



Beginning Git and GitHub

راهنمای جامع کنترل نسخه
مدیریت پروژه و کار گروهی
به صورت تصویری

خرداد
1399

عنوان : راهنما جامع کنترل نسخه و مدیریت پروژه به صورت تصویری (Beginning Git and GitHub)

ترجمه و تالیف: علی عوض زاده

پذیرنده: www.toplearn.com

نوبت چاپ : خرداد 1399

تعداد صفحات: 297 صفحه

تایپ و صفحه آرایی: علی عوض زاده

طرح روی جلد : علی عوض زاده

هر گونه کپی برداری از این کتاب شرعا و قانونا حرام است

شما اجازه ندارید:

مطالب این کتاب را به اسم خود کپی و منتشر کنید.

از این کتاب استفاده تجاری کنید.

در محتوای این کتاب دست ببرید.

در صفحات این کتاب آرم یا لوگو خود را قرار دهید.

با سپاس : علی عوض زاده

فهرست مطالب

قسمت اول: کنترل نسخه با Git

پیشگفتار

معرفی کنترل نسخه

مقدمه

تعریف تعریف نسخه کنترل

برنامه های کنترل نسخه

اصول اخلاقی مشترک

WORKFLOW شعبه ویژگی ها

FORKING WORKFLOW

NAVIGATING GIT HUB

تنظیم یک حساب حساب کاربری سنگین

بهره برداری از دو عامل

احراز هویت

سازمان ها

تنظیم یک سازمان

تنظیم یک تیم

MARKETPLACE

تنظیم کردن کد بالا

حساب ها

RUNTIME CONFIG

تنظیم اعتبار کاربر

حذف پیکربندی

پی‌کربندی SSH

تنظیم کردن SSH

ایجاد یک نمایندگی

ایجاد یک نمایندگی در A

محیط محلی

ایجاد یک نمایندگی روشن است

GIT HUB

NAVIGATING REPOSITORY

همکاران

اضافه کردن و پخش کردن

مشارکت کنندگان

واحد های ناشناخته ، کمیته ها ، و

بینش (CONTRIBUTORS ، PULSE ، Forks)

GIT HUB ETIQUETTE

نام های نماینده ، برچسب ها ، و توضیحات

اضافه کردن لیسانس

ویکیس و موارد

فعالیت 1: ایجاد یک حساب کاربری

خلاصه

فصل 2: سیستم های کنترل نسخه

کنترل نسخه چیست؟

چرا به یکی احتیاج داری؟

گزینه ها چیست؟

سیستم های کنترل نسخه محلی

سیستم های کنترل نسخه متمرکز

سیستم های کنترل نسخه توزیع شده

Git چیست؟

Git چه کاری می تواند انجام دهد؟

Git چگونه کار می کند؟

گردش کار معمولی Git چیست؟

خلاصه

فصل 3: نصب و راه اندازی

نصب و راه اندازی

windows

مک

لینوکس

راه اندازی Git

خلاصه

فصل 4: شروع کار

مخازن

دایرکتوری کار

منطقه صحنه

متعهد می شود

شروع سریع با Git

خلاصه

فصل 5: غواصی به گیت

نادیده گرفتن پرونده ها

بررسی سیاههها و سابقه

مشاهده نسخه های قبلی

بررسی تغییرات فعلی

خلاصه

فصل 6: commits

سه ایالت گیت

پیمایش بین نسخه ها

خنثی کردن تعهد

Modifying a commit

Amending a commit

خلاصه

فصل 7: بهترین روشها

پیام های متعهد

Git بهترین اقدامات را مرتکب شوید

چه کاری انجام دهیم

چه کار نکنیم

Git چگونه کار می کند (دوباره)

خلاصه

فصل 8: Git از راه دور

چرا روی ریموت کار می کنید

چگونه کار می کند

راه آسان

خلاصه

قسمت دوم: مدیریت پروژه با GitHub

فصل 9: آغازگر GitHub

بررسی اجمالی GitHub

GitHub و منبع آزاد

استفاده ی شخصی

GitHub برای مشاغل

خلاصه

فصل 10: شروع سریع با GitHub

مدیریت پروژه

چگونه مخازن از راه دور کار می کنند

پیوند مخازن

هل دادن به مخازن از راه دور

خلاصه

فصل 11: شروع مدیریت پروژه: مسائل

مروری بر موضوعات

ایجاد شماره

تعامل با یک مسئله

برچسب ها

مأمورین

پیوند دادن مسائل با تعهدات

کار روی تعهد

ارجاع یک شماره

بسته شدن یک مسئله با استفاده از کلمات کلیدی

خلاصه

فصل 12: شیرجه رفتن به مدیریت پروژه: شعب

گردش کار GitHub

شاخه ها

ایجاد شعبه

انتقال به شاخه دیگری

حذف یک شعبه

شاخه های ادغام

فشار دادن شاخه به ریموت

خلاصه

فصل 13: مدیریت بهتر پروژه: درخواست ها را بکشید

چرا از درخواستهای Pull استفاده می کنیم؟

بررسی اجمالی در مورد بکشید

کشیدن

PR چه می کند

یک درخواست Pull ایجاد کنید

بررسی کد

یک بررسی کد ارائه دهید

نظر بدهید

یک درخواست را به روز کنید

خلاصه

قسمت سوم: کار تیمی با Git

فصل 14: درگیری

چگونه یک ادغام کار می کند

در حال کشیدن

ادغام سریع به جلو

ادغام اختلافات

برداشت از مبدأ

حل اختلافات ادغام

خلاصه

فصل 15: اطلاعات بیشتر درباره درگیری ها

هل دادن پس از حل اختلاف

تغییرات را قبل از ادغام بررسی کنید

محل شعبه را بررسی کنید

شعبه بررسی

ادغام را درک کنید

کاهش درگیری

داشتن یک گردش کار خوب

سقط یک ادغام

با استفاده از یک ابزار Git visual

خلاصه

فصل 16: ابزار GUI Git

ابزارهای پیش فرض

تعهد git-gui :

مرور gitk :

ابزارهای IDE

کد ویژوال استودیو

اتم

ابزارهای تخصصی

دسک تاب GitHub

GitKraken

خلاصه

فصل هفدهم Git Advanced

بازگرداندن

Stashing

تنظیم مجدد

خلاصه

قسمت چهارم: منابع اضافی

فصل هجدهم: بیشتر با GitHub

ویکی

صفحات GitHub

منتشر شده

تابلوهای پروژه

خلاصه

فصل نوزدهم: مشکلات گیت مشترک

مخزن

شروع به کار

تغییر مبدأ

دایرکتوری کار

Git diff خالی است

تغییر در یک پرونده را لغو کنید

متعهد می شود

خطا در ارتکاب

واگرد متعهد شد

شاخه ها

جدا شده HEAD

روی شاخه اشتباه کار کرد

با شعبه والدین همگام شوید

شعب گوناگون شده اند

خلاصه

فصل بیستم: گردش کاری Git و GitHub

نحوه استفاده از این گردش کار

گردش کار GitHub

هر پروژه با یک پروژه شروع می شود

هر عملی با یک شماره شروع می شود

هیچ فشار مستقیم برای استاد نیست

توجه ، توجه» این فصل فقط برای آشنایی اولیه دانشجوی می باشد ، نگران جزئیات ان نباشید به مرور در کل کتاب با تمام قسمت ها آشنا خواهید شد.

معرفی کنترل نسخه

اهداف یادگیری

در پایان این فصل ، شما می توانید:

کنترل نسخه و انواع مختلف گردش کار را تعریف کنید

رابط کاربری GitHub را توضیح دهید

توابع مختلف GitHub مانند تیم ها و SSH را تنظیم کنید

با استفاده از قوانین و مقررات GitHub یک مخزن ایجاد کنید

در این فصل شرح کامل کنترل نسخه ، جریان کار و راه اندازی مخزن محلی و GitHub توضیح داده شده است.

مقدمه

در این فصل مفاهیم اساسی کنترل نسخه با استفاده از مثال های مصور ارائه می شود. این کتاب به جنبه هایی که کنترل نسخه را تشکیل می دهند توجه دارد که شامل ردیابی تغییرات codebase ، نحوه رمزگذاری codebase برای پشتیبانی از دسترسی از راه دور ، مدیریت مشارکت کنندگان و مشارکت ها و بهترین شیوه های پیروی از آن است. با اجرای کنترل نسخه ، ایجاد و اجرای چک و کنترل تیم محور برای بررسی دقیق ، تأیید ، ادغام ، و وارونگی تغییرات ، در صورت تحقق موارد ضروری امکان پذیر می شود.

کنترل نسخه به ردیابی و قابلیت ردیابی تغییرات اشاره می کند. به نوعی شبیه به استفاده از نشانک است که وقتی خواننده بخواهد به خواندن خود ادامه دهد ، نقطه بازگشت را به آن برگرداند. در کنترل نسخه ، این نشانک استعاری اشاره ای به عکس فوری از پایه کد را نشان می دهد. این عکس فوری وضعیت محصول یا پایه کد را در یک نقطه معین نشان می دهد

در صورت توسعه بدون استفاده از کنترل نسخه ، استفاده از کد و ایجاد تغییر در همان کد به یک محیط پر هرج و مرج تبدیل می شود که در آن تغییرات در یک پایگاه کد هماهنگ نیستند. هیچ اطلاعاتی برای ردیابی تغییرات در یک پرونده ، جدا از ابر داده ای که سیستم عامل شما از آن استفاده می کنید ، وجود ندارد و یک روش بایگانی سازی که در آن از نامگذاری پرونده برای اشاره به عکس های مختلف یک پایگاه رمز استفاده می کند.

نتیجه این تغییرات تغییر یافته و تأخیری است. این امر به این دلیل است که نیاز به تیم توسعه به طور مداوم پرونده های فیزیکی را برای بررسی موفقیت آمیز تغییرات و همچنین منابع صرف شده در اصلاح ادغام های نادرست که منجر به اشکالات ناخواسته و پیش بینی نشده در محیط تولید می شود ، دارد.

تعریف کنترل نسخه

کنترل نسخه با هدف پشتیبانی از ردیابی تغییرات در یک پرونده ، یعنی برگشت معکوس تغییرات ایجاد شده در یک پرونده و حاشیه نویسی از تغییرات معرفی شده در یک کدبندی انجام می شود. قبل از شروع نرم افزار کنترل نسخه ، کنترل نسخه خود رویکردی را اتخاذ می کرد که توسط تیمی از برنامه نویسان که روی یک بانک اطلاعات کار می کنند مورد توافق قرار گرفت.

برای معرفی و اجرای تغییر در یک محصول ، به عنوان مثال یک توسعه دهنده می تواند نسخه بایگانی شده متناسب با نسخه را در محیط تولید بازبازی کند. آنها برای ایجاد و آزمایش تغییر عمل می کنند. برای به کارگیری نسخه جدید ، یک نسخه به نسخه اختصاص داده می شود و یادداشت هایی که جزئیات آن را درج می کند در کنار نسخه حاشیه نویسی می شود. برای برگرداندن تغییرات ، از شما خواسته می شود موارد زیر را انجام دهید:

یادداشت های انتشار را با پرونده های خاص مطابقت دهید.

تغییرات واقعی معرفی شده در پرونده های مربوطه را تعیین کنید.

تغییرات را برگردانید و نسخه اصلاح شده را مستقر کنید.

برای دستیابی به انعطاف پذیری مؤثر در توسعه محصول ، محصولات توسعه دهندگان نرم افزار نیاز به هماهنگی همزمان کار هادارند.

تیمی را در نظر بگیرید که در یک سکوی فروش بلیط اتوبوس برای شهر تونس کار می کند تا بتواند یک برنامه موبایل را برای فروش بلیط تهیه کند. در این پروژه ، توسعه دهندگان ممکن است کار را به دسته های زیر تقسیم کنند:

تأیید اعتبار کاربر

خرید بلیط

این کار را می توان به اعضای مختلف تیم توسعه اختصاص داد. هر عضو می تواند تلاش های خود را بر روی یک کار متمرکز کند و کار را از طریق یک مخزن مرکزی در GitHub به اشتراک بگذارد. هر ویژگی را می توان قبل از آزمایش و ادغام در محیط

تولید برای استفاده ساکنان تونس به صورت تدریجی در بیت ها چرخاند. نگهبان اسناد برای همکاری در اسناد مشترک از کنترل نسخه استفاده می کنند. یک نگهدارنده اسناد و مدارک امکان بررسی تغییرات پیشنهادی از طرف ذینفعان ذی ربط را فراهم می کند و پس از آن نسخه نهایی یک سند برای استفاده توسط یک سازمان منتشر می شود. اسناد نرم افزار ممکن است توسط پرسنل مسئول مدیریت منابع اطلاعاتی یک سازمان با استفاده از کنترل نسخه حفظ شود. به عنوان مثال ، یک مخزن ممکن است برای بایگانی اسناد استفاده شود که دیگر قابل استفاده نیستند.

این روند در صورت عدم وجود اسناد و مدارک مانند یادداشت های منتشر شده از یک پروژه چالش برانگیزتر خواهد بود. همانطور که احتمالاً تاکنون تأسیس کرده اید و این فرایندی است که با ناامیدی ، استرس و ناکارآمدی همراه است.

ظهور نرم افزار کنترل نسخه با نیاز به پرداختن به مواردی که باعث اختلاط تغییر و انتشار نرم افزار شده بود ، تحریک شد. نرم افزار کنترل نسخه شاهد تکامل سه نسل بوده است.

در نسل اول ، نرم افزار کنترل نسخه از یک مکانیزم قفل کردن بر روی فایل ها استفاده می کرد تا امکان تغییر در پرونده را فراهم کند. یک پرونده فقط در یک زمان معین می تواند توسط یک نفر کار شود. قفل قرار داده شده بر روی پرونده هنگامی که شخصی که در حال کار بر روی پرونده مذکور بود ، برداشته می شد ، برداشته شود. مدیریت تغییر با مدیریت سابقه در هر پرونده انجام شد. در این دوره ، (SCCS (Source Source Control System و (RCS (Revision Control System نرم افزار کنترل نسخه رایج در حال استفاده بودند.

نسل دوم با استفاده از ادغام قبل از ارتکاب مکانیزم برای پشتیبانی از ویرایش همزمان یک فایل توسط چندین کاربر مشخص شد. برای ادغام تغییرات در یک پرونده ، از شما خواسته می شود تغییرات ایجاد شده توسط دیگران را در همان پرونده ادغام کنید. پس از اتمام ، شما اقدام به ارتقاء تغییر پرونده ، پرونده خود خواهید کرد. این نسل از نرم افزارها استفاده از مخازن متمرکز را معرفی کردند. توسعه دهندگان از یک مخزن مشترک می توانستند از راه دور به پایه کد دسترسی پیدا کنند و با سایر توسعه دهندگان همکاری کنند. علاوه بر این ، واحد تغییر به عنوان تغییر در مجموعه ای از پرونده ها به جای یک پرونده واحد ، ردیابی شد.

نسل سوم در طبیعت غیر متمرکز است. هر توسعه دهنده یک نسخه از مخزن را بدست می آورد. تغییرات در مخزن از راه دور از طریق ادغام معرفی می شوند. برای اینکه چندین نفر بتوانند روی همان پرونده کار کنند ، قبل از استفاده از مکانیسم ادغام ، متعهد شوند.

در اینجا ، تغییراتی را در مخزن محلی ایجاد می کنید. برای ترکیب کردن تغییرات ایجاد شده در مخزن راه دور ، شما تغییرات محلی را مرتکب می شوید ، پس از آن می توانید تغییرات ایجاد شده توسط افراد دیگر را ادغام کنید. این در بخش های بعدی نشان داده می شود. Git یک ابزار کنترل نسخه نسل سوم است.

کنترل نسخه ، بخشی جدایی ناپذیر از کار ، یعنی مدیریت تغییر را فراهم می کند. Git و GitHub ، همانطور که در این کتاب مشاهده می کنید ، ابزاری ارائه می دهند که به تیم ها و افراد امکان می دهد تا تغییر در کتاب کار را به روشی سریع و مؤثر انجام دهند. این امر از طریق امکانات تقسیم کار و ادغام تغییر ارائه شده توسط Git و GitHub حاصل می شود.

