التطبيقات الموزعة – جلسة 1 – مقدمة عن Git

أهداف الجلسة:

تهدف هذه الجلسة إلى تعريف الطلاب بنظم التحكم بالنسخ Version Control Systems بشكل عام و Git بشكل خاص.

من المتوقع في نهاسة الجلسة أن يكون الطالب قادراً على:

- تعريف نظم التحكم بالنسخ و أنواعها.
- فوائد نظم التحكم بالنسخ.
- مقدمة حول Git و آلية تخزين النسخ ضمنه.
- تعريف أقسم مشروع Git و حالات الملفات المختلفة في Git.
- تنصيب Git على جهازه و استخدام الأوامر الأساسية التالية init, status, add, commit, log.
- مقدمة حول github, إنشاء حساب عليه و إنشاء مشروع مع إضافة زوج مفاتيح SSH.
- تنزيل مشروع من github, باستخدام الأمر clone, رفع تغييرات عليه باستخدام push و تنزيل تغييرات منه باستخدام pull.

تعريف نظم التحكم بالنسخ VCS

تخزن هذه النظم حالة الملفات و التغييرات التي تطرأ عليها ضمن قاعدة بيانات خاصة, بحيث تمكن المطورين من العودة إلى أي حالة سابقة في أي لحظة و حفظ تاريخ التغييرات التي تطرأ على الملفات لديهم مما يساعدهم أكثر في فهم هذه التغييرات في المستقبل, تعتبر هذه النظم أساسية للتعاون بين المطورين كما سنرى بعد قليل.

أنواع نظم التحكم بالنسخ

تقسم هذه النظم لثلاثة أنواع هي

1. النظم المحلية: يتم فيها تخزين الملفات التي يعمل عليها المطورون و قاعدة البيانات التي تتضمن التغييرات على هذه الملفات ضمن جهاز المستخدم بشكل محلي.
2. النظم المركزية: يتم فيها تخزين الملفات التي يعمل عليها المطورون ضمن جهاز المستخدم بشكل محلي و قاعدة البيانات التي تتضمن التغييرات عليها ضمن مخدم مركزي.
3. النظم الموزعة: يتم فيها تخزين الملفات ضمن جاهز كل مستخدم و قاعدة البيانات التي تتضمن الغييرات تخزن أيضاً ضمن جهاز كل مستخدم بالإضافة لمخدم مركزي, يندرج Git ضمن هذا النوع.

الصور التالية توضح الأنواع الثلاثة لنظم التحكم بالنسخ

فوائد نظم التحكم بالنسخ

إن استخدام نظم التحكم بالنسخ لمتابعة التغييرات التي تجري على الملفات له الفوائد التالية

- تسهيل التعاون و مشاركة الملفات بين المطورين المختلفين, من دون هذه النظم من الصعب مشاركة الملفات و التعديلات التي تجري عليها بين عدد كبير من المطورين, لكن بوجودها نستطيع المشاركة بسهولة و دمج هذه الملفات و التغييرات التي تجري عليها بسهولة كبيرة و من دون أخطاء تقريباً.
- نسخ الملفات إحتياطياً: بما أن الملفات تكون موجودة عند كل المستخدمين و قاعدة البيانات عندهم أيضاً و على المخدم المركزي فإن جميع الملفات تكون محفوظة في عدة أماكن و في حال فشل أو فقدان الملفات من عند أي مستخدم أو ضمن المخدم المركزي نستطيع استخدام أي نسخة أخرى للإستعادة عند الحاجة.
- إعادة المشروع إلى أي حالة سابقة, أحياناً نحتاج أن نعيد جميع الملفات كما كانت في وقت مضى كي نقوم بالتعديل عليها أو استخدامها كأساس للعمل عليها و هذه النظم توفر هذه الميزة بسهولة دون الحاجة لنسخ الملفات يدوياً.
- فهم التغييرات: عندما نجري أي تغييرات على الملفات و نريد حفظها ضمن قاعدة البيانات, نرفق رسالة خاصة تصف التغييرات مع عملية الحفظ مما يسهل علينا في المستقبل فهم هذا التغيير و تذكر سبب القيام به.

