I'm not a robot

You signed in with another tab or window. Reload to refresh your session. You signed out in another tab or window. Reload to refresh your session. You switched accounts on another tab or window. Reload to refresh your session. Dismiss alert Instantly share code, notes, and snippets. Clone this repository at <script src=" quot;></script> Save cgonzalezdai/cc33db72a6fe5178637aabb562eae35c to your computer and use it in GitHub Desktop. Clone this repository at <script src=" quot;></script> Save cgonzalezdai/cc33db72a6fe5178637aabb562eae35c to your computer and use it in GitHub Desktop. Como subir un proyecto local a github You cant perform that action at this time. If you are in the software development industry, then, most likely you have been using GitHub, a repository hosting service for Git that also has a web-based graphical interface. In todays fast-paced and collaborative world of software development, having the right tools at your disposal can make all the difference. GitHub has emerged as one of the most popular platforms for version control, code collaboration, and project management, offering a wide range of features designed to streamline the development process and foster collaboration among teams. From individual developers to large organizations, GitHub has become an indispensable tool for anyone involved in software development. GitHub can be divided into the Git and the Hub. The service includes access controls as well as collaboration features like tools for basic task management and for all projects you handle. GitHub hosts your source code projects in different programming languages and keeps track of the various changes made to every iteration. So, the Git implies the version control system; a tool that allows developers to keep track of the constant revisions to their code. The Hub is the community of like-minded individuals who participate. It is all about the collaborative effort of the community, in reviewing, improving, and deriving new ideas from the uploaded code. Its something worth looking into! In this article, well explore seven key benefits of using GitHub and how it can help you and your team work more efficiently, collaborate effectively, and build better software. Whether youre new to GitHub or looking to maximize its potential, read on to discover how this powerful platform can transform your development workflow and take your projects to the next level. 1. It makes it easy to contribute to your open-source projectsTo be honest, nearly every open-source project uses GitHub to manage its project. Using GitHub is free if your project is open source and includes a wiki and issue tracker that makes it easy to include more in-depth documentation and get feedback about your project. If you want to contribute, you just fork a project, make your changes and then send them a pull request using the GitHub web interface. 2. DocumentationBy using GitHub, youmake it easier toget excellent documentation. Their help section and guides have articles for nearly any topic related to git that you can think of. 3. Showcase your workAre you a developer and wish to attract recruiters? GitHub is the best tool you can rely on for this. Today, when searching for recruits for their project, most companies look into GitHub profiles. If your profile is available, you will have a higher chance of being recruited even if you are not from a great university or college. 4. MarkdownMarkdown allows you to use a simple text editor to write formatted documents. GitHub has revolutionized writing by channeling everything through Markdown: from the issue tracker to user comments, everything. With so many other programming languages to learn for setting up projects, its really a big benefit to have your content inputted in a format without having to learn yet another system. 5. GitHub is a repositoryThis was already mentioned before, but its important to note, GitHub is a repository. This means that it allows your work to get out there in front of the public. Moreover, GitHub is one of the largest coding communities around right now, so its wide exposure for your project. Software requirements prioritization techniques you should know6. Track changes in your code across versionsWhen multiple people collaborate on a project, its hard to keep track of revisionswho changed what, when, and where those files are stored. GitHub takes care of this problem by keeping track of all the changes that have been pushed to the repository. Much like using Microsoft Word or Google Drive, you can have a version history of your code so that previous versions are not lost with every iteration. 