



HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.

Git para toda plataforma

<http://git-scm.com>

CONFIGURAR HERRAMIENTAS

Configura la información del usuario para todos los repositorios locales

\$ git config --global user.name "[name]"

Establece el nombre que desea esté anexo a sus transacciones de commit

\$ git config --global user.email "[email address]"

Establece el e-mail que desea esté anexo a sus transacciones de commit

\$ git config --global color.ui auto

Habilita la útil colorización del producto de la línea de comando

CREAR REPOSITORIOS

Inicia un nuevo repositorio u obtiene uno de una URL existente

\$ git init [project-name]

Crea un nuevo repositorio local con el nombre especificado

\$ git clone [url]

Descarga un proyecto y toda su historia de versión

EFFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

\$ git status

Enumera todos los archivos nuevos o modificados que se deben confirmar

\$ git diff

Muestra las diferencias de archivos que no se han enviado aún al área de espera

\$ git add [file]

Toma una instantánea del archivo para preparar la versión

\$ git diff --staged

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

\$ git reset [file]

Mueve el archivo del área de espera, pero preserva su contenido

\$ git commit -m "[descriptive message]"

Registra las instantáneas del archivo permanentemente en el historial de versión

CAMBIOS GRUPALES

Nombra una serie de commits y combina esfuerzos ya culminados

\$ git branch

Enumera todas las ramas en el repositorio actual

\$ git branch [branch-name]

Crea una nueva rama

\$ git checkout [branch-name]

Cambia a la rama especificada y actualiza el directorio activo

\$ git merge [branch]

Combina el historial de la rama especificada con la rama actual

\$ git branch -d [branch-name]

Borra la rama especificada



HOJA DE REFERENCIA PARA GITHUB GIT

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN

Reubica y retira los archivos con versión

\$ git rm [file]

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

\$ git rm --cached [file]

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

\$ git mv [file-original] [file-renamed]

Cambia el nombre del archivo y lo prepara para commit

SUPRIMIR TRACKING

Excluye los archivos temporales y las rutas

***.log
build/
temp-***

Un archivo de texto llamado `.gitignore` suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

\$ git ls-files --other --ignored --exclude-standard

Enumera todos los archivos ignorados en este proyecto

GUARDAR FRAGMENTOS

Almacena y restaura cambios incompletos

\$ git stash

Almacena temporalmente todos los archivos tracked modificados

\$ git stash pop

Restaura los archivos guardados más recientemente

\$ git stash list

Enumera todos los sets de cambios en guardado rápido

\$ git stash drop

Elimina el set de cambios en guardado rápido más reciente

REPASAR HISTORIAL

Navega e inspecciona la evolución de los archivos de proyecto

\$ git log

Enumera el historial de la versión para la rama actual

\$ git log --follow [file]

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

\$ git diff [first-branch]...[second-branch]

Muestra las diferencias de contenido entre dos ramas

\$ git show [commit]

Produce metadatos y cambios de contenido del commit especificado

REHACER COMMITS

Borra errores y elabora historial de reemplazo

\$ git reset [commit]

Deshace todos los commits después de [commit], preservando los cambios localmente

\$ git reset --hard [commit]

Desecha todo el historial y regresa al commit especificado

SINCRONIZAR CAMBIOS

Registrar un marcador de repositorio e intercambiar historial de versión

\$ git fetch [bookmark]

Descarga todo el historial del marcador del repositorio

\$ git merge [bookmark]/[branch]

Combina la rama del marcador con la rama local actual

\$ git push [alias] [branch]

Carga todos los commits de la rama local al GitHub

\$ git pull

Descarga el historial del marcador e incorpora cambios

GitHub Training

Obtenga más información sobre el uso de GitHub y Git. Envíe un e-mail al Equipo de Entrenadores o visite nuestro sitio web para informarse sobre los horarios de eventos y la disponibilidad de clases privadas.

✉ training@github.com

☞ training.github.com



HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.

