümumdünya Hörümçək Toru (World Wide Web, WWW) yaratmışdır .

Virtual mühitdə toplanan informasiyanın tez-tez dəyisdirilməsi, tipinə, formatına, strukturuna və s. parametrlərə görə müxtəlif verilənlərdən təskil olunması veb texnologiyaların üzərinə əlavə vəzifələr qoyur. Məsələn, minlərlə veb-səhifəyə malik har hansı bir internet-layihədə səhifələrin, səhifədəki sözlərin, hiperistinadların, mediafaylların sayı tez-tez dəyisir. Belə bir səraitdə müasir veb texnologiyaların tədqiqatlarda təhlükəsizlik, zaman faktoru, tranzaksiya sürəti, verilənlərin tipi mühüm rol oynayır.

# MÜASİR WEB TEXNOLOGİYALAR

Hazırda bir çox veb texnologiyalar var ki, onların üzərinə düşən əsas vəzifələr kontentin təhlüəsizliyi, istifadəçilərin rahatlığı, dizayn və rahat interfeys kimi tələblərə cavab verməkdir. Hər veb texnologiya öz üslubunda yazıldığı proqram platforması üçün tələb olunan funksiyaları həyata keçirir. Hazırda bir çox veb proqramlaşdırma mövcuddur. Onlardan misal olaraq Hypertext Preprocessor (PHP), Active Server Pages (ASP.NET), Cold Fusion Markup Language (CFML), Javə, Python və s. göstərmək olar.   
Web proqramlaşdırmanın 2 növü vər: Server tərəfdə proqramlaşdırma, Klient tərəfdə proqramlaşdırma. Server tərəfdə proqramlaşdırma dilləri kimi əsasən PHP, PERL, JAVA, C++, ASP və digərlərindən istifadə olunur. Klient tərəfdə isə JavaScript, VbScript, JScript və s. bu kimi dillərdən istifadə olunsa da, bütün Web browserlər tərəfindən əsasən JavaScript problemsiz istifadə olunur.   
**ASP-NET.** Bu texnologiya müxtəlif sahələrdə istifadə olunmaqla müştərilərin və biznes təşkilətlarının məqsədlərinə nail olmaq üçün faydalıdır.

Müasir tələblərə uyğun veb səhifələrin yaradılması üçün istifadə olunan müasir nəsil veb texnologiyadır. Microsoft şirkəti tərəfindən yaradılıb. Həmçinin server proqramlaşdırma dili kimi tanınır. ASP.NET-i Microsoftun müasir nəsil veb texnologiyası kimi adlandırmaq da olar. ASP.NET proqramlaşdırma dili aşağıdakı proqram platformaların birləşmasi ilə həyata keçirilir

* Veb dizayn dili (HTML);
* JavəScript;
* C# .NET/VB.NET,
* Web Server (IIS/ASP.NET Development Server).

**PHP**. PHP proqramlaşdırma dilinin imkanları geniş və müxtəlifdir. Saysız skriptlərdən və müxtəlif komponentlərdən ibarətdir. PHP dilində kodlaşmada kodların ümumi bir stilə uygunlasdırllmasə və müxtəlif kitabxanalara problemsiz müraciəti bu texnologiyanın üstün cəhətlərindəndir. Proqramçı tərəfindən yazilən PHP kodu ümumi bir stilə gəlmədikdə avtomatik sorğu pəncərəsi açılır və PHP veb-layihələndirmədən başqa veb tətbiqlərin (application) inkişaf etdirilməsi üçün da geniş istifadə olunur.   
ASP.NET və PHP veb texnologiyaları arasmda an populyar olanlarıdır. Şəkil qalereyalarının tətbiq edilməsi bu iki texnologiya arasasında yaradılmışdır. Bu texnologiyalar arasında seçim etmək çətindir. Müəyyən təcrübəyə malik insanlar və bir çox digər amillər bu texnologiyalardan birgə istifadəyə üstünlük verirlər. Lakin əgər bu texnologiyalardan birini seçmək lazım gələrsə istifadəçinin nə istədiyi nəzərə alınmalıdır. Məsələn dəyəri, platforma mürəkkəbliyi, təhlükəsizliyi və digər xüsusiyyətlər seçim etməyə kömək ola bilər.