7. Integration optionsGitHub can integrate with common platforms such as Amazon and Google Cloud, services such as Code Climate to track your feedback, and can highlight syntax in over 200 different programming languages. In conclusion, GitHub offers a multitude of benefits that can significantly enhance the software development process for individuals and teams alike. From simplified version control and seamless collaboration to robust project management and extensive community support, GitHub provides the tools and features needed to streamline workflows, improve productivity, and foster innovation. By leveraging the power of GitHub, developers can focus more on writing code and building great software, while also benefiting from the collective knowledge and expertise of a global community. As software development continues to evolve, having a reliable and efficient platform like GitHub becomes increasingly essential for staying competitive and delivering high-quality products. If you are working on an exciting project and you need help with software development, let us know. We would be happy to know more about it! Also, if you are interested in software development best practices, we recommend you subscribe to our monthly newsletter by clicking here. CSS Grid solution to the problems of float and flexbox $ git clone into 'ticgit'...remote: Reusing existing pack: 1857, done.remote: Total 1857 (delta 0), reused 0 (delta 0)Receiving objects: 100% (1857/1857), 374.35 KiB | 268.00 KiB/s, done.Resolving deltas: 100% (772/772), done.Checking connectivity... done.$ cd ticgit$ git remoteorigin In this guide, we are going to demonstrate how to build your own GitHub App to get a fine-grained integration with GitHubs API and to act on the behalf of its users With 56 million developers on the platform and 85 million repositories, integrating with GitHub is not just fun, its essential. Some companies have even built integrations so good, that GitHub bought them and made them part of the core platform. This has happened multiple times. Even faster than that. When we reached out to GitHub for comment, Martin Woodward, Director of DevRel told us that this year they saw 60 million new repositories created. So why is it that so many of you are doing it wrong? Many of you are still using OAuth apps which are considered legacy and have scopes which are far too broad. Many more of you are taking extreme risks by using Personal Access Tokens (PATs), most of which can do anything to your account and repositories. GitHub Apps are first-class actors within GitHub and unlike the legacy OAuth apps, allow or fine-grained actions to be performed on your users repositories. In this guide, we are going to develop a GitHub App using Go, then we deploy it as a serverless function to make use of faasd which is a lightweight & portable faas engine. We are also going to do this demo on our local environment, so we should open our function which runs on our local environment to the Internet so Github can send events to our function. In order to do that we use inlets-pro which provides secure TCP/L4 tunnels. If you are a Kubernetes or K3s user and want to follow along, you can do so and switch out faasd and inletsctl, for the inlets-operator for Kubernetes and your local cluster. Everything else will work the same. A GitHub App acts on its own behalf, taking actions via the API directly using its own identity, which means you dont need to maintain a bot or service account as a separate user. GitHub Apps can be installed directly on organizations and user accounts and granted access to specific repositories. They come with built-in webhooks and narrow, specific permissions. When you set up your GitHub App, you can select the repositories you want it to access. For example, in this guide we are going to develop a Github App that respond or close the comments for your repositories that you installed Github App for. Two examples that make thorough use of GitHub Apps are below: Derek is a GitHub bot that reduces fatigue for maintainers by automating governance and delegating permissions to your team and community. It provides the following automation: Generate changelogs for releases with PRs merged and commits added, crediting everyone invokved Let designated non-admin users manage Issues and PRs by commenting Derek or /command Enforce Developer Certificate of Origin (DCO) checking (optional) Automatically label/flag PRs without a Description OpenFaaS Cloud is designed as Multi-user version of OpenFaaS with CI/CD built in and a new dashboard. It is aimed at platform engineers who want to give functions to their team, whilst shielding them from Kubernetes at the same time. With OpenFaaS Cloud functions are managed through typing git push which reduces the tooling and learning curve required to operate functions for your team. As soon as OpenFaaS Cloud receives a push event from git