

## درس توسعه نرم افزار

مدرس ثریا عنایتی شیراز

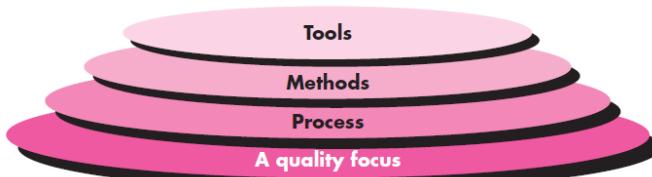
دانشگاه فنی حرفه ای سمنان

گروه کامپیوتر

## ۱- مهندسی نرم افزار به کمک کامپیوچر

مهندسی نرم افزار شاخه‌ای از مهندسی است که به تکامل محصول نرم افزاری با استفاده از اصول، تکنیک‌ها و روش‌های علمی کاملاً تعریف شده مربوط می‌شود. بنا براین مهندسی نرم افزار "تولید یک محصول نرم افزاری موثر و قابل اعتماد" است

مهندسي نرم افزار يك فن آوري لايه اي است.



با توجه به شکل سنگ بنای نگهدارنده مهندسی نرم افزار، توجه به کیفیت است. اساس مهندسی نرم افزار، لایه فرایند است که چارچوبی را برای تحويل موثر فناوری مهندسی نرم افزار تعریف می‌کند. فرایند نرم افزار، پایه ای برای کنترل مدیریتی پروژه‌های نرم افزاری تشکیل داده، بستری برای اعمال روش‌های فنی، تولید محصولات کاری (مدل‌ها، مستندات، داده‌ها، گزارشات، فرم‌ها و غیره)، تعیین مراحل، حصول اطمینان از کیفیت و مدیریت مناسب تغییرات ایجاد می‌کند.

روش‌های مهندسی نرم افزار، شیوه‌های فنی برای ساخت نرم افزار را فراهم می‌آورند. این روش‌ها شامل آرایه وسیعی از وظایف از جمله: تحلیل خواسته‌ها، طراحی، ساخت برنامه‌ها، آزمایش و پشتیبانی می‌شود.

ابزارهای مهندسی نرم افزار، متناسب پشتیبانی خودکار یا نیمه خودکار برای فرایند و روش‌ها هستند. هنگامی که ابزارها گرد هم آیند به طوری که اطلاعات ایجاد شده توسط یک ابزار، توسط ابزارهای دیگر قابل استفاده باشند، سیستمی برای پشتیبانی نرم افزار شکل می‌گیرد که مهندسی نرم افزار به کمک کامپیوچر نام دارد.

مهندسي نرم افزار به کمک کامپیوچر : ابزارهای مهندسی نرم افزار، متناسب پشتیبانی خودکار یا نیمه خودکار برای فرایند و روش‌ها هستند. هنگامی که ابزارها گرد هم آیند به طوری که اطلاعات ایجاد شده توسط یک ابزار، توسط ابزارهای دیگر قابل استفاده باشند، سیستمی برای پشتیبانی نرم افزار شکل می‌گیرد که مهندسی نرم افزار به کمک کامپیوچر نام دارد

<sup>1</sup> Computer Aided Software Engineering

## Stages in an SDLC (Software Development Life Cycle)



<sup>۱</sup> software development life cycle

## فاز اول تجزیه و تحلیل نیازمندی ها

اولین گام هر SDLC، تعریف نیازمندیهای پروژه است. برخی از سوالات مهم در این مرحله عبارتند از:

- هدف از پروژه جدید چیست؟
- انتظار کسب و کار از محصول چیست؟
- آیا تیم قصد دارد از ابتدا کد بنویسد یا قرار است سیستم موجود را ارتقا می دهیم؟
- آیا محدودیت هایی داریم؟
- آیا ما دانش لازم را در داخل داریم یا باید بخشی از پروژه را برون سپاری کنیم؟

این مرحله نیاز به تلاش ترکیبی از تجزیه و تحلیل تجاری، عملیات، رهبری، توسعه و تیم های امنیتی دارد. در برخی موارد مورد استفاده، درخواست از کاربران نهایی برای ورودی نیز منبع ارزشمندی از اطلاعات است. تمام داده های جمع آوری شده در این مرحله به بصورت سند مشخصات نیازمندیهای نرم افزار (SRS) تنظیم می شود. یک فایل SRS شامل مشخصات نرم افزار، سخت افزار، امنیت و شبکه برای محصول آینده و همچنین:

- تخصیص منابع.
- برنامه ریزی ظرفیت.
- زمانبندی پروژه.
- برآورد هزینه.
- تامین.

**خروجی تجزیه تحلیل نیازمندیها:** یک سند SRS که اهداف و محدوده پروژه را تعریف می کند، به علاوه نیازمندیهای محصول و برآوردهای تقریبی پروژه (بودجه، منابع، محدودیت ها<sup>3</sup> و غیره) را ارائه می دهد.

## فاز دوم مطالعه امکان سنجی

تحلیلگران ارشد کسب و کار یک مطالعه امکان سنجی برای تعیین قابلیت اجرا نرم افزار انجام می دهند. رویکرد معمول تمرکز بر این پنج عامل است:

- محدودیت های بودجه
- پیامدهای قانونی.
- نیازمندیهای عملیاتی.
- مهارت های موجود در خانه
- بازه زمانی مورد نیاز پروژه

تحلیلگران یافته های این مرحله را به سند SRS موجود اضافه می کنند و پس از آن تیمی از تصمیم گیرندگان گزارش را برای تأیید بررسی می کنند:

- برنامه ها و جهت پروژه.
- هزینه های تخمینی

<sup>3</sup> deadlines

برنامه های پیش بینی شده

منابع لازم.

مدیریت بالاتر یا پروژه را امضا می کند یا از تیم می خواهد که یک قدم در SDLC به عقب برگردند و پیشنهاد جدیدی ارائه دهند.

**خروجی این مرحله:** یک سند گسترش یافته SRS که توسط مدیریت بالاتر تایید شده است.

### فاز سوم طراحی طرح

بعد از تایید پروژه تیم شروع به ایجاد طراحی طرح می کند طوری که تمام جنبه های اصلی محصول جدید از جمله موارد زیر را توضیح می دهد:

معماری (زیان برنامه نویسی، پایگاه های داده، رابط ها، سیستم عامل، قالب های از پیش ساخته شده، API ها و غیره).

لیست ویژگی ها

نیاز مندیهای زیرساختی

طراحی UI.

اقدامات امنیتی لازم (به عنوان مثال، رمزگذاری SSL، حفاظت از رمز عبور، انتقال توصیه شده پایگاه داده، و غیره).

تیم اطلاعات را در مشخصات سند طراحی (DDS) جمع آوری می کند. سپس DDS جهت تایید موارد زیر مورد بررسی قرار می گیرد:

ماژولار بودن طراحی

ارزیابی ریسک

استحکام محصول

محددیت های زمانی.

برخی از شرکت ها تصمیم به ایجاد یک نمونه اولیه در این مرحله SDLC می گیرند شاید نمونه سازی زمانبر باشد اما نمونه سازی بسیار کم هزینه تر از ایجاد تغییرات اساسی پس از مرحله توسعه است.

**خروجی مرحله طراحی:** یک DDS دقیق که تمام اطلاعات مورد نیاز توسعه دهنده گان برای کدنویسی محصول را فهرست می کند.

### فاز چهارم توسعه نرم افزار

تیم توسعه براساس DDS کار روی کد را آغاز می کند. به طور معمول، این مرحله زمان برترین مرحله SDLC است، بنابراین توصیه می کنیم از متداول‌تری های چابک برای سرعت بخشیدن به کدنویسی استفاده کنید.

این مرحله منجر به نرم افزار عملیاتی می شود که تمام نیازمندیهای ذکر شده در SRS و DDS را برآورده می کند. در این مرحله تست های اولیه محصول (مانند تجزیه و تحلیل کد استاتیک و بررسی کد انواع مختلف دستگاه) انجام می شود و محصول آماده برای تست پیشرفته است.

**خروجی مرحله توسعه :** کد منبع یک نرم افزار قابل آزمایش و کاملا کاربردی.

## فاز پنجم تست عمیق نرم افزار

نرم افزاری که از فاز قبلی SDLC خارج می شود اکنون آزمایش های گسترده ای را پشت سر می گذارد. مانند:

- تست کیفیت کد
- تست واحد (تست های عملکردی).
- تست یکپارچه سازی
- تست کارایی
- تست امنیتی
- تست پذیرش
- تست غیر کاربردی

اکثر تیم ها برای سرعت بخشیدن به این مرحله به تست های خودکار تکیه می کنند، اما برخی از تست های دستی نیز ارزشمند هستند (تست های نفوذ نمونه خوبی هستند).

اگر تیم نقصی را کشف کند، کد یک مرحله از SDLC خود به عقب باز می گردد و توسعه دهندهان یک نسخه جدید و بدون نقص از نرم افزار ایجاد می کنند. مرحله آزمایش زمانی به پایان می رسد که محصول پایدار، عاری از اشکال و مطابق با استانداردهای کیفیت تعریف شده در مراحل قبلی باشد.

**خروجی مرحله تست:** یک نسخه کاملاً آزمایش شده از محصول آماده برای محیط تولید.

## فاز ششم استقرار نرم افزار

پس از مرحله تست محصول آماده تولید است تا به مشتری تحویل داده شود بنابراین در برخی از پروژه ها به تیم نیاز دارند تا قبل از اینکه نرم افزار در دسترس کاربران نهایی قرار گیرد، کتابچه راهنمای کاربر را بنویسند یا فیلم های آموزشی ایجاد کنند.. و نیز محصول تحویل داده شده را ارزیابی کرده و براساس این ارزیابی بازخورد ارائه دهد.

در حالت ایده آل، مرحله استقرار به طور خودکار اتفاق می افتد (معمولًا به عنوان بخشی از<sup>۴</sup> CI/CD). و یا ممکن است شرکت های پایین تر به تأییدیه های دستی در این مرحله SDLC نیاز داشته باشند.

CI/CD مجموعه ای از اقدامات است که مراحل ساخت، آزمایش و استقرار توسعه نرم افزار را خودکار می کند. اتوماسیون زمان بندی تحویل را کاهش می دهد و قابلیت اطمینان را در طول چرخه عمر توسعه افزایش می دهد.

**خروجی مرحله استقرار :** یک محصول کاملاً کاربردی و آزمایش شده که در دسترس کاربران نهایی است.

<sup>4</sup> continuous delivery or continuous deployment

<sup>5</sup> continuous integration

## فاز هفتم نگهداری و بهبود محصول

هر نرم افزار ارسال شده نیاز به بررسی و به روز رسانی دوره ای بر اساس بازخورد کاربر دارد. متدائل ترین فعالیت ها در این مرحله عبارتند از:

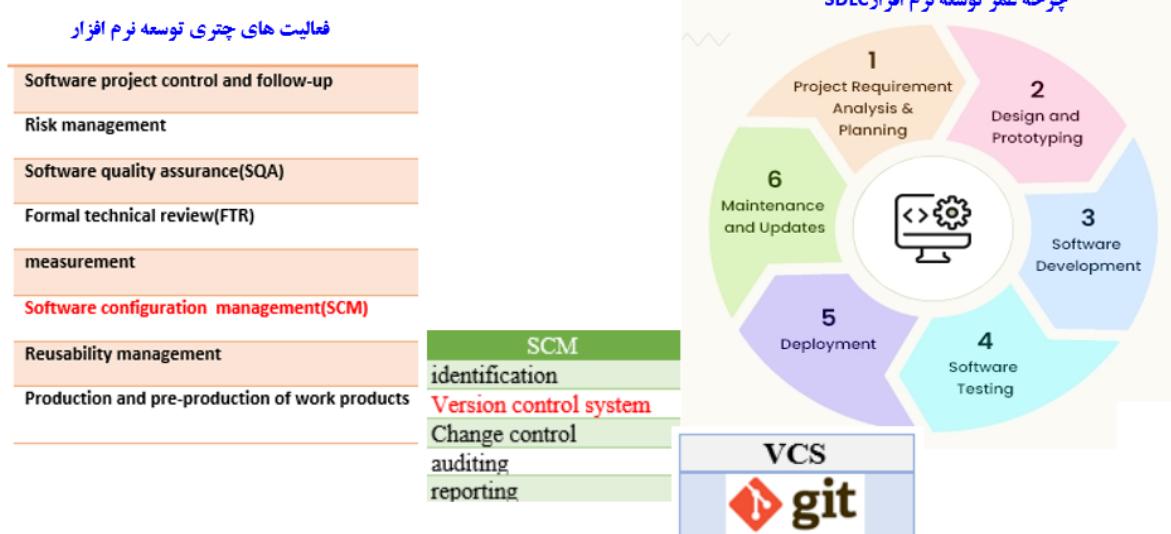
- رفع اشکال
- راه اندازی نظارت مستمر
- ارتقاء برنامه به نسخه جدیدتر.
- افزودن امکانات جدید به نرم افزار

هر زمان که کاربر اشکالی را گزارش می کند یا تیم نقص جدیدی را کشف می کند، محصول از طریق SDLC خود به تعداد مراحل لازم به عقب بر می گردد. برخی از نقص های شدید نیاز به روز رسانی در مرحله طراحی دارند، در حالی که اکثر مشکلات برنامه را به مرحله توسعه بازمی گرداند.

**خروجی مرحله نگهداری و بهبود محصول** : محصولی با نظارت کامل که به طور مداوم شاهد پیشرفت است.

## □ مدیریت پیکربندی نرم افزار

SCM فرآیندی است که به سازمان ها کمک می کند تا تغییرات محصولات نرم افزاری را در طول چرخه عمر خود کنترل و مدیریت کنند. نیاز به SCM از این واقعیت ناشی می شود که محصولات نرم افزاری اغلب پیچیده هستند و شامل بسیاری از اجزای مرتبط به هم هستند. SCM کمک می کند تا اطمینان حاصل شود که تغییرات یک مؤلفه به طور ناخواسته سایر مؤلفه ها را خراب نمی کند و می توان نسخه های مختلف نرم افزار را نگهداری و رديابی کرد.



<sup>6</sup> Software configuration management

### ۳- ابزارهای توسعه نرم افزار<sup>7</sup>

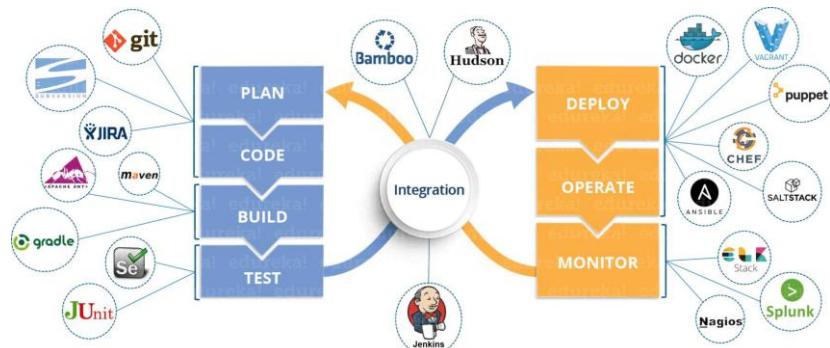
ابزارهای توسعه نرم افزار برنامه های کامپیوتری هستند که توسط توسعه دهنده گان نرم افزار برای ایجاد، ویرایش، نگهداری، پشتیبانی و اشکال زدایی سایر برنامه ها استفاده می شود. ابزارهای توسعه می توانند به اشکال مختلفی مانند پیوند دهنده، کامپایلر، ویرایشگر کد، طراح رابط کاربری گرافیکی، اسمبلر، دیباگر، ابزار تجزیه و تحلیل عملکرد و غیره باشند. بر اساس نوع پروژه، عوامل خاصی وجود دارد که باید در هنگام انتخاب ابزار توسعه مریبوطه در نظر گرفته شوند.

#### مزایای استفاده از ابزارهای توسعه نرم افزار

- ابزارهای نرم افزاری برای دستیابی و بررسی فرآیندهای تجاری، مستندسازی فرآیند توسعه نرم افزار و بهینه سازی تمامی فرآیندها استفاده می شوند.
- با استفاده از این ابزارها در فرآیند توسعه نرم افزار، نتیجه پروژه ها پریارتر خواهد بود.
- با استفاده از ابزارهای توسعه، یک توسعه دهنده می تواند به راحتی گرددش کار پروژه را حفظ کند.

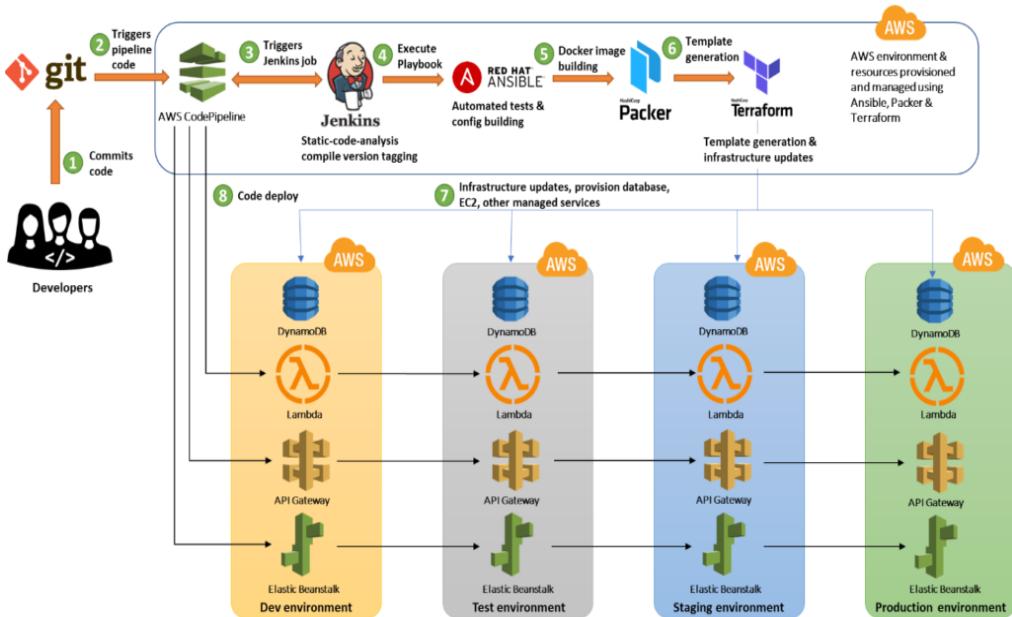
#### مهندسی نرم افزار با روش Devops

اصطلاح DevOps از دو واژه Development به معنی «توسعه» و Operations به معنی «عملیات» ساخته شده است مخلوطی از چندین نقش بوده و هدف نهایی آن کنار هم قرار دادن دولوپرها و مهندسان اجرایی است. فرهنگ دوآپس ویژگی ها و قابلیت های جدید محصول را با زیرساخت های آن سازگار می نماید و سبب می شود تا این دو بتوانند در کنار هم به خوبی عمل کنند.



<sup>7</sup> software-development-tools

تصویر زیر به خوبی گویای وظایف هر کدام از این بخش‌ها است.



