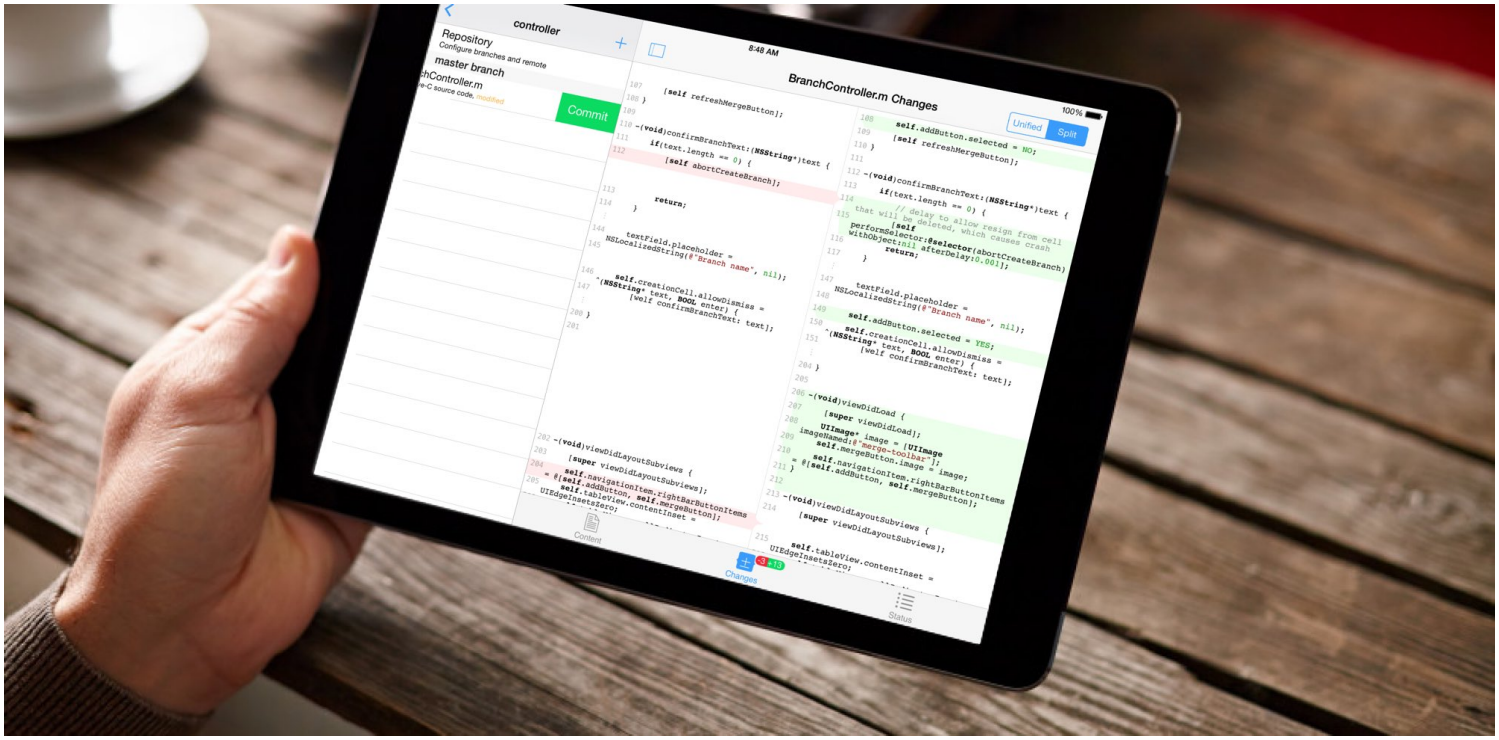
Working Copy
Users' guide

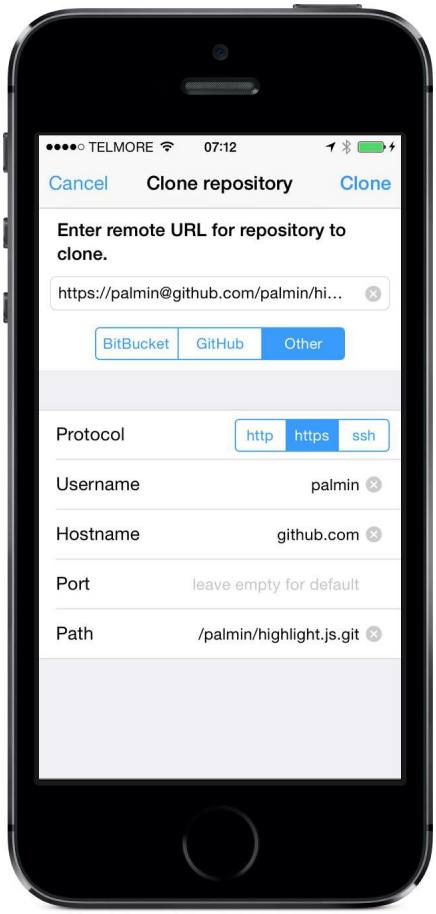


1. Introduction

Working Copy is a full featured Git client for iOS. Git is a powerful version-control system and can take some time to master. The same is true for Working Copy, and even though you will not need to work with the command-line, some understanding of Git is needed. If you are not

confident with Git's core concepts you should read the first few chapters of Pro Git by Scott Chacon or the excellent tutorials Atlassian has made available.

1.1 Cloning repositories



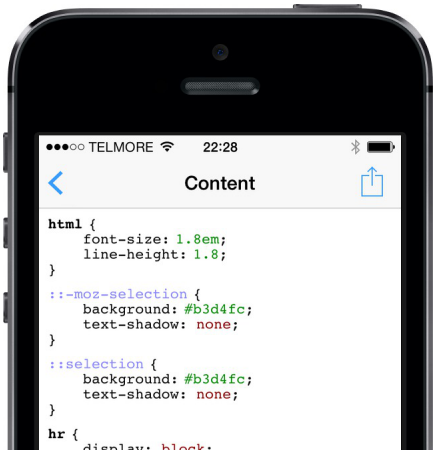
The first step is to get hold of a local copy of the Git repositories you want to access. Duplicating a repository from a remote server is known as cloning, and you do this by pressing + on the list of repositories.

You provide a URL pointing to a repository on the Git remote you wish to clone from. Working Copy can transfer data from the remote using http, https or ssh protocols. However, you should be careful using http transfer since data will be sent without encryption, which means your login credentials and your source code can be intercepted. If you are not on a trusted network you should avoid using http transfer.