**CSS3** və **Javascript**. Nəzərə almaq lazımdır ki, müasir dövrdə istifadəçilərin əksəriyyəti stolüstü kompüter vəsitəsilə internetdə fəaliyyət göstərirlər. Bu gün noutbuklar, mobil telefonlar və planşetlər internet naviqasiya üçün geniş istifadə olunur və bu cihazların siyahısı artmaqda davam edir. Bu, veb tətbiqi dizaynerlərinin qarşısına ciddi tələblər qoyur, yani artıq əvvəlki veb proqramların bəziləri bugünki yeni texnologiyalarən tələblərinə cavab vermir . Proqram internet üzərindən multimedia vasitələrindən səmərəli istifadəni tamin edir. Nəzərə almaq lazımdır ki, veb texnologiyalar multimediya fayllarının əlyetərliliyi və paylanması üçün ən yaxşı platforma olub, multimedia materiallarının müxtəlif yollarla nümayişini təmin edir.   
Yuxarıda göstərilən proqram mahsulları müasir web texnologiyalarının tələblərinə müayyən qədər cavab versələr də, onların istifadəsində müəyyən proplemlər və çatışmazlıqlar mövcuddur.

# MÖVCUD PROBLEMLƏR

* Veb texnologiyaların inkişafı ilə əlaqəli olaraq internetdə müxtəlif tipli verilənlərin həcmi hər an artmaqdadır. Bu mühitdə toplanan verilənlərin səmərəli saxlanması və emalı məsələsi ön plana keçir. Bu problemlərin öhdəsindən gəlmək üçün internet üzərində buludlar, elektron kitabxanalar, arxivlər və s.veb layihələr yaradılır. Lakin, buna baxmayaraq yaradılan yeni layihələr problemi tam həll etmək gücündə deyillər.
* Veb texnologiyalarında asas məsələlərdən biri da təhlükəsiz Veb-layihələrin yaradılmasıdır. Araşdırmalar göstərdi ki, müasir veb-texnologiyalarda informasiyanın təhlükasizlik məsələsi tam həllini tapmamışdır. Bir çox həllarda informasiyanın təhlükəsizliyi məsələlərini qismən də olsa həll etmək üçün müxtəlif proqram platformalarından birgə istifadəyə üstünlük verilir. Əsasən da reklam xarakterli spamlarla bağlı problemlər daha aktualdır.
* Müasir veb texnologiyalar istifadəçilərin tələblərini ödəyə biləcək veb səhifələr yaratmaq üçün yetərli görünsə də proqram platformasının mürəkkəbliyi bəzi problemlər yaradır. Belə ki, veb-layihələrdə axtarış sistemlərinin daha yaxşı təşkil olunması, semantika və verilənlarin strukturlasdırılması problemləri mövcuddur.
* Müasir veb texnologiyalarda axtarış sistemlərinin düzgün təşkili üçün verilənlər bazasının idarəetmə sistemi səmərəli təşkil olunmalıdır ki, bu da çoxlu vəsait tələb edir. Əksər axtarış sistemləri böyük xərclərdən qaçmaq üçün Google və Yahoo-nun bazalarından istifadə edir.
* Bir çox hallarda veb-layihələrin təhlükəsizlik məsələləri dizayn məsələlərini arxa plana atır. Lakin, nəzərə almaq lazlmdır ki, internet istifadəçilərini cəlb etmək üçün dizayn məsələləri və veb-səhifələrdən rahat istifadə önəmlidir.

