A tutorial for versioning archaeological data using Kart

Andrea Titolo Alessio Palmisano

1 Foreword

The following tutorial have been successfully reproduced on macOS 12 and 13 (Monterey and Ventura), and Pop! OS 22.04 (Ubuntu-based).

Please see the requirements below (Section 1.1) and the useful links (Section 5) for external resources or tutorials.

Tip

When using text in CAPITAL_LETTERS inside a code block/line, it usually means that you need to replace those with your own variables. For example, for a code like `mkdir YOUR_FOLDER_NAME`, if you want your folder to be called `kart_example`, you will need to write `mkdir kart_example`.

During the Getting Started (Section 2) and Getting Started with Kart (Section 3) sections, the examples will provide mostly terminal commands, since it will be impossible to provide instructions for all the existing graphical file managers (e.g. Finder, Windows Explorer, Dolphin, Nautilus, etc.). You can use a graphical file manager, of course, the textual documentation should be clear enough for that workflow too. Kart commands (identified by the `kart` prefix) should of course be run in the terminal.

In the Working with Kart (Section 3.5) section, we will instead use the graphical plugin for QGIS but, when possible, corresponding terminal commands will be given. If you are unsure about some commands, the kart command reference is an excellent documentation, otherwise man pages are available for individual commands using `kart COMMAND --help`, e.g. `kart status --help`.

While Kart can also be used to version raster data, this tutorial will only cover the vector dataset usage.

We use Github here as our git forge and we will usually mention it throughout the text. However, it is completely possible to reproduce this tutorial on other forges such as Gitlab or Codeberg. When possible we will mention guides for those forges as well.

While this tutorial is focused on QGIS, editing with ArcGIS/GDAL, etc. is supported by kart, although you will not have access to the graphical plugin.

> ℹ Note
>
> We noticed some difficulties in getting kart to work properly on older versions of macOS, likely because of outdated versions of git, xcode, xcode command line tools, etc. For a smoother experience we recommend using **AT LEAST** macOS 12 (Monterey) and XCode 13.
> On Linux, if you are running the Flatpak version of QGIS, the kart plugin (see below) might have problems finding the kart binary. Switching to a different version of QGIS (e.g. following these instructions) seems to solve the issue, otherwise you might need to see if the issue can be solved using some apps such as Flatseal.

> ℹ Note
>
> While this tutorial is presented here in a stable state, it is by no means finalized, we will keep adding and correcting things based on our continuing experience with kart and the evolution of kart itself.

1.1 Requirements

To avoid an endless guide, the instructions and tutorial below assumes the following:

- You are familiar with git and you have git installed on your machine (there are many guides available online, for example de1v.to, or Atlassian, or this other one). If you don't have git installed, please follow the instruction on the Git website.
- You have at least one authenticated profile for Github (see https://docs.github.com/en/get-started/getting-started-with-git) or for your git forge of choice (e.g.).
- You are are familiar or at least not afraid of using a terminal application (e.g. Terminal.app, Powershell, Konsole, Gnome Terminal, Kitty, Wezterm, Alacritty).
- You are familiar with QGIS and its interface, and you have **QGIS 3.16 or higher** installed on your machine (otherwise please install it from the QGIS website).

2 Getting Started

2.1 Install Kart

Go to https://kartproject.org/ and in the dropdown menu under "Download for your OS" select the appropriate package for your OS. Then, open the file and follow the instruction to install the software.

Alternatively, platform specific instructions follows below. These instructions are taken from the kart official documentation, please always double check the source website before running terminal commands:

Windows

Download the .msi installer from the release page.

macOS

Download the .pkg installer from the release page.

Or use Homebrew to install: `brew install koordinates/kart/kart --cask`

Linux

For Debian/Ubuntu-based distributions, download the .deb package from the release page and install via `dpkg -i kart_*.deb`

For RPM-based distributions, download the .rpm package from the release page and install via `rpm -i kart-*.rpm`

Testing install

Test your install by running `kart --version` in your terminal. If you want, you can follow the Quick Start guide in kart documentation and running `kart status` at one point.

3 Getting Started with Kart

3.1 Set up

If you have never used git you should run the following commands (replacing the quoted text with your actual information):

```
kart config --global user.email "you@example.com"
```

```
kart config --global user.name "Your Name"
```

To understand what the above do, you can take a look at the Getting Started with Git guide from the Github Docs, specifically the parts about Setting up your username and your commit email address.

If you have already used git, chances are you have the above already set-up, and you can continue on.

Create an empty folder where your kart projects will live and move into it:

```
mkdir kart-workflow && cd kart-workflow
```

As with git, there are different ways in which to get started with kart. Here we will take a look at the most likely (in our opinion) cases:

1. We have already existing data and we want to start versioning from scratch.
2. We have versioned data already existing on a remote repository and we want to import those data on our machine.

3.2 A word about HTTPS and SSH

HTTPS connection to github and usually other forges requires additional steps to avoid having git prompting you with a username and password request every time you interact with the server. This is usually solved by setting-up a Personal Access Token or by caching your credentials in Git. Note that failing to cache or save the credentials on your machine in some ways may result in an error when pushing to a remote repository from the QGIS interface.

For a more seamless experience (in our opinion), you can follow this guide to setup an SSH key pair for Github, but guides are also available for Gitlab and Codeberg.

3.2.1 Generating SSH keys (Optional)

The guides above for creating and adding an SSH key should be exhaustive enough, but we can repeat some steps here. These steps below are general but have been reproduced on MacOS only.

! Important

If you have no previous experience, follow the guide above, these steps are not meant to replace it.

1. Generate a new ssh key `ssh-keygen -t ed25519 -C "your_email@example.com"` (the mail should be the same used for your github account) and follow the on-screen instructions for inputting a passphrase.