Conveniently for BitBucket and GitHub users, you can enter your credentials and get a list of repositories. Cloning then amounts to picking the repository and tapping clone. You are not restricted to these Git services. If you are using some other Git hosting service or a private server to host your repositories, copy-paste your URL into the top field and Working Copy will clone just as well.

You can also import repositories from a Mac or PC by dragging directories into iTunes Document lists. If the directory does not contain a .git subdirectory at top-level, Working Copy assumes you want a new repository with all these files added. Zip-files can also be dragged into iTunes.

1.2 Accessing files



Data in Working Copy is organized as repositories, containing directories, which themselves contain either sub-directories or files. Tapping a file shows the file content, the changes to the file and the status of the file.

File content is shown with syntax highlighting for sourcecode and a preview of html and

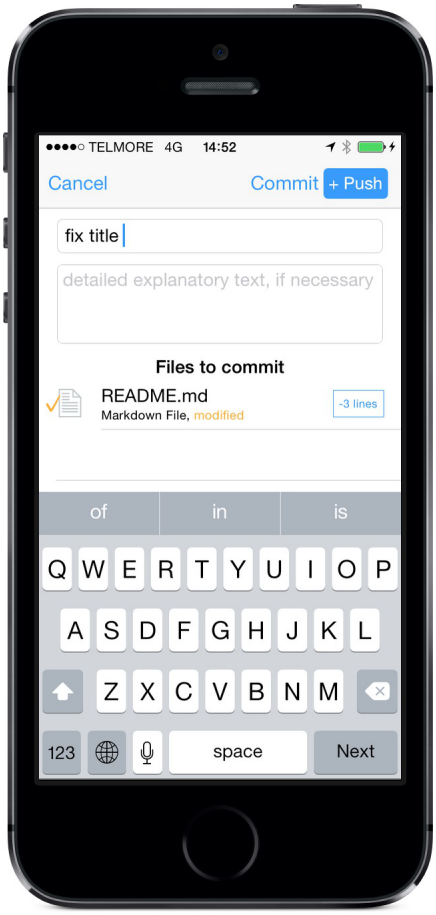
document files. To edit you will need to switch away from Preview mode with the button at the top showing your current mode. Tap the action-button in the upper-right corner to send the file to other applications such as Mail through a share sheet.

The editing inside Working Copy is bare-bones and neutral in that neither programming languages, markdown nor regular text files get special treatment. If you are performing heavy editing consider using a specialized text-editor app for programming, markdown or other purposes. You can read more about using Working Copy in combination with other applications.

To copy files use the *Copy* action from the share sheet, then navigate to the destination directory and press *+* in the upper right corner to insert *File from clipboard*.

Move files and directories by dragging them around in the list of files. Let the dragged item float on the right side of directory entries to enter these, or above the back-button in the upper-left corner to step out.

1.3 Committing changes



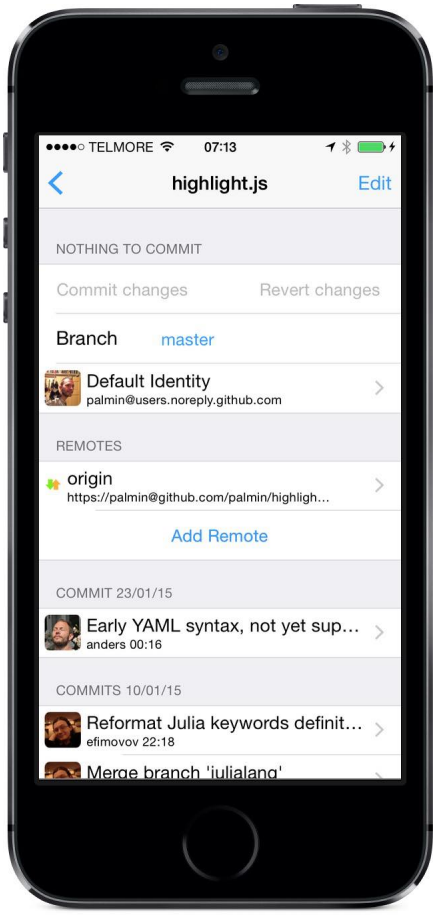
When you have file modifications the Changes tab lights up. You can see what has been added in green and what has been deleted in red. If you are satisfied with the changes you can commit them to the repository with a button on the Status tab. A faster way, however, is to swipe left on the file in the directory listing. Swiping left can generally be performed on lists of files, directories and repositories allowing convenient access to frequent actions.

You can commit a single file, multiple files or the entire repository at once, and it is considered good practice to make a commit represent one conceptual change to your repository. Following this practice also makes it easier to come up with concise yet descriptive commit descriptions.

When you have made one or more commits your on-device repository is seen as being ahead of the remote repository and you *push* these

commits to the remote. Because *Commit* and *Push* are distinct actions you can *Commit* while offline and *Push* once you get back online.

1.4 Staying up-to-date



Commits can be pushed to the remote from many sources. Other people contribute their work, or you could be doing something on a regular computer or another iOS device which results in commits that end up on the remote. You need to unlock the ability to *Push* with an in-app purchase.

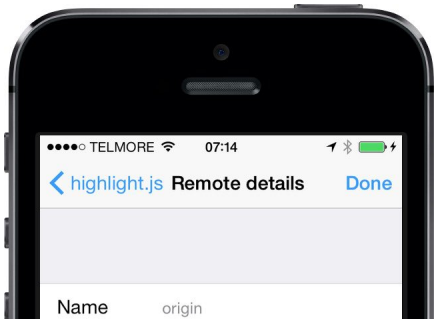
You get commits back into Working Copy through a two-step process where you *Fetch* and *Merge*. *Fetch* reads commits from the server and requires a network connection. The commits will not be integrated with the local data on your device until you *Merge*, which will combine the new commits from the server with your local data.

You can pull the list of repositories down to *Fetch* for all your repositories. If any of your repositories received new commits you will be able to *Merge* all these repositories with a single tap.

Sometimes data cannot be automatically combined because your local changes conflict with the changes from the commits fetched. These conflicts can be resolved by manually editing files and picking the wanted parts from the conflict markers and tapping the Resolve button. A faster solution is to use the Resolve tool that lets you resolve conflicts for many files at once.