### مرحله طرح‌ریزی Planning

دو سرویس مختلف چگونه می‌توانند با هم در تعامل باشند؟

برای مرتبط کردن این دو سرویس از چه پروتکلی باید استفاده نمود؟

آیا سخت‌افزاری که در اختیار ما قرار دارد پاسخگوی نیازمان هست؟

برای اینکه بتوانیم در پروسه تولید به مهندسان کمک کنیم، نیاز به چه چیزهایی داریم؟

آیا سرویس مذکور به اصطلاح Production-Ready خواهد بود؟

آیا تمام دیپننسی‌های مورد استفاده در نرم‌افزار برای ما ملموس هستند؟

چه چیزی را لازم است بسازیم و چه چیزی را باید خریداری کنیم؟

آیا یک تَسک خاص را می‌توان به صورت خودکار انجام داد؟

چه طور می‌توان در آینده از این نرم‌افزار پشتیبانی نمود؟

### مرحله توسعه Development

چگونه می‌توانیم دولوپرها را در فضای مشابه فضای محصول نهایی نگاه داریم؟

چه طور به دولوپرها اجازه دهیم تا ابزارهای مورد علاقه خود استفاده کنند؟

چگونه می‌توانیم بهره‌وری و کارایی دولوپرها را افزایش دهیم؟

چه طور باید برای دولوپرها توضیح دهیم که محیط نهایی نرم‌افزار چگونه خواهد بود؟

### مرحله تست کردن TEST

چگونه می‌توان به اصطلاح چندین Client Environment تکرارپذیر ایجاد نمود؟

از کجا بدانیم تست مورد نظر در مورد کدام نسخه از سرویس در حال انجام است؟

چگونه تاریخچه تست‌ها را دنبال کنیم و با استفاده از آن به روندهای موجود پی ببریم؟

چگونه پس از تست نمودن کدها، مشکلات احتمالی را به دولوپرها اعلام کنیم؟

داده‌های تست را از کجا به دست آوریم؟

## مرحله استقرار Deploy

این اصطلاح به معنای آپلود کردن کدها روی سرور اصلی نرم افزار است. به طور کلی، این مرحله در مورد این است که کدهای نوشته شده چه طور و با چه نظمی در محصول نهایی قرار بگیرند تا کاربر نرم افزار قادر به استفاده از سرویس ما گردد. در این مرحله نیز مهندسان دو آپس از ابزارهای CI، مشابه آنچه که در بخش قبل معرفی شد، استفاده می‌کنند و بعضی از مهم‌ترین سوالاتی که در این مرحله باید پاسخ داده شوند عبارتند از:

چه زمانی یک نسخهٔ نهایی نشده از نرم افزار آمادهٔ دیپلوی شدن است؟

چگونه بدون اینکه کاربر متوجه شود، سرویسی را دیپلوی نماییم؟

چگونه مطمئن شویم سرویسی که به تازگی دیپلوی شده منجر به ایجاد اختلال نمی‌شود؟

چگونه فرآیند دیپلوی شدن را به صورت خود کار درآوریم؟

چگونه در صورت لزوم در فرآیند دیپلوی خود کار، مراحلی را به صورت دستی و غیر خود کار انجام دهیم؟

چگونه فرآیند دیپلوی را با روشی تکرار پذیر انجام دهیم؟

معمولًا این مرحله زمان زیادی را از مهندسان دو آپس نمی‌گیرد اما بخشی که این مهندسین باید بیشترین زمان و انرژی خود را صرف آن کنند، مرحله بعدی، Maintenance است.

## Maintenance

همان‌طور که گفتیم، مرحله نگهداری نرم افزار یکی از مراحلی است که بیشترین زمان یک مهندس دو آپس را به خود اختصاص می‌دهد و این فاز تمامًا در مورد انجام کارهایی است که در نهایت موجب در دسترس قرار گرفتن یک سیستم و حفظ کارایی آن می‌شوند. در این مرحله سوالاتی مانند موارد زیر باید پاسخ داده شوند:

چگونه می‌توانیم از مشکلات و باگ‌های موجود در محصول یا سرویس آگاه شویم؟

چگونه باگ‌های مختلف موجود در محصول یا سرویس را به تیم‌های مناسب ارجاع دهیم؟

چگونه باگ‌های زیرساختی موجود در محصول را برطرف کنیم؟

به عنوان یک مهندس دو آپس چگونه می‌توانیم از سلامت و کارایی همه سرویس‌ها مطمئن شویم؟

## معرفی ابزارهای برتر در هر فعالیت فرآیند توسعه نرم افزار

ابزارهای برنامه‌ریزی و مدیریت پروژه مانند:

Jira: یک ابزار مدیریت پروژه قدرتمند که به تیم‌ها کمک می‌کند تا وظایف، مسئولیت‌ها و پیشرفت پروژه را پیگیری کنند.

Trello: یک ابزار ساده و کارآمد برای سازماندهی و پیگیری وظایف و کارها در قالب جدولها و بوردها.

Asana: یک ابزار مدیریت پروژه و همکاری تیمی که امکان ایجاد وظایف، ردیابی پیشرفت و انجام کارها را فراهم می‌کند.

ابزارهای تحلیل و مدیریت نیازمندی مانند: Enterprise architect

■ ابزارهای طراحی و معماری نرم افزار مانند: wondershare-luchid chart-dbdiagram

■ ابزارهای کدنویسی/محیط توسعه یکپارچه (IDE)

- Visual Studio Code
- IntelliJ IDEA
- PyCharm
- Eclipse
- Xcode
- Android Studio
- Codenvy
- Dreamweaver

■ ابزارهای تست : برنامه نویسان برای اطمینان از عملکرد صحیح نرم افزارها از ابزارهای تست و اشکال زدایی استفاده می کنند. برخی از این

■ ابزارها عبارت اند از:

- Selenium
- JUnit
- Pytest
- Postman

■ ابزارهای مانیتورینگ مانند: ansible

■ ابزارهای اتوماسیون انتشار و بررسی عملکرد

○ Jenkins : یک ابزار CI/CD قدرتمند که به برنامه نویسان کمک می کند فرایند انتشار و تحویل نرم افزار را اتوماسیون کند.

○ Travis CI : یک سرویس CI/CD که به ایجاد، تست و تحویل نرم افزار برای پروژه های گیت هاب کمک می کند.

○ Docker : یک پلتفرم محفظه ای که به برنامه نویسان امکان می دهد برنامه ها را به صورت مستقل از سیستم عامل و محیط اجرایی اجرا کنند.

○ New Relic : یک ابزار مشاهده و نظارت بر عملکرد نرم افزار که به برنامه نویسان کمک می کند تا مشکلات عملکردی را تشخیص داده و بهبود عملکرد نرم افزار را ارائه دهند.

○ همچنین، برنامه نویسان ممکن است از ابزارهای متفاوتی برای مستندسازی کد، تست عملکرد، تجزیه و تحلیل داده ها و طراحی واسط کاربری استفاده کنند. ابزارهایی مانند JMeter ، Swagger و Adobe XD می توانند در این حوزه مفید باشند.

■ ابزارهای کنترل نسخه : این ابزارها برای مدیریت تغییرات در کد منبع استفاده می شوند و به توسعه دهنده گان امکان می دهند تغییرات را پیگیری کنند و با همکاران خود هماهنگ شوند. نمونه هایی از سیستم های کنترل نسخه عبارت اند از:

- GitLab
- GitHub
- Git
- Subversion (SVN)
- Mercurial

■ پلتفرم های پردازش ابری AWS Cloud 9

■ سرورها و محیط های اجرایی : برنامه نویسان برای اجرای نرم افزارها نیاز به سرورها و محیط های اجرایی خاصی دارند. برخی از آنها عبارت اند از:

○ Apache HTTP Server

Enayati.5751@gmail.com

Nginx ○  
Microsoft IIS ○  
Docker ○

کتابخانه‌ها و فریمورک‌ها: این ابزارها برای تسهیل توسعه نرم‌افزارها و استفاده از ویژگی‌ها و قابلیت‌های مشترک مورد استفاده قرار می‌گیرند.

برخی از محبوب‌ترین کتابخانه‌ها و فریمورک‌ها عبارت‌اند از:

React ○  
Angular ○  
Laravel ○  
Django ○  
TensorFlow ○  
PyTorch ○

برترین ابزارهای مستندسازی کد:

- **Java برای Javadoc:** یک ابزار استاندارد در جهت تولید مستندات API برای کدهای جاوا است. با استفاده از توضیحات خاصی که در کدهای جاوا نوشته می‌شود، Javadoc مستنداتی را به صورت HTML ایجاد می‌کند که شامل توضیحات کلاس‌ها، متدها، پارامترها و مقادیر بازگشته می‌شود.
- **Sphinx برای زبان‌های مختلف از جمله Python:** یک ابزار مستندسازی قدرتمند برای زبان‌های برنامه‌نویسی مختلف است. این ابزار به برنامه‌نویسان امکان می‌دهد مستنداتی را در قالب متن ساده مانند Restructuredtext بنویسند و سپس آنها را به فرمتهای مختلفی مانند HTML، PDF و اسناد ویکی تبدیل کنند.
- **Doxxygen Python:** یک ابزار مستندسازی کد منبع است که برای زبان‌هایی مانند C، C++, Java، Objective-C، Python و به کلاس‌ها، متدها، توابع و سایر عناصر را تولید می‌کند.
- **Swagger API:** یک فرمت استاندارد برای توصیف و مستندسازی API است. با استفاده از Swagger، می‌توانید مشخصات خود را تعریف کرده و مستنداتی جامع و قابل خواندن برای آن ایجاد کنید که شامل پارامترها، روش‌ها، پاسخ‌ها وغیره است.
- **JavaScript برای JSDoc:** یک ابزار مستندسازی برای کدهای جاوااسکریپت است که قابل استفاده در محیط‌های توسعه متن باز و بسته‌های npm است. با استفاده از توضیحات خاصی که در کد نوشته می‌شود، JSDoc مستنداتی را به صورت HTML ایجاد می‌کند که شامل توضیحات کلاس‌ها، توابع، متغیرها و پارامترها می‌شود.

#### ۴- سیستم کنترل نسخه یا<sup>8</sup> VCS

قبل از اینکه به VCS پردازیم، باید به سؤالات زیر پاسخ دهیم:

- آیا تاکنون تغییری در کد ایجاد کرده اید ، متوجه شده اید که این یک اشتباه بوده است و می خواهید دوباره برگردید؟
- آیا تاکنون مجبور شده اید نسخه های مختلف یک محصول را حفظ کنید؟
- آیا می خواهید ثابت کنید که یک تغییر خاص یک قطعه کد را خراب کرده است؟
- آیا تاکنون خواسته اید تاریخ برخی از کد ها را مرور کنید؟
- آیا تاکنون کد خود را گم کرده اید یا نسخه پشتیبان تهیه کرده اید که خیلی قدیمی است؟
- آیا تاکنون خواسته اید بینید چقدر کار انجام می شود ، و کجا ، کی و توسط چه کسی؟
- آیا تاکنون خواسته اید تغییری را در کد شخص دیگری ارسال کنید؟

<sup>8</sup> version control system

- آیا تاکنون خواسته اید که خود را به اشتراک بگذارید ، یا اجازه دهید افراد دیگر به طور همزمان روی کد شما کار کنند؟
- آیا تاکنون خواسته اید بدون دخالت در کد کار با یک ویژگی جدید آزمایش کنید؟

اگر به دلیل هر یک از شرایط فوق با مشکل مواجه شده اید ، استفاده از یک VCS به شما پیشنهاد می شود.

سیستم کنترل مجموعه ای از ابزارهای نرم افزاری است که به تیم کمک می کند تا تغییرات در کد منبع را مدیریت کند. تغییرات یک فایل یا مجموعه ای از فایل ها در طول زمان ردیابی می کند تا بتوانید نسخه های خاصی را بعداً فراخوانی کنید. همچنین به شما امکان می دهد با برنامه نویسان دیگر کار کنید. و یا از نوع خاصی از پایگاه داده برای پیگیری هر تغییر در کد استفاده می کند. بنا براین توسعه دهندهای می توانند نسخه های قبلی کد را با نسخه قدیمی تر مقایسه کنند تا اشتباهات را برطرف کنند.

#### □ **مزایای سیستم کنترل نسخه**

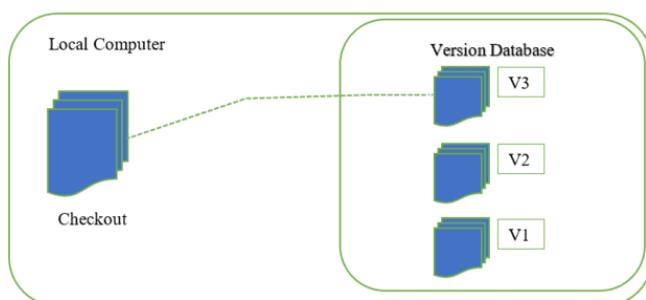
- توسعه نرم افزار امن است.
- برای عدم قطعیت پشتیبان تهیه می کند.
- یک رابط سریع به توسعه دهندهای ارائه می دهد.
- به تیم های نرم افزاری اجازه می دهد تا کارایی و چابکی را با توجه به مقیاس های تیم حفظ کنند تا توسعه دهندهای بیشتری را شامل شوند.
- تاریخچه تغییر کامل فایل
- کار کردن همزمان توسعه دهندهای
- انشعاب و ادغام
- قابلیت ردیابی

#### □ **انواع سیستم کنترل نسخه**

- سیستم کنترل نسخه محلی localized version control
- سیستم های کنترل نسخه مرکزی Centralized Version Control System
- سیستم های کنترل نسخه توزیع شده Distributed Version Control System

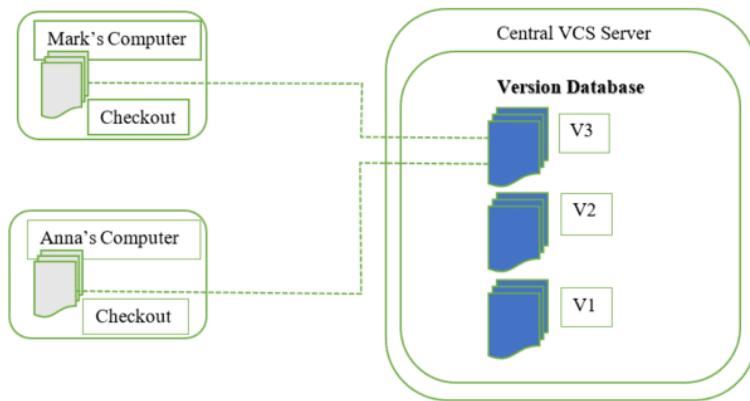
#### ▪ **سیستم کنترل نسخه محلی**

روش کنترل نسخه محلی به دلیل سادگی یک رویکرد رایج است. اما این روش منجر به احتمال بالاتر خطای شود. در این روش، ممکن است فراموش کنید که در کدام دایرکتوری هستید VCS های محلی دارای یک پایگاه داده ساده هستند. چنین پایگاه های اطلاعاتی تمامی تغییرات فایل ها را تحت کنترل بازنگری نگه می دارند. یک سیستم کنترل نسخه محلی، کپی های محلی فایل ها را نگه می دارد.



## ▪ سیستم های کنترل نسخه متمن کر

در این مدل ما یک سرور یا یک کامپیوتر اصلی خواهیم داشت که سورس اصلی پروژه روی آن قرار می گیرد (Remote Repository) و همه برنامه نویس ها (کلاینت ها) به آن متصل شده و یک کپی از سورس روی سیستم خود دریافت کرده و روی آن کار می کنند.

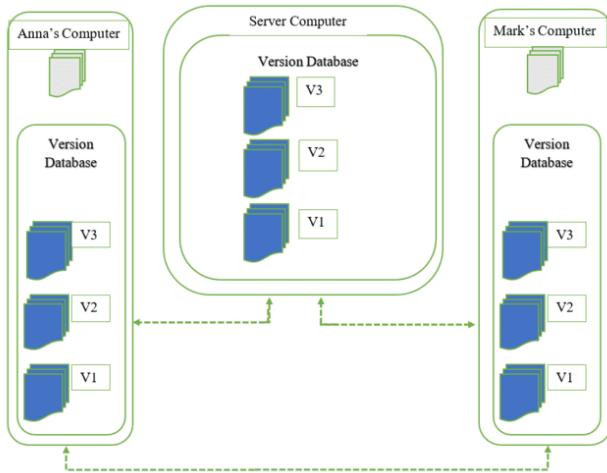


سیستم های کنترل نسخه متمن کر مزایای زیادی به خصوص نسبت به VCS های محلی دارند.

- همکاری تیم توسعه دهنده گان وجود دارد
- اطلاعاتی در مورد کاری که دیگران در پروژه انجام می دهند دارند.
- مدیران بر دیگر توسعه دهنده گان کنترل دارند.
- مقابله با یک سیستم کنترل نسخه متمن کر ساده تر از یک سیستم کنترل نسخه محلی است.
- سیستم کنترل نسخه محلی با یک جزء نرم افزار سرور که نسخه های مختلف فایل ها را ذخیره و مدیریت می کند

## ▪ سیستم های کنترل نسخه توزیع شده

سیستم کنترل نسخه متمن کر از یک سرور مرکزی برای ذخیره تمام پایگاه داده و همکاری تیم استفاده می کند. اما به دلیل خرابی تک نقطه ای که به معنای خرابی سرور مرکزی است، توسعه دهنده گان آن را ترجیح نمی دهند. بنابراین، سیستم کنترل نسخه توزیع شده توسعه یافت. در یک سیستم کنترل نسخه توزیع شده (مانند Git, Mercurial, Bazaar یا Darcs)، کاربر یک کپی محلی از یک مخزن دارد. بنابراین، مشتریان فقط آخرین عکس فوری فایل ها را بررسی نمی کنند، حتی می توانند به طور کامل مخزن را بررسی کنند. مخزن محلی شامل تمام فایل ها و ابرداده های موجود در مخزن اصلی است. در این روش ما دو Repository با نام های Local Repository بر روی سرور و Remote Repository بر روی کلاینت داریم که برنامه نویس می تواند تا مدت ها تغییرات و تاریخچه آنها را روی local repository خود داشته باشد و سپس آنها را به روی push ارسال یا remote repository کند.

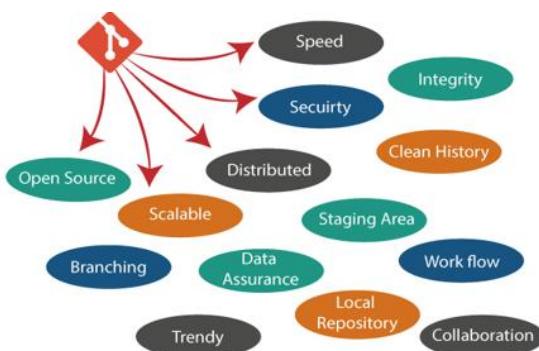


هر اکانت یک نسخه پشتیبان کامل از تمام داده ها است این مدل امکان انشعاب و ادغام مدیریت خودکار را فراهم می کند. DVCS توانایی کار آفلاین را افزایش می دهد و برای پشتیبان گیری به یک مکان تکیه نمی کند. اگر هر سروری متوقف شود سیستم های دیگر با هم همکاری میکنند، هر یک از مخازن مشتری می تواند توسط آن سرور بازیابی شود. این سیستم ها برای ذخیره های یک فایل پروژه لزوماً به یک سرور مرکزی وابسته نیستند.

## ۵- معرفی Git (گیت)

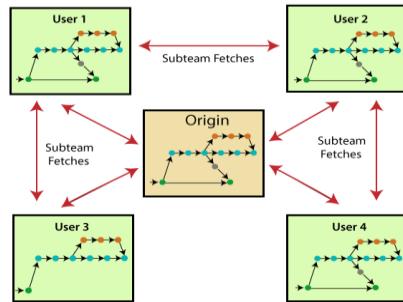
این سیستم، توسط لینوس توروالدز در سال ۲۰۰۵ میلادی ابداع شد. توسعه دهنده گیت در واقع همان سازنده هسته لینوکس است. گیت یک سیستم کنترل نسخه توزیع شده مدرن منبع باز و پر کاربرد در جهان است. گیت تنها سیستم کنترل نسخه موجود نیست، اما معروف ترین آنهاست زیرا در دنیای نرم افزار بیش از ۹۰ درصد تیم های نرم افزار در حال حاضر، گیت را به عنوان Source Control خود انتخاب کرده و با آن کار می کنند. یک ابزار توسعه مهم مورد استفاده روشن DevOps مهندسی نرم افزار است. DevOps ترکیبی از دو کلمه است، یکی توسعه نرم افزار و دومی عملیات به یک تیم اجازه می دهد تا کل چرخه عمر برنامه را از توسعه گرفته تا آزمایش، استقرار و عملیات مدیریت کند. DevOps به شما کمک می کند تا قطع ارتباط بین توسعه دهنگان نرم افزار، مهندسان تضمین کیفیت (QA) و مدیران سیستم را کاهش دهید.

### ویژگیهای گیت



برای مدیریت پروژه ها با سرعت و کارایی بالا است

- توزیع شده<sup>9</sup>:** توزیع شده به این معنی است که به جای اینکه پروژه را به ماشین دیگری تغییر دهیم، می توانیم یک `clone` از کل مخزن ایجاد کنیم. همچنین، به جای اینکه فقط یک مخزن مرکزی داشته باشید که تغییرات را به آن ارسال کنید، هر کاربر مخزن مخصوص به خود را دارد که شامل کل تاریخچه `commit` پروژه است. مانیازی به اتصال به مخزن راه دور نداریم. این تغییر فقط در مخزن محلی ما ذخیره می شود. در صورت لزوم، می توانیم این تغییرات را به یک مخزن از راه دور `PUSH` دهیم.