2. Ensure the ssh-agent is running `eval "$(ssh-agent -s)"`
3. If you are on macOS 10.12.2 or later, manually edit the ~/.ssh/config file as explained in the same guide.
4. Add the ssh key to the ssh-agent `ssh-add ~/.ssh/id_ed25519` (if you are on macOS, you can store the ssh passphrase in your keychain by running `ssh-add --apple-use-keychain ~/.ssh/id_ed25519`)
5. Add the key to your github account following the respective guide.

3.3 Example 1: use an empty Kart repository

3.3.1 Download data

Download the sample data. You can download this file anywhere on your machine.

You can get the data from the same folder where this tutorial lives, by going to this link.

Alternatively, you can use the following CLI tools by pasting the commands in your terminal (Aria2 is likely not installed in your system).

Wget

```
wget https://github.com/UnitoAssyrianGovernance/kart4arch/raw/main/static/kart-workflow-examp
```

Aria2

```
aria2c https://github.com/UnitoAssyrianGovernance/kart4arch/raw/main/static/kart-workflow-ex
```

Curl

```
curl -O https://github.com/UnitoAssyrianGovernance/kart4arch/raw/main/static/kart-workflow-e
```

3.3.2 Import data in Kart

Create an empty folder on your machine and move inside that folder (you can use “kart-tutorial” or replace it with your own folder name). Be sure to copy the absolute path of where you saved your geopackage before (the .gpkg extension **MUST** be included).

```
kart init kart-tutorial --import GPKG:path/to/your/geopackage.gpkg
```

For example, if you created the `kart-workflow` folder in the above steps, and you are on MacOS, your command will look like (run inside the `kart-workflow` folder):

```
kart init kart-tutorial --import GPKG:kart-workflow-example.gpkg
```

The terminal should output something like this (cut for convenience, your output will be longer)

```
Initialized empty Git repository in path/to/kart-workflow/kart-tutorial/.kart/
Starting git-fast-import...
Importing 4 features from kart-workflow-example.gpkg:archaeo_locationquality to archaeo_loca
Added 4 Features to index in 0.0s
Overall rate: 1390 features/s)
Closed in 0s
Importing 106 features from kart-workflow-example.gpkg:archaeo_periodisation to archaeo_peri
Added 106 Features to index in 0.0s
Overall rate: 21077 features/s)
Closed in 0s
[...]
Creating GPKG working copy at kart-tutorial.gpkg ...
Writing features for dataset 1 of 12: archaeo_status
archaeo_status: 100%|                                   | 8/8 [00:00<00:00, 11679.23F/s]
[...]
```

3.4 Example 2: Clone an existing Kart repository

Do not use standard git commands (`git clone`) or Github CLI (`gh repo clone`) or any other similar commands or CLI tools to clone the repository, as this will not work as intended. Use only the `kart clone` command.

3.4.1 Clone the repository

We provide a versioned github repo ready to clone.

3.4.1.1 Cloning with HTTPS

```
kart clone https://github.com/UnitoAssyrianGovernance/kart-tutorial.git
```

Tip

You can use your own folder name instead of the default "kart-tutorial" by adding a name at the end of the command, e.g. `kart clone https://github.com/UnitoAssyrianGovernance/kart-tutorial.git kartproject`

3.4.1.2 Cloning with SSH

```
kart clone git@github.com:UnitoAssyrianGovernance/kart-tutorial.git
```

3.5 Working with Kart

Run `kart status` , if you see something like

```
On branch main

Nothing to commit, working copy clean
```

you are good to go, the repository has been correctly initialized or cloned. If not, please repeat the above steps carefully.

3.5.1 Install the Kart QGIS Plugin

Kart comes with a useful plugin for QGIS. The source code for this plugin is available on Github. To install the latest version of the plugin, use the QGIS Plugin Manager and search for the Kart plugin. Alternatively, you can get the latest version from the release page, then open the QGIS Plugin manager and install the downloaded zip file.

As long as you have kart installed, the plugin can also take care of creating a new repo, cloning an existing one (i.e. the steps we did above), etc. The plugin **has an excellent documentation with substantial graphical examples, so we link to the official docs instead of rewriting our own.**