it will run through a build-workflow which clones your repo, builds a Docker image, pushes it to a registry and then deploys your functions to your cluster. Each user can access and monitor their functions through their personal dashboard. Pre-requisites DigitalOcean Account - We are going to use DigitalOcean as a provider to host our exit-node. arkade - arkade provides a portable marketplace for downloading your favourite devops CLIs and installing helm charts, with a single command. inletsctl - inletsctl automates the task of creating an exit-node on cloud infrastructure. inlets-pro - You can use inlets-pro to tunnel out any TCP traffic from an internal network to another network. multipass - Multipass provides a command line interface to launch, manage and generally fiddle about with instances of Linux. faas-cli - This is a CLI for use with OpenFaaS - a serverless functions framework for Docker & Kubernetes. Now we are ready to go Setup your exit-node server on DigitalOcean For this tutorial you will need to have an account and API key with one of the supported providers, or you can create an exit-server manually and install inlets PRO there yourself. For this tutorial, the DigitalOcean provider will be used. You can get free credits on DigitalOcean with this link. Create an API key in the DigitalOcean dashboard with Read and Write permissions, and download it to a file called do-access-token in your home directory or set this token as an environment variable like this: $ export INLETS_ACCESS_TOKEN="" You need to know the IP of the machine you to connect to on your local network, for instance 192.168.0.35 or 127.0.0.1 if you are running inlets PRO on the same host as SSH. You can use the inletsctl utility to provision exit-servers with inlets PRO preinstalled, it can also download the inlets-pro CLI. curl -sLSf | shsudo mv inletsctl /usr/local/bin/sudo inletsctl download --pro Finally, start your exit-node server on the fra1 region. $ inletsctl create \ --provider digitalocean \ --access-token-file ~/do-access-token \ --region lon1 \ --pro You can also change the region flag to a DigitalOcean region close to your network See the docs for inletsctl for examples of how to use other providers like AWS EC2, Azure and GCP. If everything goes well, you should see your newly created exit-node on the homepage of your the DigitalOcean account like the following: If you curious about what the exit-node or exit-server is, you can follow a link in the inlets documentation here. Create the new GitHub App on GitHub To register a new app, visit the app settings page in your GitHub profile, and click New GitHub App. Youll see a form where you can enter details about your app. See Creating a GitHub App for general information about the fields on this page. For the purposes of this guide, youll need to enter specific data in a few fields: For more details you can check it out the link The most important parts of this form are Webhook URL,Webhook Secret and Private keys: Webhook URL : You should set the IP address of the output of the command above. Webhook Secret : Define a secret for your Webhook. Private Key : Generate and download private key for your Github Application. Also, do not forget to request permissions to Read & Write the repositorys issues. Finally, we registered our application. Get up and running with your own faasd installation on your computer with multipass Multipass is a tool that not enough developers are using. Its 2020s answer to VirtualBox and Docker Desktop in one. You just run a few commands and get a Linux host with Ubuntu pre-installed. The best part is that it supports cloud-init scripts too. In order to get up and running with your own faasd installation on your Mac you can use multipass. For more details you can follow the link. Lets start our Ubuntu VM with multipass. First, we need a cloud-config.txt to set up faasd while bootstrapping VM. $ curl -sSLO \ Then, we need to update the SSH key to match your own, edit cloud-config.txt: $ ssh-keygen -t rsa -b 4096 -C "developerguyn@gmail.com" -f $PWD/id_rsa Replace the ssh_authorized_keys::ssh-rsa value with the contents of ~/.ssh/id_rsa.pub, which is defined in cloud-config.txt. Finally, boot the VM $ multipass launch \ --cloud-init cloud-config.txt \ --name faasd Note: multipass launch can be configured with more CPUs, RAM and additional disk capacity, just run --help to see how. Check the VM if it is working properly: $ multipass listName State IPv4 Imagefaasd Running 192.168.64.25 Ubuntu 20.04 LTS You can connect to the VM through SSH or via multipass exec faasd. To get the IP address simply run multipass info faasd: $ multipass info faasdName: faasdState: RunningIPv4: 192.168.64.25Release: Ubuntu 20.04.1 LTSImage hash: d68d50a4067d (Ubuntu 20.04 