- همکاری در پروژه های عمومی<sup>10</sup>:** پروژه های عمومی در GitHub در دسترس هستند. می توانیم در آن پروژه ها همکاری کنیم و خلاصه خود را به دنیا نشان دهیم. بسیاری از توسعه دهندگان در پروژه های عمومی همکاری می کنند. این همکاری به ما امکان می دهد در

- کنار توسعه دهندگان با تجربه باشیم و چیزهای زیادی از آنها بیاموزیم. بنابراین، مهارت های برنامه نویسی ما را به سطح بالا می برد سیستم کنترل نسخه (پر کاربرد):<sup>11</sup>

Git پر کاربردترین سیستم کنترل نسخه است. دارای حداکثر پروژه در بین تمام سیستم های

کنترل نسخه است. با توجه به گردش کار شگفت انگیز و ویژگی های آن، انتخاب ترجیحی توسعه دهندگان است.

- منبع باز:** یک سیستم کنترل نسخه توزیع شده مدرن منبع باز و پر کاربرد در جهان است

- یکپارچگی<sup>12</sup>:** برای اطمینان از امنیت یکپارچگی محتوا تحت کنترل نسخه توسعه یافته است. در حین انتقال یا دستکاری در سیستم فایل برای تأیید گم نشدن اطلاعات از جمیع کنترلی استفاده می کند

- منطقه صحنه سازی<sup>13</sup>:** Staging نیز یکی از قابلیت های منحصر به فرد Git است. می توان آن را به عنوان پیش نمایشی از commit بعدی ما در نظر گرفت، علاوه بر این، یک منطقه میانی است که می توان commit ها را قبل از تکمیل قالب بندی و بررسی کرد. هنگامی که یک commit ایجاد می کنید، Git تغییراتی را که در قسمت staging<sup>14</sup> هستند، می گیرد و آنها را به عنوان یک commit جدید می سازد ناحیه staging را می توان مکانی در نظر گرفت که Git تغییرات را در آن ذخیره می کند.

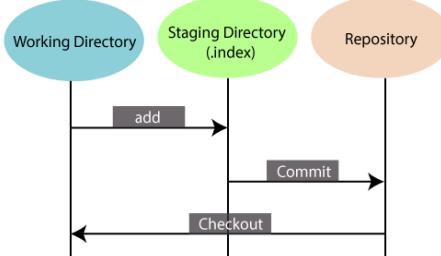
<sup>9</sup> distributed

<sup>10</sup> Collaborate to Public Projects

<sup>11</sup> Trendy Version Control System

<sup>12</sup> Integrity

<sup>13</sup> Staging area



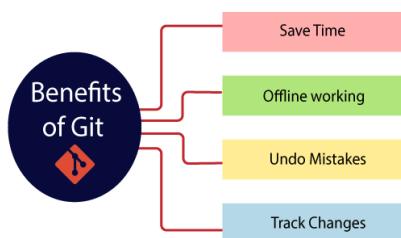
اگرچه Git دایرکتوری مرحله‌بندی اختصاصی ندارد که بتواند برخی از اشیاء را که نشان دهنده تغییرات فایل هستند، ذخیره کند. به جای این، از فایلی به نام index استفاده می‌کند.

- **تضمين داده ها<sup>۱۴</sup>**: مدل داده Git یکارچگی رمزنگاری هر واحد پروژه ما را تضمین می‌کند. Git یک شناسه commit منحصر به فرد برای هر commit از طریق یک الگوریتم SHA فراهم می‌کند. می‌توانیم commit ID را با commit بازیابی و به روز کنیم. اکثر سیستم‌های کنترل نسخه متمرکز به طور پیش فرض چنین یکارچگی را ارائه نمی‌دهند.
- **نگهداری تاریخچه تمیز<sup>۱۵</sup>**: یکی از مفیدترین ویژگی‌های Git این است که آخرین commit‌ها را از شاخه اصلی دریافت می‌کند و کد ما را در بالای آن قرار می‌دهد. بنابراین، تاریخچه ای تمیز از پروژه را حفظ می‌کند.
- **مقیاس پذیر<sup>۱۶</sup>**: این معنی که وقفی تعداد کاربران افزایش می‌یابد، Git می‌تواند به راحتی چنین موقعیت‌هایی را مدیریت کند.
- **همه چیز محلی است<sup>۱۷</sup>**: تقریباً تمام عملیات Git را می‌توان به صورت محلی انجام داد. که می‌تواند مهترین علت برای استفاده از Git باشد که در بخش محلی حتماً نیاز به اتصال اینترنت نیست.
- **انشعاب و ادغام<sup>۱۸</sup>**: شاخه‌بندی و ادغام ویژگی‌های عالی Git است که آن را از سایر ابزارهای SCM متمایز می‌کند. Git اجازه می‌دهد تا چندین شاخه را بدون تأثیر بر یکدیگر ایجاد کنید. می‌توانیم کارهایی مانند ایجاد، حذف و ادغام در شاخه‌ها را انجام دهیم و این کارها فقط چند ثانیه طول می‌کشد. در زیر برخی از ویژگی‌هایی که می‌توان با انشعاب به دست آورد. نشان داده شده است.
- می‌توانیم برای یک مژول جدید از پروژه یک شاخه جداگانه ایجاد کنیم، هر زمان که بخواهیم آن را commit و حذف کنیم.
- می‌توانیم یک شعبه تولید داشته باشیم که همیشه آنچه را که در آن وارد می‌شود می‌تواند برای آزمایش در شاخه آزمایشی ادغام شود.
- ما می‌توانیم یک شاخه آزمایشی برای تست ایجاد کنیم و بررسی کنیم که آیا کار می‌کند یا خیر. همچنین در صورت نیاز می‌توانیم آن را حذف کنیم.
- مزیت اصلی انشعاب این است که اگر بخواهیم چیزی را به یک مخزن راه دور منتقل کنیم، مجبور نیستیم همه شاخه‌های خود را push دهیم. ما می‌توانیم چند شعبه خود را انتخاب کنیم یا همه آنها را با هم انتخاب کنیم.

<sup>۱</sup> Data Assurance	4
<sup>۱</sup> Maintain the clean history	5
<sup>۱</sup> scalable	6
<sup>۱</sup> Everything is Local	7
<sup>۱</sup> branching	8

- استفاده کنندگان را تحت تاثیر قرار دهد<sup>۱۰</sup>: ما می‌توانیم با ذکر Git و GitHub در روزومه خود، استخدام کنندگان را تحت تاثیر قرار دهیم. پیوند نمایه GitHub خود را به سازمانی که می‌خواهد به آن پیوندید ارسال کنید. مهارت‌های خود را نشان دهید و از طریق کار خود بر آنها تأثیر بگذارید. شناس استفاده شدن را افزایش می‌دهد. برای مدیریت پروژه‌ها با سرعت و کارایی بالا توسعه یافته است.
- برای هماهنگی کار بین توسعه دهنده‌گان ایجاد شده است. به ما اجازه می‌دهد تا با اعضای تیم خود در یک فضای کاری رديایی و کار کنیم.
- GitLab و GitHub است، اما ما می‌توانیم بدون استفاده از سرویس‌های Git دیگر از Git استفاده کنیم.
- Git می‌تواند به صورت خصوصی و عمومی استفاده شود.
- یادگیری Git آسان است و عملکرد سریعی دارد.

### ▪ مزایای مهم استفاده از Git

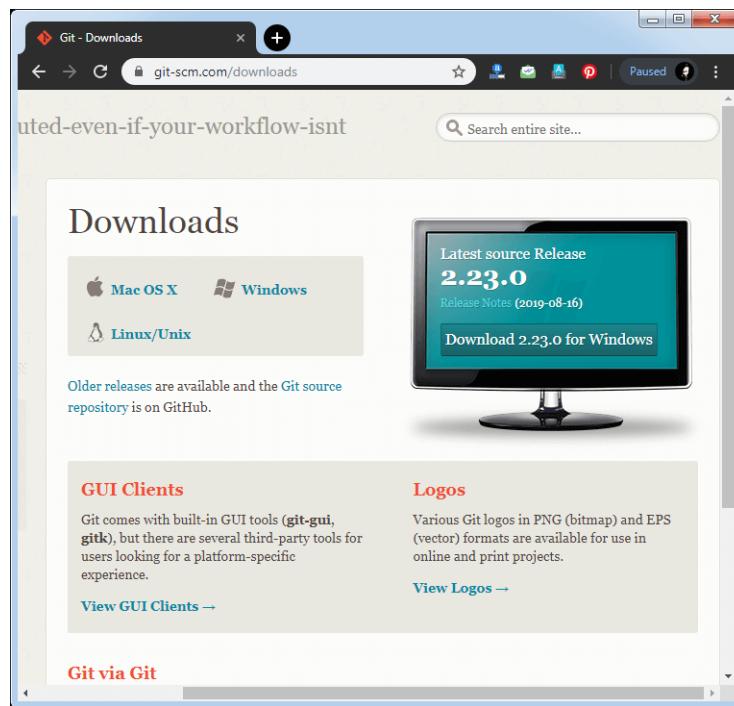


- **صرفه جویی در زمان:** یک فناوری سریع است. اجرای هر فرمان تنها چند ثانیه طول می‌کشد تا بتوانیم در زمان ورود به حساب GitHub در زمان زیاد صرفه جویی کنیم و ویژگی‌های آن را دریابیم.
- **کار آفلاین:** یکی از مهمترین مزایای Git این است که از کار آفلاین پشتیبانی می‌کند. اگر ما با مشکلات اتصال به اینترنت مواجه هستیم، تأثیری در کار ما نخواهد داشت. در Git تقریباً همه چیز را می‌توانیم به صورت محلی انجام دهیم. در مقایسه، سایر CVS مانند SVN محدود است و ارتباط با مخزن مرکزی را ترجیح می‌دهند.
- **رفع اشتباهات:** یکی از مزایای دیگر Git این است که می‌توانیم اشتباهات را لغو کنیم. گاهی اوقات خنثی سازی می‌تواند یک گزینه نجات دهنده برای ما باشد. Git تقریباً برای همه چیز گزینه لغو را فراهم می‌کند.
- **ردیابی تغییرات:** Git با برخی ویژگی‌های هیجان‌انگیز مانند Diff, Log و Status کار را آسان می‌کند، به ما امکان می‌دهد تغییرات را ردیابی کنیم تا بتوانیم وضعیت را بررسی کنیم، فایل‌ها یا شاخه‌های خود را با هم مقایسه کنیم.

## روش نصب git برای windows □

**مرحله اول دانلود نصب کننده git** برای دانلود نصب کننده git به سایت رسمی گیت به صفحه دانلود بروید . پس از ورود به وبسایت، با کلیک بر روی دکمه‌ی Download ، فایل اجرایی گیت با پسوند exe برای ویندوز دانلود می‌شود .

<https://git-scm.com/downloads>



## مرحله دوم نصب git □

روی فایل نصب کننده دانلود شده کلیک کنید و برای ادامه گزینه yes را انتخاب کنید. پس از انتخاب yes، نصب شروع می‌شود و صفحه نمایش مانند خواهد بود



## مرحله سوم □

اجزای پیش فرض به طور خود کار در این مرحله انتخاب می شوند. شما همچنین می توانید قطعه مورد نیاز خود را انتخاب کنید. . یا با تغییر گزینه ها، نصب را به شکل دلخواه خود انجام دهید. توجه داشته باشید که اگر می خواهید از گیت در خط فرمان ویندوز استفاده کنید، گزینه Git Bash Here و "Git GUI Here" را در هنگام نصب انتخاب کنید



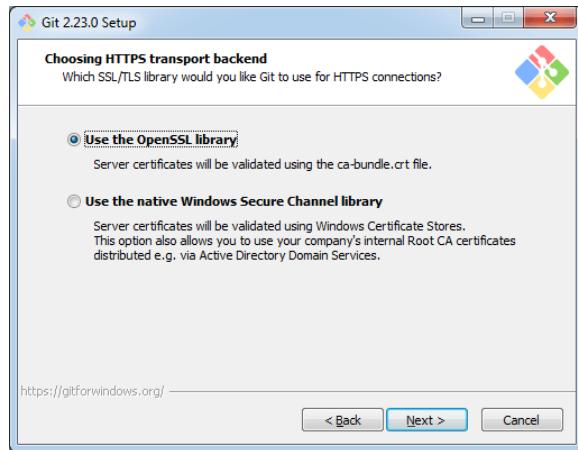
## مرحله چهارم □

این مرحله تنظیم متغیرهای محیطی است متغیرهای محیطی سیستم عامل ویندوز را تنظیم کنید تا بتوانید از گیت در خط فرمان CMD یا PowerShell استفاده کنید. گزینه های پیش فرض خط فرمان Git به طور خود کار انتخاب می شوند. شما می توانید انتخاب دلخواه خود را انتخاب کنید. برای ادامه بر روی بعدی کلیک کنید



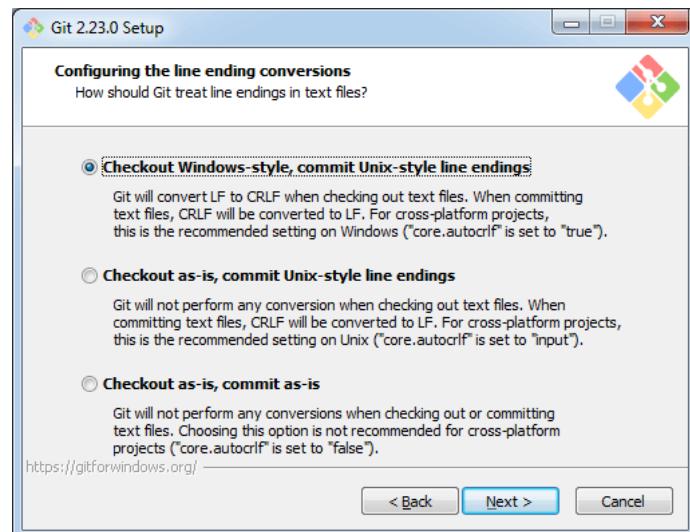
## مرحله پنجم

در این مرحله گزینه های پیش فرض انتقال back-end انتخاب می شوند. برای ادامه بر روی بعدی کلیک کنید.



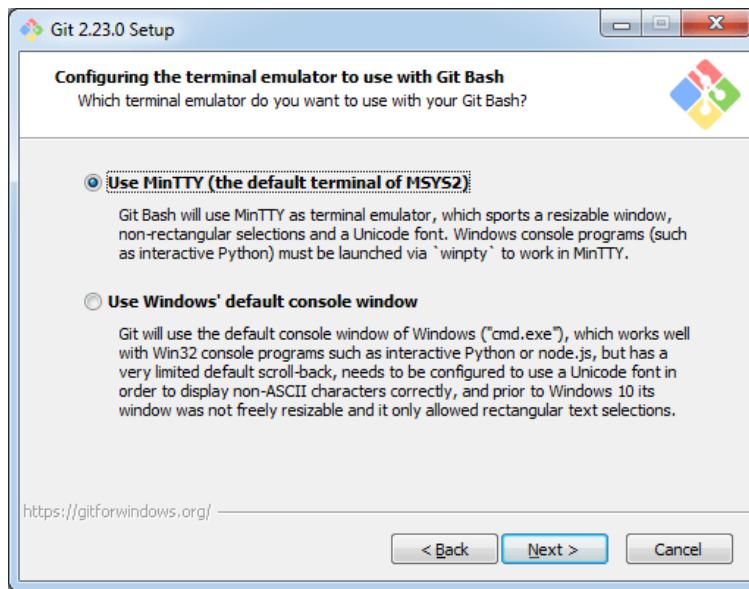
## مرحله ششم

گزینه پایان خط مورد نیاز خود را انتخاب کنید و برای ادامه روی Next کلیک کنید.



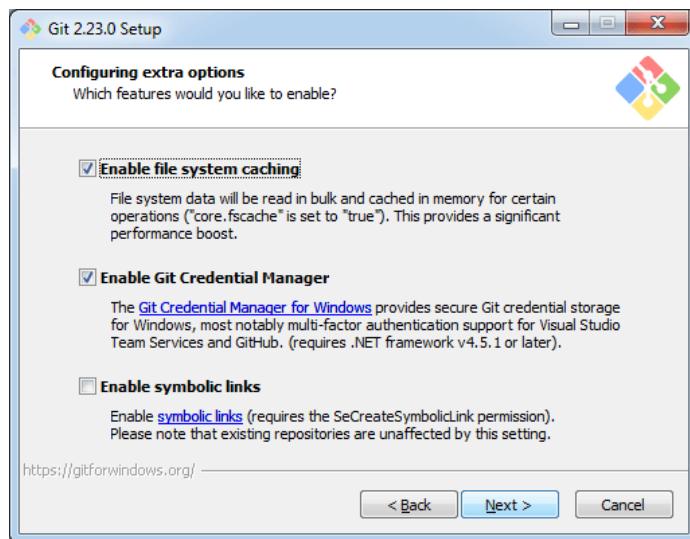
## مرحله هفتم

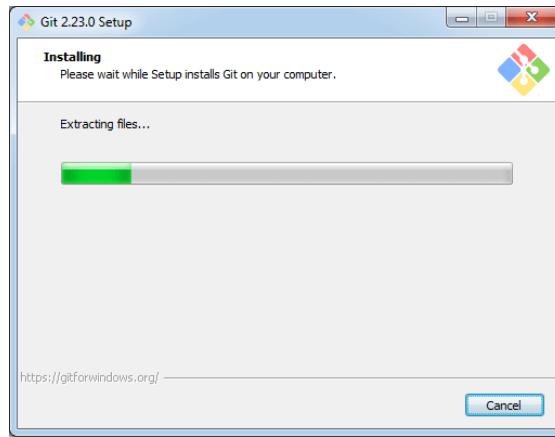
برای ادامه، شبیه ساز ترمینال ترجیحی را انتخاب کنید.



## مرحله هشتم

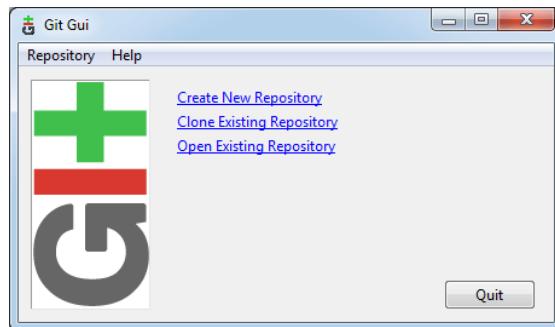
این آخرین مرحله است که برخی از ویژگی های اضافی مانند ذخیره سیستم، مدیریت اعتبار و پیوند نمادین را ارائه می دهد. ویژگی های مورد نیاز را انتخاب کرده و روی گزینه بعدی کلیک کنید.



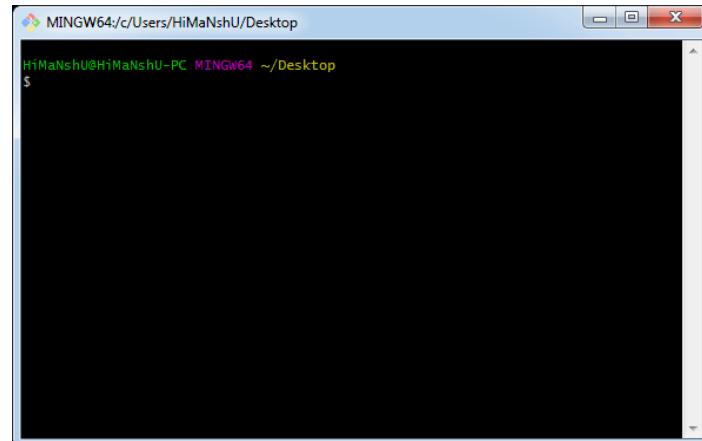


بنابراین، نصب Git به پایان رسید. اکنون می توانید به Git Gui و Git Bash دسترسی داشته باشید.

شیوه به این است **Git Gui**



**Git Bash**



**بررسی نصب**: پس از اتمام فرآیند نصب، می‌توانید با باز کردن خط فرمان ویندوز و تایپ دستور `git -version` نصب گیت را بررسی کنید. اگر نصب به درستی انجام شده باشد، ورژن نصب شده گیت نمایش داده می‌شود.

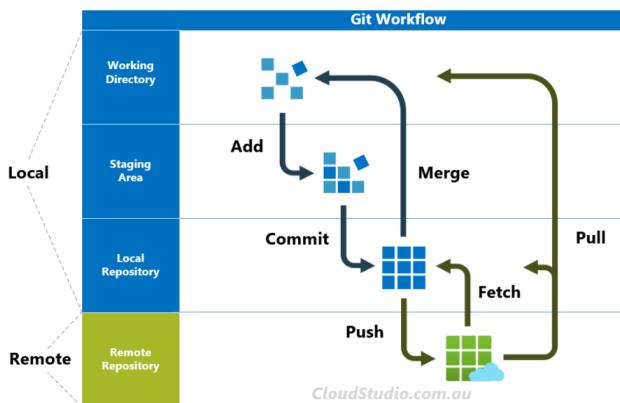
**تنظیم کاربر**: پس از نصب، توصیه می‌شود تا اطلاعات کاربری گیت خود را تنظیم کنید. با اجرای دستورات زیر نام کاربری و ایمیل خود را تنظیم کنید تا هر بار که یک کامیت ایجاد می‌کنید، اطلاعات شما به همراه آن ذخیره شود.

```
git config --global user.name "Your Name"
```

```
git config --global user.email "your@email.com"
```

با انجام این مراحل، گیت بر روی سیستم عامل ویندوز شما نصب و آماده استفاده خواهد بود. حال می‌توانید با استفاده از دستورات گیت، پروژه‌های خود را مدیریت کنید.