# GitHub bulud texnologiyasına giriş

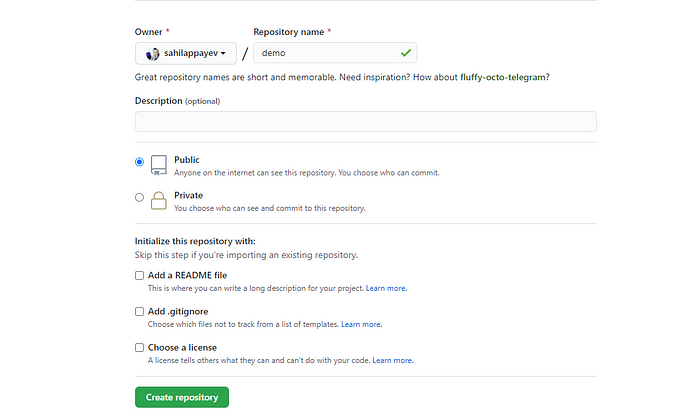
[**Git**](https://git-scm.com/) bir versiya kontrol sistemidir (Version Control System — VCS). Version control dedikdə nə nəzərdə tutulur? Biz kod yazmağa başladıqdan etibarən ilk faylımızı yaratmaqla onun ilk versiyasını yaratmış oluruq. Və hər dəfəsində onun üzərində dəyişiklik edib yaddaşda saxladıqca onun yeni versiyalarını yaradırıq. Bir fayl üzərində edilən dəyişikliklər yaddaşa qeyd edilmədiyi müddətdə həmin faylın yeni versiyası yaranmış sayılmır. Version control biz fayl üzərində hər dəyişiklik etdikdən sonra onun dəyişiklikdən əvvəlki halı və ya halları ilə sonrakı halı arasındakı fərqləri ortaya çıxarır.

Git arxitekturasından istifadə edən bir neçə serverlər mövcuddur. Bunlardan ən məşhurları [GitHub](https://github.com/), [GitLab](https://about.gitlab.com/), [Bitbucket](https://bitbucket.org/)-dir. Git texnologiyalarından istifadə edə bilmək üçün ik növbədə [git](https://git-scm.com/)i öz kompüterimizə qurmalıyıq. Bundan sonra hər hansı bir git serverində yazdığımız layihələri saxlaya biləcəyimiz özümüzə aid bir hesab açmalıyıq (bizim nümunəmizdə bu GitHub hesabıdır). Kompüterimizə qurduğumuz gitin “GitBash” proqram pəncərəsini açmaqla öz GitHub hesabımızı lokalımızdakı git bazası ilə əlaqələndiririk. Bunun üçün siz GitHub`da qeydiyyatdan keçdiyiniz emaildan istifadə etməklə GitBash pəncərəsində aşağıdakı əmrləri daxil etməlisiniz.

git config --global user.name "Ad Soyad"  
git config --global user.email [sizinGithubemailiniz@example.com](mailto:sizinGithubemailiniz@example.com)

Ardından siz doğrulamaq şərti ilə iki dəfə istifadəçi şifrənizi daxil edəcəksiniz (şifrə daxil edilərkən ekranda görünməyəcək). Bu mərhələdən sonra artıq siz GitHub serverindəki hesabınızı öz kompüterinizdəki git bazasına bağlamış olursunuz.

Giti kompüterimizdə qurub GitHub hesabımızla əlaqələndirdikdən sonra gəlin ilk remote [repository](https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/about-repositories)-imizi (repo) yaradaq. Bunun üçün biz GitHub hesabımıza daxil olub əsas səhifədə sol küncdə “Repositories” bölməsindən “New” düyməsinə sıxmaqla yeni repository yaratmaq üçün müvafiq səhifəni açırıq.



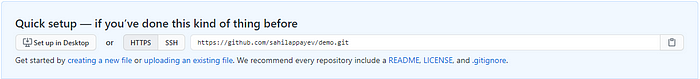
Burada yaradacağımız reponun adını (bizim nümunəmizdə: demo), proyektin hamıya açıq (public) və ya bağlı (private) olmağını təyin edə, proyekt haqqında məlumatları əks etdirən “README” faylını, proyektimizi repo-ya yüklədiyimiz zamanı yüklənməsini istmədiyimiz faylları qeyd etmək üçün “.gitignore” faylını və lisenziya sənədi əlavə edə bilərik.