مقدمة حول Git

بدأ العمل على تطوير Git عام 2005 من قبل مجتمع نواة Linux بشكل عام و مؤسس Linux بشكل خاص, ظهرت الحاجة إلى استخدام نظام تحكم بالنسخ جديد بحيث يكون قادر على التعامل مع المشاريع الضخمة حيث تعتبر نواة Linux أضخم مشروع برمجي مفتوح المصدر تم العمل عليه حيث تتألف من ملايين الأسطر البرمجية المختلفة بلغة C لتحقيق هذه الميزات يقوم Git بتخزين الملفات ك snapshots بدلاً من deltas مثل النظم الأخرى, الصورة التالية توضح الفرق.

عملية حفظ الملفات ك snapshots تحسن من الأداء بشكل كبير لكنها تزيد من مساحة التخزين قليلاً لذلك يعمد Git إلى ضغط جميع الملفات التي يخزنها.

إذا أردنا إعادة ملف إلى حالة سابقة يجب علينا في حالة التخزين ك deltas أنو نعود إلى الحالة الأولى للملف و تطبيق الفروقات عليه بشكل متتابع, أما في حالة Git ليس هناك أي حاجة للعودة إلى الحالة الأولى للملف فالملف كاملاً يتم تخزينه كل مرة يتغير فيه أي شيء, في حال لم يتغير الملف يتم تخزين مؤشر إلى حالته السابقة.

تقريباً جميع العمليات التي نقوم بتنفيذها مع Git هي عمليات محلية لا تحتاج إتصال بلإنترنت و ذلك بفضل تخزين قاعدة البيانات محلياً لدى كل مستخدم, نستطيع تخزين التغييرات ضمن قاعدة البيانات المحلية ثم نقوم برفعها إلى المخدم عند توفر الإتصال بالإنترنت فقط, بعض النظم الأخرى تتطلب إتصال بلإنترنت بشكل دائم عند أي عملية حفظ للتغييرات.

أقسام مشروع Git

يتألف أي مشروع في Git من ثلاثة أقسام هي:

1. مجلد Git أو قاعدة بيانات Git: تتضمن معلومات حول الملفات التي نتعامل معها بالإضافة لتاريخ التغييرات عليها, تكون مخزنة لدى كل مستخدم و ضمن مخدم مركزي أيضاً.
2. مجلد عمل Git (Git working directory) يتضمن الملفات التي يتعامل معها المبرمج مباشرةً.
3. منطقة الحفظ Staging area هي ملف يتم ضمنه حفظ التغييرات التي سيتم حفظها عند عملية الحفظ التالية.

حالات الملفات في Git

يمكن أن يكون أي ملف ضمن إجدى الحالات التالية:

- محفوظ Commited أي أنه تم الحفظ ضمن قاعدة البيانات و لم يتم عليه أي تغيير منذ آخر عملية حفظ.
- Staged: تم تجهيز الملف للحفظ ضمن منطقة الحفظ, سيتم إدخاله إلى قاعدة البيانات عند عملية الحفظ التالية.
- معدل modified: تم التعديل على الملف منذ آخر عملية حفظ لكن لم يتم تجهيزه للحفظ بعد أي ليس ضمن منطقة الحفظ.

يمكنك تنصيب Git من الموقع التالي https://gitforwindows.org

قبل البدء بساتخدام git لا بد أن تعرف نفسك إليه أولاً من خلال الأمرين التاليين

```
git config –global user.email “<Your email address here>”

git config –global user.name “<Your name here>”
```

عمليات Git الأساسية 1

سنتعرف الآن على بعض العمليات الأساسية ضمن git, التعليمات التالية يجب أن تنفذ ضمن git bash التي تصبح متاحة عند تنزيل Git من الموقع الموجود أعلاه.

نقوم بإنشاء مجلد جديد لكي نضع ضمنه مشروع Git الذي نريد العمل عليه و باستخدام زر الفأرة الأيمن نختار Git bash here لفتح موجه الأوامر.