## فرآیند عملیات در گیت



## مفهومی بخی از اصطلاحات در git/github □

برای اینکه بتوانید با فضای این پلتفرم به صورت حرفه‌ای تر کار کنید و به همه امکانات آن دسترسی داشته باشید بهتر است بدانید که مفاهیم اصلی در گیت و گیت هاب چیست. در این بخش در خصوص این مفاهیم و کاربرد هریک توضیح می‌دهیم.

نام	مفهوم
Repository	مخزن یا آنبار در واقع محل ذخیره سازی همه کدها و فایل‌های مربوط به یک پروژه است. در اصل مخزن‌های گیت نوعی دایرکتوری محسوب می‌شوند که در آن Git برای ردیابی و ذخیره تغییرات مقداردهی مورد استفاده قرار می‌گیرد.
Commit	هر تغییری که شما در کدها ایجاد می‌کنید نوعی Commit است؛ از طریق کامیت می‌توانید توضیح دهید که هدف شما از تغییر در این پروژه از گیت هاب چیست و فواید یا دلایل آن را بیان کنید
Branch	شاخه در Git نوعی خط توسعه جداگانه است. ایجاد شاخه‌های مختلف در توسعه به شما امکان می‌دهد تا روی ویژگی‌های مختلف یا اشکال زدایی پروژه به صورت مجزا کار کنید، بدون اینکه روی شاخه اصلی تأثیر بگذارد.
Merge	ادغام فرآیند ترکیب تغییرات از شاخه‌های مختلف به همدیگر است Merge. بخش مهمی از همکاری در Git است که به چندین مشارکت کننده اجازه می‌دهد تا کار خود را بپارچه کنند.
Pull	عمل واکنشی تغییرات از یک مخزن راه دور و ادغام آن‌ها در شاخه فعلی است. pull، مخزن محلی شما را با مخزن راه دور به روز نگه می‌دارد.
pushing/Push	پوشینگ به عمل ارسال تهدایات شما از مخزن محلی به یک مخزن راه دور اشاره دارد. پوشینگ در اصل نحوه به اشتراک گذاشتن کار خود با دیگران و همکاری در همان پروژه است
Fork	форک یک کپی از یک مخزن است که به شما امکان می‌دهد تغییرات را بدون تأثیر بر پروژه اصلی آزمایش کنید. فورک‌ها عموماً در پروژه‌های متنهای باز (پن‌سورس) استفاده می‌شوند.
Clone	کلون سازی در Git به معنای ایجاد یک کپی محلی از یک مخزن راه دور در کامپیوتر شما است. به کاربر امکان می‌دهد تا روی پروژه به صورت محلی کار کند.
Pull Request	پیشنهادی برای ادغام تغییرات از یک شاخه به شاخه دیگر است. قبل از ادغام تغییرات برای بررسی کد و issue استفاده می‌شود.

## آشنایی با دستورات اولیه Git □

در اینجا لیستی از ضروری ترین دستورات Git است که روزانه استفاده می‌شود.

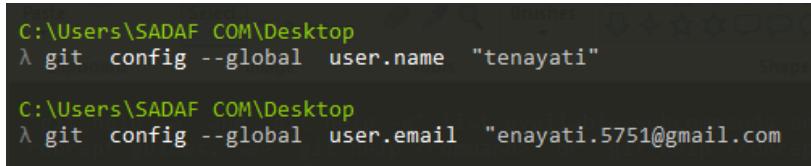
1. [Git Config command](#)
2. [Git init command](#)
3. [Git clone command](#)
4. [Git add command](#)
5. [Git commit command](#)
6. [Git status command](#)
7. [Git push Command](#)
8. [Git pull command](#)
9. [Git Branch Command](#)
10. [Git Merge Command](#)
11. [Git log command](#)
12. [Git remote command](#)

## ▪ دستور پیکربندی کاربر

دستور config Git اولین و ضروری ترین دستوری است که در خط فرمان Git استفاده می‌شود. این دستور نام نویسنده و آدرس ایمیل را برای استفاده در commit های شما تنظیم می‌کند. پیکربندی Git در سناریوهای دیگر نیز استفاده می‌شود.

## Syntax

```
$ git config --global user.name "soraya"  
$ git config --global user.email enayati.5751@gmail.com
```



```
C:\Users\SADAF COM\Desktop  
λ git config --global user.name "tenayati"  
  
C:\Users\SADAF COM\Desktop  
λ git config --global user.email "enayati.5751@gmail.com"
```

### ▪ دستور ایجاد یک مخزن محلی

دایرکتوری پروژه هایی که از سیستم های مدیریت کد منبع مثل گیت استفاده می کنند، با عنوان یک repository می خوانند. برای شروع کار با گیت در یک دایرکتوری به عنوان یک مخزن گیت، لازم است که ابتدا به گیت بگویید که میخواهید این دایرکتوری یک مخزن گیت باشد. دستور `git init` یک مخزن جدید گیت ایجاد می کند. مخزنی که درون آن میتوانید از امکانات گیت استفاده کنید و دستورات را در آن اجرا کنید. با اجرای این دستور یک دایرکتوری با نام `git`/ درون دایرکتوری حاضر شما ایجاد میشود که حاوی فایل های کانفیگ و فایل ها و تغییرات ثبت شده توسط گیت است.

▪ برای اضافه کردن گیت به یک پروژه، داخل دایرکتوری پروژه دستور زیر را وارد کنید

## Syntax

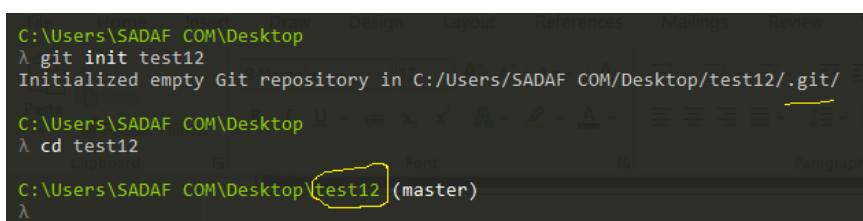
```
$ git init
```

▪ یا برای ساختن یک پروژه جدید دستور زیر را استفاده می کنیم:

## Syntax

```
$ git init namedirectory
```

که این دستور یک دایرکتوری با نامی که وارد کرده اید می سازد و گیت را به آن اضافه می کند



```
C:\Users\SADAF COM\Desktop  
λ git init test12  
Initialized empty Git repository in C:/Users/SADAF COM/Desktop/test12/.git/  
  
C:\Users\SADAF COM\Desktop  
λ cd test12  
C:\Users\SADAF COM\Desktop\test12 (master)
```

### ▪ دستور Git clone

این دستور برای کپی کردن یک مخزن از یک URL موجود استفاده می شود. اگر من یک کپی محلی از مخزن خود را از GitHub بخواهم، این دستور اجازه می دهد تا یک کپی محلی از آن مخزن را در فهرست محلی شما از URL مخزن ایجاد کنید.

## Syntax

```
$ git clone URL
```

### ▪ دستور touch

با دستور **touch** یک فایل ایجاد می کنیم.

```
$ touch filename
```

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ touch form.html

C:\Users\SADAF COM\Desktop\test12 (master)
λ touch f1.css
    GitHub
C:\Users\SADAF COM\Desktop\test12 (master)
λ touch f2.js
    برتر مخزن شما از URL مخزن ایجاد کنید

C:\Users\SADAF COM\Desktop\test12 (master)
λ git status
On branch master
    Syntax
No commits yet
    $ git clone URL

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    f1.css
    f2.js
    form.html

nothing added to commit but untracked files present (use "git add" to track)
```

تا اینجای کار یک مخزن در دایرکتوری خودمان با سه فایل ایجاد کردیم. هنوز هیچ انشعاب(شاخه) نداریم و **commit** هم نشده یعنی در مخزن راه دور یا ریموت قرار داده نشده.

### ▪ دستور git status (نمایش تغییرات)

یکی از اساسی ترین کارهایی که با گیت می کنیم بررسی و ثبت تغییرات ایجاد شده در فایل های متی پروژه است. هر زمان که تغییرات ثبت نشدهی پروژه را مایل بودید بررسی کنید، دستور زیر را وارد کنید

## Syntax

```
$ git status
```

این دستور فایل هایی که تغییر داده اید را برای شما لیست می کند. تغییرات شامل: ویرایش یک فایل، اضافه کردن فایل جدید یا حذف یک فایل و همچنین فایلهای استیج شده یا استیج نشده

### ▪ دستور git add یا دستور اضافه کردن فایل ها به stage

ثبت تغییرات در گیت بصورت معمول دو مرحله دارد: ۱. اضافه کردن فایل های تغییر داده شدهی مورد نظر ۲. ثبت تغییرات با یک پیام یا توضیح برای اضافه کردن فایل ها به گیت به مرحله **stage** دستور **add** استفاده می کنیم. به صورت زیر:

## Syntax

```
$ git add filename
```

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git add f1.css
git add <filename>
C:\Users\SADAF COM\Desktop\test12 (master)
λ git add f2.js

C:\Users\SADAF COM\Desktop\test12 (master)
λ git add form.html

C:\Users\SADAF COM\Desktop\test12 (master)
λ git status
On branch master

No commits yet

Changes to be committed:
(use "git rm --cached <file>..." to unstage)
  new file:  f1.css
  new file:  f2.js
  new file:  form.html
```

- دستور کانفیگ کردن یک ادیتور جهت باز کردن فایل در آن ادیتور مورد نظر

### Syntax

```
$ git config --global core.editor "name editor -w"
```

دستور زیر فایل را بصورت پیش فرض در ادیتور atom باز می کند تا تغییرات در ان داده شود

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git config --global core.editor "atom -w"
C:\Users\SADAF COM\Desktop\test12 (master)
λ |
```

- دستور `git commit` یا دستور ثبت در `git`

برای ثبت تغییرات یا به اصطلاح کامیت کردن تغییرات، از دستور `commit` استفاده می کنیم. در این مرحله فایل هایی را که با دستور `add` به `stage` بدهایم در سیستم گیت ثبت می کنیم. برای ثبت هر تغییر نیاز است یک پیام هم با آن ثبت شود تا معلوم شود در این قسمت از تغییرات لحاظ شده چه کار کردہ ایم، یا چه تغییراتی داده ایم.

### Syntax

```
$ git commit -m "add a description"
```

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git commit -m "edit form"
[master (root-commit) 929830f] edit form
 3 files changed, 16 insertions(+)
 create mode 100644 f1.css
 create mode 100644 f2.js
 create mode 100644 form.html

C:\Users\SADAF COM\Desktop\test12 (master)
λ
```

## ▪ دستور تغییر محتوای آخرین commit

### Syntax

```
$ git commit -- amend -m "new commit message"
```

فرض کنیم که اشتباهی در نوشتن پیام یک کامیت داشته اید و یا به هر دلیل دیگر قصد تغییر پیام آخرین کامیت را دارید و این مورد را پس از انجام کامیت متوجه شده اید. برای تغییر دوباره‌ی پیام آخرین کامیت از گزینه‌ی **amend** به همراه دستور کامیت استفاده می‌کنیم.

## ▪ دستور git ignor

ممکن است در دایرکتوری پروژه فایل‌هایی داشته باشید، که نخواهید گیت آن‌ها را در **status** نشان دهد، و همچنین نخواهید در مخزن اصلی اضافه شوند. برای این کار باید در دایرکتوری پروژه یک فایل به نام **gitignore** بسازید و در آن، لیست فایل‌ها و دایرکتوری‌هایی را که گیت باید نادیده بگیرد را بنویسید.

گیت تمام فایل‌های پوشه‌ی شما را در یکی از سه دسته زیر می‌بیند:

▪ دنبال شده – فایلی که در گذشته **Commit stage** یا **Commit** شده باشد.

▪ دنبال نشده – فایلی که هنوز **Commit stage** یا **Commit** نشده است.

▪ **ignore**-شده (نادیده گرفته شده) – فایلی که به گیت گفته شده است آن را نادیده بگیرد.

به طور معمول فایل‌هایی **ignore** می‌شوند که توسط خود سیستم تولید شده باشند یا نخواهیم آن‌ها را در مخزن خود **Commit** کنیم. چند نمونه‌ی معمول عبارت است از:

▪ فایل‌های مربوط به وابستگی‌ها و پکیج‌های پروژه مانند پوشه‌ی **node\_modules** یا **vendor**.

▪ کد‌های کامپایبل شده مانند فایل‌های **.class** و **.pyc** و ...

▪ فایل‌های ایجاد شده هنگام اجرای برنامه مانند **.log** و **.lock** و **.tmp** و ...

▪ فایل‌های مخفی شده سیستم مانند **DS\_Store** یا **Thumb.db**.

▪ پیکربندی IDE مانند پوشه‌ی **.idea**.

فایل‌های **ignore** شده در فایل مخصوصی به نام **.gitignore** ثبت می‌شوند که در مسیر اصلی مخزن قرار می‌گیرد.

```
C:\Users\SADAF COM\Desktop\test12 (master)    javatpoint.com/gitignore
λ touch .gitignore

C:\Users\SADAF COM\Desktop\test12 (master)    CDA   HTML   CSS
λ git add f2.js

C:\Users\SADAF COM\Desktop\test12 (master)
λ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   form.html
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    .gitignore
```

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git add .gitignore          From the above output, we can see that the file .gitignore has been added to the repository.

C:\Users\SADAF COM\Desktop\test12 (master)
λ git commit -m "ignored directory created."
[master efde61d] ignored directory created.
 2 files changed, 24 insertions(+), 2 deletions(-)
 create mode 100644 .gitignore

C:\Users\SADAF COM\Desktop\test12 (master)
λ git config --global core.excludesfile ~/.gitignore_global

C:\Users\SADAF COM\Desktop\test12 (master)
λ git ls-files -i --exclude-standard
fatal: ls-files -i must be used with either -o or -c

C:\Users\SADAF COM\Desktop\test12 (master)
```

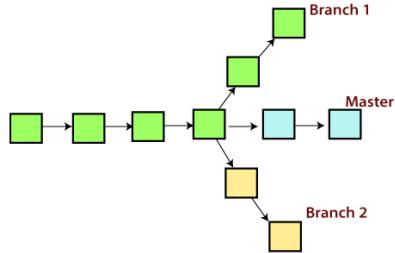
## ▪ دستور git branch

### syntax

```
$ git branch namebranch
```

کنترل ورژن گیت دارای یک تنه اصلی به نام **master** است. همانطور که قبلاً اشاره شد ساختار گیت مثل یک درخت است. هر درختی شامل یک تنه اصلی و شاخ و برگ است. به شاخ و برگ های گیت **branch** گفته می شود. شاخه یا همان **branch** یک پروژه در واقع نسخه ای کپی شده از پروژه اصلی هستند. با ایجاد یک شاخه می توانید یک کپی از پروژه اصلی را داشته باشید و تغییرات مدنظر خودتان را بر روی آن شاخه انجام دهید بدون اینکه برای شاخه اصلی مشکلی به وجود آید. یا اینکه فرض کنید در پروژه ما یک باگ ایجاد شده و می خواهیم آن را رفع کنیم. به این منظور بهتر است یک شاخه جدید ایجاد کرده و پروسه رفع خطأ را روی آن انجام دهیم. در نهایت شما می توانید تغییرات را به شاخه اصلی انتقال دهید و در واقع پروژه خودتان را **merge** کنید.

به صورت پیشفرض در هنگام ساخته شدن هر پروژه گیت با استفاده از دستور **git init** همواره یک شاخه اصلی در گیت که به آن تنه اصلی (**master**) نیز گفته می شود ایجاد می گردد. شما هر تغییراتی را که تا اینجا انجام می دادید در درون این شاخه اصلی نگهداری می شد. این تغییرات شامل **stage** ها و **commit** های پروژه است. ممکن است بخواهید تغییراتی را در درون قسمتی از کد انجام دهید بدون اینکه بر روی پروژه اصلی شما تاثیر گذار باشد. برای انجام این تغییرات می توانید یک شاخه یا همان **branch** ایجاد نمایید. برای ساختن یک شاخه کافی است دستور زیر را وارد نمایید.



مثال: پروژه projectstu2 را با مشخصات زیر در نظر بگیرد.

```
C:\Users\SADAF COM\Desktop\projectstu2>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   f1.txt
    modified:   f2.txt
    modified:   f3.txt
```

```
C:\Users\SADAF COM\Desktop\projectstu2> git log
commit b3cd0351c2812ca83254ce859464dad9fe95acd8 (HEAD -> master)
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:35:41 2023 +0330

  add a masseg to f1 &f2&f3

commit 4528261a9d3d8341670c337c1b47750dc07bf404
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:08:28 2023 +0330

  insert a cod in f1 and f1

commit 7af0598e765afb86eddef37ca9195f5cf99d6a13
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:02:26 2023 +0330

  initial project
```

در پروژه سه تا commit در شاخه master انجام شده. با دستور زیر یک شاخه branch به نام editcod ایجاد میکنیم

```
C:\Users\SADAF COM\Desktop\projectstu2>git branch editcod
C:\Users\SADAF COM\Desktop\projectstu2>git branch
  editcod
* master
```

شاخه editcod ایجاد شده و اگر دستور git branch را اجرا کنید تمام شاخه هارا نمایش میدهد هنوز هد به master اشاره می کند.

با استفاده از دستور زیر می توانیم روی شاخه جدید سویچ کنیم.

### syntax

`$ git switch namebranch`

```
C:\Users\SADAF COM\Desktop\projectstu2> git switch editcod
Switched to branch 'editcod'

C:\Users\SADAF COM\Desktop\projectstu2>git branch
* editcod
  master
```

حال در شاخه جدید تغییراتی روی پروژه ایجاد کنید.

```
C:\Users\SADAF COM\Desktop\projectstu2> git log
commit b3cd0351c2812ca83254ce859464dad9fe95acd8 (HEAD -> editcod, master)
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:35:41 2023 +0330

    add a masseg to f1 &f2&f3

commit 4528261a9d3d8341676c337c1b47750dc07bf404
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:08:28 2023 +0330

    insert a cod in f1 and f1

commit 7af0598e765afb86eddef37ca9195f5cf99d6a13
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:02:26 2023 +0330

    initial project

C:\Users\SADAF COM\Desktop\projectstu2>git status
On branch editcod
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   f1.txt
    modified:   f2.txt
```

برای اینکه تغییرات در شاخه editcod باشد باید دستور add و سپس commit را اجرا کنیم

```
C:\Users\SADAF COM\Desktop\projectstu2> git add .
C:\Users\SADAF COM\Desktop\projectstu2>git status
On branch editcod
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   f1.txt
    modified:   f2.txt

C:\Users\SADAF COM\Desktop\projectstu2>git commit -m " delt statement "
[editcod ac598af] delt statement
 2 files changed, 2 insertions(+), 3 deletions(-)

C:\Users\SADAF COM\Desktop\projectstu2>git log
commit ac598af0f8696a2c5c8fa0d60798ca9310643706 (HEAD -> editcod)
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 22:42:51 2023 +0330

    delt statement

commit b3cd0351c2812ca83254ce859464dad9fe95acd8 (master)
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:35:41 2023 +0330

    add a masseg to f1 &f2&f3

commit 4528261a9d3d8341676c337c1b47750dc07bf404
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:08:28 2023 +0330

    insert a cod in f1 and f1

commit 7af0598e765afb86eddef37ca9195f5cf99d6a13
Author: sorayaenayati <senayati640@gmail.com>
Date:   Sat Nov 18 21:02:26 2023 +0330

    initial project
```

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git branch edit
```

حالا می توانید به راحتی با استفاده از دستور زیر وارد شاخه edit شوید.

```
C:\Users\SADAF COM\Desktop\test12 (master)
λ git checkout edit
Switched to branch 'edit'
A contactus.html
```

اگر دقت کید master به edit تغییر نام داده است. محیط دستوری گیت به شکل زیر در آمده است.

```
C:\Users\SADAF COM\Desktop\test12 (edit)
λ |
```

حالا در شاخه edit قرار داریم و می خواهیم تغییراتی را بر روی پروژه ایجاد کنیم.

