# 

# Regelverk för Hajk

Innehåll

[Inledning 3](#_Toc466977776)

[Få inbjudan till GitHub och Slack 3](#_Toc466977777)

[Git 4](#_Toc466977778)

[Github 4](#_Toc466977779)

[Versionshantering 4](#_Toc466977780)

[Branches 5](#_Toc466977781)

[Kort guide för riktlinjer 5](#_Toc466977782)

[Kopiera kod för lokal utveckling 6](#_Toc466977783)

[Skapa ny gren 7](#_Toc466977784)

[Gör en commit 8](#_Toc466977785)

[Gör en Pull request 9](#_Toc466977786)

# Inledning

Hajk är ett projekt som drivs i ett samarbete mellan olika kommuner och är avsett att kunna återanvändas för generella GIS-applikationer för webb. Användningsområdet varierar för de olika kommunerna där vissa använder Hajk enbart för internt bruk och vissa även tillåter extern användning av karttjänsten. I skrivande stund är följande kommuner en del av samarbetet:

* Ale kommun
* Alingsås kommun
* Göteborgs Stad (administratör på GitHub)
* Halmstads kommun
* Jönköping kommun
* Kungsbacka kommun (administratör på GitHub)
* Lilla Edets kommun
* Uddevalla kommun
* Varbergs kommun
* Värnamo kommun

Förutom direktkorrespondens via mail så sker kommunikationen kommuner emellan med Slack som är ett chattprogram för projektsamarbeten. Här förs diskussioner om allt som berör Hajk och kommunsamarbetet. Innan man lägger en beställning av utveckling hos exempelvis en konsult, är det bra att ha en avstämning med andra kommuner för att höra om behovet finns på andra ställen och i så fall tillsammans kunna gå ihop i beställningen. Det kan även vara bra att ta reda på för att veta ifall en annan kommun redan lagt en liknande beställning.

# Få inbjudan till GitHub och Slack

Vi använder GitHub för att samordna kod för projektet och för att få en gemensam struktur kring arbetet. För att ansöka om medlemskap i Hajk2-samarbetet på GitHub kontaktar du vår kontaktperson Malsor Limani på [malsor.limani@jonkoping.se](mailto:malsor.limani@jonkoping.se).

Har du varit i kontakt med Malsor och fått ett godkännande att gå med i samarbetet bör du inom kort få en inbjudan till projektet på GitHub. Du bör även ha blivit inbjuden till gruppen på Slack (<https://hajk.slack.com>).

## Git

Git är ett versionshanteringssystem som används för att hantera och lagra olika versioner av kod. Fördelen är att det är möjligt att gå tillbaka i historiken och återställa till en tidigare version. Ett exempel på när detta skulle vara användbart är om en utvecklare laddar upp sin kod och i efterhand märker att den skapat en bugg. Vad utvecklaren gör då är att helt enkelt återställa koden till en tidigare, stabil version.

## Github

GitHub är en lagringstjänst för Git som används inom projektsamarbeten för främst källkodsprojekt. I praktiken innebär det att GitHub är ett webbgränssnitt som lagrar *Git-repositories* på sina servrar. Det GitHub erbjuder är alltså ett användarvänligt gränssnitt för att skapa projekt, ladda upp/hämta kod, skapa nya *issues* med mera.

## Versionshantering

Versionshantering innebär att det finns en inbyggd hantering i Git för att lagra olika versioner av ett projekt med syftet att kunna gå tillbaka till en tidigare version. Vill du som utvecklare göra en förändring i projektet så kopierar du hela projektkatalogen till ditt egna system och sätter upp projektet i en lokal utvecklingsmiljö på din dator. Dina ändringar på din lokala kopia kommer inte påverka något annat projekt, så det finns ingen risk att råka skriva över någon annans kod. När du gjort dina ändringar och testat att det är buggfritt, så kan du ”checka in” ändringarna till att bli en del av projektets huvudkod.

Branches  
Att arbeta med grenar handlar om att arbeta med olika versioner av ett projekt. I grunden består ett projekt av en huvudgren kallad *master*. Det är på denna gren själva kodbasen ligger. En gren skapas ifrån *master* som är en kopia med alla filer och mappar ifrån *master*. Den nyskapade grenen används till att utveckla ny kod eller experimentera på, utan att den påverka koden på huvudgrenen.