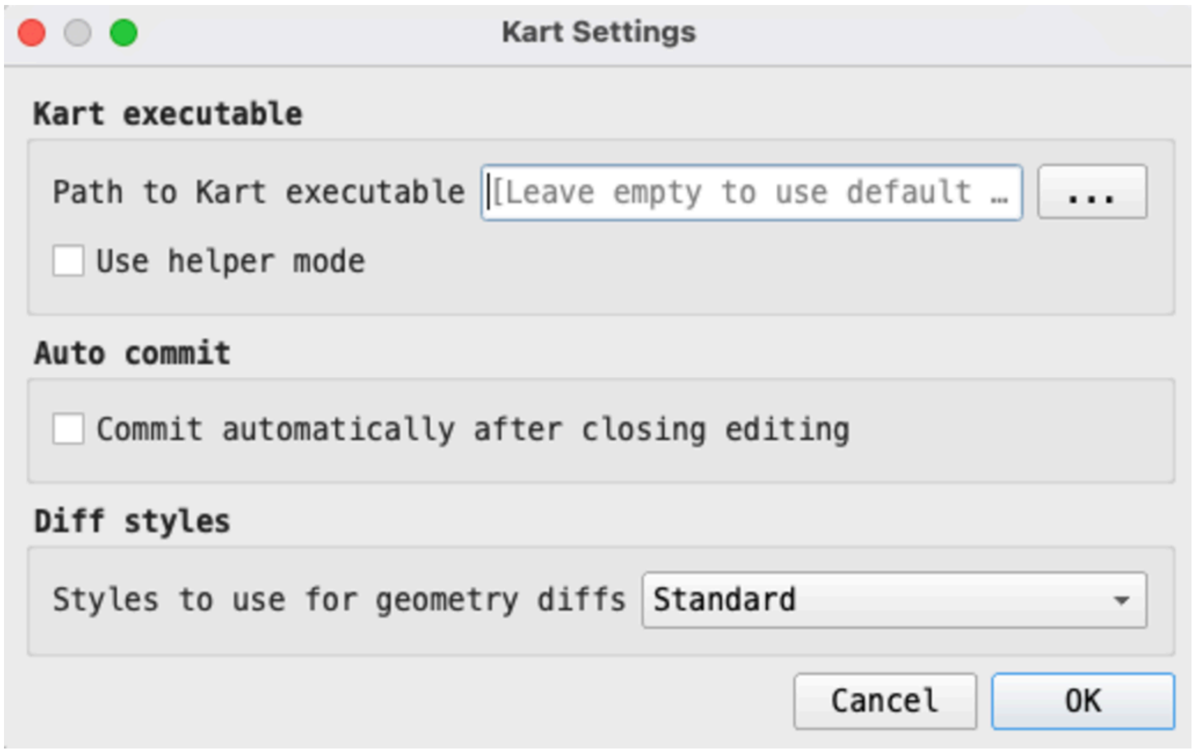
3.5.2 Set up the plugin

Not much setup is needed, the plugin should identify the kart binary automatically. If not, go to *Plugins > Kart > Settings* menu to open the settings dialog. In the *Kart executable folder* field, enter the path to your Kart executable. If the Kart folder is not configured, the Kart plugin will use the default install locations, as follows:

- **Windows**: `%PROGRAMFILES%\Kart\`
- **OSX**: `/Applications/Kart.app/Contents/MacOS/`
- **Linux**: `/opt/kart/`

We also recommend toggling off the option "Commit automatically after closing editing". Depending on your editing workflow, you might end up with hundreds of commits adding one point each without realizing. **Of course, this means that you will have to remember to commit your edits manually.**

When changes have been done, close the plugin dialog box with "Ok".


Kart Settings

Kart executable

Path to Kart executable [Leave empty to use default …

Use helper mode

Auto commit

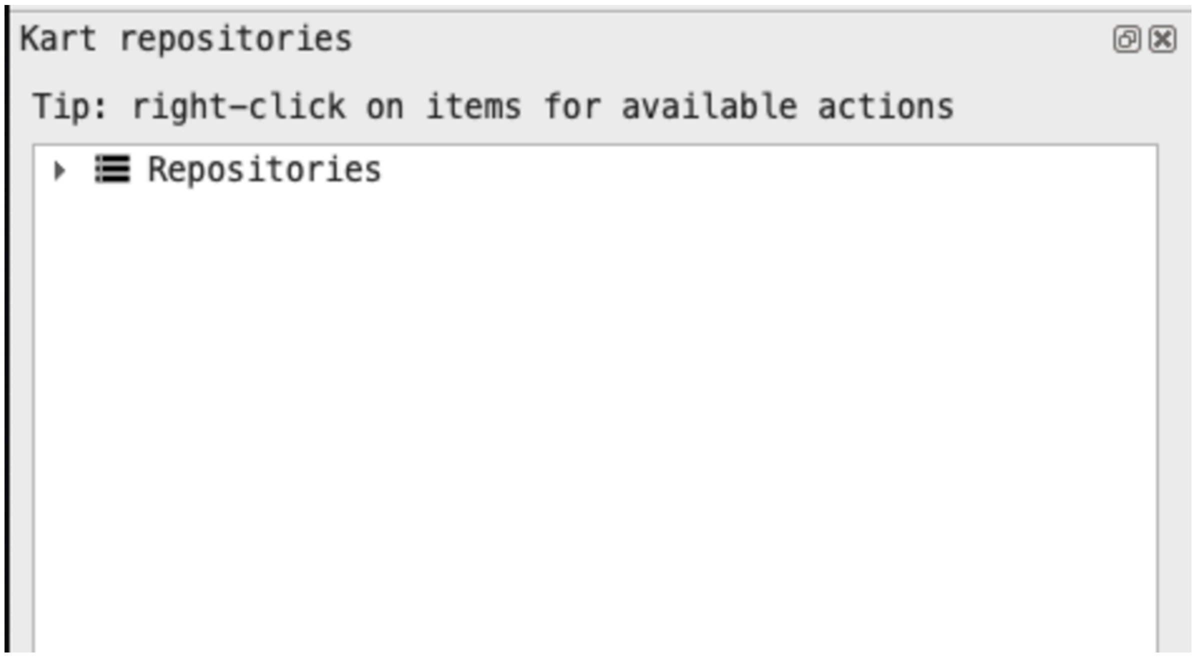
Commit automatically after closing editing