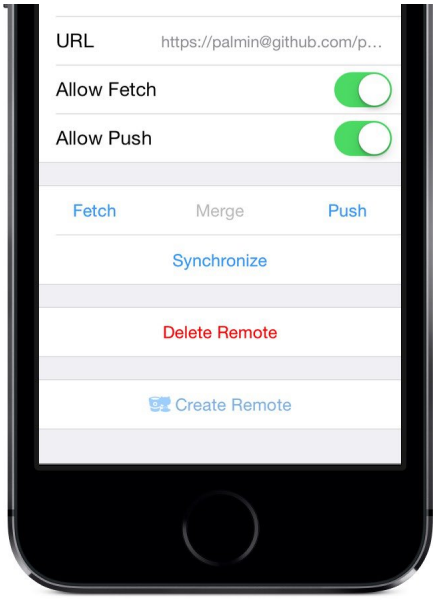
As a short-hand you can *Fetch*, *Merge* and *Push* a repository with the *Sync* button on the detail screen of your remote.

2. Remotes



Git remotes are server-side duplicates of your repositories with full history. These can be services such as GitHub, BitBucket etc. or they can be be privately hosted servers.

When you clone a repository, the URL of the remote repository is your starting point. Working



Copy supports ssh, https and http remotes and the URL consists of protocol scheme, the hostname, username and the path to the repository on the host. The following are typical examples of remote URLs:

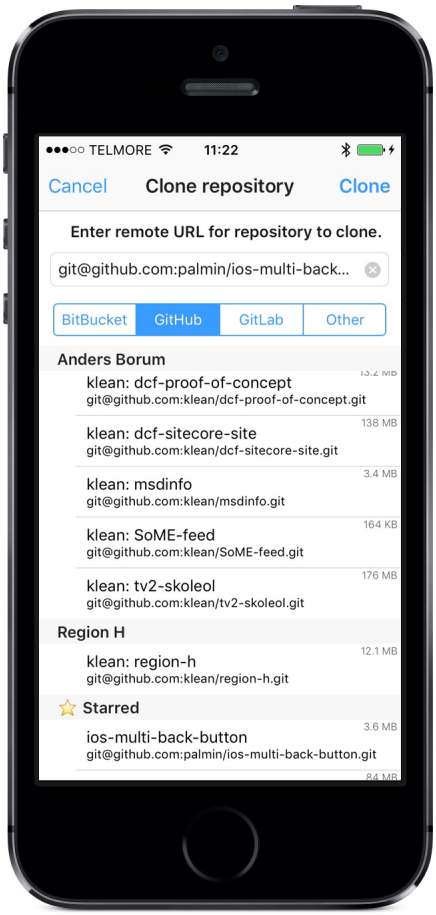
```
https://user@git.company.se/home/user/site.git
ssh://andrew@company.se/home/andrew/git/site.git/
andrew@company.se/home/andrew/git/site.git/
```

The last two URLs are equivalent since ssh is the default protocol.

Authentication will always try with a username included on the form *username@* otherwise remembering the last username for that host. The username *git* has special meaning for many hosts such that the actual user account is derived from the SSH key used to authenticate.

If you enter the Repository page you can add or delete remotes. After cloning there is only a single “origin” remote and, in many scenarios, there is no need for additional remotes.

2.1 Clone catalog



When cloning repositories from BitBucket, GitHub or GitLab you can enter your credentials to get a list of repositories to clone. Working Copy tries to show the most relevant repositories at the top, these being the ones where you have administrative or push privileges. Your GitHub Gists and BitBucket Snippets are also available from this list.

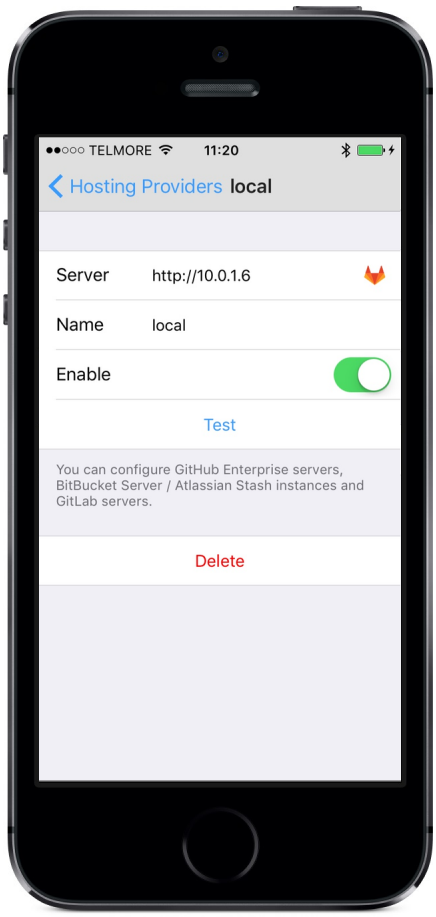
If the list is long, enter keywords in the search field in order to only see repositories containing these. If you do not see the repository you wish to clone, you can still copy-paste the clone URL into the top-field manually.

Organizations on GitHub can be configured to restrict third-party applications such that repositories are not listed and you might need to ask your administrator to approve Working Copy.

Working Copy lets you configure private

instances of BitBucket Server (previously known as Stash), GitHub Enterprise, GitLab and Go Git Service from Working Copy settings. You enter the hostname of your server and the instance type is identified automatically.

You can also configure cloud instances of the above hosting providers as well as Beanstalk and GitBook.



In addition to listing repositories, this integration lets you create new repositories from within the remote detail screen of Working Copy.

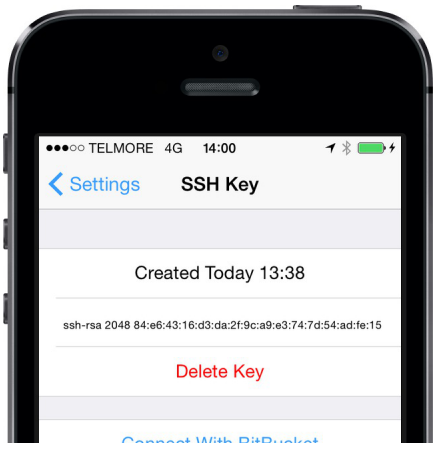
When you add hosting providers it is often a good idea to disable the built-in providers you are not using to avoid clutter in other parts of Working Copy.

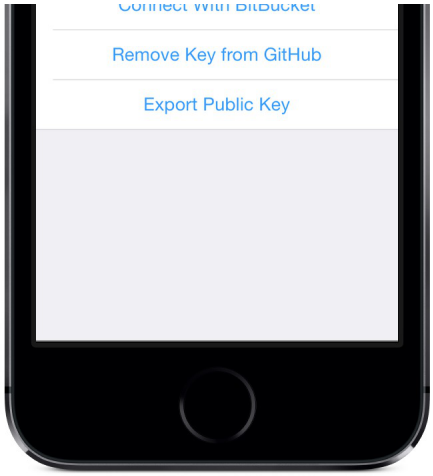
Working Copy will not store your BitBucket or GitHub password used to populate the list of remotes, but rather an authentication token that you can revoke from the BitBucket or GitHub settings. This is why you are required to login through a browser rather than inside Working Copy.