در ادامه تغییراتی دلخواه را به index.html اضافه می کنیم. حالا بر روی شاخه یکبار git status می گیریم. خروجی به صورت زیر است.

```
C:\Users\SADAF COM\Desktop\test12 (edit)
λ git status
On branch edit
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   contactus.html

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   index.html
```

**نکته:** هر بار که خواستیم از هر شاخه خارج شویم و به شاخه دیگری برویم می بایست همواره قبل از خروج تغییرات را commit کنیم تا شاخه به صورت کامل پاکسازی شود.

با استفاده از دستور git add . و سپس دستور "git commit -m "index.html updated" تغییرات را commit می کنیم. در خروجی بر روی شاخه edit یکبار git status می گیریم.

```
C:\Users\SADAF COM\Desktop\test12 (edit)
λ git add .

C:\Users\SADAF COM\Desktop\test12 (edit)
λ git commit -m"index.html updated"
[edit 25368d5] index.html updated
 2 files changed, 19 insertions(+), 1 deletion(-)
 create mode 100644 contactus.html

C:\Users\SADAF COM\Desktop\test12 (edit)
λ git status
On branch edit
nothing to commit, working tree clean

C:\Users\SADAF COM\Desktop\test12 (edit)
λ |
```

حالا دوباره کد را به contactus.html اضافه می کنیم و commit می کنیم

```
C:\Users\SADAF COM\Desktop\test12 (edit)
\ git add .

C:\Users\SADAF COM\Desktop\test12 (edit)
\ git commit -m"contactus.html updat"
[edit 7df8c56] contactus.html updat
 1 file changed, 1 insertion(+), 1 deletion(-)

C:\Users\SADAF COM\Desktop\test12 (edit)
\ git checkout master
Switched to branch 'master'

C:\Users\SADAF COM\Desktop\test12 (master)
\ git status
On branch master
nothing to commit, working tree clean

C:\Users\SADAF COM\Desktop\test12 (master)
\
```

شاخه به طور کامل پاک است. اگر یکبار log نیز بگیریم داریم

همانطور که مشاهده می کنید خبری از commit که در شاخه edit گرفته بودیم نیست. در همین شاخه اگر به فایل index.html مراجعه کنید می بینید که خبری از کد اضافه شده نیست این یکی از بزرگترین مزایای استفاده از گیت که می توانیم بر روی پروژه شاخه بندی داشته باشیم. ممکن است این سوال پیش آید که این کار به چه دردی می خورد. خوب همه این تغییرات را روی فایل اصلی انجام دهیم؟

در جواب به این سوال باید گفت که این کار چند مزیت دارد:

- اگر مشکل یا باگی در سیستم به وجود آید شما می توانید به سادگی با ایجاد کردن یک شاخه اقدام به رفع آن باگ نمائید و در صورتی که همه چیز بدون مشکل بود این تغییرات را به تنہ اصلی انتقال دهید
- شما می توانید با شاخه سازی قسمت های مختلف، یک پروژه را بین برنامه نویس ها تقسیم کنید و در نهایت بعد از اعمال تمامی تغییرات آن را بر روی شاخه اصلی ترکیب کنید
- شاخه سازی یکی از روش های ایجاد و کنترل ورژن در پروژه های مختلف است که می بایست حتما از این ویژگی در کنترل ورژن استفاده کرد.

```
C:\Users\SADAF COM\Desktop\test12 (master)
\ git log
commit 08fda3ba7347341379f7e4761a1ba1dd457a0b59 (HEAD -> master)
Author: tenayati <enayati.5751@gmail.com>
Date:   Sun Mar 5 22:25:08 2023 +0330

    chang a

commit e483d0714aa7b7e51701cfadbec5d1531e41ec7
Author: tenayati <enayati.5751@gmail.com>
Date:   Sun Mar 5 22:16:57 2023 +0330

    creat index

commit efde61d0d677cd5cb1964d481b48573dff8d8d2
Author: tenayati <enayati.5751@gmail.com>
Date:   Fri Mar 3 23:17:42 2023 +0330

    ignored directory created.

commit 929830f218a149e3a8ab3bdd461f77acbd5e69dc
Author: tenayati <enayati.5751@gmail.com>
Date:   Fri Mar 3 19:53:59 2023 +0330

    edit form

C:\Users\SADAF COM\Desktop\test12 (master)
\
```

## دستور git merge □

بعد از اینکه تمامی کدی نویسی های دلخواه بر روی پروژه را براساس قابلیت شاخه ای انجام دادیم می خواهیم تغییرات موجود بر روی شاخه edit را به تنه اصلی master اضافه کنیم بدین منظور از دستور merge برای انجام این کار بهره می بریم.

### syntax

```
$ git merge namebranch
```

همواره جهت merge کردن یک پروژه بر روی هر شاخه می بایست ابتدا به آن شاخه اصلی بروید. در اینجا ما می خواهیم شاخه edit را بر روی تنہ master ترکیب کنیم. به همین خاطر ابتدا با استفاده از دستور checkout به master می رویم. و دستور زیر را برای انجام merge شاخه edit بر روی master وارد می کنیم.

```
C:\Users\SADAF COM\Desktop\test12 (master)
└ git merge edit
Updating 08fda3b..7df8c56
Fast-forward
 contactus.html | 18 ++++++-----+
 index.html     |  2 ++
 2 files changed, 19 insertions(+), 1 deletion(-)
 create mode 100644 contactus.html
```

حالا اگر log بر روی master بگیریم خروجی شامل تمامی commit ها است.

```
C:\Users\SADAF COM\Desktop\test12 (master)
└ git log
commit 7df8c562b18477554ef8e3e90f3b6ba375fa6563 (HEAD -> master, edit)
Author: tenayati <enayati.5751@gmail.com>
Date:   Sun Mar 5 23:10:33 2023 +0330

    contactus.html updat

commit 25368d5c4e4d25cc634ffeb939e73458f665a3e1
Author: tenayati <enayati.5751@gmail.com>
Date:   Sun Mar 5 23:04:16 2023 +0330

    index.html updat

commit e483d0714aa7b7e51701cfadecbec5d1531e41ec7
Author: tenayati <enayati.5751@gmail.com>
Date:   Sun Mar 5 22:16:57 2023 +0330

    creat index

commit efde61d0d677cd5cb1964d481b48573dff8d8d2
Author: tenayati <enayati.5751@gmail.com>
Date:   Fri Mar 3 23:17:42 2023 +0330

    ignored directory created.

commit 929830f218a149e3a8ab3bdd461f77acbd5e69dc
Author: tenayati <enayati.5751@gmail.com>
Date:   Fri Mar 3 19:53:59 2023 +0330

    edit form
```

حالا اگر بخواهیم این log را به صورت شاخه ای بیسیم کافی است که دستور git log --graph را وارد کنیم. خروجی به صورت زیر در می آید

### syntax

```
$ git log --graph
```

```

C:\Users\SADAF COM\Desktop\test12 (master)
└ git log --graph
fatal: unrecognized argument: --gragh

C:\Users\SADAF COM\Desktop\test12 (master)
└ git log --graph
* commit 7df8c562b18477554ef8e3e90f3b6ba375fa6563 (HEAD -> master, edit)
| Author: tenayati <enayati.5751@gmail.com>
| Date:   Sun Mar 5 23:10:33 2023 +0330
|
|       contactus.html updat
|
* commit 25368d5c4e4d25cc634ffeb939e73458f665a3e1
| Author: tenayati <enayati.5751@gmail.com>
| Date:   Sun Mar 5 23:04:16 2023 +0330
|
|       index.html updat

* commit e483d0714aa7b7e51701cfadbec5d1531e41ec7
| Author: tenayati <enayati.5751@gmail.com>
| Date:   Sun Mar 5 22:16:57 2023 +0330
|
|       creat index
|
* commit efde61d0d677cd5cb1964d481b48573dffbd8d8d2
| Author: tenayati <enayati.5751@gmail.com>
| Date:   Fri Mar 3 23:17:42 2023 +0330
|
|       ignored directory created.
|
* commit 929830f218a149e3a8ab3bdd461f77acbd5e69dc
| Author: tenayati <enayati.5751@gmail.com>
| Date:   Fri Mar 3 19:53:59 2023 +0330
|
|       edit form

C:\Users\SADAF COM\Desktop\test12 (master)

```

در ابتدا ما commit هایی را بر روی تنه اصلی داشتیم. سپس شاخه edit را اضافه کردیم و در درون آن مواردی را commit کردیم و دوباره به شاخه اصلی برگشتمیم و مورد دیگری را در آنجا commit کردیم. تمامی این مراحل به صورت درخت وار در log graph مشخص است.

## ▪ دستور git diff

برای نشان تغییرات بین commit ها staged working directory و commit working directory با استفاده می شود. که نسخه های مختلف منابع داده را مقایسه می کند.

### Syntax

\$ git diff

**مثال:** یک پروژه جدید بنام projectstu3 در working directory ایجاد نموده و پروژه را به گیت برد و بعد در atom vs cod یا cmd دستورات یک فایل بنام name.txt ایجاد کنید فایل را به staged می برم و بعد یک commit اضافه می کنم.

```

C:\Users\SADAF COM\Desktop\projectstu3>git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    name.txt

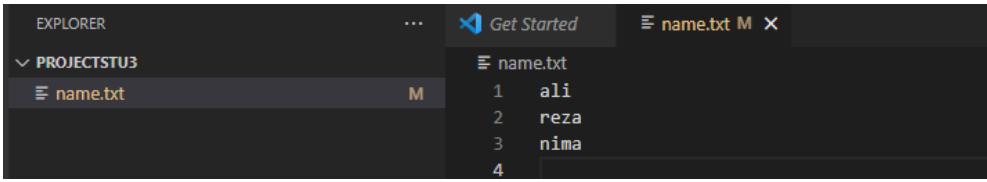
nothing added to commit but untracked files present (use "git add" to track)

C:\Users\SADAF COM\Desktop\projectstu3>git add .

C:\Users\SADAF COM\Desktop\projectstu3>git commit -m "initial project"
[master (root-commit) 20ea643] initial project
  1 file changed, 3 insertions(+)
  create mode 100644 name.txt

```

دوباره برمی گردم در vscod تغییراتی در فایل میدهم.



```
EXPLORER ... Get Started name.txt M
PROJECTSTU3
name.txt M
1 ali
2 reza
3 nima
4
```

حال اگر دستور `git status` اجرا شود به ما اعلام می کند که فایل در `staged` هست اما تغییرات `commit` نشده.

```
C:\Users\SADAF COM\Desktop\projectstu3> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   name.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

با دستور `git add` تغییرات جدید را به `staged` می بریم و کامیت جدید اضافه می کنیم.

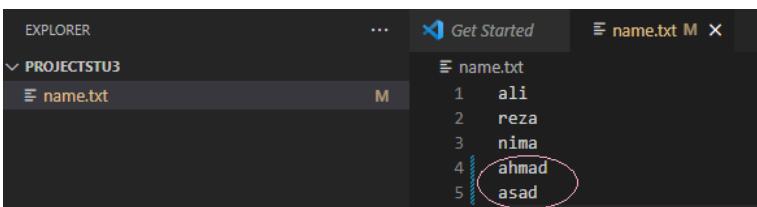
```
C:\Users\SADAF COM\Desktop\projectstu3> git add .
C:\Users\SADAF COM\Desktop\projectstu3> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    modified:   name.txt

C:\Users\SADAF COM\Desktop\projectstu3> git commit -m "add name ali-reza-nima"
[master ecf9a1a] add name ali-reza-nima
1 file changed, 1 insertion(+)
```

```
C:\Users\SADAF COM\Desktop\projectstu3> git log --oneline
ecf9a1a (HEAD -> master) add name ali-reza-nima
20ea643 initial project
```

تا اینجا کار دو تا `commit` داریم و هد رو مسٹر وی... اشاره دارد. حال دوبار در `vscod` در `working directory` `add name` را اجرا می کنیم.

تغییر جدید مثل دو اسم جدید اضافه می کنیم.



```
EXPLORER ... Get Started name.txt M
PROJECTSTU3
name.txt M
1 ali
2 reza
3 nima
4 ahmad
5 asad
```

```
C:\Users\SADAF COM\Desktop\projectstu3> git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
    modified:   name.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

دستور `git diff` را اجرا می کنیم این دستور تغییرات `staged` و `working directory` را نشان می دهد.

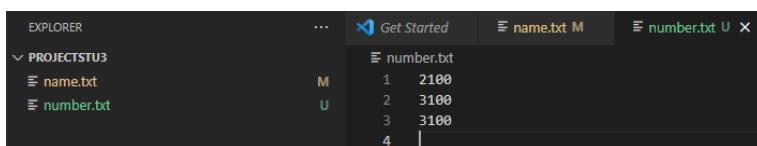
```
C:\Users\SADAF COM\Desktop\projectstu3>git diff
diff --git a/name.txt b/name.txt
index 4b2c8a6..4a20ea7 100644
--- a/name.txt
+++ b/name.txt
@@ -1,4 +1,5 @@
 ali
 reza
 nima
-
+ahmad
+asad
```

دستور `git diff --staged` stage area را commit آخرين مقاييسه می کند.

## Syntax

`$ git diff --staged`

در ادامه مثال قبل در `vs cod` يك فایل جدید به نام `number.txt` ایجاد می کنیم.



دستور `status` را اجرا کنید.

```
C:\Users\SADAF COM\Desktop\projectstu3>git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   name.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    number.txt

no changes added to commit (use "git add" and/or "git commit -a")
```

در گزارش اعلام شده نشان می دهد دو تا فایل داریم `name` و `number` در `stage` هست اما تغیرات جدید `name` در `commit` staged نشده است. با دستور `add` فایل `number` را به `staged` می بیریم.

```
C:\Users\SADAF COM\Desktop\projectstu3>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   number.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   name.txt
```

```
C:\Users\SADAF COM\Desktop\projectstu3>git diff --staged
diff --git a/number.txt b/number.txt
new file mode 100644
index 000000..3d97b92
--- /dev/null
+++ b/number.txt
@@ -0,0 +1,3 @@
+2100
+3100
+3100
```

▪ دستور `git diff head` کامیت های گذشته با working directory مقایسه می کند.

## Syntax

`$ git diff HEAD`

```
C:\Users\SADAF COM\Desktop\projectstu3>git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   number.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:  name.txt
        modified:  number.txt
```

```
C:\Users\SADAF COM\Desktop\projectstu3>git diff HEAD
diff --git a/name.txt b/name.txt
index 4b2c8a6..4a20ea7 100644
--- a/name.txt
+++ b/name.txt
@@ -1,4 +1,5 @@
 ali
 reza
 nima
-
+ahmad
+asad
diff --git a/number.txt b/number.txt
new file mode 100644
index 000000..944f588
--- /dev/null
+++ b/number.txt
@@ -0,0 +1,5 @@
+2100
+3100
+3100
+4100
+5100
```

▪ دستور بازگشت به commit

```
C:\Users\SADAF COM\Desktop\git\projectstu4>git log --oneline --all
5ccb5db (HEAD -> master) insert 3line file 4   هد به این کامیت انتشاره دارد
12401c2 add gitignor
a55a3e8 add file4
d99154e (bugfix) del line3 file 2   می خواهم به این کامیت برگرد
2da60b4 add color file3
d979fb5 insert 3 name file1 & 3 number file2
91b4970 initial project
```

ابتدا به شاخه bugfix سویچ میکنیم

```
C:\Users\SADAF COM\Desktop\git\projectstu4>git switch bugfix
Previous HEAD position was 2da60b4 add color file3
Switched to branch 'bugfix'
```

```
C:\Users\SADAF COM\Desktop\git\projectstu4>git log --oneline --all
5ccb5db (master) insert 3line file 4
12401c2 add gitignor
a55a3e8 add file4
d99154e (HEAD -> bugfix) del line3 file 2
2da60b4 add color file3
d979fb5 insert 3 name file1 & 3 namber file2
91b4970 inital project
```

حال دستور git checkout را اجرا می کنیم

```
C:\Users\SADAF COM\Desktop\git\projectstu4>git checkout 2da60b4
Note: switching to '2da60b4'.

You are in 'detached HEAD' state. You can look around, make experimental
changes and commit them, and you can discard any commits you make in this
state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may
do so (now or later) by using -c with the switch command. Example:

  git switch -c <new-branch-name>

Or undo this operation with:

  git switch -

Turn off this advice by setting config variable advice.detachedHead to false

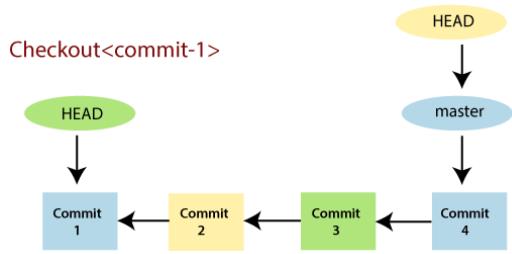
HEAD is now at 2da60b4 add color file3
```

اگر دوباره git log --oneline --all

دستور git push



دستور git push برای بارگذاری محتويات مخزن Local به مخزن remote استفاده می شود. با push کردن، شما commit های خود را به مخزن remote منتقل می کنید این دستور برعکس git fetch است. همان طور که commit، fetch کردن، local commit، remote commit توسط دستور git remote انجام می شوند.



## ۴- گیت هاب (GitHub)

گیت هاب، یک پلتفرم وب است که بر پایه‌ی گیت، سیستم کنترل نسخه، کار می‌کند و توسعه دهنده‌گان را قادر می‌سازد تا کدهای برنامه‌نویسی خود را در یک محیط ابری به اشتراک بگذارند، با سایر توسعه دهنده‌گان همکاری کنند، پروژه‌ها را مدیریت نمایند و نسخه‌ها را کنترل کنند. گیت هاب از هر دو مدل مشارکت عمومی (پروژه‌های متن باز) و خصوصی پشتیبانی می‌کند و به کاربران اجازه می‌دهد تا پروژه‌ها، سیستم‌ها و ابزارهای خود را به صورت آنلاین اداره کنند.

- توسعه پلتفرم گیت هاب در اکتبر سال ۲۰۰۷ آغاز و سال ۲۰۰۸ توسط تام پرستون. کریس ونسترت. پی جی هیت پایه گذاری گردید
- گیت هاب یکی از واژه‌های پرکاربرد در زمینه برنامه نویسی و مدیریت پروژه است. بعنوان دهکده جهانی برنامه نویس‌ها شناخته شده
- گیت هاب اساساً یک پلتفرم توسعه است که از نحوه کار توسعه دهنده‌گان الهام گرفته است. در واقع در گیت هاب شما میتوانید پروژه خود را مطرح کنید، پروژه تیم خود را توسعه دهید
- یک سرویس هاستینگ که از سیستم کنترل ورژن git استفاده می‌کند
- یکی از بزرگترین انجمن‌های توسعه دهنده‌گان وب در جهان است
- دارای سرویس رایگان برای پروژه‌های متن باز و سرویس پولی پروژه‌های تجاری
- کاربرد گیت هاب تنها اشتراک گذاری کدهای منبع نیست.
- رابط کاربری بسیار کاربرپسند طراحی شده
- کدهای پروژه‌های درسی خود را در گیت هاب قرار دهید تا برای همیشه به آن‌ها دسترسی داشته باشید
- این پلتفرم یک رزومه آنلاین برای برنامه نویسان است
- کلیه سوابق و تغییرات پروژه‌ها نگهداری می‌شوند و در هر لحظه‌ای شما می‌توانید پروژه را به حالت قبل برگردانید
- امکان ایجاد کانال‌های خصوصی برای ارتباط با تعداد خاصی از برنامه نویسان و جلوگیری از دسترسی دیگران وجود دارد

### ویژگی‌های GitHub

GitHub مکانی است که برنامه نویسان و طراحان با هم کار می‌کنند. آنها با هم همکاری می‌کنند، مشارکت می‌کنند و با گذاشت و برطرف می‌کنند. این میزبان تعداد زیادی پروژه منبع باز و کدهای زبان‌های برنامه نویسی مختلف است.



برخی از ویژگی‌های مهم آن به شرح زیر است.