برنامه های کنترل نسخه

کنترل نسخه برای هر دو بخش فنی و غیر فنی کاربرد دارد. این شرکت به پروژه های توسعه نرم افزار و همچنین به پروژه هایی که تیم ها را برای همکاری در تهیه و استفاده از اسناد نیاز دارند ، وام می دهد. مثال خوبی که باید در نظر بگیرید این است که نقشه های ساخت و ساز abuilding توسط تیمی از معماران به اشتراک گذاشته می شود و به طور مشترک مورد نقض قرار می گیرد. چنین پروژه ای می تواند با استفاده از Git و GitHub برای به اشتراک گذاشتن اسناد و برنامه ها مدیریت شود.

بیایید نگاهی به برخی از اصطلاحات متداول که در این کتاب با آنها می پردازیم بیاندازیم:

مخزن (Repository)

واحدی از ذخیره سازی و ردیابی تغییر که نماینده دایرکتوری است که محتویات آن توسط Git ردیابی می شود.

شاخه (Branch)

نسخه ای از مخزن که بیانگر وضعیت فعلی مجموعه پرونده هایی است که یک مخزن را تشکیل می دهند. در یک مخزن ، یک شاخه پیش فرض یا اصلی وجود دارد که منبع حقیقت واحد را نشان می دهد.

استاد (master)

شعبه پیش فرض یا اصلی. نسخه ای از مخزن که منبع واحد حقیقت به حساب می آید.

برای استفاده از قیاس یک رودخانه ، استاد اصلی رودخانه است. شاخه های دیگر دقیقاً مانند یک توزیع کننده از جریان اصلی حرکت می کنند ، اما به جای بازگشت به جریان اصلی ، شاخه ها دوباره مانند جریان یک شاخه بازگانی به جریان اصلی وصل می شوند. از این روند پیوستن به جریان اصلی به عنوان ادغام یاد می شود.

Reference

مرجع یا مرجع Git نامی است که مربوط به یک هش متعهد است. منابع در یک پرونده در فهرست git / refs یک مخزن ذخیره می شوند.

Head

ارجاع به تعهد اخیر در شعبه. جدیدترین تعهد معمولاً به عنوان نوک شعبه گفته می شود.

Working Tree

این مربوط به بخشی است که در آن پرونده ها را در یک شاخه مشاهده و تغییر می دهیم. پرونده هایی که تغییر کرده اند ، پس از آماده شدن برای ارتکاب ، به یک منطقه مرحله بندی منتقل می شوند.

Index

این ناحیه ای است که Git پرونده هایی را تغییر داده ، اضافه کرده و یا آمادگی لازم را برای مرتکب شدن داده است. این منطقه صحنه ای است که از آن جا تغییرات ایجاد می کنید.

Commit

این یک ورود به تاریخ Git است که نشانگر تغییر و تحول در مجموعه ای از پرونده ها در یک زمان معین است. **Commit** پرونده هایی را که به فهرست اضافه شده اند ارجاع می دهد و HEAD را به روز می کند تا به وضعیت جدید شعبه اشاره کند.

Merge

با استفاده از قیاس یک رودخانه ، ادغام به فرآیندی اطلاق می شود که از طریق آن یک بوته به رودخانه اصلی می پیوندد. در گیت ، ادغام فرآیند ترکیب تغییرات از یک شاخه به شاخه دیگر است.

Workflows

گردش کار به رویکردی است که یک تیم برای معرفی تغییرات در یک پایگاه داده استفاده می کند. گردش کار با یک روش مشخص در استفاده از شعب (یا فقدان آن) برای معرفی تغییرات در یک مخزن مشخص می شود.

گردش کار Gitflow

با استفاده از دو شاخه **master and develop** . از شاخه استاد برای ردیابی تاریخ انتشار استفاده می شود ، در حالی که از شاخه توسعه برای ردیابی ویژگی های یکپارچه شده در محصول استفاده می شود.

Centralized workflow

این روش از شاخه استاد به عنوان شاخه توسعه پیش فرض استفاده می کند. تغییرات مربوط به شعبه استاد است. این یک گردش کار مناسب برای تیم ها و تیم هایی با اندازه کوچک است که از Apache Subversion در حال گذر هستند. در Apache Subversion ، تنه معادل شاخه اصلی است.

FEATURE BRANCH WORKFLOW

در این گردش کار ، توسعه ویژگی ها در یک شعبه اختصاصی انجام می شود. پس از تصویب تغییرات مورد نظر ، شاخه سپس به master ادغام می شود.

FORKING WORKFLOW

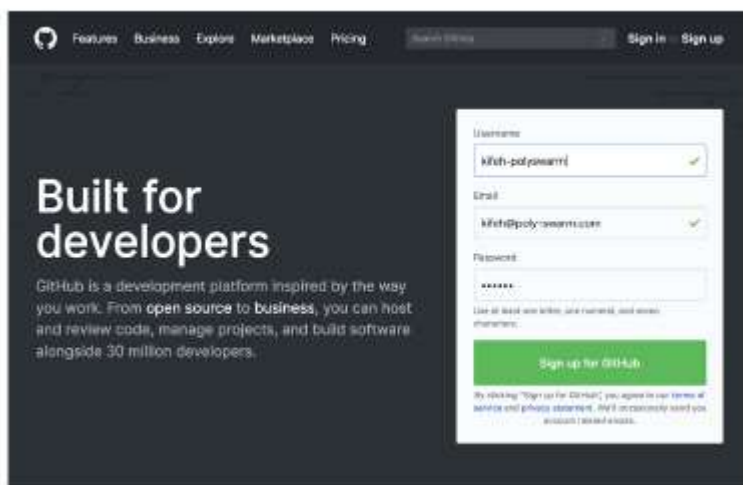
در این روش ، فردی که به دنبال ایجاد تغییر در مخزن است ، یک کپی از مخزن مورد نظر را در حساب GitHub مربوطه خود ایجاد می کند. تغییرات در یک کپی از مخزن منبع ایجاد شده و از طریق درخواست کشش به مخزن منبع ادغام می شوند.

کنترل نسخه با Git یک ماهیت توزیع شده دارد. این کد بر روی هر رایانه محلی که پایه کد در آن کار می شود ، و همچنین در یک نقطه مرکزی از راه دور قرار دارد که در آن هر شخصی که مایل به کار بر روی پایه کد است می تواند آن را بدست آورد. GitHub یک نقطه از راه دور مرکزی است. GitHub میزبان مخازن است و کاربران را قادر می سازد از طریق Git تغییرات یک پایگاه کد را بدست آورند ، تغییر دهند و ادغام کنند:

تنظیم یک حساب Github

برای درک نحوه عملکرد GitHub به عنوان یک ابزار میزبانی ، اکنون می خواهیم ویژگی های ارائه شده را با کاوش در رابط کاربری بررسی کنیم:

1. مطابق شکل زیر جزئیات کاربری خود را وارد کنید ، دکمه را مطابق تصویر ر فشار داده و ثبت نام را برای GitHub انجام دهید. توجه داشته باشید از طرف GitHub برای تأیید آدرس ایمیلی که هنگام ثبت نام استفاده کرده اید از شما خواسته می شود ، لطفاً برای راهنمایی ایمیل خود را بررسی کنید.



مانند تصاویر زیر عمل کنید

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Welcome to GitHub

You've taken your first step into a larger world, @klifeh-polyswarm.



Completed
Set up a personal account



Step 2:
Choose your plan



Step 3:
Tailor your experience

Choose your personal plan



Unlimited public repositories for free.



Unlimited private repositories for \$7/month. [\(view in KES\)](#)

Don't worry, you can cancel or upgrade at any time.

☐ Help me set up an organization next

Organizations are separate from personal accounts and are best suited for businesses who need to manage permissions for many employees.

[Learn more about organizations](#)

☐ Send me updates on GitHub news, offers, and events

Unsubscribe anytime in your email preferences. [Learn more](#)

Continue

Both plans include:

- ✓ Collaborative code review
- ✓ Issue tracking
- ✓ Open source community
- ✓ Unlimited public repositories
- ✓ Join any organization

You'll find endless opportunities to learn, code, and create, @kifeh-polyswarm.

✓ Completed
Set up a personal account

🔧 Step 2:
Choose your plan

⚙️ Step 3:
Tailor your experience

How would you describe your level of programming experience?

☐ Totally new to programming ☐ Somewhat experienced ☐ Very experienced

What do you plan to use GitHub for? (check all that apply)

☐ Project Management ☐ Development ☐ Research
☐ Design ☐ School projects ☐ Other (please specify)

Which is closest to how you would describe yourself?

☐ I'm a student ☐ I'm a professional ☐ I'm a hobbyist
☐ Other (please specify)


What are you interested in?

e.g. tutorials, android, ruby, web-development, machine-learning, open-source




Submit

skip this step

دکمه submit را بزنید با ثبت نام حساب کاربری ، باید صفحه را به شرح زیر مشاهده کنید:

 Search or jump to...

[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)


  

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and open a pull request.

Read the guide

Start a project

 Our new Terms of Service and Privacy Statement are in effect.

Repositories

New repository

You don't have any repositories yet!

Browse activity

Discover repositories

Discover interesting projects and people to populate your personal news feed.




Your news feed helps you keep up with recent activity on repositories you [watch](#) and people you [follow](#).

Explore GitHub

برای پیکربندی بیشتر حساب خود ، لطفاً سمت راست را انتخاب کنید، دکمه کشویی و گزینه تنظیمات را انتخاب کنید.
تنظیمات حساب باید مطابق تصویر زیر نمایش داده شود:



[Pull requests](#) [Issues](#) [Marketplace](#) [Explore](#)

Signed in as
kifeh-polyswarm

[Your profile](#)

[Your repositories](#)

[Your stars](#)

[Your gists](#)

[Help](#)



[Settings](#)

[Sign out](#)

Learn Git and GitHub without any code!

Using the Hello World guide, you'll create a repository, start a branch, write comments, and make a pull request.

[Read the guide](#) [Start a project](#)

 Our new Terms of Service and Privacy Statement are in effect. 

[Repositories](#) [New repository](#)

You don't have any repositories yet!

Browse activity

[Discover repositories](#)

Discover interesting projects and people to populate your personal news feed.

Your news feed helps you keep up with recent activity on repositories you [watch](#) and people you [follow](#).

[Explore GitHub](#)

TopLeads

Personal settings

- Profile
- Account
- Emails
- Notifications
- Billing
- SSH and GPG keys
- Security
- Blocked users
- Repositories
- Organizations
- Saved replies
- Applications
- Developer settings

Public profile

Name

Public email

Select a verified email to display

You have set your email address to private. To toggle email privacy, go to email settings and uncheck "Keep my email address private."

Bio

Tell us a little bit about yourself!

You can mention other users and organizations to link to them.

URL

Company

You can mention your company's GitHub organization to link it.

Location

All of the fields on this page are optional and can be deleted at any time, and by filling them out, you're giving us consent to share this data whenever your user profile appears. Please see our [privacy statement](#) to learn more about how we use this information.

Update profile

Contributions

☐ **Include private contributions on my profile**

Get credit for all your work by showing the number of contributors to private repositories on your profile without any repository or organization information. [Learn how we count contributions.](#)

Update contributions

نتیجه

شما با موفقیت یک حساب GitHub را به عنوان ابزار میزبانی تنظیم کرده اید. اکنون می خواهیم با استفاده از منوی تنظیمات ، دو عامل را فعال کنیم ، احراز هویت و ایجاد سازمان.

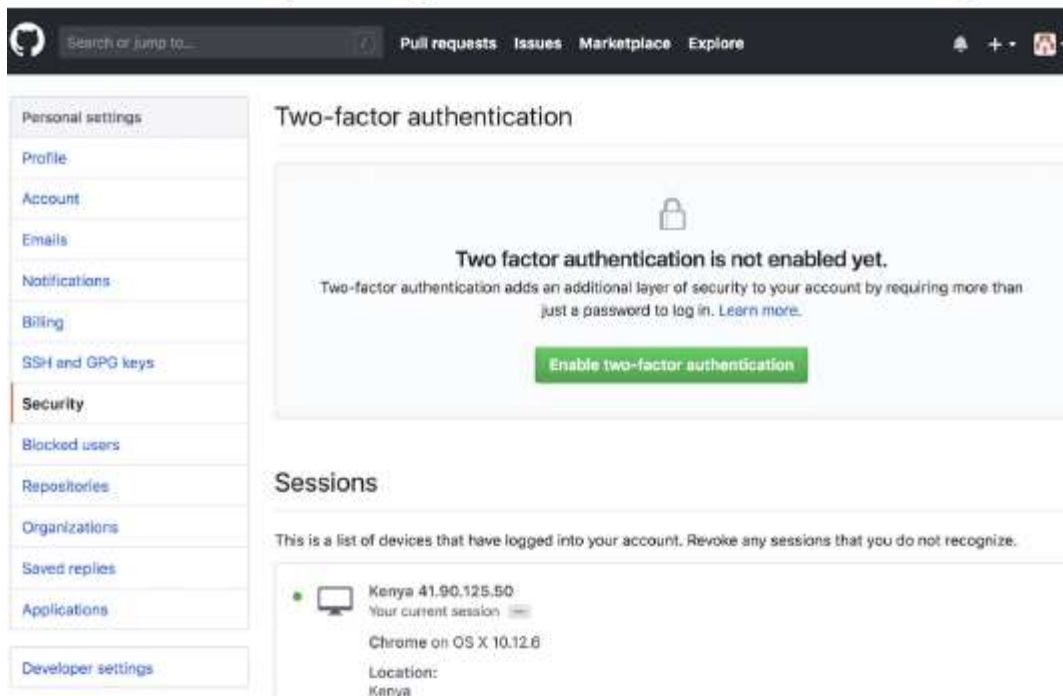
بهره برداری از اعتبار دو فاکتور

باید Google Authenticator را بر روی تلفن خود نصب کنید. برنامه در Android و iOS موجود است.

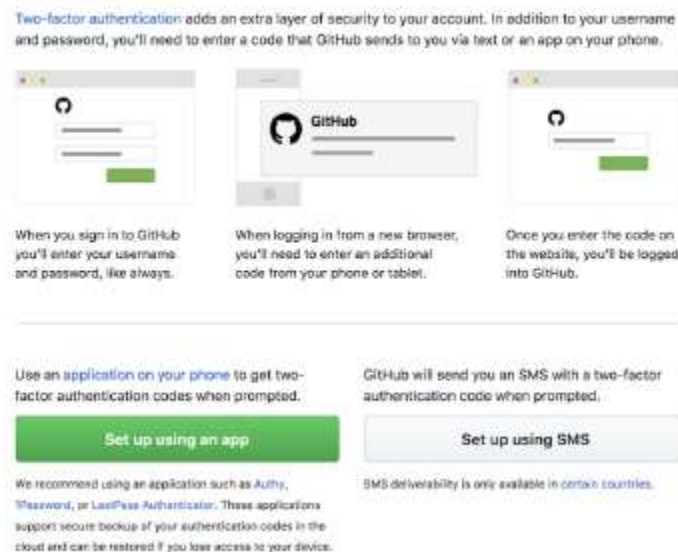
برای فعال کردن تأیید هویت دو عاملی در حساب به منظور افزایش امنیت ، این مراحل را دنبال کنید:

1. به تنظیمات بروید و Security را انتخاب کنید.

2. روی دکمه فعال کردن تأیید اعتبار دو عاملی کلیک کنید و رمز عبور خود را در فوریت بعدی وارد کنید:



روشی را که می خواهید استفاده کنید برای تنظیم تأیید هویت دو عاملی انتخاب کنید.
ما باید از یک برنامه برای این تنظیمات استفاده کنیم گزینه پیامک توسط همه مناطق پشتیبانی نمی شود.
تنظیم را کلیک کنید 😞 دکمه سبز



5- کدهای بازیابی را بارگیری کرده و آنها را در مکان دلخواه خود ذخیره کنید.

6. برای ادامه به مرحله بعدی ، Next را فشار دهید.

7. در تلفن خود ، تنظیم یک حساب را انتخاب کنید.