When actually cloning the repository you cannot use the authentication token for the transfer and will either need to configure a SSH authentication key or enter your password inside Working Copy. This password is stored in an encrypted keychain maintained by iOS that is only accessible when your phone is unlocked. If you are using the two-factor authentication, the password authentication will not work and you are required to use SSH key authentication for transfer.

2.2 SSH keys

SSH transfers support password authentication but also public/private key authentication for improved security. The public part of SSH key corresponds to a padlock that you use to lock-down resources. The private part of the SSH key corresponds to the physical key that opens the padlock. Your private key must be kept secret and the public key can be distributed to servers where you want to store remote repositories.





If you tap “Connect with BitBucket” or “Connect with GitHub” your public key will automatically be registered with BitBucket or GitHub. For other Git hosting providers such as OpenShift or GitLab you need to enter your public key in the settings page for that service. The details will depend upon the service in question, but your first step is to Export the public key. When using a Linux server, you need to append the public key to $HOME/.ssh/authorized_keys file.

Working Copy can import private keys in either OpenSSH or Putty format. This choice is available when looking at new SSH Keys that have not yet been generated.

2.3 SSH Troubleshooting

If you are having problems authenticating with an SSH server you should make sure the public key installed on the server matches the private key in Working Copy. If you have some other SSH client on your device or computer, you should make sure you can connect from these without problems. If this works, you must also make sure you use the same SSH key in Working Copy, possibly importing the private key from the other application.

When exporting the public key you end up with something on the form:

```
ssh-rsa AAAAB3NzaC1...g+y4Pfz9 Working Copy - device - 2015-24-12 08:10:00
```

Everything after the second space is just a comment, that makes it easier to determine where and how the key was created. It can sometimes help to remove this comment, before registering the key with your Git server.

2.4 Heroku Remotes

It is possible to deploy to Heroku using Git. You need to setup a second remote for your repository that points to your Heroku application. Since the Heroku Toolbelt is not available on iOS, you must manually configure the remote.

The easiest way to do this is to setup your remote on a regular computer and determine the remote details on the command line by entering:

```
git remote -v
```

You then create a new remote from this URL in Working Copy. Note that username and password for remote is not your Heroku account credentials. You will find this information in your home directory in the hidden file:

`~/.netrc`

2.5 AWS CodeCommit

To use Working Copy with AWS CodeCommit you need to create an IAM user with the *IAMUserSSHKeys* Policy. You also need to add some Policy allowing repository access and *AWSCodeCommitPowerUser* is a good choice, unless you only need to clone and fetch code in which case *AWSCodeCommitReadOnly* is preferable.

You must also export your SSH Key from Working Copy and *Upload SSH Public Key*. This is done in the *Security Credentials* tab of your IAM User. When your SSH Key has been uploaded it will be listed with a *SSH Key Id* that will be needed for the next step.

Your repositories are listed in the CodeCommit Dashboard. When looking at individual repositories you can request the Clone URL in the SSH format, which can be used inside Working Copy, but you need to use the *SSH Key Id* as your username. Your URL should end up looking something like:

```
ssh://APKAJMDDPOLPSL7OAYOA@git-codecommit.us-east-1.amazonaws.com/v1/repos/test
```

3. Viewing and editing

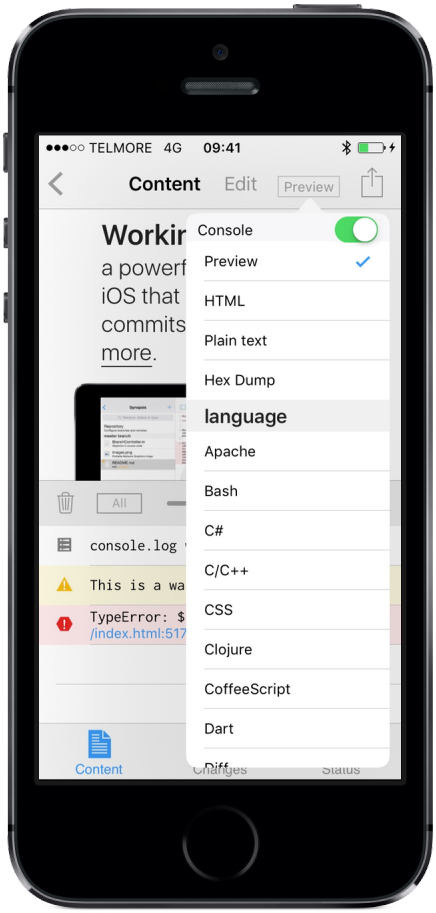
A repository is presented as a hierarchy of directories and files where you tap a directory to enter and view the contents.

Some repositories have deep directory hierarchies and to avoid having to go back repeatedly, you can tap and hold the back button to choose how far back you want to step.

If you tap a file and pick the Content tab you can view the file in different modes. This is controller by the button in the top bar indicating the current mode. The top choices are the recommended mode for

this file and depends on both filename and file content. When you pick a mode for a given file, Working Copy remembers this choice for other files with the same file extension.

3.1 Text Editing



Text files can be viewed as plain text or with syntax highlighting for one of the supported languages. Font size is adjusted in the popup switching between modes and is remembered individually for different modes.

Looking at source code you can tap anywhere to start editing. *Done* in the upper right corner stops editing. You undo latest changes with the undo-button above the keyboard on iPad or by shaking your iPhone. If you want to undo all your changes, switch to the Status tab where you can revert the file to how it was at last commit.

File search is only case sensitive if there are uppercase letters in your query. Wrapping your query in /slashes/ turns it into a regular expression search. The up & down arrows are used to scroll to search results not currently visible. The editor will scroll just enough to bring a new search result into view.

Any text selection can be transformed with action extensions, which will let you do things such as URL encode text from within the editor. Workflow is the prime example of such a application.

When your selection matches a valid CSS color you can adjust this. Placing the caret where a color is expected the popup menu also lets you use the color picker to make a new color.

If you have Dash installed your text selection can be used to make a lookup. Just make sure you have the relevant docsets downloaded in Dash.