LTS)Load: 0.68 1.09 0.54Disk usage: 1.9G out of 4.7GMemory usage: 259.8M out of 981.4M Build and deploy a webhook receiver function For this demo, we are going to use Golang to develop a function that responds to any webhooks sent to us from the GitHub App. In order to do that, first, we need to pull the corresponding function template for the Golang. We need to install faas-cli tool for that but we need to install arkade first because arkade is the marketplace for our favourite devops CLIs. $curl -sLS | sudo sh Then, lets install our faas-cli tool. We can now find the Golang template we want and continue to create our function. # let's look at the available Go function templates within the OpenFaaS store$ faas-cli template store list | grep -i "go"go openfaas Classic Golang templategolang-http openfaas Golang HTTP templategolang-middleware openfaas Golang Middleware template# We are going to use golang-middleware function template, let's pull it.$ faas-cli template store pull golang-middleware# Then, create the function itself.$ faas-cli new issues-bot --lang golang-middleware --prefix You can find all the code details in my GitHub repository: developer-guy/faasd-github-bot. After youve created the function, you need to define a build-arg to use Go modules, an environment variable for the GitHub App ID (found in the GitHub UI) and a secret for the the webhook secret (for verifying genuine payloads) and the private key (for acting on the behalf of a user). Lets add them to the YAML file created by faas-cli new: build_args: GO111MODULE: on secrets: - webhook-secret # your secret goes here - private-key-secret # your private key goes here environment: APP_ID: "" # your app id goes here Next we need to create the above two secrets. Download the private key for the GitHub app to your host using the GitHub UI. $ export WEBHOOK_SECRET="sup3rs3cr3t"$ faas-cli secret create webhook-secret --from-literal $WEBHOOK_SECRET$ faas-cli secret create private-key-secret --from-file .pem Now its time to authenticate faas-cli so that we can do a deployment to our faasd instance from our laptop. This would work the same if you were deploying faasd to a cloud instance. # Get the IP into a variable$ export IP=$(multipass info faasd --format json| jq '.info.faasd.ipv4[0]' | tr -d '\"') # Let's capture the authentication password into a file for use with faas-cli$ ssh ubuntu@$IP "sudo cat /var/lib/faasd/secrets/basic-auth-password" > basic-auth-password # Login from your laptop (the host)$ export OPENFAAS_URL=http://$IP:8080 cat basic-auth-password | faas-cli login -s You can add the OPENFAAS_URL entry to your shells profile if you like, so you get the variable set every time you open a new terminal. Connect your Exit Node from the faasd instance We need to establish connection between our client, and the inlets-pro server in order to get events from there. $ export UPSTREAM=$IP # faasd gateway ip, we have already grap the URL above$ export PORTS=8080 # faasd gateway port$ export LICENSE="eyJhbGciOiJFUzI..." # Notice that this command is the output of the "inletsctl create" command above$ inlets-pro client --url "wss://XX.XXX.XXX.XX:8123/connect" \ --token "$TOKEN" \ --license "$LICENSE" \ --upstream $UPSTREAM \ --ports $PORTS Lets test your GitHub App In order to test it we need to install this app to selected repositories. Create a repository called test-issues-bot, then install this app for it. If you create an issue on the test-issues-bot repository, then you will see a message like this: Hello, issue opened by: developer-guy Finally, lets close the issue by typing in a command, just like how Derek works: Tear down up the resources (optional) $ multipass delete --purge faasd$ inletsctl delete --provider digitalocean --id "YOUR_INSTANCE_ID" So what next? Now that you can develop first-class integrations with GitHub, with fine-grained permissions and do things the right way. OAuth works are very broadly defined and personal access tokens (PATs) are just the wrong tool for the job and very risky if someone were to find it. Each user that installs your app is called an installation and has their own API limit of 1000 calls per hour. So what will you build? You could develop a bot, an integration, a linter as a service, integrate a machine-learning model from the OpenFaaS function store, and a whole host of other things. Its up to you to decide. Do you already have a preferred way to deploy and run HTTP servers which isnt faasd? Were OK with that and want you to know that what you learned here about inlets and GitHub Apps can apply whether you run with a Docker container or just deploy a binary directly to a server. Other GitHub Apps: Blog posts I found useful: SugerenciaTambin puedes crear un repositorio con la GitHub CLI. Para ms