Git para toda plataforma

<http://git-scm.com>

CONFIGURAR HERRAMIENTAS

Configura la información del usuario para todos los repositorios locales

\$ git config --global user.name "[name]"

Establece el nombre que desea esté anexo a sus transacciones de commit

\$ git config --global user.email "[email address]"

Establece el e-mail que desea esté anexo a sus transacciones de commit

\$ git config --global color.ui auto

Habilita la útil colorización del producto de la línea de comando

CREAR REPOSITORIOS

Inicia un nuevo repositorio u obtiene uno de una URL existente

\$ git init [project-name]

Crea un nuevo repositorio local con el nombre especificado

\$ git clone [url]

Descarga un proyecto y toda su historia de versión

EFFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

\$ git status

Enumera todos los archivos nuevos o modificados que se deben confirmar

\$ git diff

Muestra las diferencias de archivos que no se han enviado aún al área de espera

\$ git add [file]

Toma una instantánea del archivo para preparar la versión

\$ git diff --staged

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

\$ git reset [file]

Mueve el archivo del área de espera, pero preserva su contenido

\$ git commit -m "[descriptive message]"

Registra las instantáneas del archivo permanentemente en el historial de versión

CAMBIOS GRUPALES

Nombra una serie de commits y combina esfuerzos ya culminados

\$ git branch

Enumera todas las ramas en el repositorio actual

\$ git branch [branch-name]

Crea una nueva rama

\$ git checkout [branch-name]

Cambia a la rama especificada y actualiza el directorio activo

\$ git merge [branch]

Combina el historial de la rama especificada con la rama actual

\$ git branch -d [branch-name]

Borra la rama especificada



HOJA DE REFERENCIA PARA GITHUB GIT

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN

Reubica y retira los archivos con versión

\$ git rm [file]

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

\$ git rm --cached [file]

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

\$ git mv [file-original] [file-renamed]

Cambia el nombre del archivo y lo prepara para commit

SUPRIMIR TRACKING

Excluye los archivos temporales y las rutas

***.log
build/
temp-***

Un archivo de texto llamado `.gitignore` suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

\$ git ls-files --other --ignored --exclude-standard

Enumera todos los archivos ignorados en este proyecto

GUARDAR FRAGMENTOS

Almacena y restaura cambios incompletos

\$ git stash

Almacena temporalmente todos los archivos tracked modificados

\$ git stash pop

Restaura los archivos guardados más recientemente

\$ git stash list

Enumera todos los sets de cambios en guardado rápido

\$ git stash drop

Elimina el set de cambios en guardado rápido más reciente

REPASAR HISTORIAL

Navega e inspecciona la evolución de los archivos de proyecto

\$ git log

Enumera el historial de la versión para la rama actual

\$ git log --follow [file]

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

\$ git diff [first-branch]...[second-branch]

Muestra las diferencias de contenido entre dos ramas

\$ git show [commit]

Produce metadatos y cambios de contenido del commit especificado

REHACER COMMITS

Borra errores y elabora historial de reemplazo

\$ git reset [commit]

Deshace todos los commits después de [commit], preservando los cambios localmente

\$ git reset --hard [commit]

Desecha todo el historial y regresa al commit especificado

SINCRONIZAR CAMBIOS

Registrar un marcador de repositorio e intercambiar historial de versión

\$ git fetch [bookmark]

Descarga todo el historial del marcador del repositorio

\$ git merge [bookmark]/[branch]

Combina la rama del marcador con la rama local actual

\$ git push [alias] [branch]

Carga todos los commits de la rama local al GitHub

\$ git pull

Descarga el historial del marcador e incorpora cambios

GitHub Training

Obtenga más información sobre el uso de GitHub y Git. Envíe un e-mail al Equipo de Entrenadores o visite nuestro sitio web para informarse sobre los horarios de eventos y la disponibilidad de clases privadas.

✉ training@github.com

☞ training.github.com



HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.