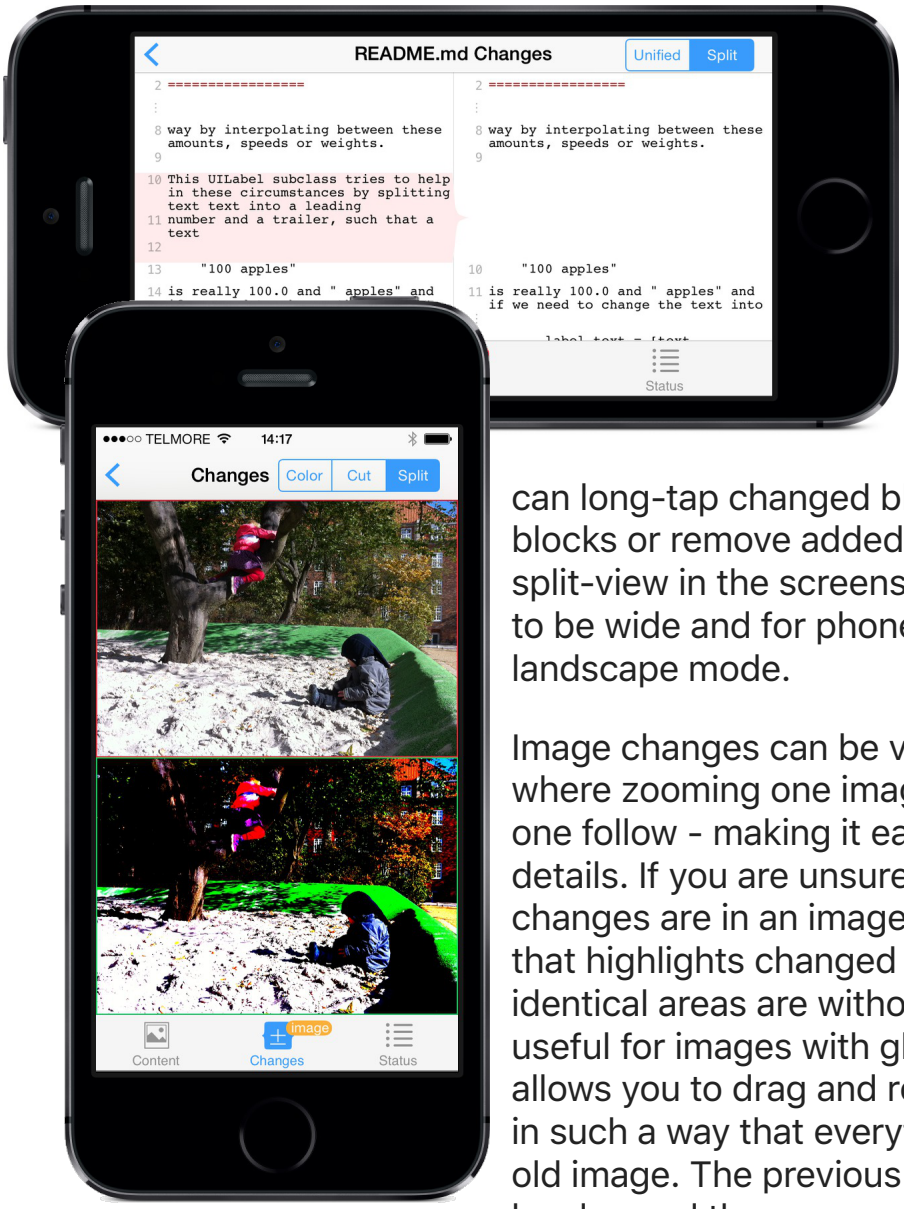
3.2 Preview

When previewing HTML files, relative links to images, javascript and stylesheets resolve to files inside repository and will work without Internet connection. External assets require a Internet connection to load. If offline preview is important to you, consider including javascript

frameworks inside repository.

You can use the Javascript Console to check errors, warnings or log statements. Errors that occur in javascript files inside repository can be tapped taking you to the line in the source file and editing will place the caret at this location. You can evaluate javascript in the context of the HTML page and when external keyboard is attached, the ↑↓ keys let you step through evaluation history.

3.3 File changes



A badge on the Changes tab shows the number of lines added or deleted from a file. The Changes tab itself shows the differences between the last version committed and the current version. You can long-tap changed blocks to recover deleted blocks or remove added blocks. The two-panel split-view in the screenshot requires the screen to be wide and for phones to be turned to landscape mode.

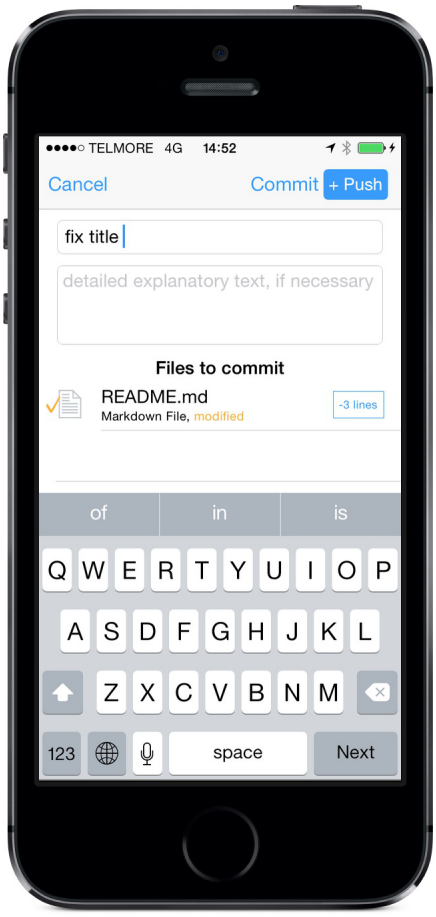
Image changes can be viewed in a split-mode where zooming one image will make the other one follow - making it easy to focus on the details. If you are unsure as to where the changes are in an image, use the Color mode that highlights changed areas in green where identical areas are without color. Cut mode is useful for images with global changes and allows you to drag and rotate a partitioning line in such a way that everything on one side is the old image. The previous image will have a red border, and the new one a green border.

Files can sometimes be conflicted because there have been changes made to the file that could not be automatically merged. This will occur when two people work on the same part of a file at once and they commit changes without knowing about the other changes made. In such cases, you will then need to pick one or the other, or a combined

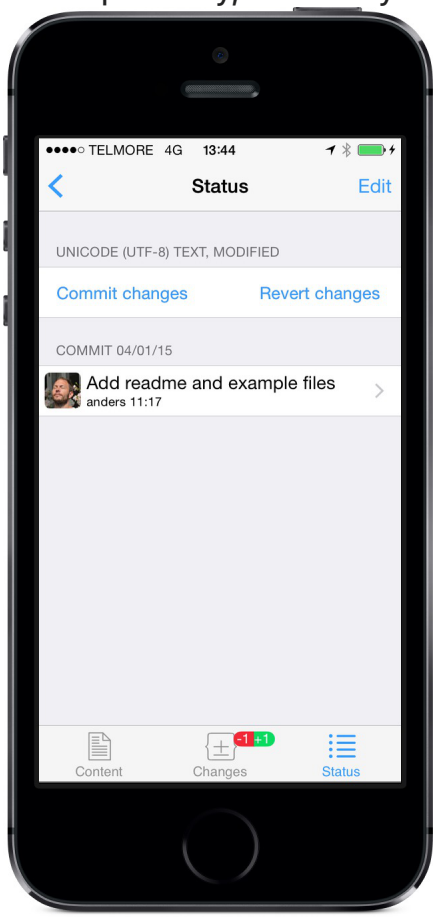
version of the file. When you edit the file there are conflict markers showing how the two different versions of the file disagree on content. You need to sort this out and mark the file as Resolved, which can be done with the action button.