informacin, vea gh repo create en la documentacin de GitHub CLI.En la esquina superior derecha de cualquier pgina, selecciona y luego haz clic en Nuevo repositorio.Otra opcin para crear un repositorio con la estructura de directorios y los archivos de un repositorio existente es seleccionar el men desplegable Escoger una plantilla y hacer clic un repositorio de plantilla. Vers repositorios de plantillas que te pertenecen a ti y a las organizaciones de las que eres miembro o bien repositorios de plantillas que has usado anteriormente. Para ms informacin, consulta Crear un repositorio desde una plantilla.Si decidi elegir una plantilla, tambin puede incluir la estructura de directorios y los archivos de todas las ramas en la plantilla, y no nicamente aquellos de la rama predeterminada, seleccione Include all branches.Usa el men desplegable Propietario para seleccionar la cuenta que quieres que sea propietaria del repositorio.Teclea el nombre de tu repositorio, y una descripcin opcional.Elige la visibilidad del repositorio. Para ms informacin, consulta Acerca de los repositorios.Si no ests utilizando una plantilla, hay varios elementos opcionales que puedes pre-cargar en tu repositorio. Si ests importando un repositorio existente a GitHub, no elijas ninguna de estas opciones, ya que producirs un conflicto de fusin. Puedes agregar o crear nuevos archivos usando la interfaz de usuario o elegir agregar nuevos archivos usando luego la lnea de comando. Para ms informacin, consulta Importacin de un repositorio de Git externo mediante la lnea de comandos, Agregar un archivo a un repositorio y Cmo abordar los conflictos de combinacin. Opcionalmente, si la cuenta personal o la organizacin en la que va a crear usa cualquier GitHub Apps de GitHub Marketplace, seleccione las aplicaciones que le gustara usar en el repositorio.Haga clic en Create repository (Crear repositorio).En la parte inferior de la pgina de Configuracin rpida resultante, en "Importar el cdigo del repositorio anterior", puedes elegir importar un proyecto en tu nuevo repositorio. Para ello, haga clic en Import code. Puedes usar parmetros de consulta para rellenar previamente los campos de formulario al crear un nuevo repositorio. Los parmetros de consulta son partes opcionales de una URL que puedes personalizar para compartir una vista de pgina web especfica, como los resultados de filtro de bsqueda o una plantilla de propuestas en GitHub. Para especificar valores para los parmetros de consulta predefinidos, deben coincidir con el par clave y valor.Es posible que los campos de formulario rellenados previamente con una consulta de direccin URL sean tiles si a menudo quieres crear repositorios con la misma configuracin predeterminada. Por ejemplo, un profesor puede querer que cada alumno de una clase cree un repositorio en su cuenta personal con el mismo nombre, descripcin y visibilidad. Con una consulta de direccin URL, el profesor puede crear un vnculo que rellene previamente el nombre del repositorio, los campos de descripcin y visibilidad, y compartirlo con toda la clase.Debes tener los permisos adecuados para cualquier accin para usar el parmetro de consulta equivalente. Por ejemplo, debes tener permiso para crear un repositorio en una organizacin para especificar la organizacin como propietario del repositorio en un parmetro de consulta. Para ms informacin, consulta Roles de repositorio para una organizacin.Si creas una direccin URL no vlida mediante parmetros de consulta o si no tienes los permisos adecuados, se omitirn los parmetros de consulta no vlidos y el resto de la direccin URL funcionar de la manera habitual. Si crea una direccin URL que supere el lmite del servidor, la direccin URL devolver una pgina de error414 URI Too Long. Parmetro de consultaEjemploValores vlidosname crea un repositorio denominado "test-repo" propiedad de la organizacin "avocado-corp".Nombre del repositorio. Los espacios deben reemplazarse por + o %20.description crea un repositorio con la descripcin "Un repositorio emocionante" con visibilidad privada propiedad de @octocat.Cualquier cadena Los espacios deben reemplazarse por + o %20.visibility crea un repositorio con visibilidad privada.public privateowner crea un repositorio pblico propiedad de la organizacin "avocado-corp".Cualquier nombre de organizacin o nombre de usuario vlido. Como alternativa, mientras has iniciado sesin, utiliza @me para especificar la cuenta de usuario como propietario.template_owner y template_name crea un repositorio que pertenece a "avocado-corp" mediante la plantilla de avocado-corp "octo-repo".Nombre de usuario del propietario de la plantilla y el nombre de la plantilla del repositorio.Creacin de un repositorio desde Copilot ChatPuedes crear un repositorio desde Copilot Chat en Visual Studio Code con