LICENSE sənədi repoda saxladığınız sənədləri qoruyan lisenziyanızı müəyyənləşdirir. Onun şərtləri seçdiyiniz lisenziyadan asılı olaraq dəyişir. [Bu ünvanda](https://choosealicense.com/) 30 pulsuz lisenziyaya baxa bilərsiniz.

GitHub repo yaratdığınız zaman bu lisenziyalardan birini seçməyə imkan verir və avtomatik olaraq bu lisenziyanı repoya əlavə edir. Hər kəs lisenziya icazə verdiyi və lisenziyada göstərilən şərtlərə əməl edildiyi müddətdə lisenziyadan istifadə edə bilər. Pulsuz lisenziyaların mənası buradan qaynaqlanır. Lisenziyalar pulsuzdur ki, qoruduqları şeylər pulsuz olsun.

Description bölməsində isə proyektiniz haqqında qısa açıqlama yaza bilərsiniz.

**I üsul**



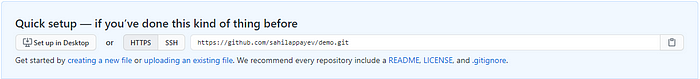
Bundan sonra “Quick setup” bölməsində HTTPS seçərək (SSH seçilərsə SSH kodu generasiya edilməli və Github-a əlavə edilməlidir, “passphrase” mütləq yadda saxlanmalıdır) repomuzun GitHub ünvanını kopyalayır, kompüterimizdə remote repo-nu yadda saxlamaq istədiyimiz yerə kontekst menyudan “Git Bash Here” düyməsini sıxmaqla GitBash terminalını açırıq və

**git clone *kopyaladigimiz\_unvan*** yazırıq. Beləliklə remote repomuz lokal kompüterimizə əlavə edilir. Bundan sonra lokal repoda edilən dəyişiklikləri remote repoya yükləmək üçün ardıcıl olaraq tək-tək aşağıdakı əmrləri yerinə yetirməliyik.

|  |  |
| --- | --- |
|  | git add .  git commit -m "My first commit" |
|  | git push |

Kodların izahı aşağıda verilmişdir.

**II üsul**



Əgər lokal kompüterinizdə hazır repo varsa və bu reponu remote repo ilə əlaqələndirmək lazımdırsa aşağıdakı qaydada edilməlidir:

“Quick setup” bölməsində HTTPS seçərək (SSH seçilərsə SSH kodu generasiya edilməli və Github-a əlavə edilməlidir, “passphrase” mütləq yadda saxlanmalıdır) repomuzun GitHub ünvanını kopyalayır, kompüterimizdə proyektimiz olan qovluqda (lokal repo) kontekst menyudan “Git Bash Here” düyməsini sıxmaqla git bash terminalını açırıq və

**git clone *kopyaladigimiz\_unvan*** yazırıq. Beləliklə remote repomuz lokal kompüterimizə əlavə edilir. Bundan sonra lokal repoda edilən dəyişiklikləri remote repoya yükləmək üçün ediləcək Lokal repomuzu yaratmaq üçün **git init**əmrini daxil edirik. Nəticədə gizli “.git” qovluğu yaranacaq ki, bu qovluq bizim gitlə bağlı bütün konfiqurasiya fayllarını və lokal repomuzu özündə saxlayır.