Diff styles

Styles to use for geometry diffs: Standard

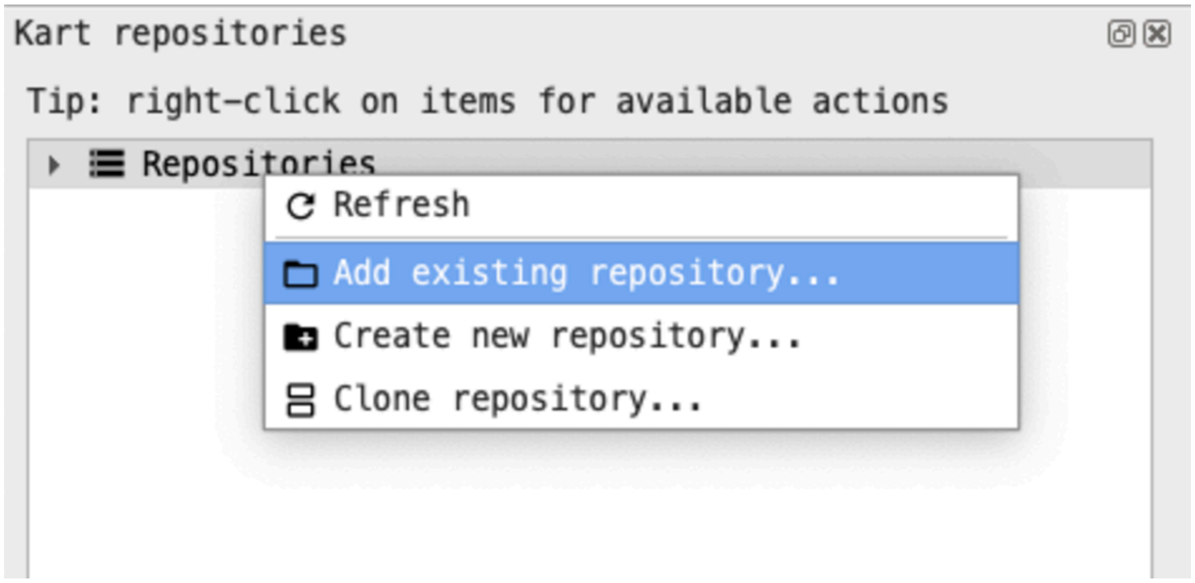
3.5.3 Connect to a local repository

If a side panel has not already appeared, click on *Plugins* > *Kart* > *Repositories* to open the kart panel. The panel will look empty, but we will populate it with the repository we created before.

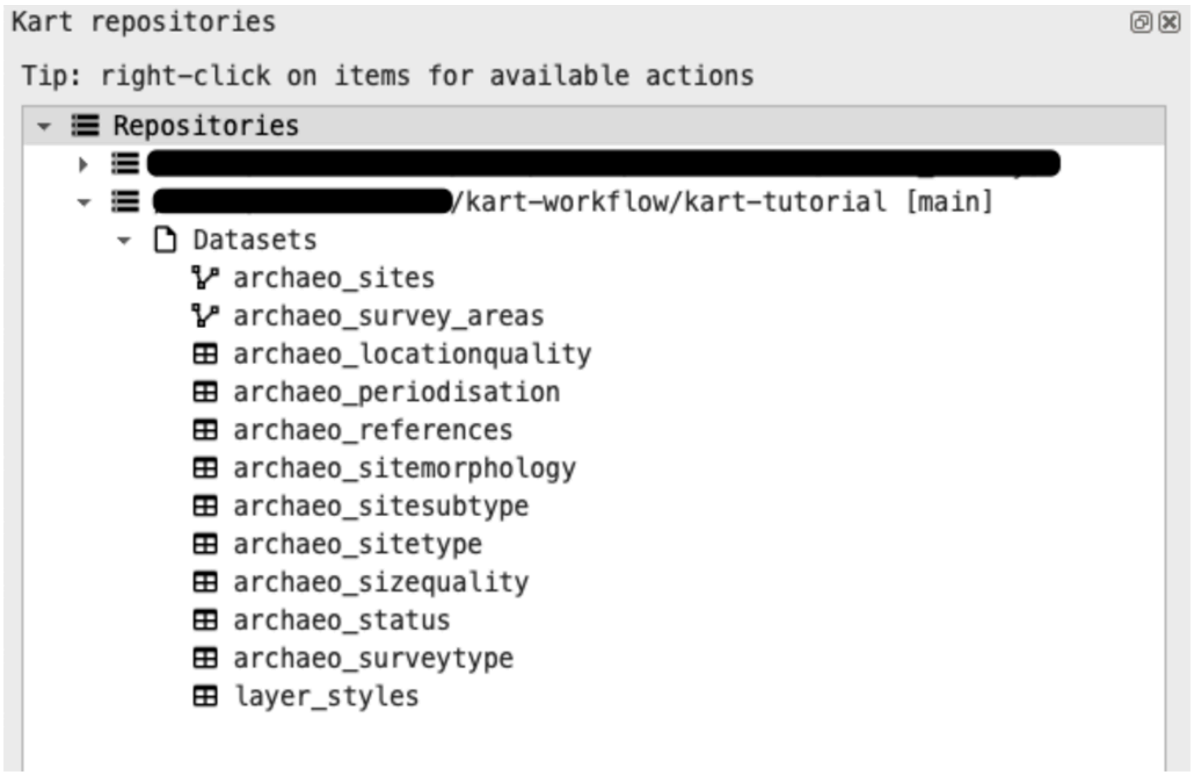


In order to do so, right click on “Repositories”. You will see a list of options: the last two (*Create new repository* and *Clone repository*) are graphical replacement to the steps we did before (Section 3.3 and Section 3.4).

Click on *Add existing repository...* and navigate to the *kart-workflow* folder we created before, then click on “Open”.



You will see the repository path now listed inside the panel under the name `path/kart-workflow/kart-tutorial` `[main]`. The `[main]` bit indicates the branch on which we are working (just as the git branches, more on this on the kart documentation and below in Section 3.6.1).



Kart repositories

Tip: right-click on items for available actions

Repositories
[redacted]
[redacted]/kart-workflow/kart-tutorial [main]
Datasets
archaeo_sites
archaeo_survey_areas
archaeo_locationquality
archaeo_periodisation
archaeo_references
archaeo_sitemorphology
archaeo_sitesubtype
archaeo_sitetype
archaeo_sizequality
archaeo_status
archaeo_surveytype
layer_styles