- |                         |                     |
|-------------------------|---------------------|
| ▪ همکاری                | ▪ میزبانی مخازن Git |
| ▪ ردیابی مشکل           | ▪ مدیریت پروژه      |
| ▪ نمایش گرافیکی شاخه‌ها | ▪ مدیریت تیم        |

## گفتگو

میزانی کد

ردیابی و تعین وظایف

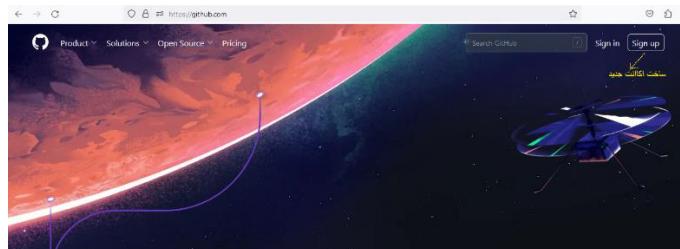
## استفاده از گیت هاب

نکته‌ای که باید مد نظر داشته باشید این است که گیت هاب به عنوان یک سرویس وب، نیازی به نصب در سیستم عامل شما ندارد. برای استفاده از این سرویس، شما می‌توانید به وب‌سایت <http://github.com> مراجعه کنید و یک حساب کاربری ایجاد نمایید. البته ابتدا باید گیت در سیستم شما نصب شده باشد با استفاده از آدرس ایمیل خود و وارد نمودن نام کاربری و پسورد در گیت هاب ثبت نام نمایید تا حساب کاربری شما ایجاد شود.

پس از ایجاد حساب از قابلیت‌های گیت هاب استفاده نمایید

### مرحله اول ساختن اکانت در گیت هاب

روش ساخت اکانت یا ثبت نام در git hub به این صورت است که ابتدا با آدرس <http://github.com> وارد سایت گیت هاب شوید سپس در سایت گیت هاب sign up را اجرا کنید.

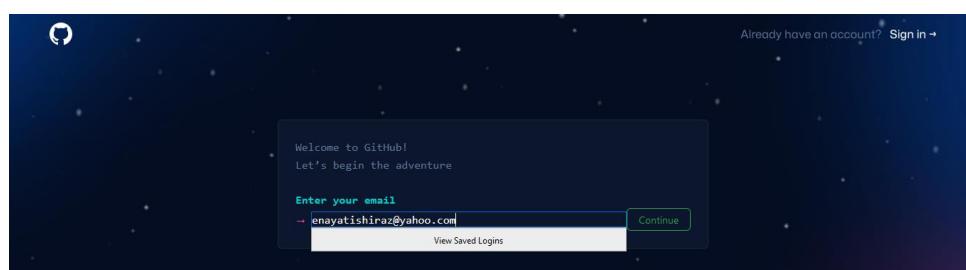


پس از اجرای sign up ایمیل خود را وارد کنید.



در صفحه بعد ایمیل خود را وارد کنید و continue را برای ادامه و بررسی ایمیل انتخاب کنید.

### پرکردن فرم ثبت نام



در صورت تایید ایمیل وارد مرحله زیر خواهد شد. در این بخش یک پسورد جهت ورورد به git hub اکانت خود وارد کنید.

#### ■ انتخاب plan

plan مناسب خود را انتخاب کنید. برای کارهای شخصی و پروژه‌های عمومی متن باز، plan رایگان کفایت می‌کند.

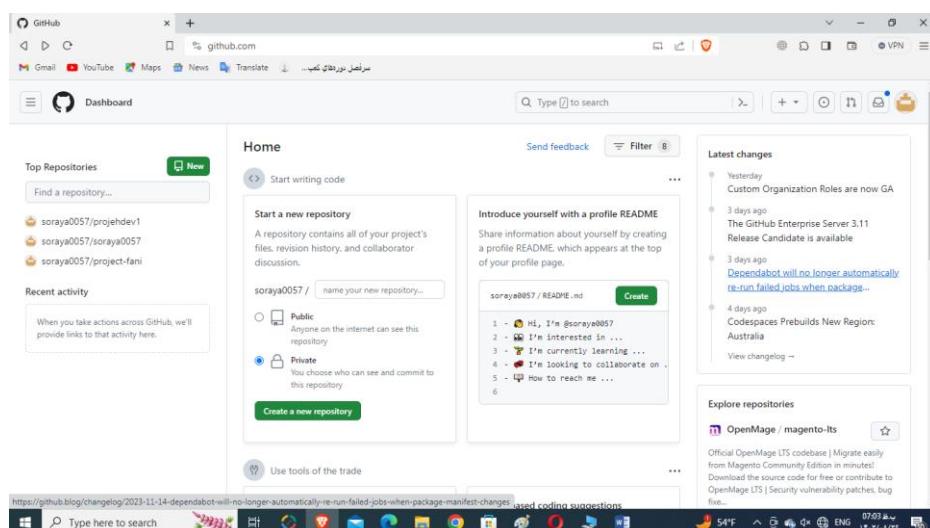
در صورتی که بخواهید در پروژه‌های خصوصی کار کنید یا به ویژگی‌های پیشرفته‌تری نیاز داشته باشید، ممکن است بخواهید یکی از plan های پرداختی را انتخاب کنید.

#### ■ تنظیمات اولیه

در این مرحله، گیت‌هاب ممکن است از شما برخی سوالات در مورد تجربیات، علایق و نحوه استفاده‌ی قصد دارید از پلتفرم پرسد. این اطلاعات به گیت‌هاب کمک می‌کند تا پیشنهادهای بهتری به شما ارائه دهد. همچنین می‌توانید تنظیمات اولیه‌ی پروفایل خود را انجام دهید، مانند افزودن یک تصویر پروفایل.

#### ■ تأیید ایمیل

به ایمیل خود رفته و بر روی لینک تایید که گیت‌هاب به شما فرستاده است کلیک کنید. حالا اکانت شما فعال و آماده استفاده است! حالا که یک اکانت کاربری روی گیت‌هاب دارید، می‌توانید پروژه‌های خود را ایجاد کنید، به پروژه‌های دیگران ملحق شوید، و با استفاده از این ابزار قدرتمند، توانایی‌های توسعه‌ی خود را به حداقل برسانید.



## نحوه استفاده از گیت هاب □

گیت هاب یک پلتفرم وب است که از کنترل نسخه گیت پشتیانی می کند و به توسعه دهنده گان امکان می دهد که کدهایشان را در یک محیط اشتراکی و همکاری محور برای پروژه های نرم افزاری قرار دهند. در ادامه، نحوه استفاده ابتدایی از گیت هاب را بررسی خواهیم کرد.

### مرحله ۱: ساخت ریپازیتوری در گیت هاب

- وارد حساب کاربری خود در گیت هاب شوید.
- روی دکمه **New** کلیک کنید تا یک ریپازیتوری جدید ایجاد کنید.
- به ریپازیتوری خود یک نام دهید، توضیحاتی ارائه دهید، مثلاً این که این مخزن حاوی چه پروژه ای است و یا چه کسانی روی آن کار می کنند و تصمیم بگیرید که آیا می خواهید ریپازیتوری شما عمومی باشد یا خصوصی. گرینه **public** به این معناست که به دیگران اجازه دسترسی به مخزن و ایجاد اصلاحات در داده هایتان را می دهید. اما **private** بودن مخزن شما به این معناست که فقط شما می توانید داده های مخزن را تغییر دهید.
- پس از طی کردن مراحل فوق شما می توانید یک فایل با عنوان **Readme** بسازید که به عنوان آرشیو پروژه باشد.
- آخرین گام برای ایجاد یک مخزن زدن دکمه **Create Repository** است تا مخزن شما با ویژگی هایی که مشخص کردید ایجاد شود.

### مرحله ۲: افزودن فایل به ریپازیتوری

می توانید مستقیماً از طریق وب سایت فایل ها را به ریپازیتوری خود اضافه کنید یا اینکار را با استفاده از کلاینت گیت انجام دهید. برای افزودن فایل از وب سایت، در ریپازیتوری خود به بخش **Code** بروید و روی دکمه **Upload files** کلیک کنید.

### مرحله ۳: کار با Branch ها

می توانید برای کار بر روی ویژگی های مختلف یا بخش های مختلف پروژه، شاخه های متفاوت ایجاد کنید. برای ایجاد یک شاخه جدید، در صفحه اصلی ریپازیتوری به بخش **Branch** بروید، نام جدیدی وارد کنید و اینتر بزنید.

### مرحله ۴: ایجاد Pull Request

وقتی که بر روی یک شاخه کار کردید و تغییراتی اعمال کردید، می توانید یک **Pull Request** ایجاد کنید تا تغییرات شما به شاخه اصلی ادغام شوند. برای این کار، به صفحه **Pull requests** بروید و روی دکمه **New pull request** کلیک کنید.

### مرحله ۵: همکاری و کنترل بیوشن

می توانید در پروژه های دیگران مشارکت کنید با این کار که به آن ها ستاره دهید، فورک کنید و یک **Pull Request** ارسال کنید. یا می توانید با مشاهده تب **Explore** پروژه های جدیدی را پیدا کنید و در آن ها مشارکت کنید.

با توجه به اینکه گیت هاب دارای ویژگی ها و امکانات بسیار زیادی است، این تنها یک شروع است و توصیه می شود با استفاده از راهنمای داکیومنت های موجود بر روی وب سایت گیت هاب، بیشتر با این ابزار آشنا شوید.

## ▪ در گیت هاب Fork

یکی از ویژگی های کلیدی در گیت هاب است که به کاربران این امکان را می دهد تا یک کپی از یک ریپازیتوری (مخزن) موجود بسازند و آن را به حساب کاربری خودشان منتقل کنند، بدون آنکه نیاز به دسترسی نویسنده به ریپازیتوری اصلی باشد. وقتی شما یک ریپازیتوری را فور کی می کنید، یک نسخه کاملا مشابه از آن ریپازیتوری، همراه با تمام تاریخچه کامیت ها و تغییرات، در حساب کاربری شما ساخته می شود.

این کپی، یک نسخه مستقل است که شما می توانید بر روی آن کار کنید، تغییرات ایجاد کنید و پول ریکوئست ها (PR) را به ریپازیتوری اصلی بفرستید. برای مثال، اگر شما بخواهید به یک پروژه متن باز کمک کنید اما دسترسی نوشتن به آن را نداشته باشید، می توانید یک فور کی از پروژه ایجاد کنید، تغییرات خود را در فور ک خود اعمال کنید و سپس یک پول ریکوئست به پروژه اصلی بفرستید تا تغییرات شما مورد بررسی قرار گیرد و احتمالاً به پروژه اصلی ادغام شود.

## ▪ پا پول ریکوئست در گیت هاب PR

که به اختصار PR نیز شناخته می شود در واقع یک درخواست برای ادغام تغییرات از یک شاخه به شاخه ای دیگر در یک ریپازیتوری گیت است. در محیط های گرافیکی مانند GitHub ، GitLab ، Bitbucket یا PR امکان ایجاد یک توضیح مفصل درباره تغییرات ایجاد شده، دیدگاه ها و بررسی کدها Code Review و همچنین ردیابی تغییراتی که با تغییر درخواست داده شده است را فراهم می آورد.

روال کار به این صورت است: شما تغییراتی را بر روی یک شاخه جداگانه ایجاد کرده و سپس یک PR ایجاد می کنید تا این تغییرات به یک شاخه دیگر (معمولًا master یا main) ادغام شود. این درخواست بازیبینی کد را فراهم می آورد و اعضای تیم می توانند کدهای پیشنهادی شما را مورد بررسی قرار دهند، نظراتی ارائه دهند و حتی تغییراتی پیشنهاد کنند. سپس، بر اساس توافق و پس از ارائه نظرات و تایید تغییرات، PR می تواند به شاخه اصلی پروژه ادغام شود.

Pull Request ها ابزاری بسیار قدرتمند برای توسعه هی همزمان و همکارانه بر روی پروژه های نرم افزاری هستند چرا که به توسعه دهنده ها امکان می دهد تا به صورت همزمان و بدون تداخل با کد هم دیگر کار کنند، و همچنین توسعه دهنده ها دیگر را به صورت ساخت یافته و کنترل شده به بازرسی و ارتقا کد خود دعوت کنند.

## ▪ بررسی کد یا Code Review در گیت هاب

بررسی کد یا Code Review یکی از فرآیندهای اصلی در توسعه نرم افزار است که در آن توسعه دهنده ها کد ارسالی یکدیگر را مورد بررسی و ارزیابی قرار می دهند تا اطمینان حاصل کنند که کد به درستی عمل می کند و با استانداردها و الگوهای طراحی نرم افزار سازگار است. این فرآیند می تواند به صورت همزمان و در زمان واقعی یا به صورت غیر همزمان و با استفاده از ابزارهایی مانند pull requests در سیستم های کنترل ورژن مانند Git انجام شود.

بررسی کد به چند دلیل مهم است.

- این فرآیند به کاهش احتمال خطأ و بهبود کیفیت نهایی محصول کمک می کند.
- این فرآیند فرهنگ یادگیری و به اشتراک گذاری دانش در بین توسعه دهنده ها را تقویت کرده و ممکن است مشکلات و مسائلی را که در مراحل ابتدایی قابل مشاهده نیستند، کشف کند.

■ به توسعه دهنده‌گان کمک می‌کند تا با کد و تغیرات جدید آشنا شوند، سبک نویسنده‌گان کد دیگر را یاد بگیرند و در نهایت به ساخت یک کد پایدار و قابل نگهداری کمک کنند.

بررسی کد می‌تواند به صورت داخلی در یک تیم یا میان تیمهای مختلف انجام شود و شامل بررسی کلی کد، ورودی/خروجی، عملکرد، معیارهای کیفی، استانداردهای نویسنده‌گی کد و غیره باشد. این فرآیند ممکن است به صورت رسمی و با دستورالعمل‌های خاص یا به صورت غیررسمی و به عنوان بخشی از فرهنگ توسعه‌ی تیم انجام شود.

## ۵-متدولوژی کانبان و متدولوژی اسکرام

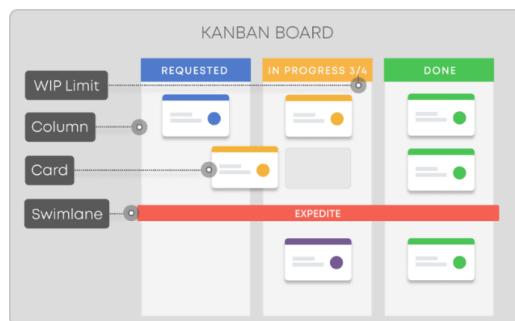
یکی از روش‌های مدیریت پروژه، روش چابک یا Agile است. این متد، با تقسیم پروژه به مراحل کوچکتر، انجام آن را ساده‌تر و عملی‌تر می‌کند. برای پیشرفت و موفقیت روش مدیریت چابک، کسب و کارها می‌توانند چارچوب‌های مختلفی مانند متدولوژی اسکرام یا متدولوژی کانبان را انتخاب کنند.

### متدولوژی کانبان (Kanban) □

کانبان یک کلمه ژاپنی است که از دو کلمه Kan (دیدنی) و Ban (کارت) تشکیل شده است. این سیستم برای اولین بار در دهه ۱۹۴۰ به وسیله تایچی اووهنو در گروه خودروسازی تویوتا مورد استفاده قرار گرفت. سیستم کانبان (Kanban)، چارچوبی مصور است که برای پیاده‌سازی و اجرای سیستم مدیریت چابک مورد استفاده قرار می‌گیرد. کانبان یک روش محبوب مدیریت گرددش کار ناب برای تعریف، مدیریت و بهبود خدمات برای سیستم‌های کاری است به زبان ساده سیستمی است که نشان می‌دهد چه محصولی، در چه زمانی و به چه مقدار باید تولید شود. این سیستم اطلاع رسانی تولید، از «کارت» به عنوان سیگنال بصری برای به حرکت درآوردن جریان مواد و قطعات در فرآیند تولید استفاده می‌کند. می‌توانید سیستم کانبان را یک کارت اطلاعاتی در نظر بگیرید که در کنترل تولید، بهخصوص در سیستم‌های جدید تولید کاربرد زیادی دارد. این کارت برای برنامه‌ریزی، هدایت فرآیند و زمان‌بندی عملیات تولید محصولات به کار می‌رود و معمولاً در جریان ساخت، در کنار سایر مواد قرار می‌گیرد. در واقع کانبان یک ابزار است که با به شرایط متفاوتی که در کسب و کارهای مختلف وجود دارد، نحوه استفاده از آن نیز متفاوت است.

#### تابلوی کانبان ▪

تابلوی Kanban ابزاری برای تجسم گرددش کار است که به شما کمک می‌کند تا روند کار خود را شفاف کنید و با محدود کردن کار در حال پیشرفت کارایی را افزایش دهید. به سرعت مراحل کار مشکل ساز را شناسایی خواهید کرد، کارایی تیم را افزایش می‌یابد به این صورت حداقل مقدار کارهایی که در هر مرحله باید انجام شوند، مشخص می‌کند.



▪ **کارت کانبان** – این نمایش بصری وظایف است. هر کارت حاوی اطلاعاتی در مورد وظیفه و وضعیت آن است، مانند مهلت، تعیین کننده، توضیحات و غیره.

ستون های کابنан – هر ستون روی تابلو نشان دهنده مرحله متفاوتی از گردش کار شما است. ساده‌ترین شکل تابلو شامل ۳ ستون است که

ubar tnd az:

- ستون انجام دادن (To Do)
- ستون در حال پیشرفت (In Progress)
- ستون انجام شده (Done)

حدودیت های کار در حال پیشرفت<sup>۱</sup> – wiplimit حداکثر مقدار کارها را در مراحل مختلف گردش کار محدود می‌کند. محدود کردن

WIP به شما این امکان را می‌دهد که با کمک به تیم خود فقط بر روی کارهای فعلی تمرکز کنید، موارد کاری را سریع‌تر به پایان برسانید.

خطوط افقی هستند که می‌توانید از آنها برای جداسازی فعالیت‌های مختلف، تیم‌ها، کلاس‌های خدمات و موارد

دیگر استفاده کنید.

– کامیت نقطه‌ای را در فرآیند کار نشان می‌دهد که در آن یک کاری آماده است تا به سیستم کشیده شود.

نقطه تحويل – نقطه‌ای در گردش کار که در آن موارد کار تمام شده در نظر گرفته می‌شود.

### مزایای روش کابنان

تصویری بودن، مهم‌ترین مزیت روش کابنان برای چابک سازی پروژه‌ها است. عملکرد تابلوی کابنان بسیار ساده و سریع است و با بهبود گردش کار، زمان چرخه را به حداقل می‌رساند. اما روش کابنان مزایای دیگری هم دارد عبارتنداز

روشی انعطاف‌پذیر: کابنان یک مدل شناور، منعطف و در حال گسترش است. به این صورت که اگر اطلاعات جدیدی وارد سیستم شود، می‌توانید اولویت‌ها را دوباره ارزیابی کنید.

یک مدل ساده و قابل فهم: کابنان، روشی مبتنی بر تصویر است، با این روش تصویری، فهم و درک پروژه به میزان قابل توجهی ساده می‌شود. همچنین Kanban می‌تواند به سادگی بر روی سایر سیستم‌های مدیریتی پیاده سازی شود.

صرفه جویی در منابع مختلف: تمرکز روش کابنان بر کاهش اتلاف منابع زمانی و مالی است. بنابراین می‌توانید مطمئن باشید که تیم شما کار غیرضروری یا اشتباهی انجام نمی‌دهد و منابع شما هدر نمی‌رود.

بهبود جریان تحويل به مشتری: کابنان، جریان تحويل محصولات و خدمات به مشتری را بهبود می‌بخشد و کار را در دوره‌های زمانی منظم ارائه می‌کند. این روش بر تحويل به موقع یا "JIT" متمرکز است.

به حداقل رساندن زمان چرخه: زمان چرخه، مدت زمانی است که فرآیند به طور کامل از جریان کار عبور می‌کند. در پروژه‌های کابنان اعضای تیم تلاش می‌کنند که مراحل کار با سرعت و موقفيت طی شود.

### معایب روش کابنان

ایجاد مشکلات با تخته‌های به روز نشده: اعضای تیم باید اطلاعات تابلوی کابنان را به طور مرتباً به روز رسانی کنند. در غیر این صورت، کار خود را بر اساس اطلاعات ناقص و اشتباه انجام می‌دهند. اگر کار با این اطلاعات غلط پیش برود، بازگشت به عقب، بسیار مشکل و در مواردی غیرممکن خواهد بود.

<sup>2</sup> wiplimit  
<sup>2</sup> just in time delivery  
<sup>2</sup> Workflow

1

2

3

- **ایجاد مشکلات با تخته‌های پیچیده:** تابلوی کابان باید ساده، تمیز و به سادگی قابل خواندن و درک کردن باشد. تخته‌های شلوغ و پیچیده نه تنها هیچ فایده‌ای ندارند بلکه فقط پیچیدگی فرآیندها و سردرگمی اعضاً تیم را افزایش می‌دهند.
- **ایجاد مشکلات به دلیل عدم توجه به زمانبندی:** یکی از مشکلات متداول در سیستم کابان، مشخص نبودن زمان اجرای فعالیت‌ها است. ستون‌های تخته کابان بر مبنای «فاز فرآیندها» نوشته می‌شوند و چون برای اجرای هر فاز، زمانی در نظر گرفته نشده، شما نمی‌توانید بفهمید چقدر زمان لازم است تا مثلاً فاز «انجام دادن» به پایان برسد.