The Status tab says whether the file is modified and allows you to commit or revert changes.

4. Committing or reverting



You can commit changes to your files for the entire repository, for all files in a sub-directory or for a specific file. If you do not wish to commit some of your files you can Revert to how they where at the last commit. The files taken into account are determined by where in the directory structure you initiate the commit. As a short-hand you can swipe left on a repository, directory or file to commit.



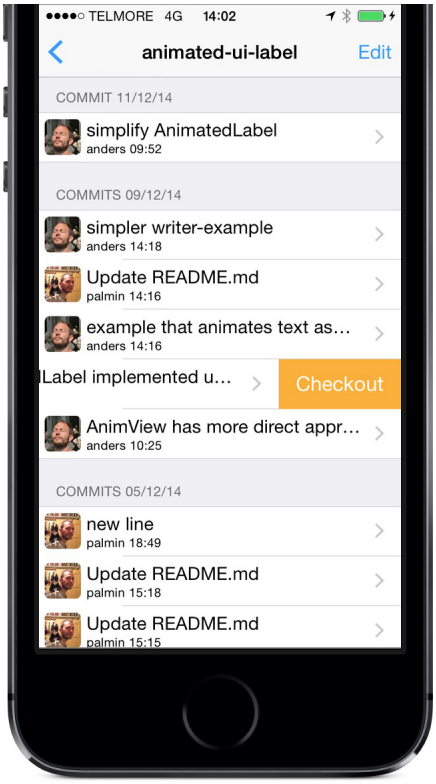
During commit you are shown a list of changed files and can view differences for individual files by pressing the button that shows the number of lines added or deleted. Files with a checkmark will be included in the commit and you toggle the checkmark by tapping the file. Working Copy will push the commit to the remote right away if you enable the *+Push* button.

As a general rule you should make commits with a single purpose and only include the changed files that helped achieve this purpose. You should write a message in the top line describing this purpose; if it is hard to write something short but concrete you might need to break your commit into smaller parts.

4.1 Commit history

The value of well-drafted commit messages

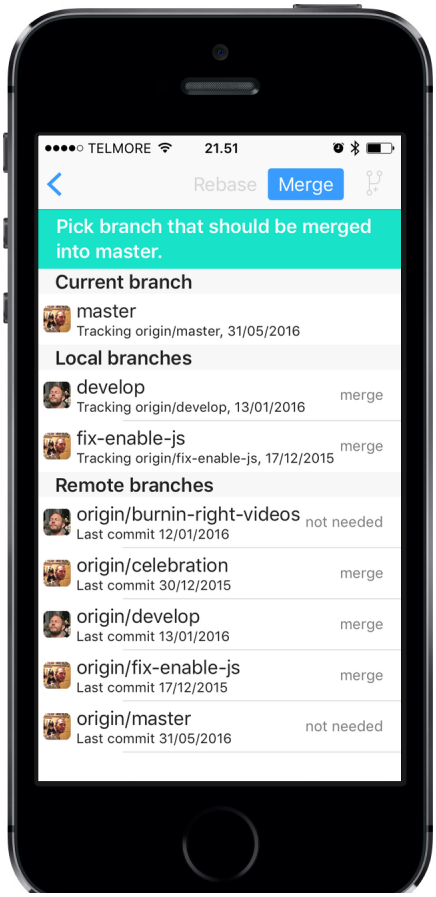
becomes apparent when looking at a log of previous commits. You may do this for either the entire repository, a directory with all its files or for single files. If your commit messages are meaningful, even if you return to a project after months or years you have a much better chance of making sense of the source-code. Tap a commit to see specific changes this commit made to the files in question.



The images shown in commit-logs are determined from the email-address of the person making the commit with the help of gravatar.com. At the commit-list for the entire repository you can Checkout old versions of your files by swiping left on a commit. Your repository will be in a “detached head” state where you cannot commit any changes, but if you make modifications and wish to keep these, you should create a new branch.

Checking out the topmost commit will reattach the head in such a way that your repository is back to normal.

4.2 Branches



A great advantage of Git compared to other version-control software is the ease at which you can branch your repository to work independently on different things. Once you are confident with the work undertaken in a branch, you can merge it back to one of your main branches.

In Working Copy you can do this from the Repository screen by tapping the current branch name to access a list of branches. Tapping a branch brings up a detail view where you can checkout the branch (make it current), rename or delete it.

You can swipe left on branches to *Checkout*, *Rename* or *Delete* without having to go to the detail screen and when a local branch is ahead of its remote, you can *Push* as well.

You create new branches from the current one

with the upper-rightmost button. To put commits on a branch you can either *Merge* or *Rebase*. Atlassian has a great tutorial describing the differences. In both situations you change your current branch to include commits from some other branch.

Merge will create a merge-commit as needed, while rebase will rewrite commits from your current branch on top of the commits from the other remote. Working Copy will not rebase if this requires rewriting commits that have already been pushed to a server. You can override this behaviur by configuring the branch for *History Rewriting*, but this in turn requires you to *Force Push*.

and you merge back another branch into your current branch with the other button in the upper-right. Swipe left to rename or delete branches.

4.3 Commit Graph

To explore repository changes across branches use the Commit Graph available from the Repository screen.

Commits are presented in chronological order with lines showing which commits are based on each other, with tags and branch heads displayed as well as the commit message summary, date and information about the author of the commit.

Pinch to zoom will let you explore additional details, such as the full commit message, the full name of the author rather than initials, a commit identifier and the files modified by this commit. If you connect an external screen or projector to your device, you will get a full-screen Commit Graph without any interface elements obscuring the view. This makes for a convenient tool when your team needs to discuss the project.