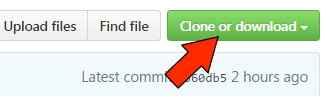
Bilden nedan illustrerar flödet:

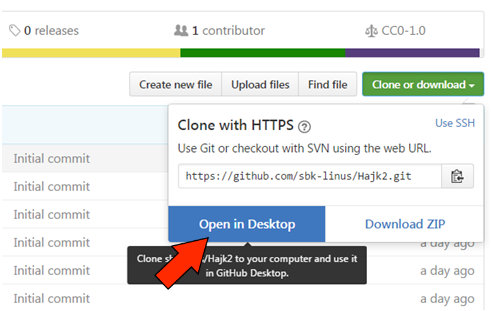
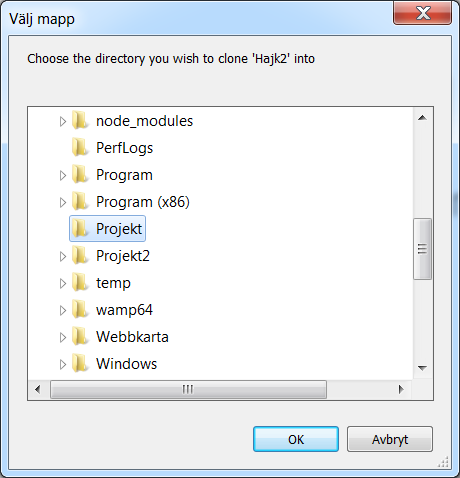
* Huvudgrenen *(master)*
* Grenen *feature* skapas ifrån *master*
* Resan *feature* gör innan den laddas upp till *master*

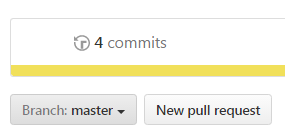
**Namngivning av ny gren**Viktigt med bra namngivning

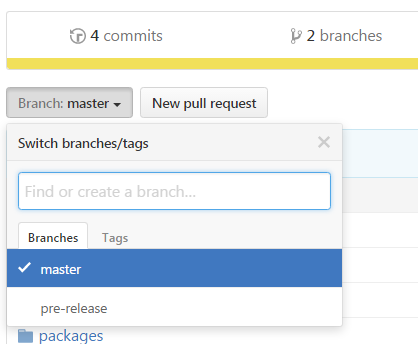
Kort guide för riktlinjer1. Klona koden ifrån <https://github.com/hajkmap/hajk>   
2. Skapa en **branch** ifrån ***master*** och namnge den med en beskrivning för vad det är som utvecklas och vilken kommun som gör det, ex: **feature-drawpanel/gbg**3. Utveckla och arbeta lokalt med koden  
4. Gör en **commit** med dina ändringar till din branch  
5. Öppna en **Pull request** ifrån din branch till branchen ***pre-release***6. Om inga konflikter förekommer kan du göra en **merge** av din **Pull request**7. Avsluta med att **radera** branchen efter din merge

Kopiera kod för lokal utveckling  
Det finns en rad olika verktyg för att använda Git och GitHub, men i denna guide används gratisprogramvaran GitHub Desktop som är ett grafiskt gränssnitt för att ladda ner, ladda upp och hantera koden.

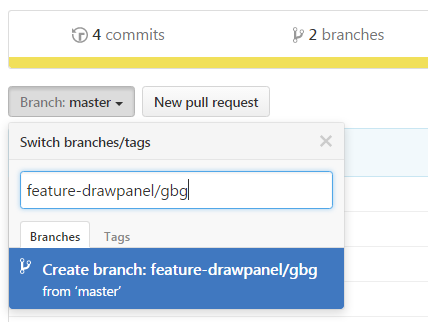
1. Ladda ner GitHub Desktop till Windows (<https://desktop.github.com/>)  
2. Se till att du är inloggad på GitHubs hemsida och logga därefter in med samma konto på GitHub Desktop.  
3. Navigera till projektsidan på GitHub ([https://github.com/hajkmap/hajk](https://github.com/hajkmap/hajk2)) och klicka på **Clone or download**

4. Klicka på **Open in Desktop**, projektet klonas och öppnas i programmet GitHub Desktop  
  
6. Välj en mapp där du vill att projektet ska lagras  


Skapa ny gren1. Gå till projektet på [https://github.com/hajkmap/hajk](https://github.com/hajkmap/hajk2)  
2. Klicka på rullgardinsmenyn till vänster där det står **Branch: master**  
3. Se till så att grenen **master** är vald i listan.