8- اسکن بارکدی را برای اسکن کد QR ارائه شده در مرورگر خود ، مطابق شکل زیر ، انتخاب کنید

تصویر صفحه:



کد شش رقمی نشان داده شده در برنامه را در قسمت متن زیر برچسب قرار داده شده وارد کنید و کد شش رقمی را از برنامه وارد کنید و Next را بزنید:

با احراز هویت دو عاملی که برای حساب کاربری خود پیکربندی شده است ، باید بتوانید با استفاده از گذرواژه و کدی که توسط برنامه ارائه شده است ، وارد سیستم شوید. می توانید با استفاده از گزینه شماره پیام کوتاه برای تماس با ما ، حساب خود را برای استفاده از یک سیم کارت پیکربندی شده تنظیم کنید. علاوه بر این ، می توانید برای استفاده از پیام کوتاه به عنوان روش ورود پیش فرض برای دریافت کد احراز هویت به جای برنامه ، حساب خود را تغییر دهید. با استفاده از گزینه های Delivery می توان به این نتیجه رسید.

ORGANIZATIONS

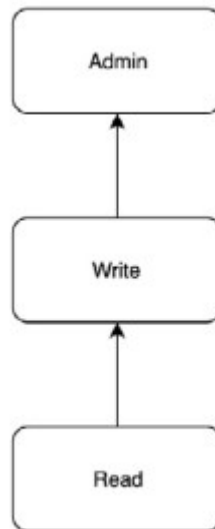
GitHub توانایی مدیریت چندین پروژه و با استفاده از یک حساب مشترک را که به عنوان یک سازمان خوانده می شود ، از طریق مخازن پسوند ارائه می دهد. با استفاده از یک سازمان ، می توانید مشارکت کنندگان را در یک پروژه ترتیب دهید تا ساختار سازمان خود را منعکس کند. این ساختار با تیم هایی که در پروژه های مربوطه کار می کنند و همچنین حق دسترسی اختصاص یافته به همکاران شخصی در هر تیم مطابقت دارد.

سازمان ها هماهنگی یکپارچه کار را از طریق ویژگی های زیر که از GitHub بهره می برد ، تشویق می کنند:

1. عضویت مبتنی بر نقش.

سه نقش در عضویت مبتنی بر نقش وجود دارد ، یعنی مالک ، مدیر صورتحساب و عضو. هر حساب شخصی که به سازمان اضافه می شود می تواند به یکی از نقشهای فوق الذکر تعلق داشته باشد. نقش مالک امتیاز برتر است و برای انجام مراحل اداری استفاده می شود.

مجوزهای سطح مخازن. تیم ها یا اعضای مربوطه می توانند مجوزهای سطح خواندن ، نوشتن یا مدیریت را به یک مخزن اختصاص دهند. هر سطح فعالیت‌هایی را که اعضای واگذار شده انجام می دهند ، با درجه محدودیت های مختلف دیکته می کند. نمودار زیر ، به منظور افزایش قابلیت ها ، در سطوح مربوطه ، سه سطح اجازه را نشان می دهد:



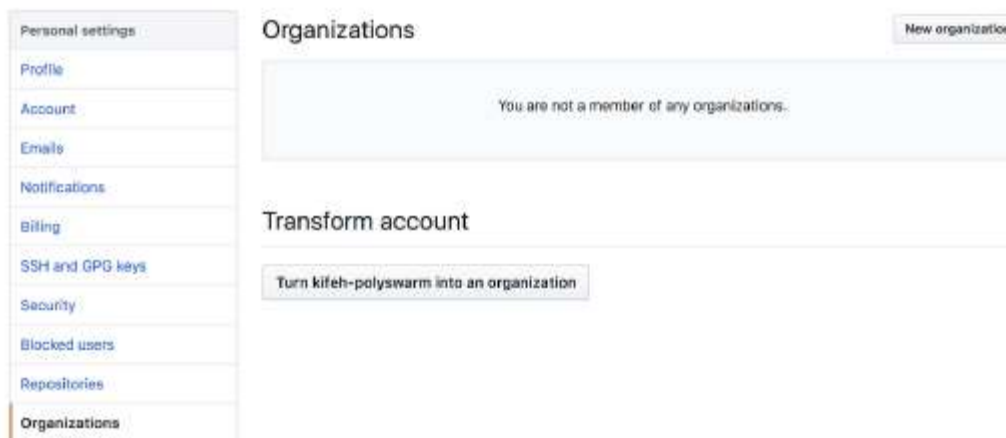
تیم ها: اینها اعضای سازمانی هستند که می توانند در تیم ها گروه بندی شوند ، با این گزینه که تیم ها را برای سازگاری با ساختار یک سازمان قرار می دهند.

تأیید هویت چند عاملی: سازمانها از اجرای احراز هویت دو عاملی و همچنین رویکردهای ورود به سیستم منفرد تجاری خاص مانند امنیت (SAML) و سیستم مدیریت هویت متقابل دامنه (SCIM) پشتیبانی می کنند.

تنظیم یک سازمان

برای ایجاد یک سازمان ، شما می توانید حساب شخصی خود را به یک سازمان تبدیل کنید یا سازمانی را ایجاد کنید که از آن طریق می توانید با حساب شخصی خود در ارتباط باشید.

در GitHub ، به تنظیمات رفته و سازمانها را انتخاب کنید. سپس ، بر روی سازمان جدید کلیک کنید:



جزئیات سازمان را وارد کنید ، برنامه ای را انتخاب کنید ، و روی ایجاد سازمان کلیک کنید:

Sign up your team

✓ Completed
Create personal account

Step 2:
Create organization

Step 3:
Invite members

Create an organization account

Organization name

✓

This will be your organization name on <https://github.com>.

Billing email

We'll send receipts to this inbox.

Choose your plan

☒ **Free**
Unlimited users and public repositories
\$0

☐ **Team**
Starts at \$35 / month which includes your first 5 users.
Unlimited public repositories
Unlimited private repositories
\$9
per user / month

☐ **Business**
Includes everything in the Team plan, plus:
SAML based single sign-on (SSO)
Access provisioning
99.95% uptime SLA
24/5 email support with < 10-hour response time
Learn more about our Business Plan or contact our team.
\$21
per user / month

☐ This account is owned by a business.
[See our Corporate Terms of Service for details.](#)

By clicking on "Create organization" below, you are agreeing to the [Terms of Service](#) and the [Privacy Policy](#).

Create organization

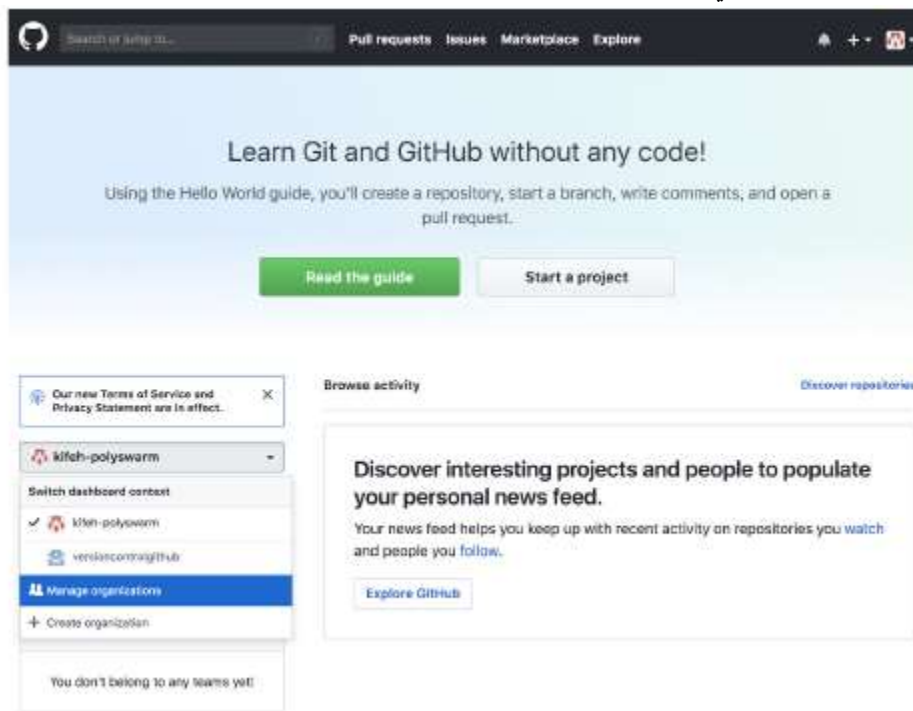
کاربران را جستجو و اضافه کنید ، یا برای پایان دادن به روند در سریعترین مرحله ، روی Finish کلیک کنید.

تنظیم یک تیم

با سازمان های GitHub ، می توانید همانطور که قبلاً گفته شد ، مشارکت کنندگان را در تیم ها سامان دهید و مجوزها و محدودیت ها را در سطح تیم و سطح مخازن مدیریت کنید.

برای ایجاد تیم تحت کنترل نسخه GitHub این مراحل را دنبال کنید:

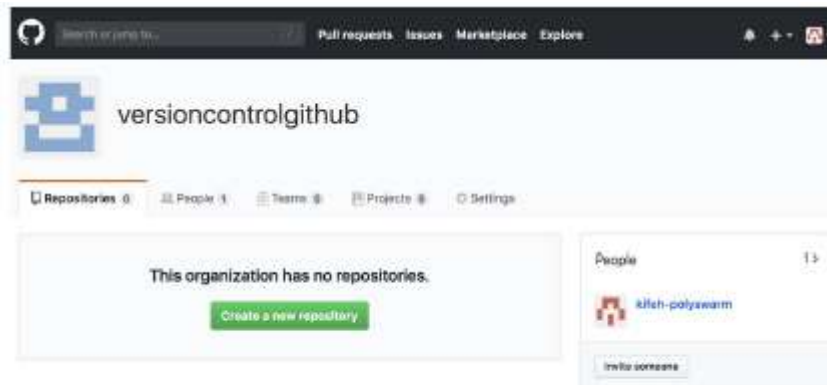
1. به <https://github.com/> بروید.
2. در سمت چپ خود ، باید یک منوی کشویی با نام کاربری خود پیدا کنید. برای نشان دادن سازمانهایی که به آن تعلق دارید ، روی کشویی کلیک کنید:



3.

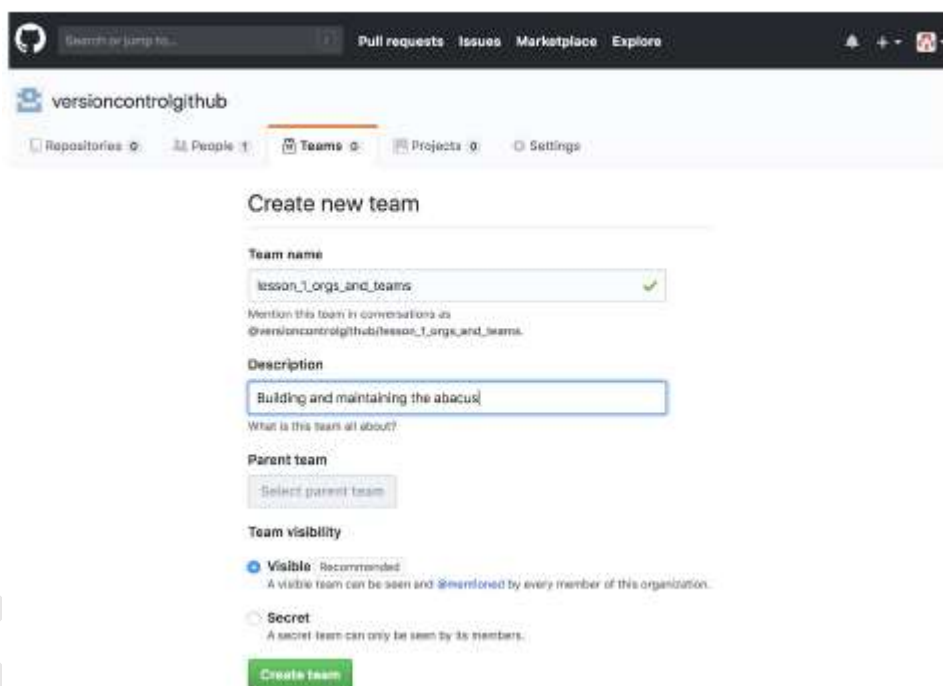
3. بر روی مدیریت سازمانها کلیک کنید و سپس در قسمت بعدی بر روی گزینه کنترل نسخه GitHub کلیک کنید.

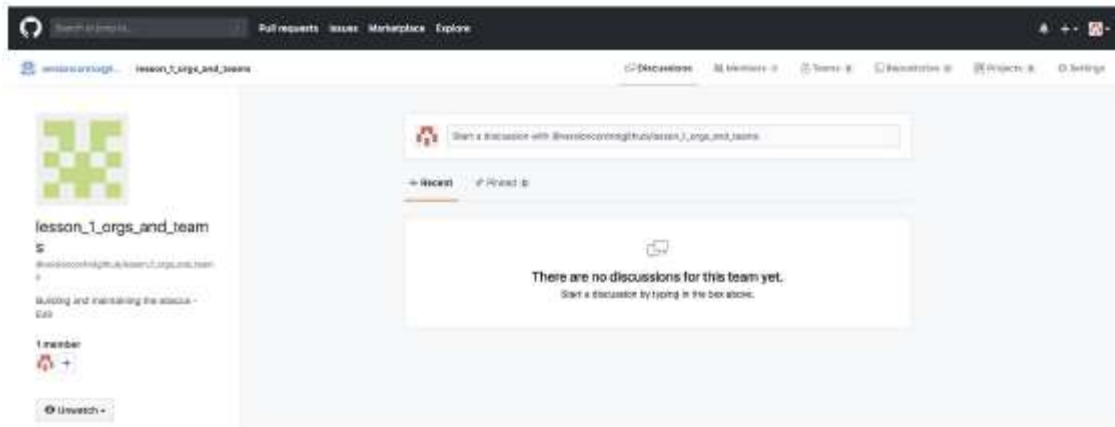
4- بر روی زبانه تیمها در داشبورد سازمان کلیک کنید:



5- در قسمت بعدی بر روی دکمه تیم جدید کلیک کنید.

6. جزئیات تیم را تنظیم کرده و روی ایجاد تیم کلیک کنید:





شما با موفقیت یک تیم تحت کنترل نسخه سازمانی Github ایجاد کرده اید. همانطور که از مراحل قبل مشاهده می شود ، یک تیم می تواند: برای مدیریت کارهایی که تحت مخازن سازمان یافته اند ، مورد استفاده قرار گیرد.

آیا اعضا به آن اختصاص داده شده است تا در مخازن خاص همکاری کنند؟ تیم های فرزندی را که در زیر آن ایجاد شده اند ایجاد کنید. این را می توان از برگه Team ، که در تصویر قبل نشان داده شده است ، بدست آمد.

MARKETPLACE

GitHub امکان ادغام برنامه ها با حساب کاربری شما را ارائه می دهد. این برنامه ها طیف وسیعی از نقش ها مانند ادغام مداوم ، تجزیه و تحلیل کیفیت کد و تجزیه و تحلیل مدیریت وابستگی را ارائه می دهند. بعد کد حساب را برای حساب های خود تنظیم می کنیم. کد برای تجزیه و تحلیل کدی که در یک پروژه وارد شده است برای شناسایی مناطقی از پیشرفت که در آن می توان قبل از ادغام تغییرات انجام داد ، استفاده می شود.

تنظیم قوانین مربوط به حسابهای مربوطه

استفاده از متن زیر را به عنوان متن توضیحات تمرین در نظر بگیرید.

برای راه اندازی Codacy مراحل زیر را دنبال کنید.

1. به <https://github.com> بروید.
2. بر روی Marketplace در نوار پیمایش بالا کلیک کنید.
3. از لیست دسته بندی ها بر روی کیفیت Code کلیک کنید.
4. روی Codacy که در سمت راست ذکر شده است کلیک کنید:

Marketplace

Code quality

Search

Automate your code review with style, quality, security, and test-coverage checks when you need them.

LGTM
Find and prevent zero-days and other critical bugs, with customizable alerts and automated code review.

codebeat
Code review expert on demand. Automated for mobile and web.

CodeFactor
Automated code review for GitHub.

Better Code Hub
A shared Definition of Done for code quality.

Codecov
Group, merge, archive and compare coverage reports.

Sider
Detect anti-pattern instances and apply project best practices in your pull requests automatically. Easy and quick setup.

ImgBot
A GitHub app that optimizes your images.

Code Climate
Automated code review for technical debt and test coverage.