- الأمر git init يستخدم لإنشاء مشروع Git جديد ضمن المجلد الحالي يتألف فقط من قاعدة بيانات ضمن مجلد يسمى git.
- الأمر git status يستخدم لطباعة حالة جميع الملفات الموجودة ضمن المجلد الحالي.
- الأمر git add <filename> تستخدم لإضافة الملف المسمى filename إلى Git كي يتم تتبع التغييرات التي تجري عليه.
- الأمر git commit –m “message” يستخدم لحفظ التغييرات الموجودة ضمن منطقة الحفظ إلى قاعدة بيانات Git, يتم استخدام message لوصف التغيير الذي نريد حفظه.
- الأمر git log يستخدم لعرض تاريخ التغييرات ضمن المشروع على شكل قائمة من ال commits التي تم إضافتها إلى قاعدة بيانات Git.

مجموعة الأوامر التالية تنشأ مشروع git بالإضافة ملف README.md و حفظه ضمن Git.

```
git init

echo “Read me file” > README.md

git status

git add README.md

git status

git commit –m “Add readme file”

git status

git log
```

مقدمة إلى github

تحدثنا سابقاً عن وجود مخدم مركزي لكي يتم ضمنه تخزين قاعدة بيانات Git و مشاركتها مع عدة مستخدمين, يعتبر github مثال عن هذه المخدمات حيث يوفر واجهة ويب يمكن من خلالها إنشاء مشاريع git و مشاركتها مع المستخدمين بالإضافة لميزات أخرى كثيرة سنتعرف عليها ضمن الجلسات القادمة.

لكي نستخدم github لا بد من إنشاء حساب جديد ضمن الموقع أولاً ثم يفضل أن نعمل على إضافة زوج مفاتيح SSH إلى الحساب كما توضح المقالة التالية https://help.github.com/articles/adding-a-new-ssh-key-to-your-github-account/

عمليات Git الأساسية 2

سنتعرف الآن على العمليات التي تمكننا من تنزيل مشاريع من github و التعديل عليها ثم مشاركة التعديلات مع المستخدمين الآخرين.

- الأمر git clone <url> يستخدم لتنزيل مشروع جديد من github إلى المجلد الحالي, نحصل على url من صفحة المشروع الأساسية على github.
- الأمر git push origin master يستخدم لرفع أي تغييرات قمنا بها إلى github.
- الأمر git pull origin master يستخدم لتنزيل أي تغييرات قام بها غيرنا من github.

وظيفة

- قم بإنشاء حساب جديد على github
- أنشأ مشروع جديد بأي اسم تختاره.
- تنزيل المشروع الجديد إلى جهازك.
- أضافة ملف README.md يتضمن اسمك.
- إضافة الملف السابق إلى قاعدة بيانات git.
- رفع التغييرات السابقة إلى المشروع على github.

العلامات 0 لكن استخدام github ضروري جداً للوظائف القادمة.

روابط مساعدة

- للحصول على مقدمة حول git استخدم الرابط التالي https://mohsensy.github.io/developer-tools/2017/08/25/introduction-to-git-and-version-control.html
- للمساعدة في إنشاء مشروع ضمن github و تنزيله استخدم الرابط التالي https://mohsensy.github.io/developer-tools/2017/09/01/git-startup-guide-part-1.html
- للمساعدة في إضافة زوج مفاتيح SSH تستطيع استخدام الرابط التالي https://mohsensy.github.io/developer-tools/2019/02/21/git-startup-guide-part-2.html
- الكتاب الموجود في الرابط يتضمن معلومات كاملة عن git https://git-scm.com/book/en/v2
- https://guides.github.com/introduction/git-handbook/

Contents

أهداف الجلسة: 1
تعريف نظم التحكم بالنسخ VCS 1
أنواع نظم التحكم بالنسخ 1
فوائد نظم التحكم بالنسخ 2
مقدمة حول Git 2
أقسام مشروع Git 3
حالات الملفات في Git 3
عمليات Git الأساسية 1 3
مقدمة إلى github 4
عمليات Git الأساسية 2 4
وظيفة 4
روابط مساعدة 5