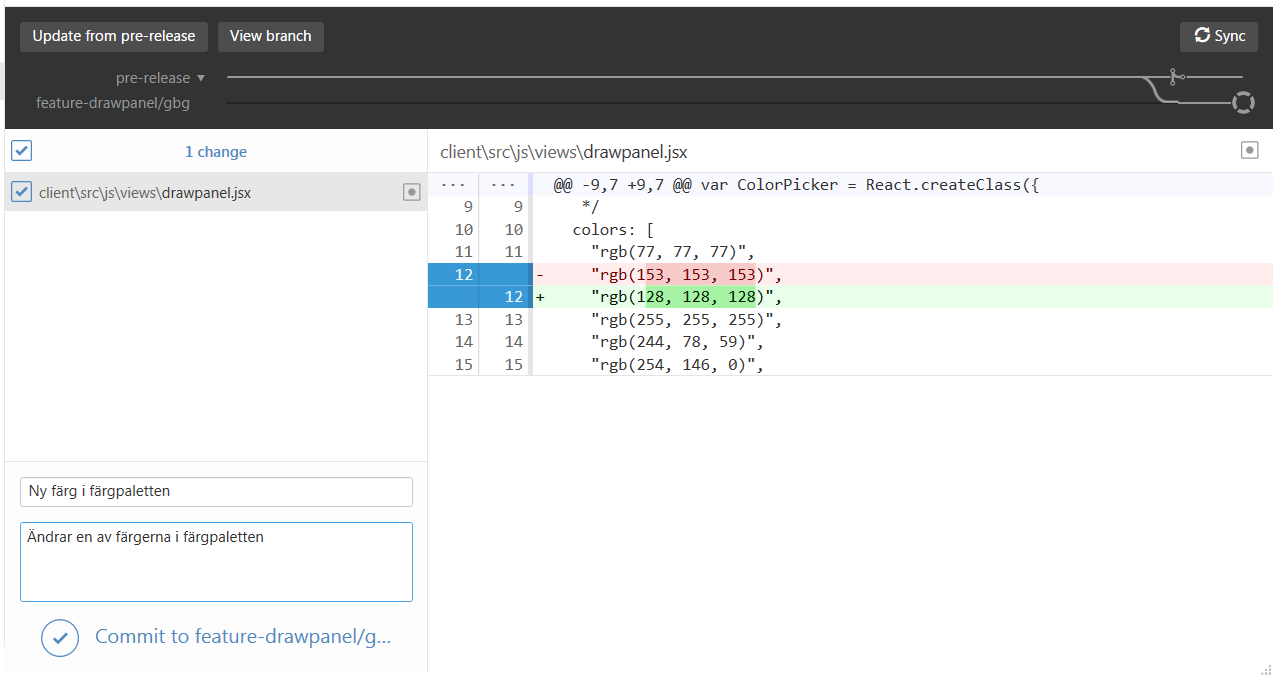
4. Ange namnet för grenen och klicka på den blåa knappen med texten **Create branch:...**



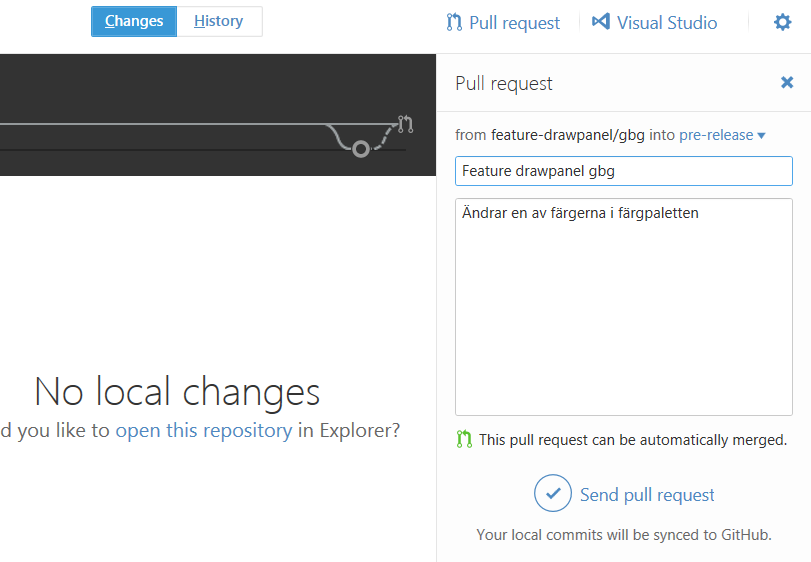
Gör en commitNär du ändrar någonting i koden så uppdateras GitHub Desktop och talar om hur tidigare kod såg ut (rödmarkerad text) och hur den nya koden ser ut (grönmarkerad text).

Första gången du gör en ändring måste du publicera din lokala gren till fjärrarkivet. Detta gör du genom att klicka på **Publish**.  


Varje gång du gör ändringar till det lokala projektet, måste du synka ändringarna genom att trycka på knappen **Sync** i vänstra hörnet. De blir då synbara online på GitHub för andra.

Gör en commit till din branch och ange en rubrik som beskriver vilka förbättringar du gjort och en beskrivande text som är mer djupgående än rubriken. Klicka på **Commit to...**  


Gör en Pull request  
Det sista steget handlar om att dina ändringar laddas upp till pre-release. Du gör en Pull request, vilket innebär att gör en förfrågan om att göra en sammanslagning av din branch till pre-release.

Rubrik och text kommer att anges automatiskt utifrån det du angett i din **commit**, men det går bra att ändra texterna.

Klicka på knappen **Merge pull request** för att slå samman grenarna.