Coveralls
Ensure that new code is fully covered, and see coverage trends emerge. Works with any CI service.

Codacy
Automated code reviews to help developers ship better software, faster.

TestQuality
Modern, powerful, test plan management.

Looking for more tools that work with GitHub?
Browse the [Works With GitHub](#) directory for services that integrate with the GitHub platform.

4. از آنجا ، روی تنظیم یک طرح کلیک کنید و آن را به صورت رایگان نصب کنید:

The screenshot displays the Codacy website. The top navigation bar includes links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. The main header features the Codacy logo and a 'Get up a pilot' button. Below this, the 'Categories' section lists 'Code quality' and 'Code review'. The 'Supported languages' section mentions 'Go, Java, JavaScript and 7 other languages supported'. The 'Developer links' section includes 'Support', 'Blog', 'Documentation', 'Privacy Policy', and 'Terms of Service'.

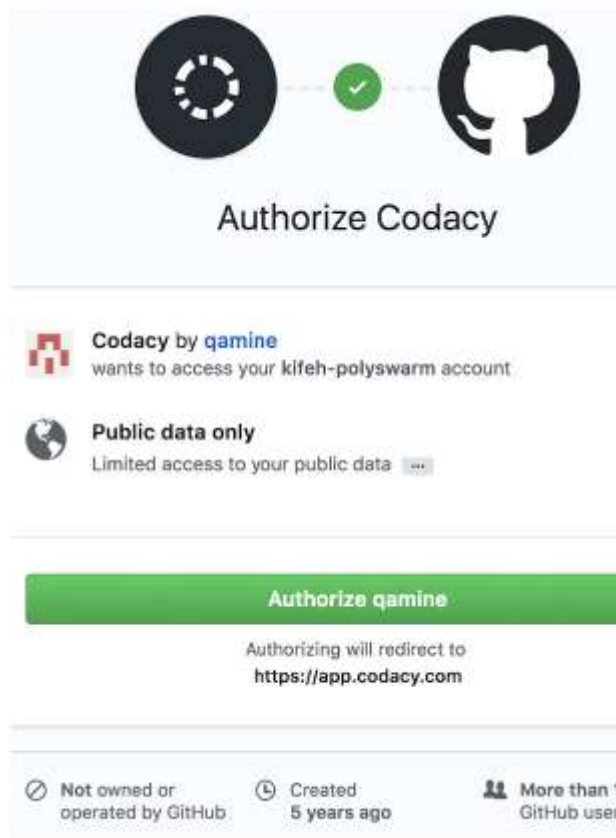
The main content area is divided into two columns. The left column, titled 'Track your project quality evolution', describes how Codacy helps track code quality over time and provides a link to 'Read more...'. The right column, titled 'Integrated in your workflow', explains how Codacy integrates with GitHub repositories to get quality analysis of every pull request. Below these columns are several smaller screenshots showing the Codacy interface.

The bottom section, titled 'Pricing and setup', shows two pricing options: 'Open Source' (Free for open source projects) and 'Pro' (\$18 per year, 3 months). A 'Get up a pilot' button is also present. The footer mentions that Codacy is provided by a third party and is governed by separate terms, privacy policy, and support documentation.

5.

6. بر روی Complete Order کلیک کنید و نصب را شروع کنید.

7. با کلیک کردن روی دکمه Autorizqamine ، برنامه را مجاز کنید:



8- رمز عبور خود را وارد کرده و روی تأیید گذرواژه کلیک کنید تا مراحل تنظیم آن انجام شود.

RUNTIME CONFIG

Git از پیکربندی گزینه های زمان اجرا پشتیبانی می کند. این گزینه ها و / یا مقادیر توسط سایر دستورات Git برای دیکته رفتار استفاده می شود. تنظیمات زمان اجرا با استفاده از دستور git config تنظیم می شود.

دستور تنظیمات git امکان تنظیم ، بازیابی ، حذف و جایگزینی تنظیمات را فراهم می کند. تنظیمات Git در سه سطح تنظیم شده است ، یعنی: پیکربندی گسترده سیستم این گزینه ها در پرونده پیکربندی / git / etc تنظیم شده اند. ایستگاه از پیش تنظیم شده در این گروه برای همه کاربران در رایانه استفاده می شود. برای دسترسی به این تنظیمات ، از پرچم سیستم پیکربندی git استفاده می کنید که باید از پیکربندی گسترده سیستم استفاده شود.

پیکربندی خاص کاربر

این گزینه ها در پرونده ~ / gitconfig تنظیم شده اند. ایستگاه از پیش تنظیم شده در اینجا برای حساب کاربری که در رایانه استفاده می شود ، استفاده می شود.

کاربر خاص

تنظیمات از طریق پرچم جهانی پیکربندی git قابل دستیابی است و مشخص می کند باید از پیکربندی اختصاصی کاربر استفاده شود.

پیکربندی ویژه مخزن

تنظیمات ویژه مخزن در پرونده `path_to_repository / .git / config` تنظیم شده است. گزینه های تنظیم شده در اینجا در سطح مخزن استفاده می شوند. نمونه ای از پیکربندی در اینجا URL GitHub یک مخزن است که در این سطح تنظیم شده است. این تنظیمات از طریق پرچم محلی پیکربندی git قابل دسترسی است و مشخص می کند که باید از پیکربندی اختصاصی مخزن استفاده شود. شما می توانید با استفاده از گزینه `for file` یک فایل پیکربندی مشخص را مشخص کنید.

تنظیم اعتبار کاربر

برای تنظیم اعتبار کاربر برای یک حساب ، موارد زیر را دنبال کنید

مراحل:

1. راه اندازی سریع ترمینال یا فرمان.
 2. با استفاده از دستور زیر نام کاربری را تنظیم کنید:
- git config global user.name kifehpolyswarm

```
alexmagana@ALEXs-MacBook-Pro ~ % git config --global user.name kifeh-polyswarm
alexmagana@ALEXs-MacBook-Pro ~ %
```

با استفاده از دستور زیر ایمیل را تنظیم کنید:

git config global user.email kifeh@polyswarm.com

```
alexmagana@ALEXs-MacBook-Pro ~ % git config --global user.email kifeh@poly-swarm.com
alexmagana@ALEXs-MacBook-Pro ~ %
```

پیکربندی را با استفاده از یکی از دستورات زیر لیست کنید:

```
git config --global --list
```

Or, `git config --list` , to fetch all the available presets

```
credential.helper=osxkeychain
core.excludesfile=/Users/alexmagana/.gitignore
user.name=kifeh-polyswarm
user.email=kifeh@poly-swarm.com
(END)
```

حذف پیکربندی

با استفاده از پیکربندی git ، می توانیم پیکربندی را در صورت نیاز به تغییر حذف کنیم. ابزار `git config` با استفاده از گزینه `unset` از این پشتیبانی می کند.

نحو فرمان به شرح زیر است:

```
git config --global --unset
[section_name].[section_variable]
```

Example

```
git config --global --unset user.name
```

دستور قبلی مقدار تعیین شده برای نام کاربری را حذف می کند.

SSH CONFIGURATION

برای تعامل با یک مخزن و / یا انجام کارهایی که در GitHub از محیط محلی شما انجام می شود ، باید این ادعا کنید که شما فردی هستید که می گوید هستید. Git از این طریق با استفاده از ترکیبی از نام کاربری و رمز عبور یا استفاده از کلید SSH برای تأیید صحت اتصال یا درخواستهای ارسال شده به GitHub از محیط محلی شما پشتیبانی می کند.

استفاده از کلیدهای SSH باعث افزایش امنیت می شود و از لزوم تهیه نام کاربری و رمز عبور برای هر درخواست جلوگیری می کند.

تنظیم کردن SSH

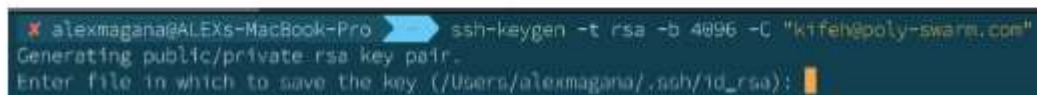
برای تنظیم کلید SSH برای حساب ، این مراحل را دنبال کنید:

1. راه اندازی سریع ترمینال یا فرمان.

2. با استفاده از دستور زیر یک کلید SSH ایجاد کنید:

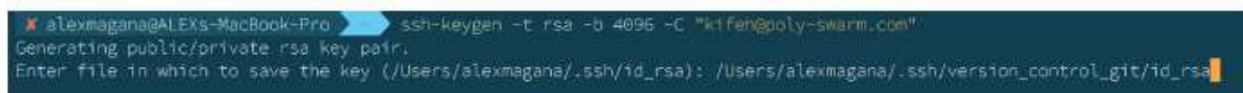
```
ssh-keygen -t rsa -b 4096 -C "[email_address]"
```

Example: `ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"`



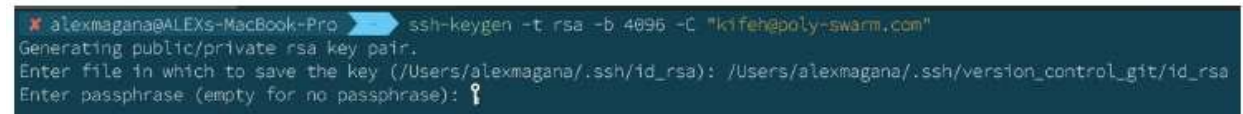
```
* alexmagana@ALEXs-MacBook-Pro ➤ ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alexmagana/.ssh/id_rsa):
```

مکانی که کلید تولید شده در آن ذخیره می شود را مشخص کنید. شما می توانید Enter را فشار دهید تا به ژنراتور اصلی دستور دهید از موقعیت پیش فرض استفاده کند:

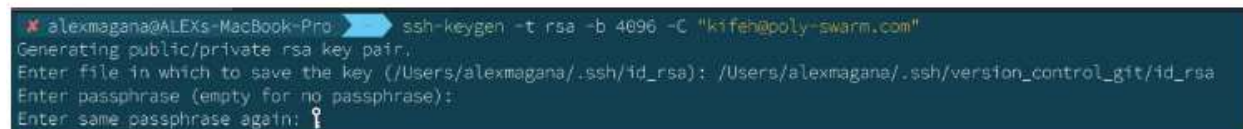


```
* alexmagana@ALEXs-MacBook-Pro ➤ ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alexmagana/.ssh/id_rsa): /Users/alexmagana/.ssh/version_control_git/id_rsa
```

برای تأمین امنیت کلید ایجاد شده ، همانطور که در تصاویر زیر مشاهده می شود ، یک عبارت عبور را تایپ کنید:



```
* alexmagana@ALEXs-MacBook-Pro ➤ ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alexmagana/.ssh/id_rsa): /Users/alexmagana/.ssh/version_control_git/id_rsa
Enter passphrase (empty for no passphrase):
```



```
* alexmagana@ALEXs-MacBook-Pro ➤ ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alexmagana/.ssh/id_rsa): /Users/alexmagana/.ssh/version_control_git/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

استفاده از عبارات عبور برای قسمت امنیتی 2

```
* alexmagana@ALEXs-MacBook-Pro ssh-keygen -t rsa -b 4096 -C "kifeh@poly-swarm.com"
Generating public/private rsa key pair.
Enter file in which to save the key (/Users/alexmagana/.ssh/id_rsa): /Users/alexmagana/.ssh/version_control_git/id_rsa
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /Users/alexmagana/.ssh/version_control_git/id_rsa.
Your public key has been saved in /Users/alexmagana/.ssh/version_control_git/id_rsa.pub.
The key fingerprint is:
SHA256:vE/jbrE53uyVl2Z5snozZIHG8zBc1l2Bz2fVlNg-BJI kifeh@poly-swarm.com
The key's randomart image is:
+----[RSA 4096]-----+
|      . . . . o=0 |
|      E. =..* |
|      o = o o |
|      - o o +o |
|      S. =.oo. |
|      .. ++.. |
|      . o+ooo+. |
|      +=+. =+. |
|      +=+ o |
+-----[SHA256]-----+
alexmagana@ALEXs-MacBook-Pro
```

همانطور که در عکس های قبلی نشان داده شده ، کلید در مکان مشخص شده ذخیره می شود

1. عامل SSH را با استفاده از دستور زیر شروع کنید:

eval "\$(ssh-agent s)"

2. در macOS ، ssh ، / ~ / پیکربندی را ویرایش کنید تا عامل ssh بتواند به طور خودکار کلیدها را بارگیری و ذخیره کند

عبارات کلیدی در keychain:

Host *

AddKeysToAgent yes

UseKeychain yes

IdentityFile

[location_of_the_generated_private_key]

Example: Host *:

AddKeysToAgent yes

UseKeychain yes

IdentityFile

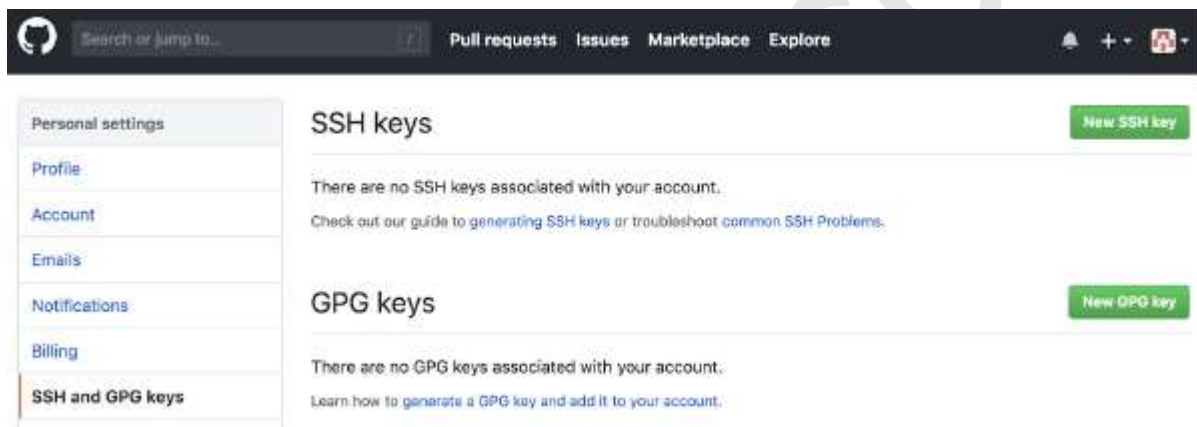
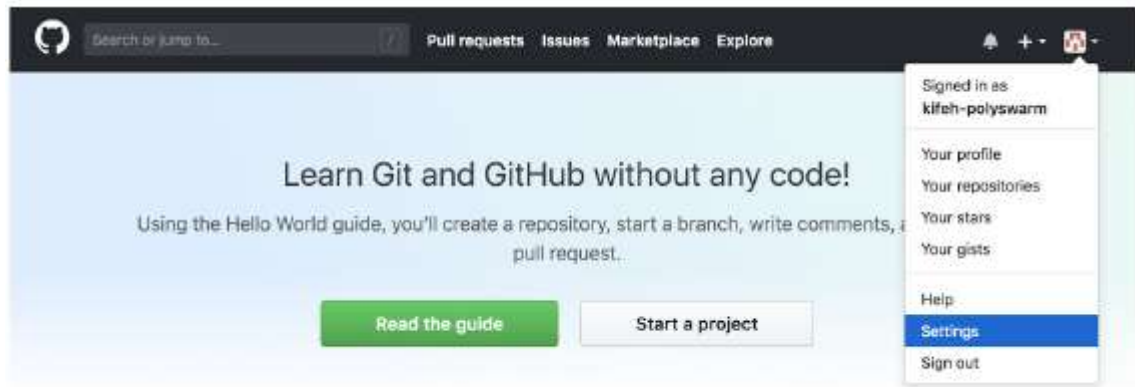
~/.ssh/version_control_git/id_rsa

کلید خصوصی SSH را به عامل ssh اضافه کنید.

Example: ssh-add -K

~/.ssh/version_control_git/id_rsa

کلید عمومی SSH را به حساب GitHub خود اضافه کنید. به <https://github.com> بروید و سپس مطابق تصویر زیر به تنظیمات بروید:

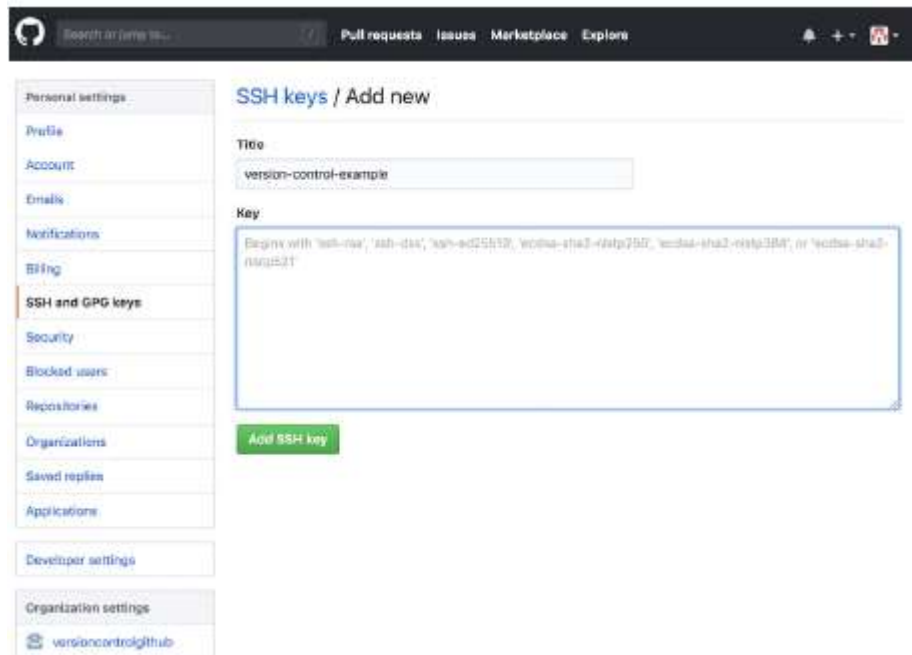


6. روی کلید SSH جدید کلیک کنید و عنوانی را برای کلید عمومی SSH خود تنظیم کنید.

7. راه اندازی ترمینال و کپی کردن مطالب کلید عمومی SSH در کلیپ بورد با استفاده از دستور زیر:

```
.pbcopy <~ / .ssh / version_control_git / id_rsa.pub
```

8- محتوای کلید عمومی را در قسمت زیر برچسب Key قرار دهید:



9. برای افزودن کلید عمومی به حساب GitHub خود ، روی افزودن کلید SSH کلیک کنید.

10. آزمایش کنید که با استفاده از دستور زیر ، کلید SSH به درستی تنظیم شده است:

```
ssh -T git@github.com
```

```
alexmagania@ALEXs-MacBook-Pro ➤ ssh -T git@github.com
Hi kifeh-polyswarm! You've successfully authenticated, but GitHub does not provide shell access.
alexmagania@ALEXs-MacBook-Pro ➤
```

شما با موفقیت کلید SSH را برای حساب تنظیم کرده اید.

CREATING A REPOSITORY

کنترل نسخه مستلزم آن است که پرونده ها و تغییرات مرتبط با آن که باید ردیابی شوند در یک مخزن سازماندهی می شوند که واحدی است که Git بعنوان نامزد کنترل منبع شناسایی می کند. برای شروع یک کار ، باید یک مخزن ایجاد کنیم.

در این بخش ، دورویکرد را که ممکن است شما برای اولیه سازی یک مخزن استفاده کنید ، بررسی خواهیم کرد.

CREATING A REPOSITORY IN A LOCAL ENVIRONMENT

اولیه سازی یک مخزن بصورت محلی مستلزم استفاده از git init و نقشه برداری از مخزن محلی به مخزن از راه دور مربوطه می باشد:

1. راه اندازی ترمینال.

آ. در رایانه لینوکس: Ctrl + Alt + T را فشار دهید.

ب در رایانه macOS: bar + نوار فاصله را فشار دهید ، تایپ کنید

Terminal یا iTerm و سپس روی برنامه کلیک کنید

آرم برای راه اندازی ترمینال:



در رایانه Microsoft Windows: Win + R را فشار دهید

روی صفحه کلید خود برای راه اندازی پنجره Run. سپس،

cmd.exe را تایپ کرده و Enter را روی صفحه کلید خود فشار دهید ، یا

روی پنجره Run کلیک کنید.

2. با استفاده از. یک دایرکتوری برای برنامه ایجاد کنید

دستور زیر: mkdir abacus



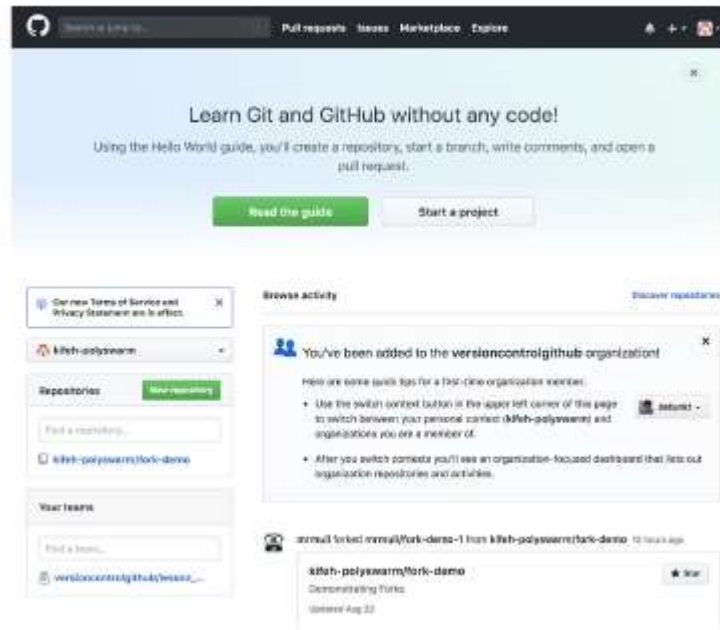
3. با استفاده از دستور زیر ، فهرست کار را به فهرست پروژه تغییر دهید: cd abacus

4- مقدار اولیه مخزن را با استفاده از دستور زیر شروع کنید: git init.

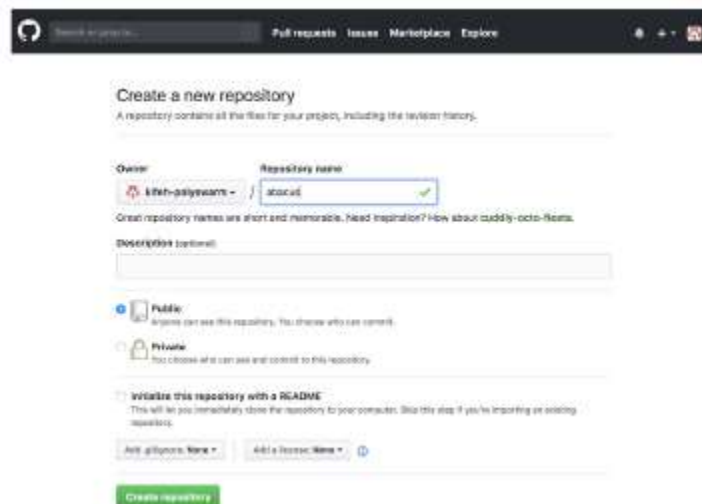
5. به <https://github.com/> بروید

6. روی مخزن جدید در قسمت سمت چپ صفحه کلیک کنید

صفحه:

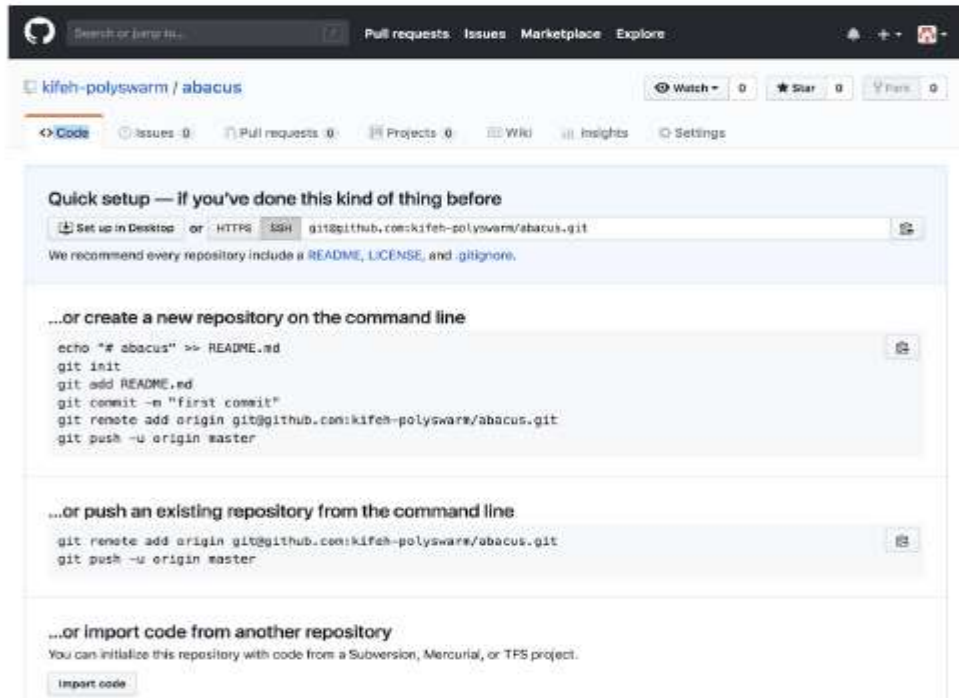


همانطور که در شکل نشان داده شده است ، نام مخزن را مشخص کنید
تصویر زیر:



روی ایجاد مخزن کلیک کنید.

برای به دست آوردن URL SSH بر روی دکمه SSH کلیک کنید. اگر قبلاً این نوع کار را انجام داده اید ، این دکمه در زیر متن است که تنظیمات سریع را می خواند:



در ترمینال ، URL GitHub مخزن ، یعنی **abacus** را مشخص کنید:

```
git remote add origin [repository_url]
```

Here is an example: **git remote add origin**
git@github.com:kifeh-
polyswarm/abacus.git

```
alexmagana@ALEXs-MacBook-Pro: ~/Documents/03-03-2020/abacus [master] $ git remote add origin git@github.com:kifeh-polyswarm/abacus.git
```

پرونده ای را که میزبان کلاس خود خواهیم بود با استفاده از کد زیر همانطور که در تصاویر زیر مشاهده می کنید میزنید.


```
mkdir -p src/lib
```

```
touch src/lib/compute.py
```

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master mkdir -p src/lib
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

Hosting the code

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master touch src/lib/compute.py
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

Hosting the code part

برای اولین بار پرونده ها را با استفاده از دستور زیر آماده کنید:

```
git add src/lib/compute.py
```

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master git add src/lib/compute.py
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

Preparing the files for commit

با استفاده از دستور زیر پرونده ها را commit کنید:

```
Git commit -m "Initial commit"
```

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master git commit -m "Initial commit"
[master (root-commit) f4e4e8d] Initial commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 src/lib/compute.py
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

گزینه m که با فرمان commit استفاده می شود پیامی را که ما می خواهیم برای یک commit استفاده کنیم مشخص می کند.
با استفاده از دستور زیر می توانید پرونده های مخزن را به مخزن موجود در GitHub فشار دهید:

`git push-u origin master`

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master git push -u origin master
Counting objects: 5, done.
Writing objects: 100% (5/5), 292 bytes | 292.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To github.com:kifeh-polyswarm/abacus.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

Pushing to the GitHub repository

گزینه u که با استفاده از فرمان push استفاده می شود ، شاخه از راه دور مخزن از راه دور را که مخزن محلی به آن وصل می شود ، تنظیم می کند. از این گزینه برای ایجاد مرجع ردیابی بین یک شعبه محلی و از راه دور استفاده می شود. این امکان را به شما می دهد تا بدون نیاز به مشخص کردن استدلال هایی از قبیل نام شعبه همانطور که در تصویر زیر نشان داده شده است ، git pull یا git push را انجام دهید:

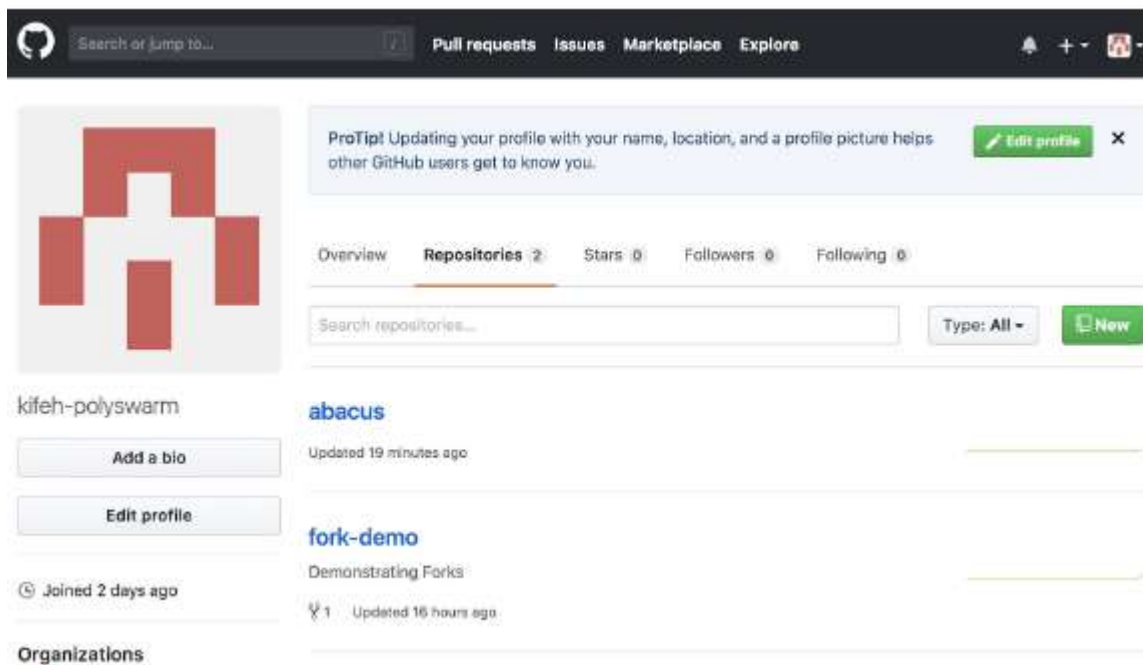
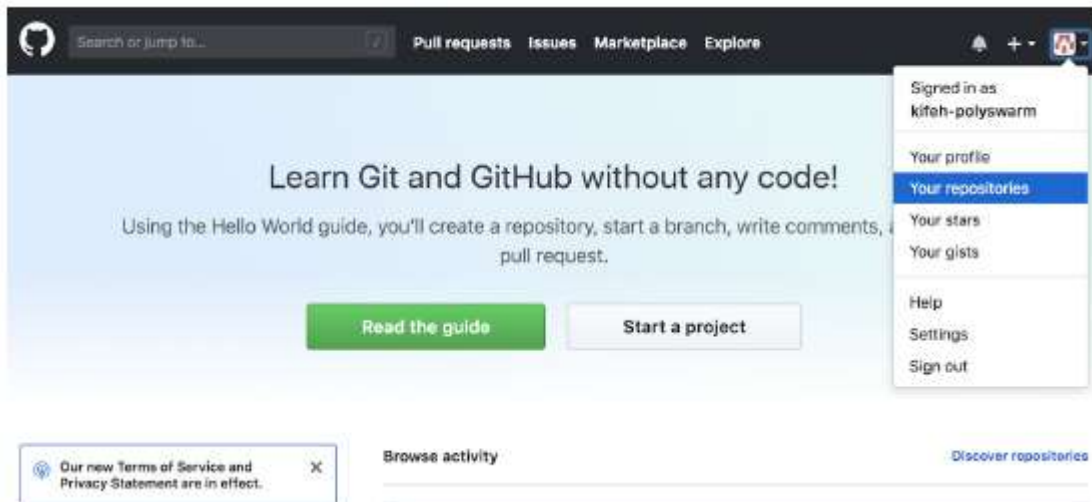
```
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master git pull
Already up to date.
alexmagana@ALEXs-MacBook-Pro ~/Documents/Github/abacus master
```

ایجاد یک نمایندگی در GITHUB

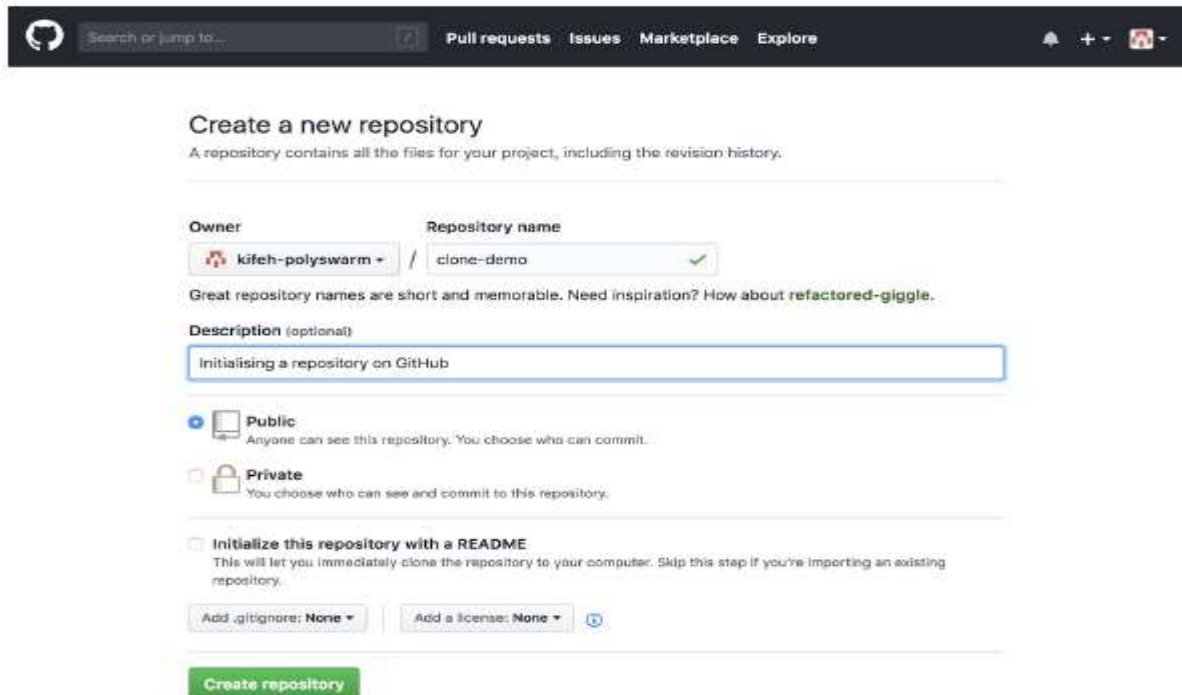
برای شروع یک کار ، می توانید به جای شروع یک مخزن به صورت محلی ، مخزن را در GitHub ایجاد کنید ، پس از آن می توانید آن را بصورت محلی کلون کنید:

به <https://github.com> بروید.

2. با کلیک روی مخازن خود به فهرست مخازن حساب خود بروید و سپس مطابق تصویر زیر روی دکمه New کلیک کنید:



نام و توضیحات مخزن را مشخص کنید:



Search or jump to... Pull requests Issues Marketplace Explore

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: **kifeh-polyswarm** / Repository name: **clone-demo** ✓

Great repository names are short and memorable. Need inspiration? How about **refactored-giggle**.

Description (optional): **Initialising a repository on GitHub**

☒ **Public**
Anyone can see this repository. You choose who can commit.

☐ **Private**
You choose who can see and commit to this repository.

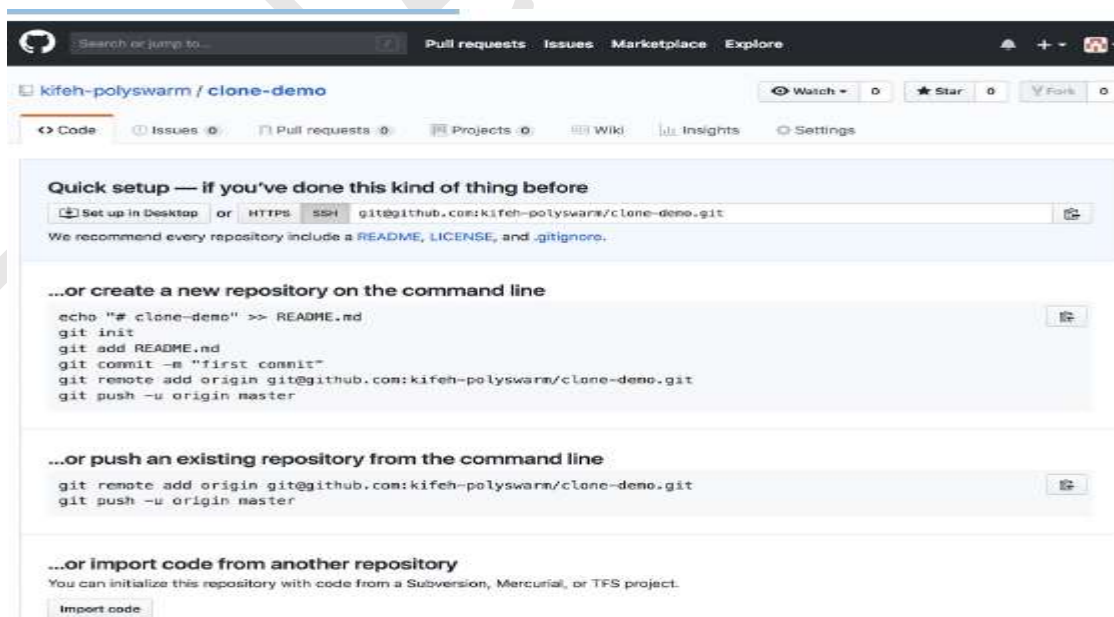
☐ **Initialize this repository with a README**
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** Add a license: **None**

Create repository

روی دکمه ایجاد مخزن کلیک کنید.

URL SSH را بازیابی کنید:



kifeh-polyswarm / clone-demo Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

Quick setup — if you've done this kind of thing before

Set up in Desktop or **HTTPS** **SSH** `git@github.com:kifeh-polyswarm/clone-demo.git`

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# clone-demo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin git@github.com:kifeh-polyswarm/clone-demo.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin git@github.com:kifeh-polyswarm/clone-demo.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

مخزن موجود در محیط محلی خود را با استفاده از دستور زیر همانطور که در تصویر زیر مشاهده می شود کlon کنید:

```
git clone [repository_url].
```

Example: `git clone git@github.com:kifeh-polyswarm/clone-demo.git`