## متدولوژی اسکرام (scrum) □

اسکرام توسط تیم توسعه نرم‌افزار مورداستفاده قرار می‌گیرد. در واقع این مشهورترین متدولوژی اجیل است. برطبقی دوازدهمین گزارش سالانه Agile ۷۰ درصد تیم‌های نرم‌افزار از اسکرام یا اسکرام ترکیبی استفاده می‌کنند. اسکرام یک فریم ورک است که به تیم‌ها کمک می‌کند با هم کار کنند، تیم را تشویق می‌کند تا تجربه بیندوزند، در حالی که در حال رفع یک مشکل هستند، خودسازماندهی کنند و از برد و باخت‌ها درس بگیرند تا پیشرفت مداوم داشته باشند.



## ارزش‌های یک تیم اسکرام □

اعضاً یک تیم اسکرام باید ارزش‌های زیر را یاد بگیرند:



- **تعهد:** هر کدام از اعضای تیم، شخصاً باید متعهد به دستیابی به اهداف تیم باشند.
- **جرأت، شجاعت و رشادت:** اعضای تیم باید شجاعت رو به رو شدن با چالش‌ها را داشته باشند و قادر باشند کار را به درستی انجام دهند و روی مشکلات سخت کار کنند.
- **تمه‌کوز:** هر شخص باید تمرکز خود را بر روی کار مشخص شده برای پیشرفت و اهداف تیم بگذارد.
- **گشودگی و رکبودن:** اعضای تیم و ذینفعان در مورد همه کارها و چالش‌هایی که تیم با آن روبرو است، رکب باشند و با دید باز آن را پذیرند.
- **احترام گذاشتمن:** اعضای تیم به توانایی و استقلال یکدیگر، احترام بگذارند.

## ▪ شش اصل کلی در اسکرام

چارچوب اسکرام بر شش اصل اساسی تکیه دارد. دستورالعمل هایی که باید در هر پروژه دنبال شوند تا تمرکز تیم از دست نرود و پروژه به موفقیت برسد.

۱. **کنترل فرآیند تجربی**: در اسکرام، فرآیند تجربی به جای تئوری مبتنی بر مشاهده شواهد سخت و آزمایش است. برای کنترل فرآیند تجربی سه ایده وجود دارد:

- شفافیت
- بازرگانی
- انطباق

به این معنا که هر مرحله- کاملاً واضح و روشن باشد، فعالیت‌های مرتبط مورد بازرگانی قرار گیرد تا امکان خطای خطا را به حداقل برساند و با هدف کلی و نهایی منطبق باشد.

۲. **خودسازماندهی**: از آنجایی که فرآیند اسکرام به افراد زیادی متکی است، خودسازماندهی اصلی ضروری و مهم است. همه افراد در گیر در پروژه، این اختیار را دارند که به طور مستقل کار کنند. ولی اصل خودسازماندهی امکان تعهد و جلب مشارکت در بین اعضاء را فراهم می‌کند.

۳. **همکاری**: اسکرام یک فرآیند مشارکتی است. بنابراین اصل همکاری در اسکرام بر سه بعد متصرکز است:

- آگاهی
- بیان
- تخصیص

اعضا با آگاهی از عملکرد یکدیگر و همکاری در پیش‌برد کار سریع تر به نتیجه می‌رسند.

۴. **اولویت‌بندی بر ارزش**: این اصل شامل سازماندهی و اولویت‌بندی وظایف براساس ارزش آنها و چگونگی تکمیل آنها می‌شود.

۵. **تایم اسپرینت<sup>۱۶</sup>**: در اسکرام، کارها به صورت اسپرینت انجام می‌شوند و مدت زمان مشخصی برای هر کدام تعیین می‌شود. سایر عناصر، از جمله برنامه‌ریزی اسپرینت و جلسات روزانه نیز زمان شروع و توقف مشخصی دارند. این زمان‌بندی تضمین می‌کند که همه افراد در گیر می‌دانند با احتساب حذف زمان تلف شده و تأخیرات، دقیقاً چه مقدار زمان برای هر مرحله اختصاص داده شده است.

تایم اسپرینت: مدت زمانی که تیم پروژه برای انجام یک سری اقدامات بر روی محصول دارد تا وظایف خود را انجام دهد

۶. **توسعه تکراری**: این اصل نهایی بیانگر این درک است که یک پروژه ممکن است در طول فرآیند توسعه نیاز به پالایش چندین بار داشته باشند. توسعه تکراری درواقع به تیم اجازه می‌دهد تا تنظیمات را انجام دهد و تغییرات را آسان‌تر مدیریت کند.

## ▪ ساختار متدولوژی اسکرام

چارچوب اسکرام اکتسابی است و بر مبنای یادگیری مداوم و سازگاری با عوامل متغیر قرار گرفته است. اسکرام واقع است به اینکه همه تیم از همان شروع پروژه همه چیز را نمی دانند و در خلال تجربه بهتر می شوند. اسکرام طوری ساختار یافته که به تیم ها کمک کند تا به طور طبیعی با شرایط در حال تغییر و نیازهای کاربر سازگار شوند و این کار با اولویت بندی مجدد که در پروسه صورت می گیرد، انجام می شود، به همین خاطر تیم شما مدام در حال یاد گرفتن و پیشرفت خواهد بود. در شکل زیر ساختار متدولوژی اسکرام نشان داده شده است.



متدولوژی اسکرام توسط نقش های تیم (Role)، رویدادها (Event)، مصنوعات (Artifact) و قوانین (Rule) تعریف می شود.

### ۱. نقش های تیم :

تیم های اسکرام معمولاً از ۷ عضو (با ۲ عضو بیشتر یا ۲ عضو کمتر) تشکیل می شوند و هیچ رهبر تیمی برای تفویض وظایف یا تصمیم گیری در مورد حل مسائل وجود ندارند. تیم به عنوان یک واحد تصمیم می گیرد که چگونه به مسائل پردازد و مشکلات را حل کند. هر یک از اعضای تیم اسکرام، بخش جدایی ناپذیری از راه حل هستند و از ابتدا تا انتهای محصول، در آن نقش دارند و تصمیم می گیرند. سه نقش اصلی در یک تیم اسکرام وجود دارد.

## ▪ مالک محصول (Product Owner)

مالک محصول، کسی است هدف و خروجی محصول را تعیین میکند، چون ذینفع اصلی است. مالک محصول، ایده یا هدفی را تصور کرده و تیم را موظف به اجرای آن می کند. همچنین نیازهای تجاری محصول را کشف و درک می کنند. مسئولیت های مالکان محصول شامل موارد زیر است.

- توسعه و بیان صریح هدف محصول
- صورت حساب واضح از بکلاگ محصول (یا همان لیست رتبه بندی شده از تمام فاکتورهای گنجانده شده در یک محصول)
- سفارش بکلاگ محصول براساس اولویت بندی
- حصول اطمینان از شفاف بودن، قابل مشاهده بودن و قابل درک بودن بکلاگ محصول
- ارتباط با ذینفعان خارجی و انتقال نیازها و انتظارات آنها به تیم

## ▪ مدیر محصول(Scrum Master)

حفظ نظم و همسویی در تیم اسکرام با مدیر محصول است اکثر اعضای تیم برای گرفتن مشاوره و راهنمایی‌های لازم از شخص اسکرام مستر کمک می‌گیرند. این وظیفه اسکرام مستر است که تیم را منظم و اکتیو نگه دارد، موانع پیشرفت کار را بردارد و ارتباط بین مالک محصول و سایر اعضای تیم را توسعه دهد و تسهیل کند. پایبندی به اصول، فرآیندها و شیوه‌های چابک، هسته اصلی مأموریت اسکرام مستر است. مدیر محصول همان‌طور که به تیم توسعه خدمت می‌کند، به سازمان و صاحب محصول نیز خدمات ارائه می‌دهد. از جمله وظایف اسکرام مستر:

- کمک به یافتن تکنیک‌هایی برای تعریف هدف محصول و مدیریت بک‌لاگ محصول
- رهبری، آموزش و مریبی گری تیم
- برنامه‌ریزی و مشاوره‌ی اجرای اسکرام در طول پروژه است.

## ▪ تیم توسعه

اجرای اسپرینت‌ها اینکه چطور و چگونه انجام شود با تیم توسعه است. تیم توسعه مسئول اجرای طرحی هستند که توسط مالک محصول ارائه و تحت هدایت مدیر محصول قرار دارد. تیم توسعه مشکل از افرادی است که هر یک مهارت خاص خود را دارد و به وظیفه تعیین شده‌ای مشغول می‌شود. به طور مثال تیم توسعه شامل؛ تولید محتوا، گرافیست مدیر خلاق برای تأیید طرح کلی آگهی و همچنین مهندسان کامپیوتر و تحلیلگران است.

به طور کلی یک تیم توسعه وظایف زیر را به عهده دارد:

- کمک به برنامه‌ریزی و رسیدن به هدف
- استفاده از داده‌ها برای پیدا کردن بهترین شیوه‌ها برای توسعه-ی محصول
- تست محصولات و نمونه‌های اولیه
- رفع اشکالات و تضمین کیفیت

## ۲. رویدادهای اسکرام

متداول ترین رویدادهای اسکرام به شرح زیر است

- اسپرینت (sprint)
- برنامه ریزی اسپرینت (sprint planning)
- اسکرام روزانه (Daily Scrum)
- بررسی اسپرینت‌ها (Sprint Review)
- بازنگری اسپرینت‌ها (Sprint Retrospective)

**اسپرینت:** مدت زمانیست که حداقل آن یک ماه است و در طی این مدت زمان تکرار شونده، تیم توسعه موظف است که موارد ذکر شده در بک‌لاگ اسپرینت را پیاده سازی کنند.

## ویژگیهای اسپرینت:

- ۱- هر اسپرینت، زمان-ثابت است: اسپرینت در حقیقت از مفهومی به نام روش زمان ثابت<sup>۵</sup> که تکنیکی در مدیریت زمان است، گرفته شده به این معنا که تاریخ شروع و پایان مشخصی دارد. طی دوره زمان ثابت، بایستی تیم با سرعت و آهنگی پایدار، مجموعه منتخبی از کارها را مطابق با هدف اسپرینت انجام دهد.
- ۲- هر اسپرینت، کوتاه است: همانطور که گفتیم، هر اسپرینت بین یک هفته تا یک ماه است که به نوبه خود کوتاه است؛ این کوتاهی چندین مزیت دارد عبارتند از: **سهولت برنامه ریزی، بازخورد سریع، محدود کردن اشتباه، کاهش زمان بازگشت سرمایه، حفظ شور و هیجان، نقاط کنترلی متوالی**
- ۳- مدت اسپرینت‌ها یکسان است: وقتی برای اسپرینت‌ها از مدت زمان یکسان استفاده می‌کنیم، منجر به افزایش بهره‌گیری از ریتم و همچنین سادگی برنامه ریزی می‌شود.
- ۴- هدف اسپرینت تغییر ناپذیر است: یکی از قواعد مهم اسکرام این است که پس از تعیین هدف و آغاز اجرای اسپرینت، مجاز نیستیم هدف اسپرینت را تغییر دهیم

- **برنامه ریزی اسپرینت (sprint planning)**: در متد چاپک قرار است فعالیت‌های تیم اسکرامی به بخش‌های کوچک‌تر یا همان اسپرینت‌ها تقسیم شوند. برای هر اسپرینت باید هدفی درنظر گرفته شود و بعد از پایان هر اسپرینت باید بازخوردها درمورد همان بخش کاملاً واضح بیان شوند. بنابراین به فرایندی به نام برنامه ریزی اسپرینت نیاز داریم.
- **قدم اول؛ چه چیزی What**: هر مالک محصول، لیستی از ویژگی‌های محصولش را دارد. مالک محصول می‌خواهد محصولی که تولید می‌شود ویژگی‌های مشخصی داشته باشد تا به کمک آن ویژگی‌ها بتواند مزیت‌های مورد نظرش را ایجاد کند. به این ویژگی‌ها یوزر استوری (user story) می‌گویند. که در هر چرخه اسپرینت قرار است مجموعه‌ای از یوزر استوری‌ها به سرانجام برسند. بنابراین باید لیستی تهیه شود و یوزر استوری‌های درخواستی به ترتیب اهمیت در لیستی مشخص و مرتب شوند. به این لیست بکلاگ محصول می‌گویند.
- **قدم دوم؛ چطور How**: بعد از اینکه هدف‌ها کاملاً مشخص شدند، باید برنامه ریزی برای چگونگی انجام کارها صورت بگیرد. در این مرحله تیم توسعه یا همان تیم انجام دهنده‌ی مراحل برای چگونگی انجام کارها برنامه ریزی می‌کنند و با مالک محصول درمورد مدت زمان رسیدن به اهداف به توافق می‌رسند.
- **قدم سوم؛ چه کسی Who**: مسئولیت هر کدام از اعضاء در این مرحله به‌طور دقیق و کامل مشخص می‌شود.
- **قدم چهارم؛ Input**: در این مرحله باید نیازهای پروژه معلوم شود و اعضاء مسئولیت تهیی نیازها را به‌عهده می‌گیرند. اینجا دقیقاً مشخص می‌شود که هر کس وظیفه‌ی تهیی چه نیازی را به‌عهده دارد.
- **قدم پنجم؛ Output**: در این مرحله هدف، بازه‌ی زمانی اسپرینت‌ها، چگونگی انجام کارها همه و همه دقیقاً معلوم و مشخص است و مانعی برای شروع وجود ندارد. مدت زمان هر اسپرینت به نوع محصول تاب بستگی دارد. برای توسعه و طراحی نرم‌افزارها اسپرینت‌های یک‌ماهه و برای پروژه‌های بازاریابی و ... در بیشتر مواقع اسپرینت‌های دو هفته‌ای درنظر گرفته می‌شوند. به‌طور متوسط برای هر هفته از اسپرینت به برگزاری جلسه‌ای یک تا دو ساعته نیاز داریم. مثلاً اگر قرار باشد پروژه‌ی شما در ۴ اسپرینت دو هفته‌ای به سرانجام برسد باید جلسه‌ی برنامه ریزی اسپرینت نهایتاً شانزده ساعته برگزار شود.

**اسکرام روزانه(Daily Scrum):** اسکرام روزانه رویدادی است که توسط تیم توسعه انجام می شود، برنامه‌ای ۱۵ دقیقه‌ای است که طی آن، اعضاً تیم توسعه محصول فعالیت‌های خود را با هم هماهنگ می‌کنند. این امکان را به تیم می‌دهد تا با یک برنامه جدید برای ۲۴ ساعت آینده، کارهای انجام شده در ۲۴ ساعت گذشته را بازرسی کند و با رسیدن به هدف اسپرینت سازگار شود.

#### ▪ انواع جلسات اسکرام:

۶ نوع جلسه‌ی اسکرام وجود دارد که در یک زمان خاص در طول چرخه اسپرینت برگزار می‌شود و هر نوع به یک هدف متمایز می‌پردازد.

- جلسه برنامه ریزی اسکرام (Sprint Planning Meeting)
- جلسه روزانه اسکرام (Daily Scrum Meeting)
- جلسه نقد و بررسی اسپرینت (Sprint Review Meeting)
- جلسه بازنگری اسپرینت (Sprint Retrospective Meeting)
- جلسه اصلاح بک لاغ (Backlog Refinement Meeting)

**جلسه برنامه ریزی اسکرام (Sprint Planning Meeting):** در ابتدای هر اسپرینت، جلسه برنامه ریزی اسپرینت با کل تیم برگزار می‌شود. هدف این جلسه توسعه موارد بک لاغ اسپرینت به صورت واقع‌بینانه و تعریف اولویت‌بندی وظایفی است که باید در طول هر اسپرینت انجام شود. در طول جلسه، اعضاً تیم میزان کارهایی را که می‌توانند در اسپرینت انجام دهند را تعیین می‌کنند و کار را از بک لاغ محصول به بک لاغ اسپرینت منتقل می‌کنند. این مرحله نیاز به برنامه ریزی زیادی دارد و چندین ساعت طول می‌کشد تا گروه در مورد اسپرینت نهایی تصمیم‌گیری کند. بنابراین در پایان این جلسه، تیم توسعه یک هدف اسپرینت و همچنین یک بک لاغ اسپرینت در اختیار دارد.



**جلسه روزانه اسکرام (Daily Scrum Meeting):** جلسات روزانه اسکرام هر روز به مدت ۱۵ دقیقه، اعضاً تیم دور هم جمع می‌شوند تا هرگونه مشکل یا پیشرفت در وظایف خود را گزارش دهند. با اینکه استنداپ های روزانه کوتاه است اما این جلسه بخشی اساسی از فرآیند اسکرام محسوب می‌شود. این برنامه طوری طراحی شده است که همه اعضاً گروه را به صورت منسجم در مسیر خود نگه دارد. معمولاً مالک محصول هم در جلسات روزانه اسکرام حضور دارد تا به هر طریقی به تیم کمک کند. در واقع در این جلسات به سه پرسش زیر پاسخ داده می‌شود:

- دیروز به چه نتیجه‌ای رسیدید و چه کاری انجام دادید؟
- امروز مشغول چه کاری هستید؟
- آیا موانعی بر سر راه شما وجود دارد؟

این سوالات منع عالی برای کسب اطلاعات در مورد پیشرفت کار و نحوه همراهی افراد با کار توسعه است.

■ جلسه نقد و بررسی اسپرینت (Sprint Review Meeting): در پایان هر اسپرینت، یک جلسه نقد و بررسی اسپرینت برگزار می شود. هدف اصلی این جلسه نشان دادن عملکرد محصول و آنچه که در یک اسپرینت خاص به دست آمده است. در این جلسه صاحب محصول، اسکرام مستر و ذینفعان برای بررسی محصول و پیشنهاد تغییرات یا بهبودها حضور دارند.

■ جلسه بازنگری اسپرینت (Sprint Retrospective Meeting): هدف از برگزاری این جلسه بررسی درست و غلط های انجام شده در طول اسپرینت است. این جلسه فرصتی عالی را برای همه اعضا تیم فراهم می کند تا در مورد کار و پیشرفت هایی که باید انجام شود، بازنگری کنند. تیم آشکارا درباره نگرانی های سازمانی و کار گروهی خود صحبت می کند. به طوری که این گفتگو باید بی طرفانه و بدون قضاوت باشد. جلسه بازیبینی بخش مهمی از ایجاد و توسعه تیم است و همچنین برای پروژه های بعدی اسکرام بسیار مهم است.

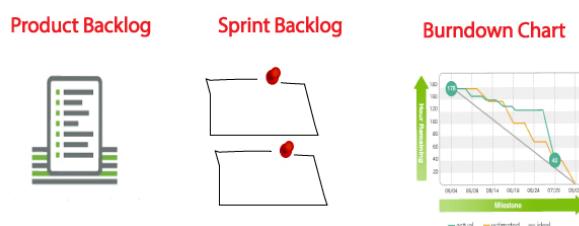
■ جلسه اصلاح بک لاغ (Backlog Refinement Meeting): آخرین نوع جلسات اسکرام، جلسه اصلاحات بک لاغ است. در این جلسه، اعضای تیم بر کیفیت و مهارت کار در طول اسپرینت تمرکز می کنند. این جلسه برای ارتباط صاحبان مشاغل با تیم توسعه ضروری است و برای ارزیابی کیفیت محصول نهایی مورد استفاده قرار می گیرد. در جلسه اصلاحات، آیتم های بک لاغ طبقه بندی شده و پس از بحث فنی با تیم اولویت بندی می شوند تا مطمئن شوند که تیم درک کاملی از این که دقیقاً چه چیزی قابل تحویل است و الزامات آن چیست، داشته باشد. برگزاری جلسه اصلاح بک لاغ نیاز به یک جلسه برنامه ریزی طولانی مدت اسپرینت را کاهش می دهد و به تیم این فرصت را می دهد تا قبل از تعهد کامل به اهداف اسپرینت، در مورد موارد بک لاغ تأمل کند.

■ بررسی اسپرینت ها (Sprint Review): رویداد Sprint Review در پایان هر اسپرینت انجام می شود و هدف آن کمک به پیشرفت مدام است. این رویداد به تیم کمک می کند تا افزایش بهره وری را تعیین کنند و در آینده نیز روی خصوصیات مثبت تمرکز کنند.