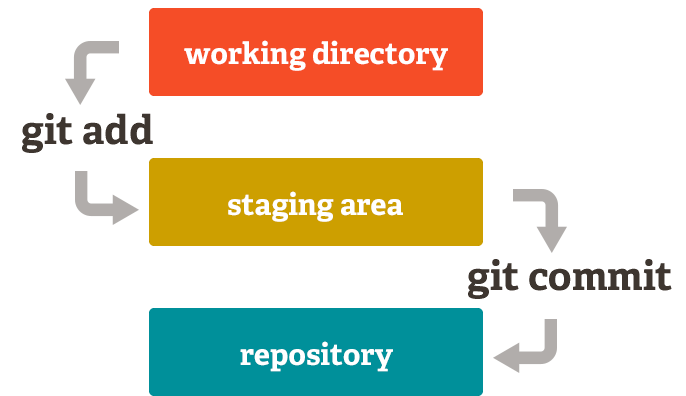
İndi isə yaratdığımız lokal ropomuzu remote repomuz ilə əlaqələnirməliyik. Bunun üçün ardıcıl olaraq tək-tək aşağıdakı əmrləri yerinə yetirməliyik.

|  |  |
| --- | --- |
|  | git add .  git commit -m "My first commit" |
|  | git branch -M master |
|  | git remote add origin https://github.com/sahilappayev/demo.git |
|  | git push -u origin master |

Burada, 2-ci sətirdə dırnaq işarələri ilə ayrılmış commit mətnini etdiyiniz əməliyyata uyğun olaraq təyin edirsiz və 4-cü sətirdəki url sizin remote github reponuzun ünvanı olmalıdır.

Repodakı fayllarımız gitin versiya nəzarəti altına keçmədən öncə üç mərhələdən keçir:

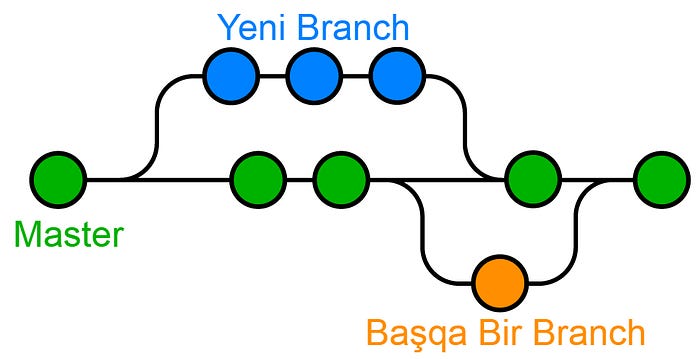
**Untracked**: fayl mövcuddur, lakin git versiyası nəzarətinin bir hissəsi deyil.  
**Staged**: fayl git versiyasını idarə etməyə əlavə edildi, lakin dəyişikliklər təslim olunmayıb.  
**Commited**: dəyişikliklər versiya nəzarəti üçün repoya təslim olunmuşdur.



Biz fayllarımız üzərində hər hansı bir dəyişiklik və ya yeni fayl əlavə etdikdə bu dəyişikliklər bizim “working directory”də qeydə alınır və bu zaman fayllarımız git tərəfindən untracked olaraq qeyd olunur. Biz **git status**əmriniverməklə faylların hansı mərhələdə olduğuna baxa bilərik. Bundan sonra untracked fayllarımızı staging area-ya əlavə etmək üçün **git add faylın\_adı\_uzantısı\_ilə**və ya bütün faylları əlavə etmək üçün **git add .**əmrini daxil edirik. Amma yenə də fayllarımız tam olaraq gitin nəzarəti üçün təslim olunmuş sayılmır, bunun üçün biz etdiyimiz dəyişiklikləri ehtiva edən qısa mesajla **git commit -m “commit mesajı”** əmrini verməklə təslimatı tamamlamalıyq. Artıq bizim fayllarımız tam olaraq lokal repoda gitin nəzarətindədir.

Bundan sonra biz **git push -u origin master**əmrini verməklə lokal repoda olan fayllarımızı remote repomuza yükləyə bilərik. Bu əmrdə **-u** parametrin ilk dəfə push edəndə verməklə lokaldakı branch-in müvafiq remote branch-ini təyin edirik və növbəti push proseslərində sadəcə olaraq **git push**əmri fayllarımızı müvafiq branch-a yükləməyə kifayət edəcək.