```
alexmagana@ALEXs-MacBook-Pro ~/Documents/GitHub$ git clone git@github.com:kifeh-polyswarm/clone-demo.git
Cloning into 'clone-demo'...
warning: You appear to have cloned an empty repository.
alexmagana@ALEXs-MacBook-Pro ~/Documents/GitHub$
```

NAVIGATING A REPOSITORY

GitHub ویژگی هایی را در سطح مخازن ارائه می دهد. این ویژگی ها با ارائه بینش های مربوط به سرعت ، پیروی از استانداردهای جامعه و استفاده از یک مخزن توسط جامعه ، چشم انداز کار و پیشرفت را در مخازن ایجاد می کنند. همچنین GitHub از اضافه کردن و حذف مشارکت کنندگان به / از یک مخزن پشتیبانی می کند.

ADDING AND DELETING CONTRIBUTORS

برای افزودن یا حذف مشارکت کنندگان از یک مخزن معین ، این مراحل را دنبال کنید:

1. به عنوان مثال به یک مخزن حساب خود بروید.

<https://github.com/kifehpolyswarm/abacus>

2. روی تنظیمات کلیک کنید و سپس بر روی Collaborators کلیک کنید:

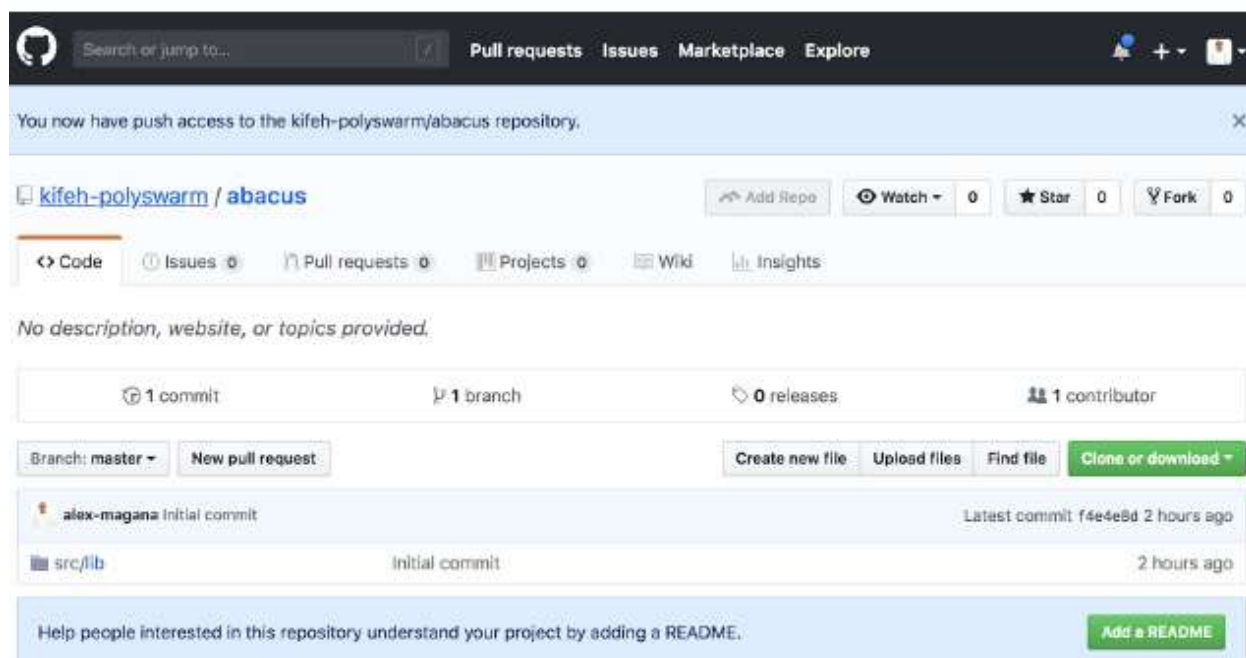
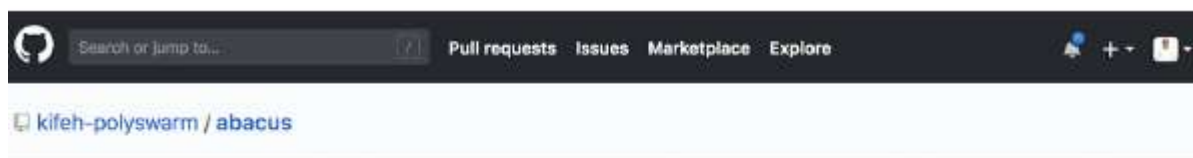
GitHub repository page for **kifeh-polyswarm / abacus**. The page shows the repository name, navigation tabs (Code, Issues, Pull requests, Projects, Wiki, Insights, Settings), and repository statistics (1 commit, 1 branch, 0 releases, 1 contributor). It also displays the latest commit by alex-magana and a button to add a README.

GitHub repository page for **kifeh-polyswarm / abacus**, showing the **Settings** tab. The left sidebar has **Collaborators** selected. The main content area shows a message: "This repository doesn't have any collaborators yet. Use the form below to add a collaborator."

3. جستجو در مورد کاربر توسط آدرس ایمیل ، نام کاربری یا نام کامل آنها ، به عنوان مثال Alex Magana همانطور که در تصویر زیر نشان داده شده است:

GitHub repository page for **kifeh-polyswarm / abacus**, showing the **Settings** tab. The left sidebar has **Collaborators** selected. The main content area shows a search form for adding a collaborator with the text "Search by username, full name or email address" and a button "Add collaborator".

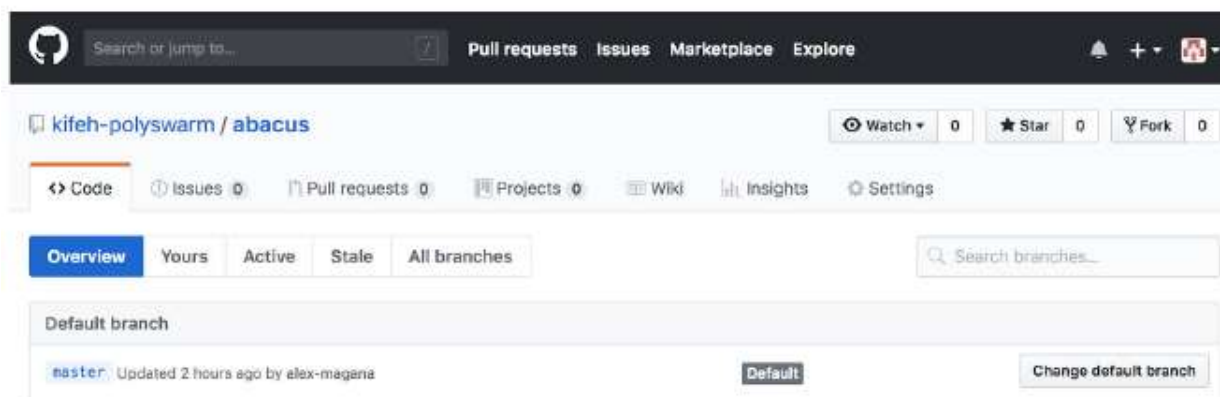
4- کاربر حاصل را انتخاب کنید و همانطور که در تصاویر زیر مشاهده می کنید ، افزودن همکار را کلیک کنید:
هنگامی که کاربری که برای آن دعوت نامه ارسال کرده اید ، دعوت نامه را بپذیرد ، می تواند به مخزن کمک کند.



Repository home page

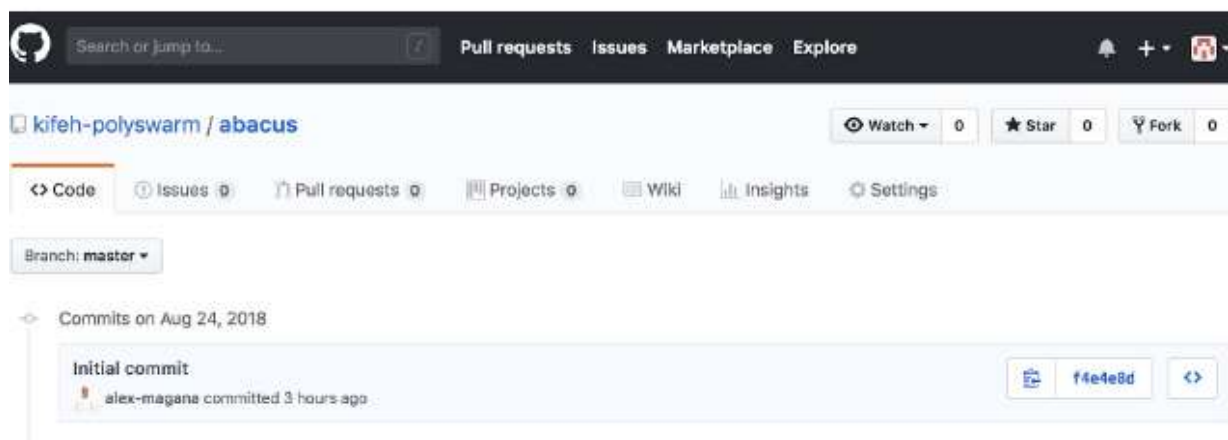
NAVIGATING BRANCHES, COMMITS, AND INSIGHTS (CONTRIBUTORS, PULSE, FORKS)

مطابق با مراحل خانه داری می توان شاخه ها را مشاهده و حذف کرد. این می تواند با مراجعه به مخزن موجود در حساب شما انجام شود ، به عنوان مثال ، <https://github.com/kifehpolyswarm/abacus> در مرحله بعد ، برای مشاهده لیستی از شعب ، باید بر روی دکمه Forked 1 Branch در نوار بالا کلیک کنید. سپس ، شما باید به صفحه ای با نمای کلی از شعب موجود در مخزن هدایت شوید:

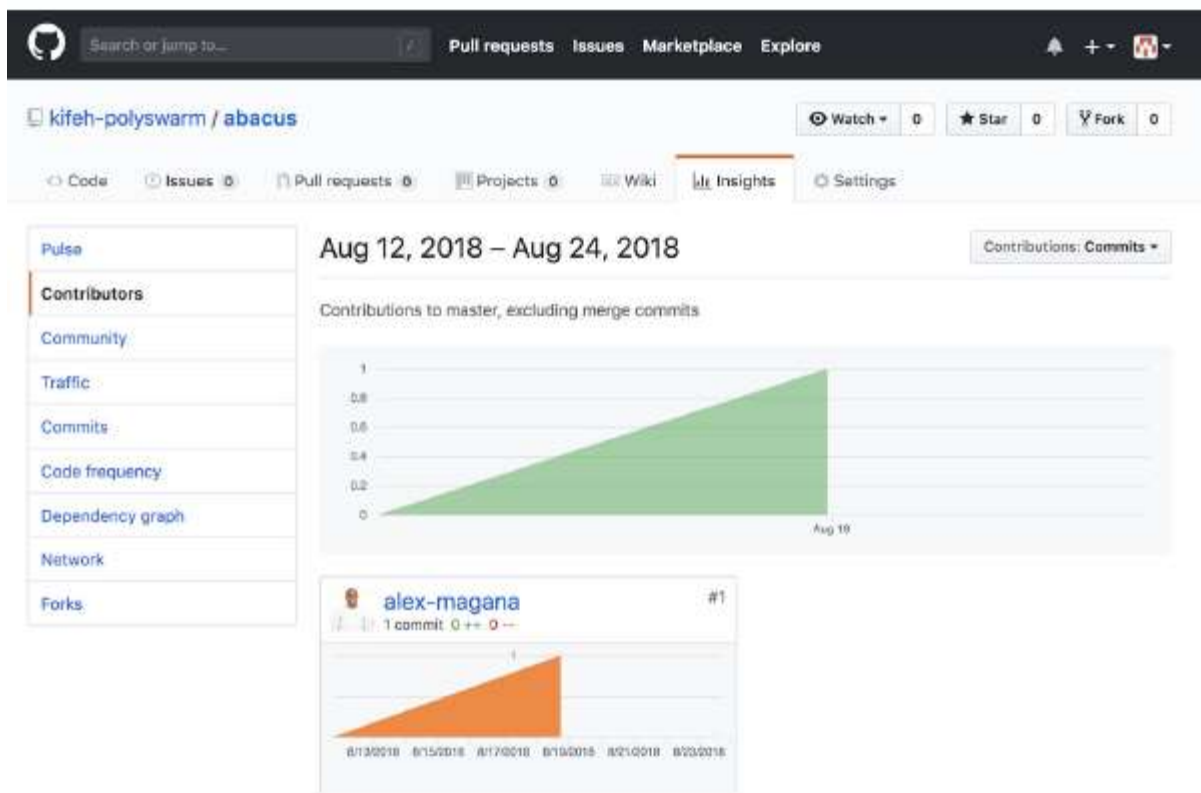


commits وضعیت مخزن را در زمان ایجاد آنها ارائه می دهند. می توانید مخزن را در یک نقطه از تاریخ مرور کنید و پرونده هایی را تغییر دهید که توسط یک تعهد تغییر یافته است. سپس ، به یک مخزن در حساب خود بروید ، به عنوان مثال ، <https://github.com/kifehpolyswarm/abacus>

برای دیدن commits موجود در مخزن ، روی نماد 1 commit کلیک کنید. برای مشاهده تغییرات پرونده در مورد تعهد ، روی هش مرتبه کلیک کنید ، به عنوان مثال f4e4e8d همانطور که در تصویر زیر نشان داده شده است:



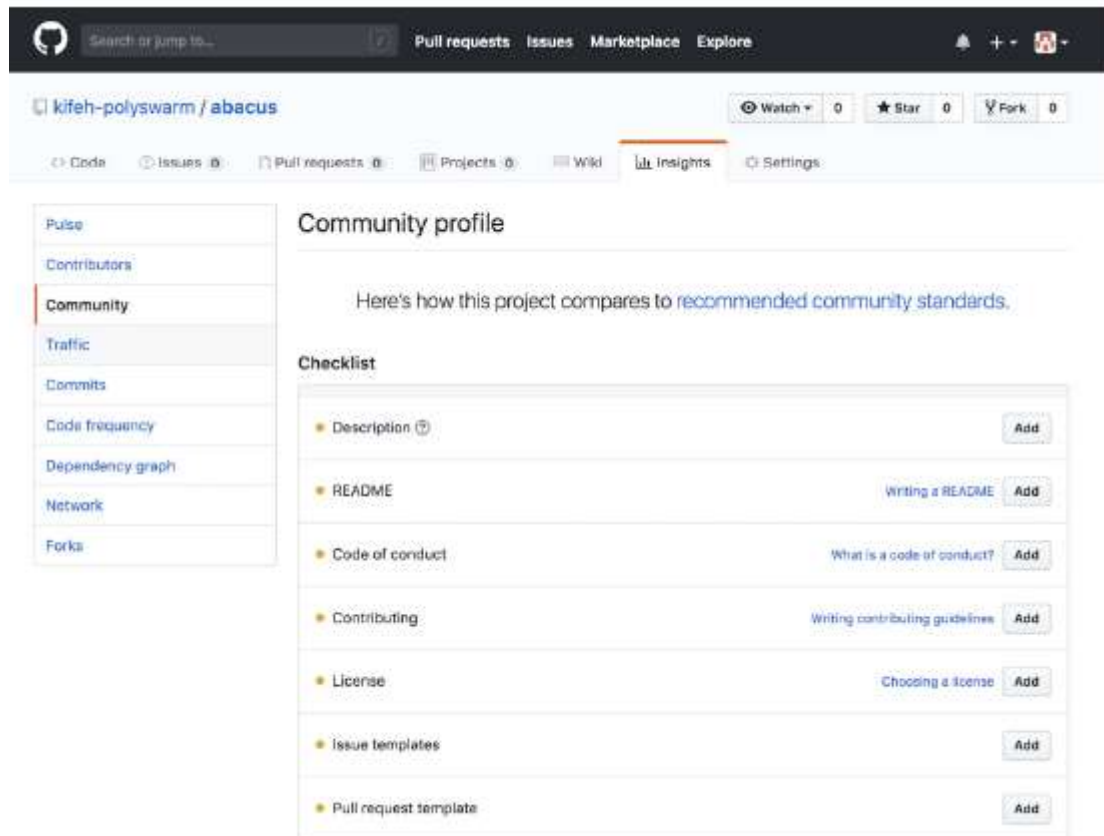
سپس برای مرور وضعیت یک مخزن در یک نقطه معین ، بر روی نماد پیکان دوبار کلیک کنید. این را می توان در سمت راست هر تعهدی که ذکر شده است ، یافت. در آخر ، می توانید تعهداتی را که به منظور همکاران مربوطه ترتیب داده شده اند ، مشاهده کنید. این نقش ضروری در هماهنگی موضوعات مانند اشتراک دانش در تیمها با استفاده از نقاط قوت نقاط مختلف مشارکت کنندگان دارد. در مخزن مورد نظر خود ، روی insights کلیک کنید و سپس برای مشاهده مشارکت کنندگان ، به همراه جزئیات مربوط به مشارکت های خاص که مطابق تصویر زیر نشان داده شده است ، روی مشارکت کنندگان کلیک کنید:



GITHUB ETIQUETTE

انتظار می رود دکوراسیون خاصی از مخازن و کاربران در استفاده از کنترل نسخه و همکاری در GitHub باشد. این ویژگی اطمینان حاصل می کند که مشارکت ها به صورت منظم انجام می شود و محیط های کاری سازنده را که بر روی ارائه ارزش متمرکز هستند ، ترویج می کند.

GitHub یک لیست چک را از طریق نمایه انجمن فراهم می کند که از طریق برگه Insights ناوبری مخزن همانطور که در تصویر زیر نشان داده شده است قابل دسترسی است:



REPOSITORY NAMES, TAGS, AND DESCRIPTIONS

به تعاریف زیر نگاهی بیندازید:

نامها

یک مخزن باید دارای یک اسم توصیفی باشد که مربوط به کارایی و کاربردی است که می خواهد تحویل دهد.

برچسب ها

اینها به منظور شناسایی نکات مهم خاص در تاریخ یک مخزن مورد استفاده قرار می گیرند ، برای مثال ، نرم افزار منتشر شده Git از ایجاد نشانگرهایی که به آنها برچسب گفته می شود ، برای مطابقت با نسخه های نرم افزاری پشتیبانی می کند.

برچسب ها دو نوع هستند: سبک و حاشیه نویسی.

برچسب های سبک وزن به عنوان نشانگر یک commit خاص عمل می کنند. این فقط مرجع commit را ذخیره می کند:

git tag v2.5

برچسب های بدون حاشیه به عنوان نشانگر یک commit خاص عمل می کنند و علاوه بر این اطلاعات مربوط به سازنده برچسب ، ایمیل و تاریخ ایجاد را نیز ذخیره می کنند:

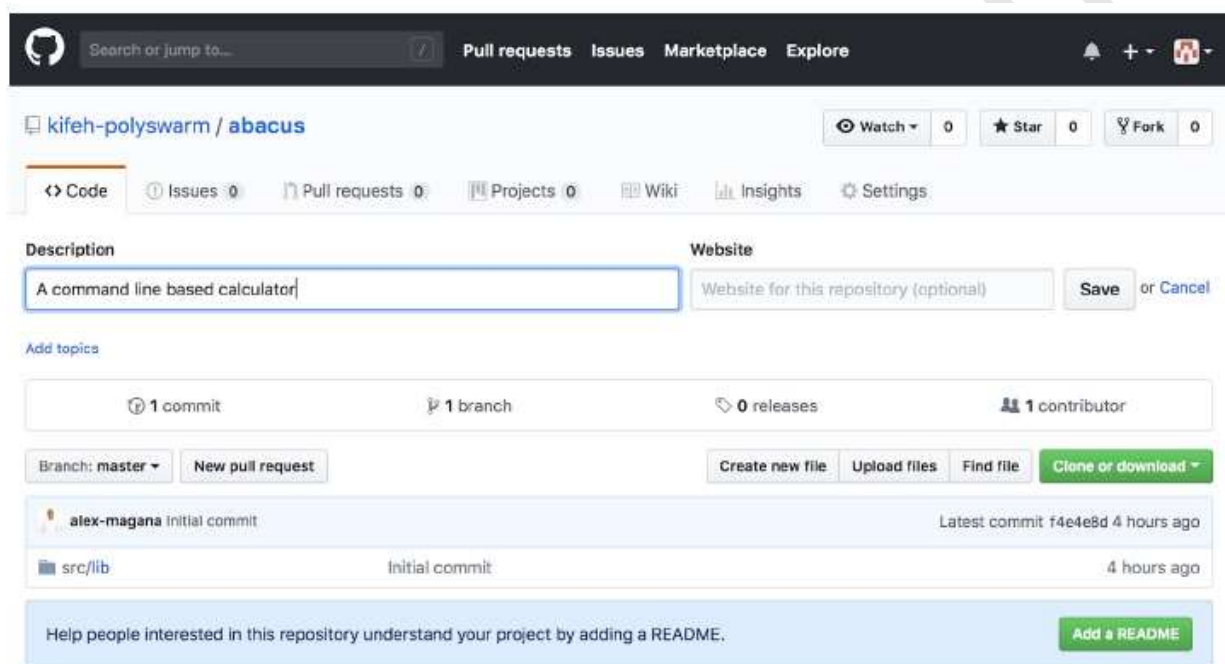
```
git tag -a v2.6 -m "Support sdk version 3"
```

توضیحات Descriptions

توصیف مخزن به عنوان بهترین روش مورد نیاز است. این به عنوان اولین نکته معرفی و مبنایی را مبنی بر اینکه عملکرد یک مخزن قابل درک است ، تعیین می کند.

اضافه کردن توضیحات

این کار با کلیک بر روی دکمه Add در سمت راست توضیحات در پروفایل Community و سپس اضافه کردن توضیحات و کلیک بر روی Save:



README.md

این سند مختصراً درمورد پروژه ای که توسط یک مخزن اداره می شود ، ارائه می دهد. این شامل یک راهنمای شروع ، منابع ، ویکی ها ، و راهنمایی های مربوط به رفتار و مشارکت است.

Adding a README.md, CODE_OF_CONDUCT.md and CONTRIBUTING.md

این با کلیک کردن بر روی دکمه Add در سمت راست README در پروفایل Community حاصل می شود. از اینجا ، شما باید جزئیات لازم را اضافه کنید و تغییرات را بدهید:

Search or jump to... Pull requests Issues Marketplace Explore

kifeh-polyswarm / abacus Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

abacus / README.md or cancel

Edit new file Preview Spaces 2 No wrap