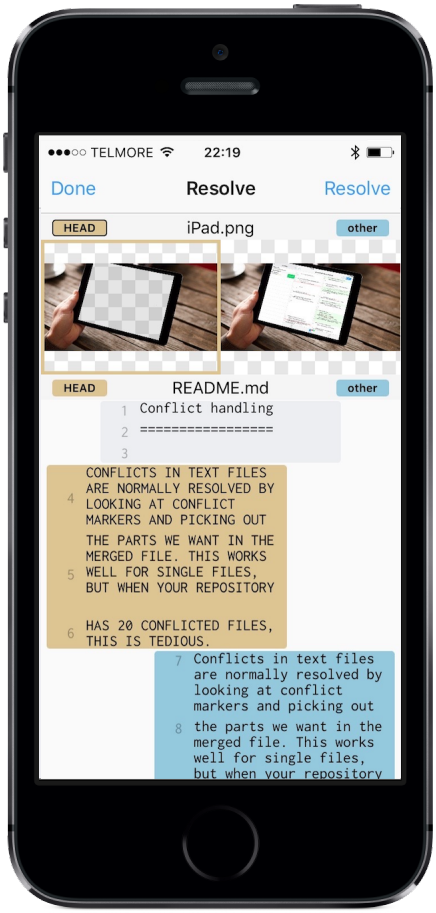
You can tap and hold or double-tap the elements of a commit for additional actions, where the commit message itself takes you to a detail view.

You can copy the author email to the clipboard or start composing a email to the author.

It is also possible to view the commit date in your calendar to see what

else happened around this time and when Fantastical is installed, it will be used instead of the built-in Calendar app.

4.4 Resolving conflicts



Working Copy has a built-in Resolve tool that can be used to fix conflicts for the entire repository, a subdirectory or single files, depending on where in the directory hierarchy the tool is invoked. Swipe left on cells for repository, directories or files to get started.

Text files are shown as chunks of text, where everything both files agree on start out in the center, and everything from *our* version is to the left and *their* version to the right. You swipe these chunks towards the center to include them and away to exclude them. This way your final file will be lined up at the center. Chunks at the border are pending your decision.

If you want to use all chunks from one version of a file and discard the other version of the file entirely, you can tap one of the branch names at the header of the file. These are *HEAD* and *other* in the screenshot.

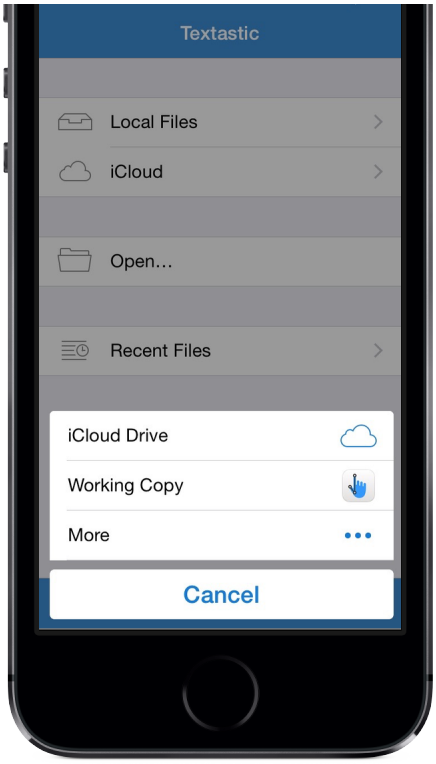
There is no way to combine conflicted images and other binary files. You need to pick one or the other by tapping the one you want. The selected version is marked by a thick border.

When all chunks have been either accepted or rejected and no binary files are awaiting your choice, tap *Resolve* to verify your choice and mark files as resolved. Next commit will conclude the merge. If there are chunks or files pending your decision, the *Resolve* button will scroll to indicate this.

You can also resolve conflicts by manually editing files. The Content tab for a conflicted file has a *Resolve* button for marking the file as resolved, when you have picked the content you want and removed conflict markers.

5. Extending iOS

Working Copy takes advantage of recent improvements to iOS allowing richer cooperation between applications. All



repositories in Working Copy can be accessed by other applications using the iCloud document picker. Once a file inside Working Copy is picked, the other application is allowed to read and make changes to this file and these changes stay inside Working Copy. You can perform editing in this application and switch to Working Copy to review and commit changes. On the iPad you can use Slide Over or Split View to avoid switching between Working Copy and the editing application.

Textastic is a very good general purpose/programmers editor that works well with Working Copy and you use Open... to open the document picker. Alexander Blach the developer of Textastic has made a video showing how. You can pick individual files, directories or entire repositories. Initially you need to enable Working Copy as a Location choice by picking *More*.

Byword is a special purpose Markdown editor that can also use the iCloud document picker to edit inside Working Copy by choosing *Open Other...* but only individual files are supported.

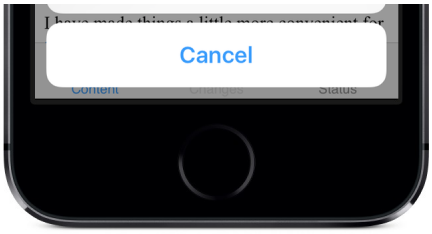
1Writer is another good Markdown editor and you open files in Working Copy by tapping ⊕ in the lower right corner. If you need to work with images Pixelmator is a good choice.

5.1 Saving to Working Copy

Saving files into Working Copy can be accomplished by way of a Share sheet, the mechanism also used to share files with Mail or Messages. Picking Working Copy on a Share sheet will present a list of repositories, where you drill down to the directory where the file should be saved. To overwrite existing files you tap a file before confirming. Otherwise you will be prompted to enter a new file name.

After saving a file you can optionally *Commit* this change immediately and *Push* to the remote right away.

When saving zip-arhives into Working Copy you can either import the archive as a new



repository, extract to a existing repository overwriting existing files as needed, or you can save the zip-file as is.

5.2 Edit in App

In some situations you browse files in Working Copy and need to pick out a file you want to edit in another application. The action-button in the upper-right corner used to present a share-sheet also lets you send files to other applications.

Open In... is the most basic choice and lets you pick any app supporting that type of file. Editing will be on a copy of the file and it must be transferred back to Working Copy with "Save in Working Copy".

If Textastic is installed you can use the *Edit in Textastic* action. A copy of the file is transferred to Textastic with a filename such that Working Copy can recognise the file later on, saving you the trouble of picking the destination file.

Edit in Editorial works similarly, but requires a workflow inside Editorial to facilitate file transfer. It is installed the first time you use *Edit in Editorial* or if you delete or rename the workflow. This workflow serves a dual purpose in that it receives files when activated from Working Copy, and returns files to Working Copy when activated from Editorial.