Git para toda plataforma

<http://git-scm.com>

CONFIGURAR HERRAMIENTAS

Configura la información del usuario para todos los repositorios locales

\$ git config --global user.name "[name]"

Establece el nombre que desea esté anexo a sus transacciones de commit

\$ git config --global user.email "[email address]"

Establece el e-mail que desea esté anexo a sus transacciones de commit

\$ git config --global color.ui auto

Habilita la útil colorización del producto de la línea de comando

CREAR REPOSITORIOS

Inicia un nuevo repositorio u obtiene uno de una URL existente

\$ git init [project-name]

Crea un nuevo repositorio local con el nombre especificado

\$ git clone [url]

Descarga un proyecto y toda su historia de versión

EFFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

\$ git status

Enumera todos los archivos nuevos o modificados que se deben confirmar

\$ git diff

Muestra las diferencias de archivos que no se han enviado aún al área de espera

\$ git add [file]

Toma una instantánea del archivo para preparar la versión

\$ git diff --staged

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

\$ git reset [file]

Mueve el archivo del área de espera, pero preserva su contenido

\$ git commit -m "[descriptive message]"

Registra las instantáneas del archivo permanentemente en el historial de versión

CAMBIOS GRUPALES

Nombra una serie de commits y combina esfuerzos ya culminados

\$ git branch

Enumera todas las ramas en el repositorio actual

\$ git branch [branch-name]

Crea una nueva rama

\$ git checkout [branch-name]

Cambia a la rama especificada y actualiza el directorio activo

\$ git merge [branch]

Combina el historial de la rama especificada con la rama actual

\$ git branch -d [branch-name]

Borra la rama especificada



HOJA DE REFERENCIA PARA GITHUB GIT

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN

Reubica y retira los archivos con versión

\$ git rm [file]

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

\$ git rm --cached [file]

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

\$ git mv [file-original] [file-renamed]

Cambia el nombre del archivo y lo prepara para commit

SUPRIMIR TRACKING

Excluye los archivos temporales y las rutas

***.log
build/
temp-***

Un archivo de texto llamado `.gitignore` suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

\$ git ls-files --other --ignored --exclude-standard

Enumera todos los archivos ignorados en este proyecto

GUARDAR FRAGMENTOS

Almacena y restaura cambios incompletos

\$ git stash

Almacena temporalmente todos los archivos tracked modificados

\$ git stash pop

Restaura los archivos guardados más recientemente

\$ git stash list

Enumera todos los sets de cambios en guardado rápido

\$ git stash drop

Elimina el set de cambios en guardado rápido más reciente

REPASAR HISTORIAL

Navega e inspecciona la evolución de los archivos de proyecto

\$ git log

Enumera el historial de la versión para la rama actual

\$ git log --follow [file]

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

\$ git diff [first-branch]...[second-branch]

Muestra las diferencias de contenido entre dos ramas

\$ git show [commit]

Produce metadatos y cambios de contenido del commit especificado

REHACER COMMITS

Borra errores y elabora historial de reemplazo

\$ git reset [commit]

Deshace todos los commits después de [commit], preservando los cambios localmente

\$ git reset --hard [commit]

Desecha todo el historial y regresa al commit especificado

SINCRONIZAR CAMBIOS

Registrar un marcador de repositorio e intercambiar historial de versión

\$ git fetch [bookmark]

Descarga todo el historial del marcador del repositorio

\$ git merge [bookmark]/[branch]

Combina la rama del marcador con la rama local actual

\$ git push [alias] [branch]

Carga todos los commits de la rama local al GitHub

\$ git pull

Descarga el historial del marcador e incorpora cambios

GitHub Training

Obtenga más información sobre el uso de GitHub y Git. Envíe un e-mail al Equipo de Entrenadores o visite nuestro sitio web para informarse sobre los horarios de eventos y la disponibilidad de clases privadas.

✉ training@github.com

☞ training.github.com



HOJA DE REFERENCIA PARA GITHUB GIT

Git es el sistema de control de versiones distribuido de fuente abierta que facilita las actividades de GitHub en su computadora portátil o de escritorio. Esta hoja de referencia rápida resume las instrucciones de las líneas de comando de Git más comúnmente usadas.

INSTALAR GIT

GitHub le ofrece a los clientes de computadoras de escritorio que incluye una interfaz gráfica de usuario para las acciones de repositorio más comunes y una edición de línea de comando de actualización automática de Git para escenarios avanzados.

GitHub para Windows

<https://windows.github.com>

GitHub para Mac

<https://mac.github.com>

Hay distribuciones de Git para sistemas Linux y POSIX en el sitio web oficial Git SCM.