■ بازنگری اسپرینت ها (Sprint Retrospective): در این بخش که می توان آن را دور پایانی اسپرینت تلقی کرد، تیم اسکرام تلاش می کند تا بداند چه مواردی برای آینده اثربخش است و باید بهبود پیدا کند. این رویداد به اعضای تیم و مدیریت فرصتی می دهد تا در زمان تعیین شده به صورت تعاملی روی بهبود کیفیت اسپرینت های آتی کار کنند و طرح های عملی پیشنهاد دهند.

### ۳. مصنوعات اسکرام (Artifact scrum)

انواع Artifact در اسکرام عبارت اند از:



■ بک لاغ محصول (Product Backlog): بک لاغ یا سند محصول در ابتدای لیست قرار گرفته و باید توسط صاحب محصول یا مدیر محصول انجام شود. این لیستی پویا از ویژگی ها، نیازمندی ها، پیشرفت ها و اصلاحاتی است که به عنوان ورودی بک لاغ اسپرینت عمل می کنند.

در واقع لیستی از تکالیف یا کارهایی است که تیم باید انجام دهد. بک لاگ محصول مدام بازیبینی و اولویت‌بندی مجدد می‌شود و توسط صاحب محصول نگهداری می‌شود.

**بک لاگ اسپرینت (Sprint Backlog)**: بک لاگ یا سند اسپرینت لیستی از آیتم‌ها، story‌های کاربران یا رفع اشکالات است که توسط تیم توسعه برای اجرا در چرخه اسپرینت فعلی انتخاب شده است. پیش از هر اسپرینت، در جلسه برنامه‌ریزی اسپرینت، تیم انتخاب می‌کند کدام آیتم از بک لاگ محصول بر روی اسپرینت کار خواهد کرد. سند اسپرینت می‌تواند انعطاف‌پذیر باشد و در طول اسپرینت تکامل یابد

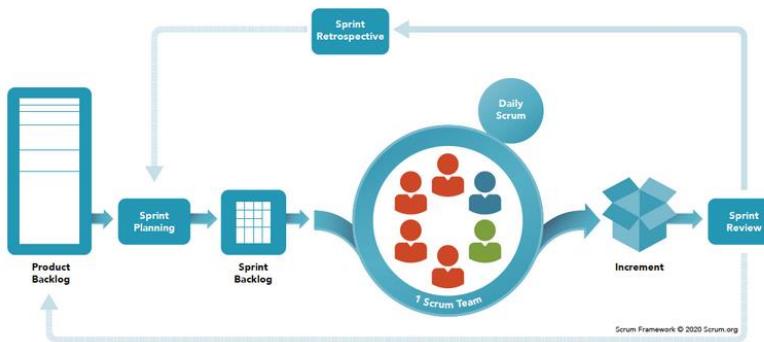
**نمودار فرسودگی burndown chart**: نمودار فرسودگی نتیجه اسپرینت است که پیشرفت را در یک اسپرینت نشان می‌دهد. نمودار فرسودگی نشان می‌دهد که چگونه روی سرعت کار می‌کنید. در نمودار فرسودگی، نمودار از زمانی شروع می‌شود، یعنی از جایی که فعالیت شروع می‌شود، و در پایان اسپرینت، نمودار به صفر می‌رسد، جایی که فعالیت به پایان می‌رسد. به طور کلی یک خط مایل از بالا به پایین است.

**افزایش یا جمع بندی (increment)**: جمع بندی یا افزایش در ماموریت انجام شده یا وظایف انجام شده معنا پیدا می‌کند. در واقع جمع بندی هدف نهایی اسپرینت است. البته بستگی دارد تیم شما کار انجام شده و هدف اسپرینت را چه تعریف می‌کنند. ممکن است آنچه شما تعریف کرده‌اید خیلی واقع‌بینانه نباشد. تصور کنید بر روی یک محصول مبتنی بر سرور کار می‌کنید که هر سه ماه برای مشتریان ارسال می‌شود. ممکن است تصمیم بگیرید در اسپرینت‌های دو هفته‌ای کار کنید، اما تعریف شما از "ماموریت انجام شده" ممکن است پایان دادن به یک ورژن بالاتر باشد که می‌خواهید با هم ارسال کنید، اما مطمئناً هر چه ارسال نرم‌افزار از لحاظ زمانی طول بکشد، ریسک از رده خارج شدن آن بیشتر می‌شود

**قوانین اسکرام (Rules)**: قوانین اسکرام چابک باید کاملاً متناسب با تیم باشد و بر اساس آنچه برای فرایندهای آن‌ها بهتر کار می‌کند، تنظیم شود. بهترین مریبان چابک (Agile coaches) به تیم‌ها می‌گویند که با رویدادهای اولیه اسکرام که در بالا ذکر شده شروع کنند و سپس بر اساس نیازهای منحصر به فرد تیم، همراه شده و پیشرفت کنند.

## چرخه زندگی اسکرام (Scrum Lifecycle) □

چرخه زندگی متدولوژی اسکرام با Backlog اولویت بندی شده شروع می شود، اما هیچ راهنمایی در مورد چگونگی توسعه یا اولویت بندی آن ارائه نمی دهد. در شکل زیر چرچه اسکرام نشان داده شده است



چرخه زندگی اسکرام از یک سری Sprint تشکیل شده است، که در آن نتیجه نهایی، یک محصول قابل انتشار است. در داخل این Sprints، تمام فعالیت های لازم برای توسعه محصول، در زیر مجموعه کوچکی از محصول کلی اتفاق می افند. در زیر شرح مراحل اصلی چرخه زندگی اسکرام آورده شده است:

۱. ابتدا Backlog محصول ایجاد می شود .
۲. صاحب محصول و تیم توسعه در نیمه ای اول محدوده Sprint را تعیین می کنند و در نیمه ای دوم برنامه ریزی Sprint و برنامه ای تحويل آن را مشخص خواهند کرد .
۳. با پیشرفت Sprint، تیم توسعه کارهای لازم را برای تحويل موارد Backlog محصول انتخاب شده، انجام می دهد .
۴. به طور روزانه، تیم توسعه کار خود را در Daily Scrum هماهنگ می کند .
۵. در پایان Sprint، تیم توسعه مواردی را که در طول برنامه ریزی Sprint انجام شده اند، ارائه می دهد. تیم توسعه برای نشان دادن پیشرفت کار به مشتری و دریافت بازخورد، یک Sprint Review برگزار می کنند. در بررسی اسپرینت تیم توسعه دهنده و صاحب محصول در مورد چگونگی پیشروی Sprint صحبت کرده و فرایند های آن را مطابق با گذشته بررسی می کنند .
۶. تیم توسعه مراحل ۲ تا ۵ را تکرار می کند تا نتیجه مطلوب برای محصول، برآورده شود.

## نحوه اجرای اسکرام در مدیریت پروژه □

یک اسکرام می‌تواند در مراحل زیر انجام شود

مرحله‌ی اول؛ تشکیل گروه اسکرام و مشخص نمودن اهداف و محصولات نهایی پروژه (increment)

مرحله‌ی دوم؛ تهیه سند بگ لاگ محصول

**بگ لاگ محصول:** لیست اصلی اقداماتی است که باید در یک پروژه انجام شود و توسط مالک محصول تهیه می‌شود. این لیست شامل ویژگی ها، نیازمندیها، پیشرفت‌ها و اصلاحاتی است که باید انجام شود و مالک محصول به طور مداوم آن را بازبینی و بروزرسانی می‌کند زیرا با گذشت زمان، جمع‌آوری اطلاعات بیشتر و یا تغییر بازار، امکان دارد بعضی از موارد لیست دیگر مورد نیاز نباشند یا موانع برطرف شده باشند.

مرحله‌ی سوم؛ فاز بندی: بررسی و مشخص نمودن فازهای پروژه و اسپرینت‌هایی که باید موازی با یکدیگر انجام شوند.

مرحله‌ی چهارم؛ برنامه‌ریزی اسپرینت (Sprint Planning) و تهیه سند اسپرینت (Sprint Backlog)

مرحله‌ی پنجم؛ اجرای اسپرینت

مرحله‌ی ششم؛ تحويل و بررسی اسپرینت (Sprint Review)

مرحله‌ی هفتم؛ بازنگری اسپرینت (Sprint Retrospective)

پس از پایان یافتن یک اسپرینت بلافارسله اسپرینت دیگری برنامه‌ریزی و شروع می‌شود تا این که گروه اسکرام به محصول نهایی خود دست یابد.

نموده ای از بگ لاگ محصول					
نکات	چگونگی دمو	برآورد	اهمیت	نام	ش
لاغین - باز کردن صفحه سپرده گذاری - به دیاگرام توالی UML نیاز است . نیازی به استفاده از encryption	لاغین - باز کردن صفحه سپرده گذاری - سپرده گذاری میغ 10 دلار - باز کردن صفحه موجودی که باید موجودی 10 دلار اضافه شده باشد.	5	30	سپرده گذاری	1
Paging	لاغین - کلیک بر روی تراکشن ها و منلپده تراکشن آخر - باز کردن صفحه سپرده گذاری و انجام سپرده گذاری - باید این سپرده گذاری در لیست تراکشن ها نشان داده شود. مشاهده کاربران طراحی شود.	8	10	مشاهده تراکشن های شما	2

## ۶-مستندات در مهندسی نرم افزار

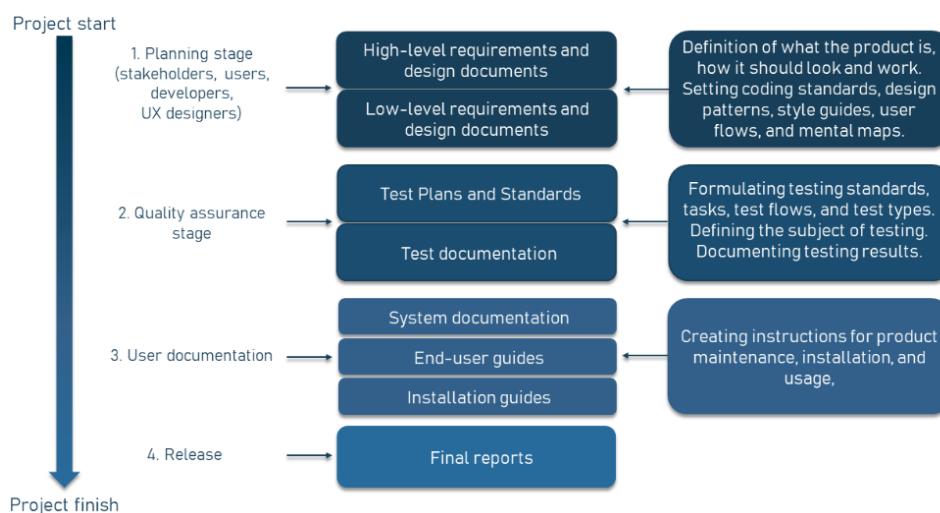
مستندسازی بخش بزرگی از روند تولید نرم افزار است خوب است یادآوری کنم که در شرکت‌های مطرح نرم افزاری دنیا، مستندسازی جزو پژوهشی ترین واحد هاست. روش کار به این صورت است که کارشناسان مستندسازی در جلسات متعدد و منظم با تیم توسعه، به تدریج اسناد لازم را که شامل مستندات تجربه کاربری، مستندات فنی و راهنمای کاربر است را تهیه، آرشیو و بروزرسانی می‌کنند. ما در ایران کمتر تولیدکننده‌ای می‌شناسیم که به طور جدی به این ردیف شغلی مهم توجه کند و شاید واقعاً علت عمر پایین محصولاتی که تولید می‌کنیم همین باشد.

▪ **تعريف مستند سازی** مستند سازی، تهیه مجموعه اسناد و مدارکی است که سیر تکوین و تحقق یک فعالیت از شروع تا خاتمه آن و چگونگی بهره برداری و نگهداری را، با تحلیل و ارزیابی مربوط، نشان می‌دهد.

▪ **اهمیت مستندسازی** در مدت اجرای پروژه وقایع و رویدادها، ابتکارها، خلافت‌ها، تنگناها و مشکلات فنی و حقوقی و اجرایی و مالی اتفاق می‌افتد و برای هریک راه حل‌ها و تدابیر اتخاذ می‌شود. چنانچه این تدابیر مستند شوند، کمک مؤثری به پرهیز از دوباره کاری و تجربه‌گیری در اجرای طرح‌ها و پروژه‌ها می‌شود.

▪ **مستند سازی یک پروژه باید به پرسش‌های زیر پاسخ دهد:**

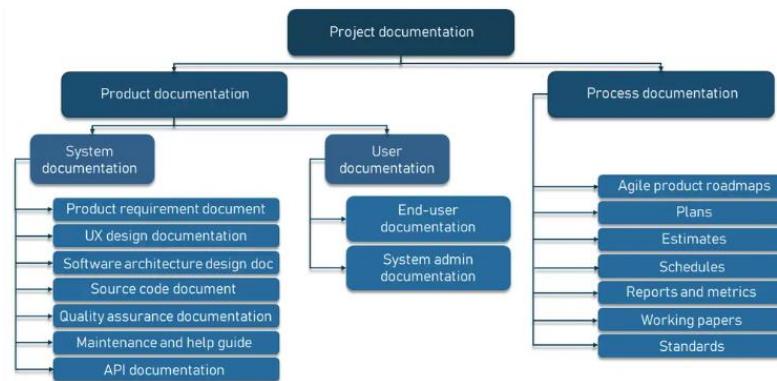
چه کسی؟، چرا؟، چه موضوعی؟، چه وقت؟ و چطور؟



<https://document360.com/blog/software-documentation-tools/>

## انواع مستندات توسعه نرم افزار

هدف اصلی مستندات مؤثر این هست که اطمینان حاصل کند ذینفعان و توسعه دهنده‌گان در جهت یابی اهداف پروژه، هم جهت باشند و در یک جهت هدایت شوند. برای دستیابی به این هدف، تعداد زیادی از انواع مستندات وجود دارد. نرم افزار را از دیدگاه کاربر نهایی، کسب و کار و ذینفعان برنامه ریزی می‌کنند نیازمندیها مشخص می‌کنند که نرم افزار باید چه کاری انجام دهد. تمامی مستندات نرم افزار را می‌توان به دو دسته بنده‌ی اصلی تقسیم کرد:



### ۱. مستندات محصول

مستندات محصول، محصولی را که در حال توسعه است توصیف می‌کند و دستورالعمل‌هایی را در مورد نحوه انجام وظایف مختلف با آن ارائه می‌دهد. به طور معمول، مستندات محصول شامل نیازمندیهای مشخصات فنی، منطق تجاری و دفترچه راهنمایها است. دو نوع اصلی مستندات محصول وجود دارد:

▪ **مستندات سیستم** اسنادی را نشان می‌دهد که خود سیستم و قسمت‌هایش را توصیف می‌کند. که شامل اسناد نیازمندیها، تصمیمات مربوط به طراحی، تشریح معماری، سورس کد برنامه و پرسش و پاسخ‌های متداول است.

▪ **مستندات کاربر** شامل کتابچه‌های راهنمایی شود که عمدتاً برای کاربران نهایی محصول وادمین‌های سیستم تهیه شده‌اند. مستندات کاربر شامل دستورالعمل‌ها، راهنمایی‌های کاربر، کتابچه راهنمایی عیب‌یابی، کتابچه‌های راهنمای نصب و مرجع است.

### ۲. مستندات فرآیند

تمامی اسنادی که در طول توسعه و نگهداری تولید می‌شوند را شامل می‌شود که به خوبی فرآیند توسعه را تشریح می‌کنند. مثال رایج برای اسناد مرتبط با فرآیند استاندارها، مستندات پروژه هستند مانند برنامه ریزی های پروژه، برنامه های آزمایشی، گزارشات، یاداشت های جلسات و یا حتی مکاتبات تجاری می‌باشد. تفاوت اصلی بین مستندات فرآیند و محصول این هست که اولی فرایند توسعه را ذخیره می‌کند و دومی محصولی که در حال توسعه است را تشریح می‌کند.

▪ **مستندات سیستم**: مستندات سیستم یک دیدگلی از سیستم می‌دهد و به مهندسین و ذینفعان در فهم و درک تکنولوژی‌های اساسی کمک می‌کند. معمولاً شامل اسناد نیازمندیها، طراحی معماری، سورس کد، اسناد اعتبارسنجی، اطلاعات تست و تایید، و راهنمای نگهداری می‌شود.

## ▪ سند نیازمندیهای محصول<sup>۲</sup> PRD

سند نیازمندیهای محصول اطلاعاتی در مورد عملکرد سیستم فراهم می‌کند. به طور معمول، نیازمندیها عبارت است از آنچه که یک سیستم باید انجام دهد. که شامل قوانین بیزینس، سناریوهای کاربر، موارد استفاده (Use Cases) و غیره می‌شود. این سند باید واضح و روشن باشد و به اندازه کافی برای توصیف محصول شامل: ویژگی‌های آن، عملکردهای آن، نگهداری و رفتار آن باشد. بهترین کار نوشتن یک سند نیازمندیها با استفاده از یک الگوی منسجم می‌باشد که تمامی اعضای تیم آن را رعایت کنند.

برخی از اصطلاحات رایج مورد استفاده برای سند مشخصات به شرح زیر است:

- سند نیازمندیهای تجاری BRD<sup>۳</sup>
- سند نیازمندیهای سیستم SRD
- مشخصات نیازمندیهای عملکردی FRS
- مشخصات نیازمندیهای سیستم SRS<sup>۴</sup>
- مشخصات نیازمندیهای پیکربندی CRS
- مشخصات نیازمندیهای محصول PRS
- مشخصات نیازمندیهای قابلیت اطمینان RRS
- مشخصات نیازمندیهای سازگاری CRS

- **سند BRS**: برای ایجاد سند BRS، تحلیلگر تجاری با مشتریان در ارتباط است. سند BRS شامل قوانین کسب و کار، محدوده پروژه و نیازمندیهای دقیق مشتری است. در این نوع سند، مشتری نحوه عملکرد کسب و کار خود یا نرم افزار مورد نیاز خود را شرح می‌دهد.
- **سند SRS**: در این سند، تحلیلگر کسب و کار مشخصات مورد نیاز مشتری را از مشتری جمع آوری کرده و آنها را به مشخصات نیازمندیهای نرم افزار SRS ترجمه می‌کند. SRS شامل نحوه توسعه نرم افزار و ارائه آن توسط تحلیلگر کسب و کار است. به عبارت دیگر، می‌توان گفت که سند SRS برای پنهان کردن اطلاعات مشتری در یک سند دقیق استفاده می‌شود که توسعه دهنده‌گان و مهندسان آزمایش می‌توانند آن را به راحتی درک کنند.

برخی از ویژگی‌های مهم سند SRS به شرح زیر است:

- سند SRS برای تعیین هزینه اولیه محصول نرم افزاری استفاده می‌شود.
- پیوند دادن شکاف بین توسعه دهنده و کاربر مفید است.
- مشخصات مورد نیاز نرم افزار به عنوان یک توافق بین طرف‌های در ارتباط عمل می‌کند.

<sup>2</sup> Product requirement document <sup>6</sup>

<sup>2</sup> Business Requirement Specification

<sup>2</sup> Software Requirement Specification<sup>8</sup>

## تفاوت اصلی بین اسناد BRS و SRS

- SRS به عنوان مشخصات مورد نیاز سیستم نشان داده می شود در حالی که BRS به عنوان مشخصات مورد نیاز تجاری نشان داده می شود.
- نیازهای کاربردی و غیر کاربردی نرم افزار را تعریف می کند. از سوی دیگر، BRS یک سند رسمی است که نیازهای ارائه شده توسط مشتری را مشخص می کند.
- سند SRS توسط معمارسیستم توسعه یافته است. از سوی دیگر، BRS توسط تحلیلگر کسب و کارت توسعه یافته است.
- از BRS به دست می آید، در حالی که BRS از اظهارات مشتریان و نیازهای تجاری آنها به دست می آید.
- **سند سورس کد**:<sup>2</sup> یک سند سورس کد بخشی فنی است که تشریح می کند کدها چگونه کار می کنند. کاربران اصلی اسناد سورس کد مهندسین نرم افزار یا برنامه نویسان دیگر هستند. اسناد سورس کد ممکن است شامل موارد زیر باشد
  - چارچوب تولید، زبان و دیگر فریمور که های استفاده شده
  - نوع اتصال داده ها (data binding)
  - الگوهای طراحی مانند (model-view-controller)
  - معیارهای امنیت
  - دیگر الگوها و اصول