**Branch**— ingilis dilindən tərcümədə budaq, şaxə, qol mənasına gəlir. Bu hərfi mənalar gitdəki branch anlayışını demək olar ki, izah edir. Proyektimizi ilk dəfə git repoya əlavə edəndə onu master barnchı üzərindən push edirik və master branchı layihənin əsas qolu olur. Biz layihədə hər hansı yenilik edəndə master branchdan yeni branchlar yaradıb dəyişikliklərimizi həmin branchlar üzərində həyata kechirdirik və hər şey qaydasında gedərsə həmin branchları yenə master brancha **merge** edirik, yəni birləşdiririk.



Repomuzda mövcud branchlara baxmaq üçün repomuz olan ünvanda terminaldan **git branch** əmrini daxil etməliyik. Bu zaman mövcud olan bütün branchların siyahısı ekranda görünəcək(hal-hazırda aktiv olan branch xüsusi işarələnmiş şəkildə). Əgər yeni branch yaratmaq istəyiriksə **git branch yeni\_branch\_adi** şəklində əmr daxil edirik. Bu zaman yeni yaranmış branch bizim cari aktiv branchı istinad olaraq götürəcək onun bir kopyasını ehtiva edən bir şaxə yaradacaq. Biz yeni yaratdığımız brancha keçid etmək üçün isə **git checkout yeni\_branch\_adi** şəklində əmr daxil edirik. Və artıq bizim working directory yeni yaratdığımız brancha dəyişəcək və etdiyimiz bütün dəyişikliklər yeni branchda həyata keçəcək. Yuxarıda əmrləri ikisini özündə birləşdirən bir əmr də vardır ki, bu əmr yeni branch yaradır və cari iş sahəsini yeni yaranmış brancha keçirir: **git checkout -b yeni\_branch\_adi**

Lokalda yaratdığımız branchı remote-a daşımaq üçün **git push -u origin yeni\_branch\_adi** şəklində əmr daxil edirik və artıq lokalda yaratdığımız branch remote olaraq da yaranmış olacaq. Yaratdığımız branch üçərində nəzərdə tutduğumuz dəyişikliklər tamamlandıqdan sonra master və yaxud da birləşdirməyi nəzərdə tutduğumuz brancha keçid etdib (checkout) **git merge yeni\_branch\_adi** əmrini daxil edirik. Merge prosesi uğurla başa çatdıqdan sonra artıq yaratdığımız lokal və remote branchları silmək üçün müvafiq olaraq aşağıdakı əmrlərdən istifadə edirik.

**Git branch -d yeni\_branch\_adi**

**Git push origin --delete yeni\_branch\_adi**

Git haqqında danışmalı olduğumuz digər bir məsələ **git clone** əmridir. Hər hansı bir git reposu üzərində bir neçə şəxs və ya bir şəxs bir neçə kompüter üzərindən iş görürsə, bu zaman remote repodakı faylları başqa kömpüter üzərindən də iş görməyə əlçatan etmək üçün git clone əmrindən istifadə edirik. Bunun üçün clone etmək istədiyimiz reponun url ünvanını kopyalayıb yükləmək istədiyimiz qovluqda terminaldan **git clone reponun\_urli** şəklində əmr daxil edirik. Bununla da remote ropadakı proyektin bir kopyası bizim kompüterimizdə yaranmış olur.

Biz kompüterimizdəki fayllar üzərində yenilik etməzdən əvvəl remote repoda hər hansı yeniliklər varsa onları lokal repoya əlavə etmək üçün **git pull** əmrini daxil etməliyik. Əgər biz git pull əmrini daxil etmədən lokal repoda sonradan etdiyimiz dəyişikilikləri push edə bilməyəcəyik. Buna alternativ kimi force push anlayışı (git push -f origin branch\_adi) var ki, bu zaman remote repodakı commit-lər silinəcək və lokal repodakı dəyişikliklər remote repoya yüklənəcək.