```
1 # abacus
2 A command line based calculator
3
```

Commit new file

Create a README.md

Add an optional extended description...

☒ Commit directly to the master branch.

☐ Create a new branch for this commit and start a pull request. [Learn more about pull requests.](#)

Commit new file Cancel

با افزودن ویژگی هایی به برنامه ما در طول کتاب ، این پرونده تغییر خواهد کرد. سپس بر روی دکمه Add در سمت راست کد رفتار در پروفایل جامعه کلیک کنید. در مرحله بعد ، کد راهنمایی رفتاری را به شرح زیر انتخاب کنید:

Search or jump to... Pull requests Issues Marketplace Explore

kifeh-polyswarm / abacus Watch 0 Star 0 Fork 0

Code Issues Pull requests Projects Wiki Insights Settings

Add a code of conduct to your project

Contributor Covenant
Recommended for projects of all sizes

Citizen Code of Conduct
Suitable for large communities or events

Code of Conduct

1. Purpose

A primary goal of **abacus** is to be inclusive to the largest number of contributors, with the most varied and diverse backgrounds possible. As such, we are committed to providing a friendly, safe and welcoming environment for all, regardless of gender, sexual orientation, ability, ethnicity, socioeconomic status, and religion (or lack thereof).

This code of conduct outlines our expectations for all those who participate in our community, as well as the consequences for unacceptable behavior.

We invite all those who participate in **abacus** to help us create safe and positive experiences for everyone.

2. Open Source Citizenship

A supplemental goal of this Code of Conduct is to increase open source citizenship by encouraging participants to recognize and strengthen the relationships between our actions and their effects on our community.

Communities mirror the societies in which they exist and positive action is essential to counteract the many forms of inequality and abuses of power that exist in society.

If you see someone who is making an extra effort to ensure our community is welcoming, friendly, and encourages all participants to contribute to the fullest extent, we want to know.

To adopt Citizen Code of Conduct, enter your details. You'll have a chance to review before committing a CODE_OF_CONDUCT.md file to a new branch or the root of your project.

Community name
abacus

Contact email address
kifehpolyswarm.com

Link to reporting guidelines

Governing body
kifeh-polyswarm

Link to policy

Review and submit

سپس ، سند را به مخزن خود اختصاص دهید:



بعد ، روی دکمه Add در سمت راست مشارکت در نمایه انجمن کلیک کنید. سرانجام ، شما می توانید این سند را ایجاد و آن را متناسب با پروژه خود تغییر دهید.

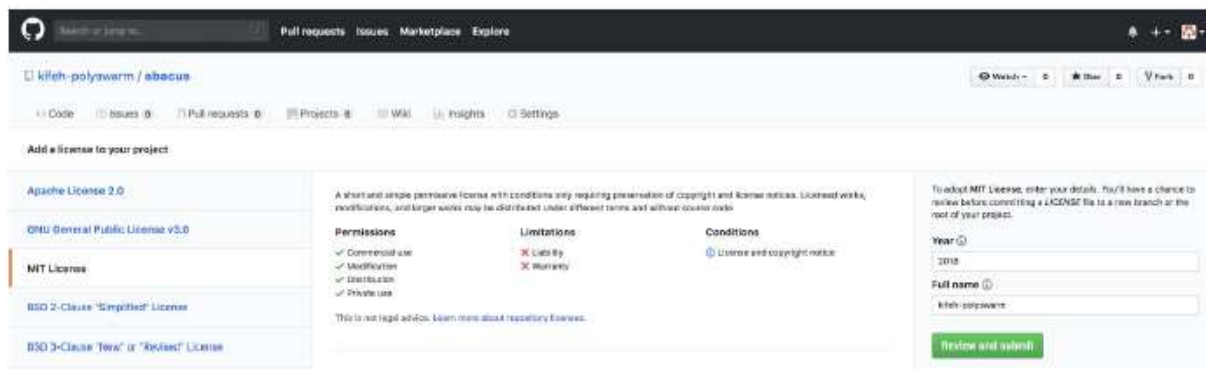
مجوز

پروانه استفاده از برنامه توسط کاربران خود با بیان تعهدات و مسئولیت های ایجاد کننده و کاربر یک برنامه نرم افزاری ، حاکم است. به عنوان مثال مجوزها برای جلوگیری از جبران خسارت یک کاربر در هنگام استفاده از یک برنامه بر خلاف شرایط استفاده از آن ، به کار می روند.

اضافه کردن مجوز

برای افزودن مجوز مناسب به سند ، این مراحل را دنبال کنید:

1. بر روی دکمه Add در سمت راست لایسنس در نمایه انجمن کلیک کنید.
2. مجوز ترجیحی را انتخاب کنید. در این حالت ، مجوز MIT را انتخاب می کنیم.
3. بر روی نقد و بررسی کلیک کنید و پس از اتمام بررسی سند ، ارسال کنید:

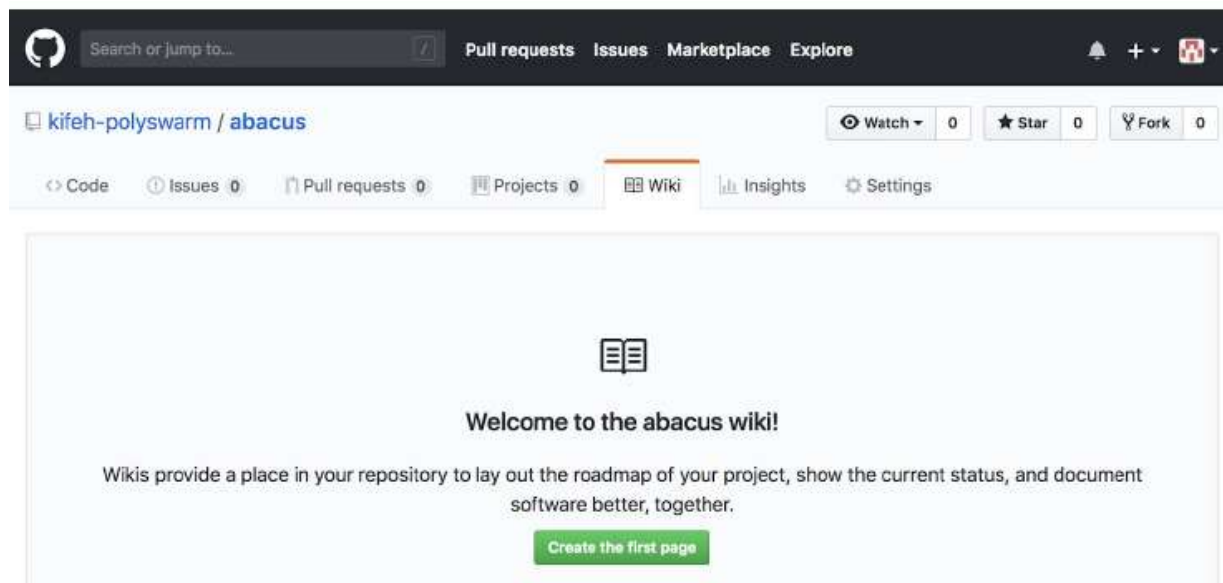


متعهد سند شوید. می توانید این متن را متناسب با پروژه خود تغییر دهید.

WIKIS و موارد

ویکی ها راهی برای مستند سازی فرآیندها و راهنمای استفاده از یک قطعه نرم افزار خاص فراهم می کند. می توانید از ویکی ها برای مستند کردن قراردادهایی استفاده کنید که باید در ایجاد شعب ، گزارش گزارش مسائل و درخواست درخواست از ویژگیها استفاده شوند.

برای دسترسی و / یا ایجاد ویکی ، از دکمه ویکی در نوار پیمایش بالا استفاده کنید:

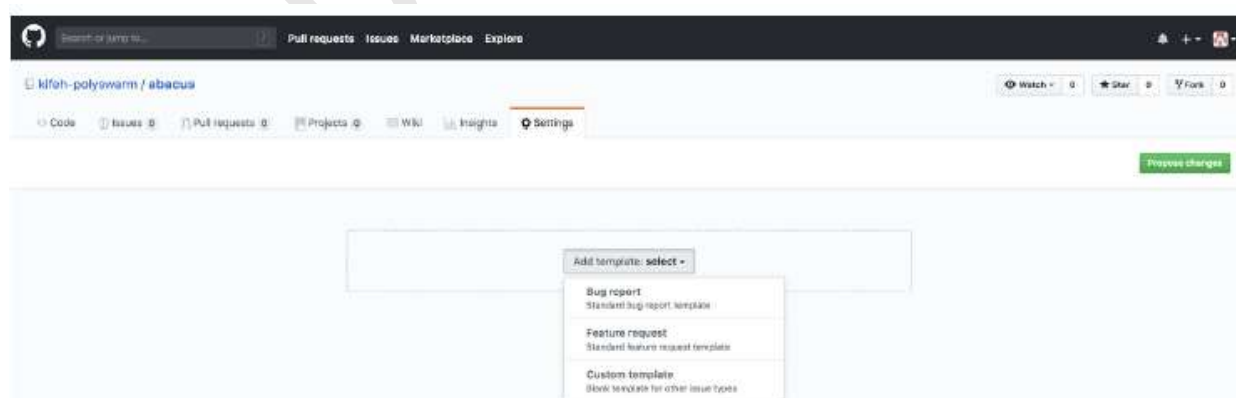


ما از ویکی ها استفاده خواهیم کرد تا بهترین شیوه ها و کنوانسیون هایی را برای برنامه هایی که در این کتاب می خواهیم بسازیم ، مستند سازیم.

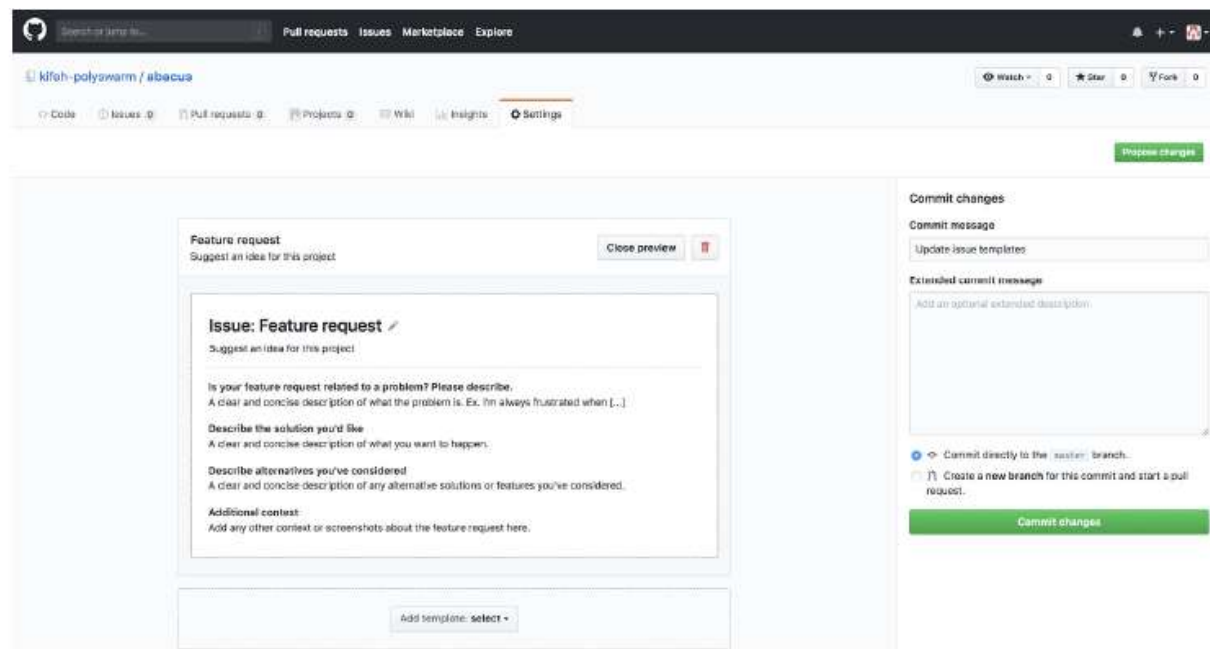
مسائل

مسائل GitHub مناسب برای گزارش چالش ها یا اشکالات موجود در هنگام استفاده از برنامه ، و همچنین در درخواست از ویژگی های مناسب است. برای پیگیری کارهایی که به آنها کمک می کند صادر می کند: لازم است که بخشی از نقشه راه محصولات به درخواست های کاربر و چالش های گزارش شده ، برای مثال ، اشکالات پاسخ دهد.

برای تسهیل در تشکیل پرونده ها ، باید الگوهای ایجاد کنیم که به سهولت روند گزارش دهی کمک می کند. شما باید بر روی دکمه Add در سمت راست از الگوهای Issues در نمایه انجمن کلیک کنید. سپس ، گزارش اشکال و الگوهای درخواست را انتخاب کنید. در آخر ، در صورت لزوم همانطور که در تصاویر زیر مشاهده می کنید ، الگوهای را مشاهده و ویرایش کنید:



Abacus Bug report

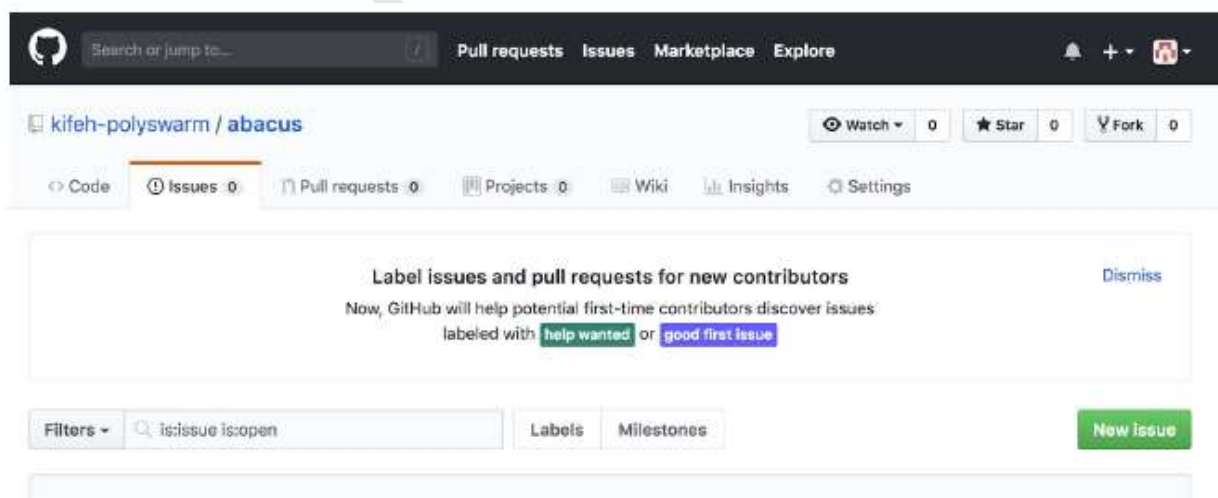


Issue: Feature request

با کلیک بر روی دکمه پیشنهاد تغییرات و commit change پس از آن برای افزودن الگو به مخزن خود ، می توانید درخواست مشکلات را ارسال کنید.

مطرح کردن شماره Raising an Issue

این کار به راحتی با انتخاب برگه Issues در نوار پیمایش بالای مخزن ذخیره می شود:



Labeling issues and pulling requests

سپس روی پرونده جدید کلیک کنید تا یک مسئله ایجاد شود. در قسمت بعدی، نوع مسئله مناسب را انتخاب کنید، به عنوان مثال گزارش اشکال. در آخر، می توانید جزئیات مربوط به مشکل خود را ارائه داده و بر روی ارسال یک شماره جدید کلیک کنید تا روند آن کامل شود، همانطور که در تصویر زیر نشان داده شده است:

The screenshot shows the GitHub interface for creating a new issue in the repository 'kifeh-polyswarm / abacus'. The 'Issues' tab is selected. The form is titled 'Issue: Bug report' and includes a prompt: 'Create a report to help us improve. If this doesn't look right, choose a different type.' The form fields include: a 'Title' input field, a 'Write' tab (with a 'Preview' tab also visible), and a large text area for the issue description. The text area contains a template with sections: '**Describe the bug**' (A clear and concise description of what the bug is.), '**To Reproduce**' (Steps to reproduce the behavior: 1. Go to '...', 2. Click on '...', 3. Scroll down to '...', 4. See error), '**Expected behavior**' (A clear and concise description of what you expected to happen.), '**Screenshots**' (If applicable, add screenshots to help explain your problem.), and '**Desktop (please complete the following information):**' (with sub-fields for OS, Browser, and Version). At the bottom of the text area, it says 'Attach files by dragging & dropping, selecting them, or pasting from the clipboard.' To the right of the form, there are sections for 'Assignees' (No one—assign yourself), 'Labels' (None yet), 'Projects' (None yet), and 'Milestone' (No milestone). At the bottom right, there are links for 'Helpful resources' (Contributing, Code of conduct) and a green 'Submit new issue' button. A footer note says 'Styling with Markdown is supported'.

Issues: Bug report

ایجاد یک نمایندگی

از شما خواسته شده است که برنامه ای بسازید که کاربر آن را قادر به تهیه غذا از یک رستوران و تحویل آن می کند. برای شروع این کار، باید در حالی که کنترل نسخه را اعمال می کنید، برنامه را بسازید. شما باید یک مخزن ایجاد کنید که میزبان برنامه باشد. این مخزن برای ردیابی تکمیل کار و استقرار برنامه استفاده خواهد شد.

برای شروع، شما باید خط فرمان Git را داشته باشید

ابزار نصب شده بر روی رایانه شما علاوه بر این ، باید در <https://github.com> یک حساب کاربری داشته باشید و در GitHub به حساب خود وارد شوید:

1. راه اندازی ترمینال.
2. دایرکتوری به نام [dinein] را برای برنامه ایجاد کنید و به فهرست dine-in بروید.
3. اولیه مخزن را شروع کنید.
- 4- به <https://github.com> بروید تا یک مخزن جدید با نام dine-in ایجاد کنید.
- 5- از HTTPS یا SSH URL مخزن از GitHub استفاده کنید.
6. URL از راه دور را روی مخزن محلی تنظیم کنید:

```
alexmagana@ALEXs-MacBook-Pro ~ % git remote add origin git@github.com:kifeh-polyswarm/dine-in.git
```

7. یک فایل README و یک پرونده gitignore ایجاد کنید ، که هر دو به فهرست اضافه می شوند.
8. پرونده ها را commit و سپس آنها را به مخزن راه دور هدایت کنید:

```
alexmagana@ALEXs-MacBook-Pro ~ % git push -u origin master
Counting objects: 4, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 304 bytes | 304.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0)
To github.com:kifeh-polyswarm/dine-in.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

نگران این موضوع نباشین ما در فصل های بعد به طور مفصل تر به این موارد خواهیم پرداخت.

خلاصه

در این فصل ، ما کنترل نسخه و انواع مختلف کار را که استفاده می شود تعریف کردیم. سپس ، ما بررسی کردیم که آن چیست و GitHub و چگونه آنها به یکدیگر ارتباط دارند. آخرین ، اما نه مهم ، ما یک مخزن در GitHub ایجاد کردیم ، آن را کlon کردیم و یک مخزن را در محلی تنظیم کردیم. قبل از بارگذاری (فشار دادن) آن به GitHub. بلوک های اساسی ساخت کنترل کنترل نسخه و متن برنامه نیز معرفی شدند. در فصلهای بعد ، در نهایت خواهید دید که چگونه مراحل ارتقاء و مراحل برای ردیابی افزایش پرونده ها را انجام دهید. شما همچنین یک درخواست جمع آوری مطرح کرده و آن را به شعبه اولیه یک مخزن ادغام می کنید.