el Protocolo de contexto de modelo (MCP). Para ms informacin, consulta Extensin de CopilotChat con el protocolo de contexto de modelo (MCP).Informacin adicionalPara empezar, quiero informarte que este es el segundo artculo de una secuencia de tres; donde el primero es Iniciando un repositorio con Git y el tercero es Clonado un repositorio con git y GitHub.Ahora bien, primeramente debes registrar una cuenta en GitHub, si es que an no la tienes lista. As que, teniendo creada la cuenta, se proceder a iniciar sesin con usuario y contrasea.Creando repositorioSe acceder al conjunto de repositorios y se clicar sobre el boton New, como se muestra en la imagen:Luego se colocar el nombre del repositorio preferiblemente con el mismo nombre del repositorio local, en este caso "portafolio" para evitar inconvenientes a la hora de enviar los archivos.Y por ltimo se da clic en el boton create repositoryUna vez creado el repositorio remoto, inicialmente se encuentra vaco, en otras palabras, sin archivos y sin carpetas, por lo que se copiar la primera lnea de comando proporcionada por el propio GitHub; tal como se muestra en la imagen siguiente:ste comando permitir crear una direccin del repositorio remoto en la mquina local, siendo ejecutado a travs del terminal y tendr la siguiente estructura :$ git remote add origin :TuUsuarioDeGitHub/NombreRepositorio.gitPosteriormente sern ejecutadas otras lneas de comando para visualizar el nombre y la direccin del repositorio remoto a donde sern enviados los archivos existentes del repositorio local. Estos comandos son descritos con ms detalles en el punto abajo.Enviando datos del repositorio local al repositorio remoto en GitHub a travs de lneas de comandoDentro de este marco, se crear una conexin entre el repositorio local y el remoto. Para ello se abrir el terminal Git Bash, donde se aplicarn los siguientes pasos:Primer paso: Conectando el repositorio remoto con el repositorio local.En la lnea de comando del terminal, se pegar la instruccin que fue copiada en el punto arriba:Para verificar que el repositorio fue adicionado correctamente, debe visualizarse el nombre y la direccin del repositorio remoto al ser ejecutados los siguientes comandos:$ git remote /***Muestra el nombre del repositorio remoto***/$ git remote -V /***Muestra ms detalles del repositorio remoto***/Segundo paso: Enviando archivos desde el repositorio local al remoto.Se empujar todos los archivos desde el repositorio local para el repositorio remoto con la siguiente lnea de comando:git push -u origin mainSeguidamente se comprobar en el repositorio remoto de GitHub si estn apareciendo los archivos que fueron enviados o empujados para la branch main desde la mquina local, recordando que el repositorio remoto estaba vaco. Por lo tanto, se clicar directamente sobre el repositorio portafolio donde se visualiza todos los archivos siendo actualizados:Observemos detalladamente el ltimo resultado del repositorio remoto:Finalmente, en esta pantalla se puede ver que los archivos enviados desde el repositorio local, fueron recibidos con xito en el repositorio remoto.Importancia de tener un archivo readme.mdPor lo general en todo proyecto bien hecho en GitHub vers un archivo con el nombre readme.md, que contiene informacin relevante del mismo. Esta extensin .md (Markdown), es un lenguaje de marcado de texto parecido al html, con la diferencia de que es mucho ms sencillo.Ahora, por qu razn tendra que tener este archivo en mi repositorio? Bien! hay muchas razones por la que deberas tenerlo, sin embargo, solo te mencionar algunas: en este archivo podrias llevar un registro sobre el contenido de tu proyecto, como por ejemplo: cul es el problema y como fue resuelto, las tecnologas o lenguajes utilizados, como debera ser su uso para que funciones correctamente, etc. Hay muchas cosas que podras especificar aqu, con el fin de dejar el proyecto bien comprensible y organizado, para quien lo est mirando.Cabe resaltar, que teniendo ya tu proyecto listo en el repositorio remoto, puedes hacer una publicacin de tu pgina totalmente gratuita a traves de GitHub Pages y linkarla en tu archivo readme.md.Como recurso final coloco a disposicin las siguientes imagenes que ensean a como crear el archivo readme.md y como hacer este enlace a la pgina web:Una vez dentro del repositorio, donde se encuentra publicada la pgina deseada, se clicar sobre Settings, luego se deslizar hacia abajo hasta hallar la opcin Pages, y clicando sobre ella, se copiar la *URL (* proporcionada por el propio GitHub Pages, tal como se muestra en la imagen siguiente:Ahora se presiona sobre la opcin code; y en la parte inferior derecha vers un botn escrito Add a Readme, clicando en l, se abrira el editor de GitHub para ste archivo, donde se escribir el siguiente cdigo:1 # portafolio2 3