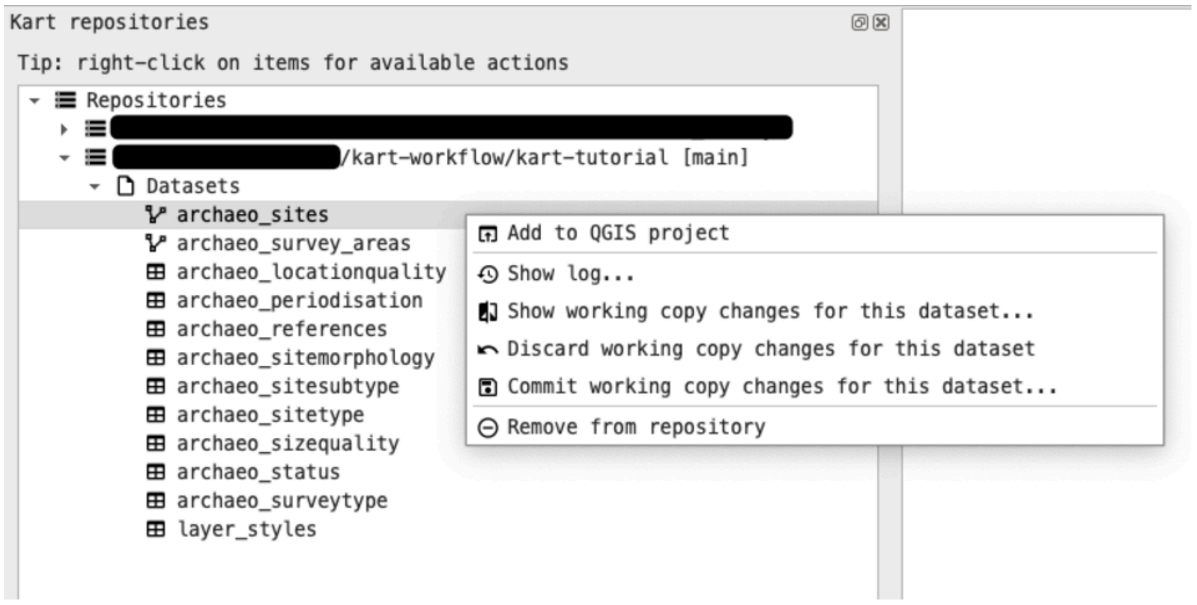
Tip

For more information on the data, see the Appendix or our wiki section.

3.5.4 Load data inside the project

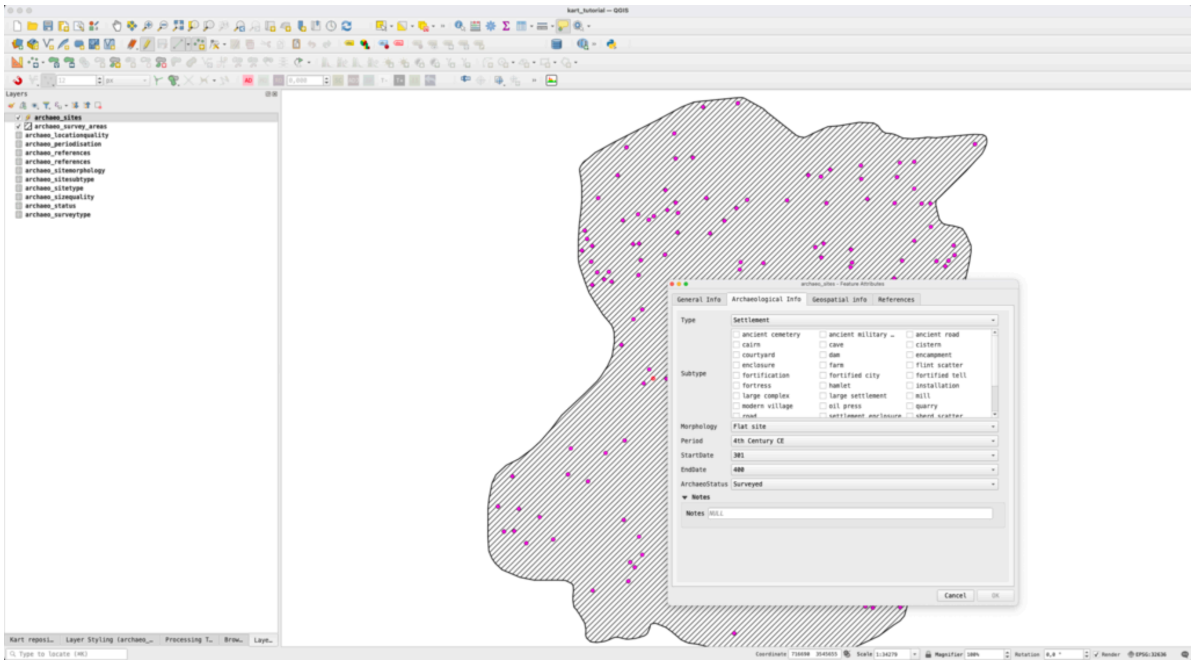
What you connected to is what kart calls working copy, i.e. a file living on your computer that you can interact with through GIS. Kart uses different working copy types, in our case the Geopackage working copy.

To add the layers to our QGIS project, expand the repository tree by clicking on the arrow button on the left in the kart panel. Right-click on the `archaeo_sites` layer and click on *Add to QGIS Project*. A point layer should appear on the map and the CRS should change to EPSG:32636. Add also all the other layers (you should have 10 tabular layer and another 1 geometry layer. Add all of them except the `layer_styles` layer. These table layer are background data useful for populating records in the main layer (archaeo_sites) through QGIS value relation widgets.



3.5.5 Add new geometry to existing layers

You can now proceed to add new data as you would do normally in QGIS. Toggle the editing for the `archaeo_sites` layer and click on the *Add Point Feature* button. Click anywhere and add a new point. Fill the attribute table however you like. The tab structure you will see in the new feature prompt is thank to QGIS attribute form feature, and because the style of the layer is saved inside the geopackage, and versioned in kart as well (this was the `layer_styles` layer present in the repository listing in the kart panel.