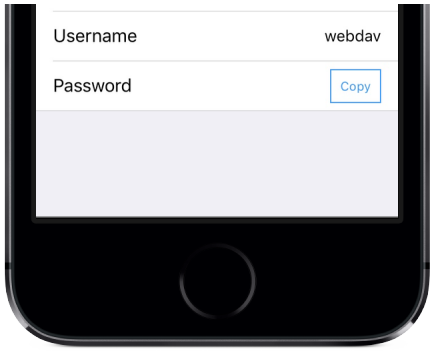
The *Edit in Byword* action has one extra step, where you need to use the share sheet to save back to Working Copy.

You will only see actions for apps you have already installed and you can change the order by picking *More* to the far right.

5.3 WebDAV access

In situations where you need to transfer entire directory hierarchies, a good way to get files into and out of Working Copy is the built-in WebDAV server. It is also the best way to let Coda edit the files in your Git repositories.

WebDav cababilities are built into the macOS Finder and most versions of Windows also let you access WebDAV by mapping to a network drive. You will also be able to use third party WebDAV clients for macOS, Windows, Linux, Android & iOS.



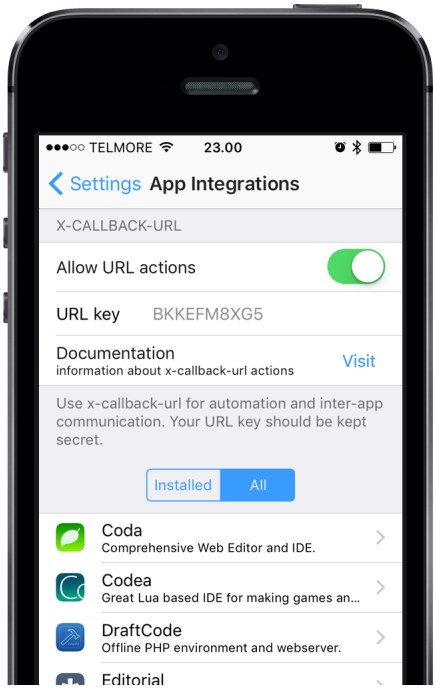
As a security precaution you need to start the server before each use and it automatically shuts down after five minutes of inactivity. You should be cautious of using the WebDAV server on untrusted networks as the transfers are unencrypted. In these cases you should restrict yourself to connections from applications on the same device, as the traffic cannot be intercepted when it never leaves your device. Local connections are also possible in situations without an Internet connection. You need to specify *localhost* as the hostname in these situations.

Applications on iOS are restricted as to how long they are allowed to run in the background. If you start the WebDAV server and switch to some other application, you will be given a grace period before Working Copy and its WebDAV server is terminated and a notification will inform you of this.

On newer iPads you can use Split View multitasking to keep two applications in the foreground.

By using applications such as Textastic and Transmit that can work with both WebDAV and SSH servers, you can deploy from a Git repository to a standard Linux server by transferring the directory hierarchy from Working Copy into the other application and then from the other application onto the server. Here you can watch a short video illustrating this.

5.4 Workflows and Callbacks

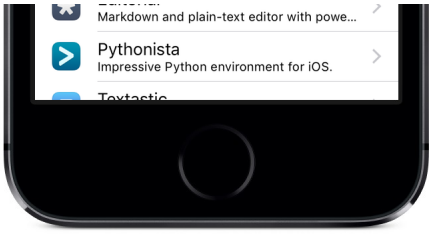


Working Copy supports the x-callback-url mechanism for inter-app coorporation. This allows reading and writing files from repositories as well as committing changes and pushing commits to your remotes. As a security measure this mechanism needs to be enabled from the settings page and is protected by a random key.

By using applications such as Workflow, Drafts and Editorial you can achieve some of the power of shell-scripting on iOS by combining actions of several applications into one action.

5.5 DraftCode

DraftCode is comphrehensive editor and



runtime for PHP, that lets you run entire websites on your iPad or iPhone. Everything happens on your device and works even when offline.

You can transfer a directory to DraftCode by picking *Serve in DraftCode* from the share sheet of a repository summary or directory status screen.

6. Help and Support

A lot of work has gone into making Working Copy as trouble-free to use as possible, but despite that, problems will sometimes occur. Please send your questions by email to anders@workingcopyapp.com and I will do my very best to assist.

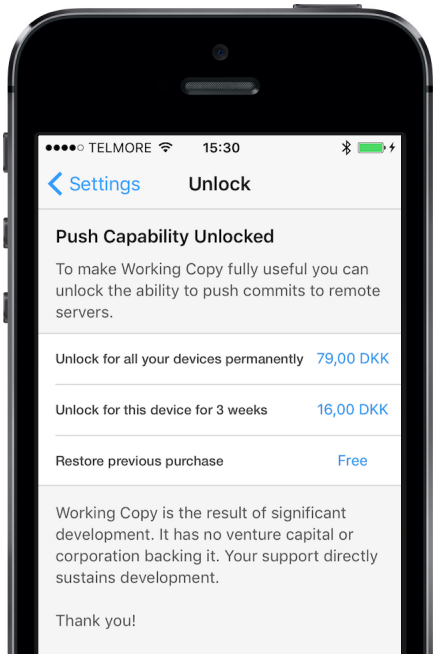
6.1 File Backup

Files stored in Git are often very safe, since repositories are stored in multiple locations some on local computers and some on remote servers. Files that have not yet been committed and pushed to a server are somewhat vulnerable to data loss.

These files takes part in regular iOS backup to either iCloud or a local computer through iTunes. You access the files using iTunes File Sharing and since this uses a external computer, you can recover your files in situations where Working Copy itself has problems running. Your files will be in a directory called *changes*.

6.2 How to purchase

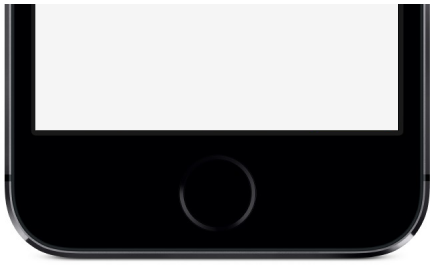
Working Copy is a free download but you need to pay to unlock the ability to push commits back to remote servers. You can do this for 3 weeks or permanently where the permanent unlock is remembered by Apple such that you can restore this purchase on other devices using the same Apple ID.



Alternatively you can use Working Copy Enterprise that has the same features and includes the in-app purchase. Avoiding in-app purchases makes it easier for IT departments to deploy the application to employee devices.

If you are purchasing Working Copy for yourself,

there is no advantage or disadvantage in picking the Enterprise version. Note that repositories in one app will not show up in the other.



Get in touch by email or on Twitter.