Git para toda plataforma

<http://git-scm.com>

CONFIGURAR HERRAMIENTAS

Configura la información del usuario para todos los repositorios locales

\$ git config --global user.name "[name]"

Establece el nombre que desea esté anexo a sus transacciones de commit

\$ git config --global user.email "[email address]"

Establece el e-mail que desea esté anexo a sus transacciones de commit

\$ git config --global color.ui auto

Habilita la útil colorización del producto de la línea de comando

CREAR REPOSITORIOS

Inicia un nuevo repositorio u obtiene uno de una URL existente

\$ git init [project-name]

Crea un nuevo repositorio local con el nombre especificado

\$ git clone [url]

Descarga un proyecto y toda su historia de versión

EFFECTUAR CAMBIOS

Revisa las ediciones y elabora una transacción de commit

\$ git status

Enumera todos los archivos nuevos o modificados que se deben confirmar

\$ git diff

Muestra las diferencias de archivos que no se han enviado aún al área de espera

\$ git add [file]

Toma una instantánea del archivo para preparar la versión

\$ git diff --staged

Muestra las diferencias del archivo entre el área de espera y la última versión del archivo

\$ git reset [file]

Mueve el archivo del área de espera, pero preserva su contenido

\$ git commit -m "[descriptive message]"

Registra las instantáneas del archivo permanentemente en el historial de versión

CAMBIOS GRUPALES

Nombra una serie de commits y combina esfuerzos ya culminados

\$ git branch

Enumera todas las ramas en el repositorio actual

\$ git branch [branch-name]

Crea una nueva rama

\$ git checkout [branch-name]

Cambia a la rama especificada y actualiza el directorio activo

\$ git merge [branch]

Combina el historial de la rama especificada con la rama actual

\$ git branch -d [branch-name]

Borra la rama especificada



HOJA DE REFERENCIA PARA GITHUB GIT

NOMBRES DEL ARCHIVO DE REFACTORIZACIÓN

Reubica y retira los archivos con versión

\$ git rm [file]

Borra el archivo del directorio activo y pone en el área de espera el archivo borrado

\$ git rm --cached [file]

Retira el archivo del control de versiones, pero preserva el archivo a nivel local

\$ git mv [file-original] [file-renamed]

Cambia el nombre del archivo y lo prepara para commit

SUPRIMIR TRACKING

Excluye los archivos temporales y las rutas

***.log
build/
temp-***

Un archivo de texto llamado `.gitignore` suprime la creación accidental de versiones de archivos y rutas que concuerdan con los patrones especificados

\$ git ls-files --other --ignored --exclude-standard

Enumera todos los archivos ignorados en este proyecto

GUARDAR FRAGMENTOS

Almacena y restaura cambios incompletos

\$ git stash

Almacena temporalmente todos los archivos tracked modificados

\$ git stash pop

Restaura los archivos guardados más recientemente

\$ git stash list

Enumera todos los sets de cambios en guardado rápido

\$ git stash drop

Elimina el set de cambios en guardado rápido más reciente

REPASAR HISTORIAL

Navega e inspecciona la evolución de los archivos de proyecto

\$ git log

Enumera el historial de la versión para la rama actual

\$ git log --follow [file]

Enumera el historial de versión para el archivo, incluidos los cambios de nombre

\$ git diff [first-branch]...[second-branch]

Muestra las diferencias de contenido entre dos ramas

\$ git show [commit]

Produce metadatos y cambios de contenido del commit especificado

REHACER COMMITS

Borra errores y elabora historial de reemplazo

\$ git reset [commit]

Deshace todos los commits después de [commit], preservando los cambios localmente

\$ git reset --hard [commit]

Desecha todo el historial y regresa al commit especificado

SINCRONIZAR CAMBIOS

Registrar un marcador de repositorio e intercambiar historial de versión

\$ git fetch [bookmark]

Descarga todo el historial del marcador del repositorio

\$ git merge [bookmark]/[branch]

Combina la rama del marcador con la rama local actual

\$ git push [alias] [branch]

Carga todos los commits de la rama local al GitHub

\$ git pull

Descarga el historial del marcador e incorpora cambios

GitHub Training

Obtenga más información sobre el uso de GitHub y Git. Envíe un e-mail al Equipo de Entrenadores o visite nuestro sitio web para informarse sobre los horarios de eventos y la disponibilidad de clases privadas.

✉ training@github.com

☞ training.github.com