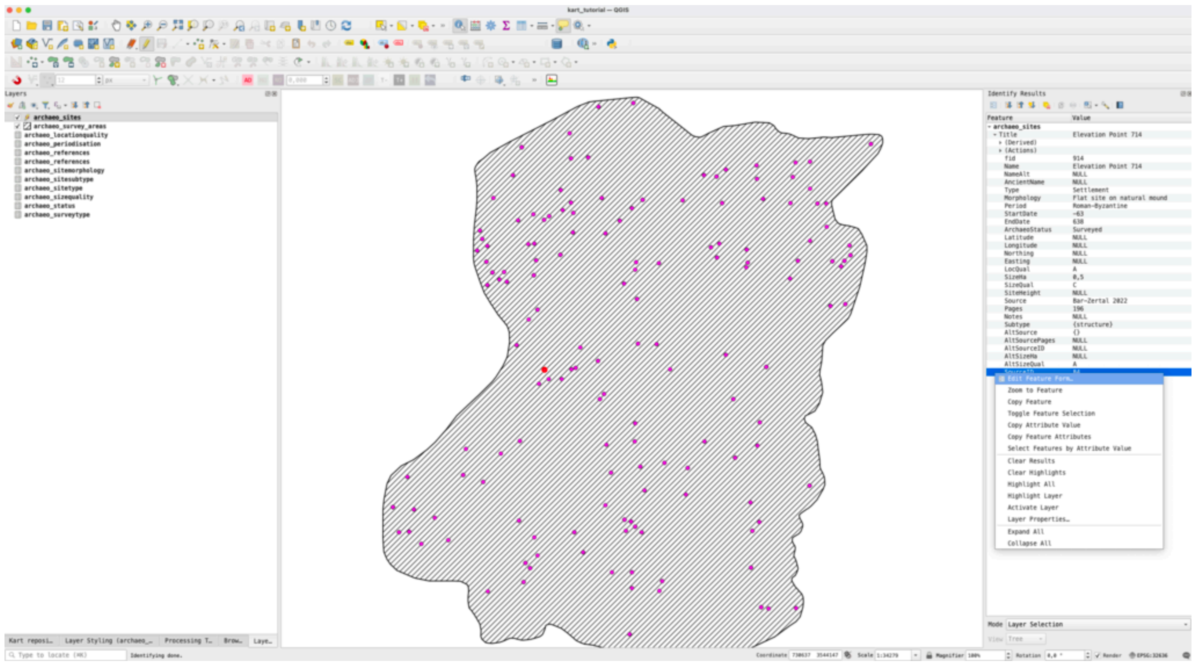


Type	Settlement
Morphology	Flat site
Period	4th Century CE
StartDate	301
EndDate	400
ArchaeoStatus	Surveyed
Notes	NULL

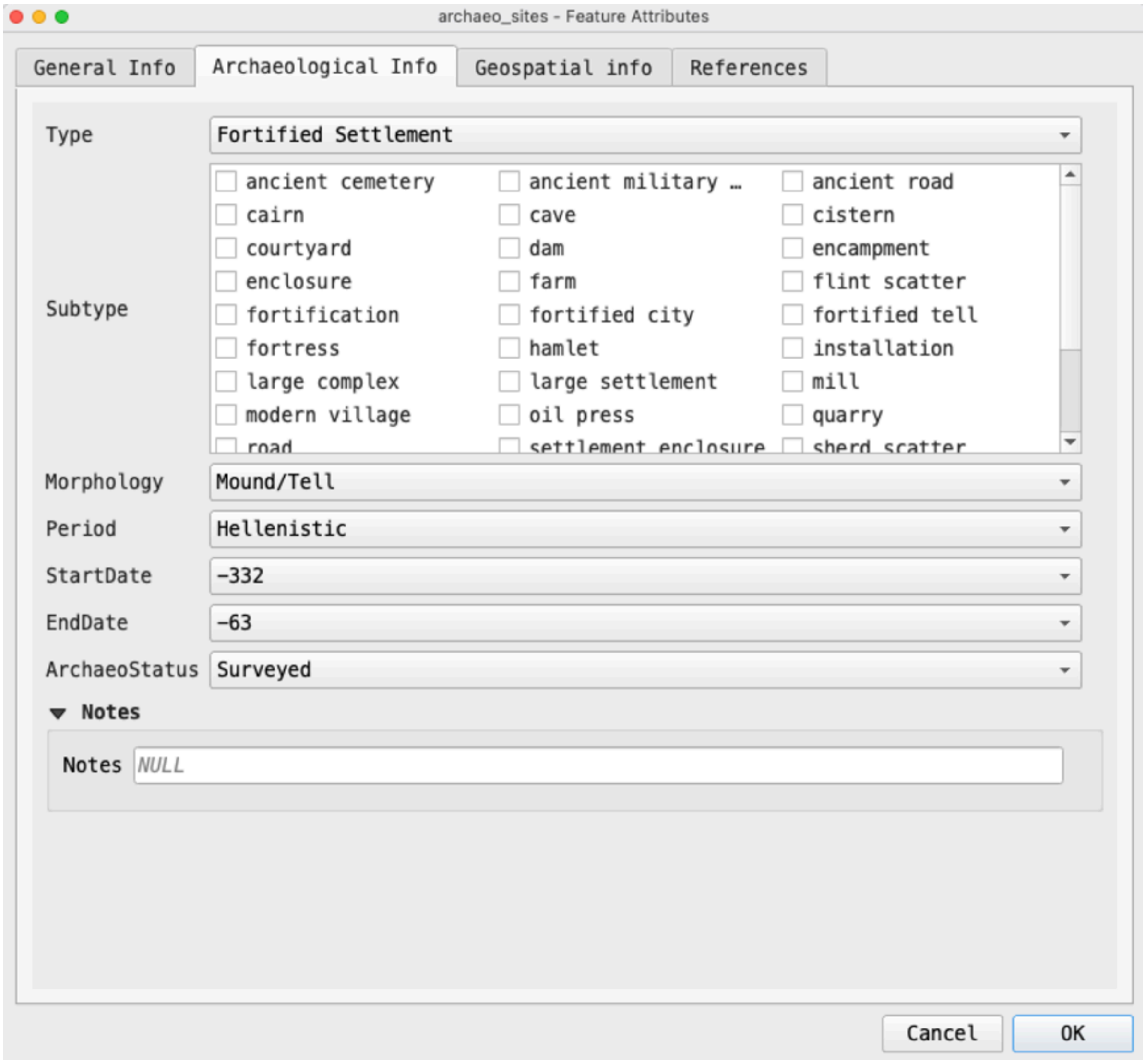
Save the edits by either closing the editing tool or clicking on the *Save Layer Edits* button.

3.5.6 Edit an existing geometry

Let's edit an existing point now, for example by changing its position and modifying some attributes. Use the identify features tool to quickly access the attribute table of a point of your choice, and right click on the identify panel to bring up a contextual menu. From this menu select *Edit Feature Form*. Edit the attribute table, e.g. changing the site type or the periodization, and saving your edits.



Feature	Value
archaeo_sites	
Title	Elevation Point 714
(Derived)	
(Actions)	
fid	914
Name	Elevation Point 714
NameAlt	NULL
AncientName	NULL
Type	Settlement
Morphology	Flat site on natural mound
Period	Roman-Byzantine
StartDate	-63
EndDate	638
ArchaeoStatus	Surveyed
Latitude	NULL
Longitude	NULL
Northing	NULL
Easting	NULL
LocQual	A
SizeHa	0,5
SizeQual	C
SiteHeight	NULL
Source	Bar-Zertal 2022
Pages	196
Notes	NULL
Subtype	{structure}
AltSource	{}
AltSourcePages	NULL
AltSourceID	NULL
AltSizeHa	NULL
AltSizeQual	A
SourceID	[illegible]


archaeo_sites - Feature Attributes

General Info | Archaeological Info | Geospatial info | References

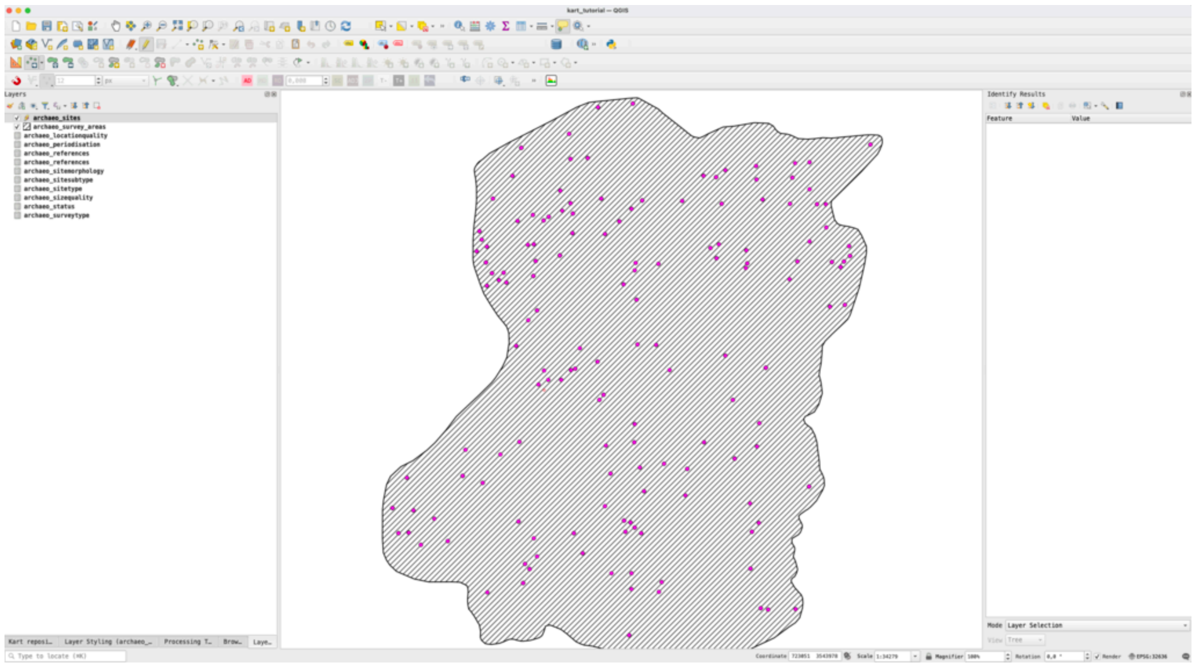
Type	Fortified Settlement		
Subtype	☐ ancient cemetery	☐ ancient military …	☐ ancient road
	☐ cairn	☐ cave	☐ cistern
	☐ courtyard	☐ dam	☐ encampment
	☐ enclosure	☐ farm	☐ flint scatter
	☐ fortification	☐ fortified city	☐ fortified tell
	☐ fortress	☐ hamlet	☐ installation
	☐ large complex	☐ large settlement	☐ mill
	☐ modern village	☐ oil press	☐ quarry
	☐ road	☐ settlement enclosure	☐ sherd scatter
Morphology	Mound/Tell		
Period	Hellenistic		
StartDate	-332		
EndDate	-63		
ArchaeoStatus	Surveyed		

▼ Notes

Notes NULL

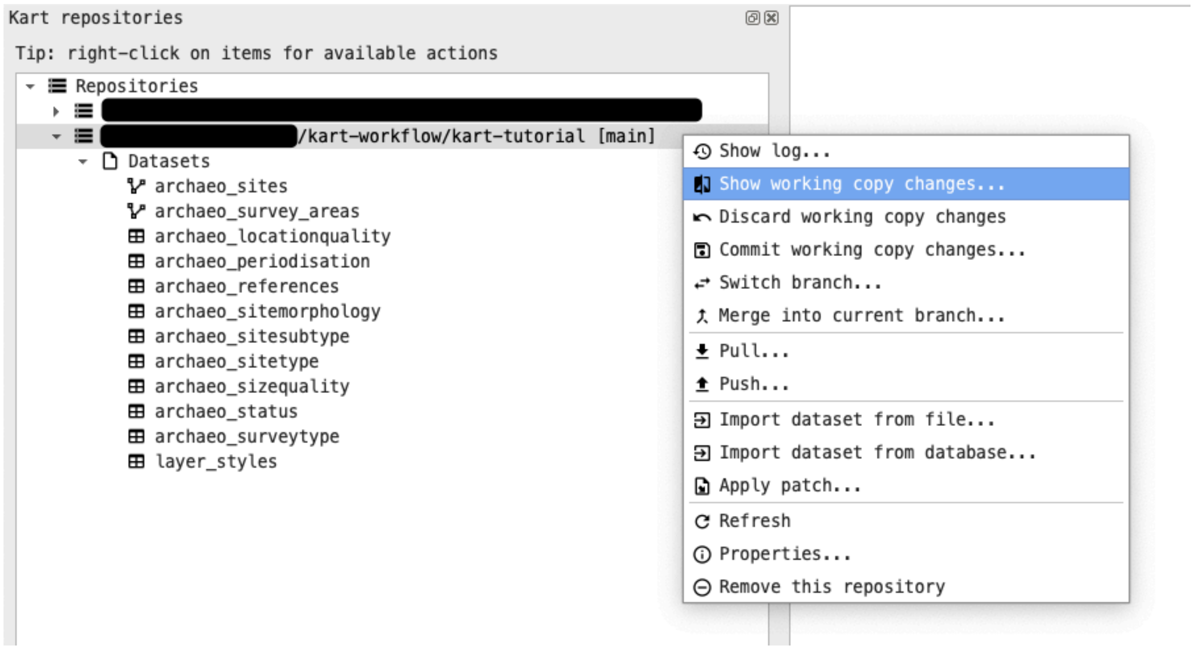
Cancel | OK

Now let's also change the location of one of the points. It can either be the same one or another, but for an easier visualization later, it might be best to edit points near the one we added before. Use the move feature tool to do it and move a point a bit further from its actual location, saving the edits after.

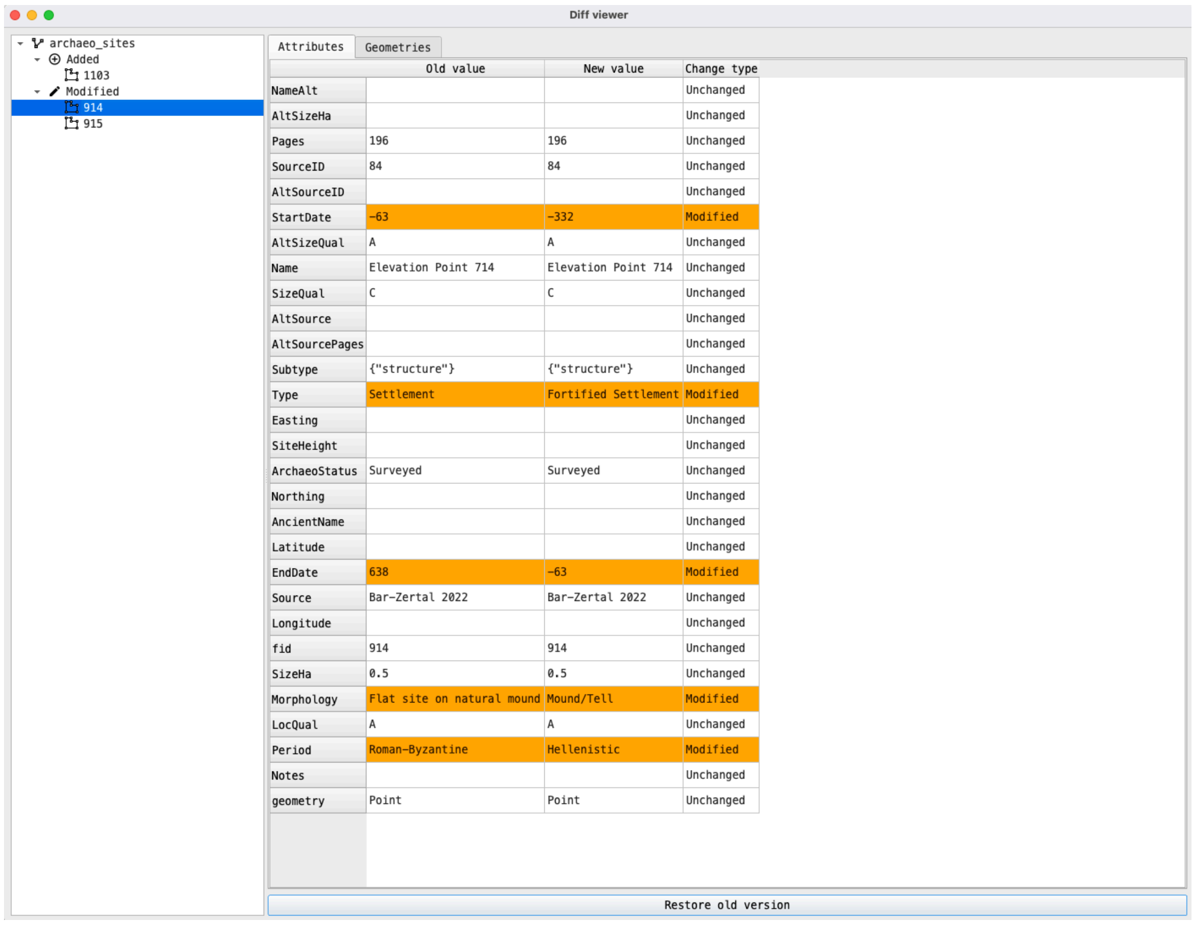


3.5.7 Inspect working copy changes

One of the interesting aspects of Kart is the possibility of inspecting changes before commits. The QGIS plugin in particular offers a way of graphically visualize changes, both on a 2D map or in a table layout. To do so, return to the kart repository panel and right-click on the repository name, then click on the *Show working copy changes...* button (you can also right-click on a single layer to inspect changes only for that one).



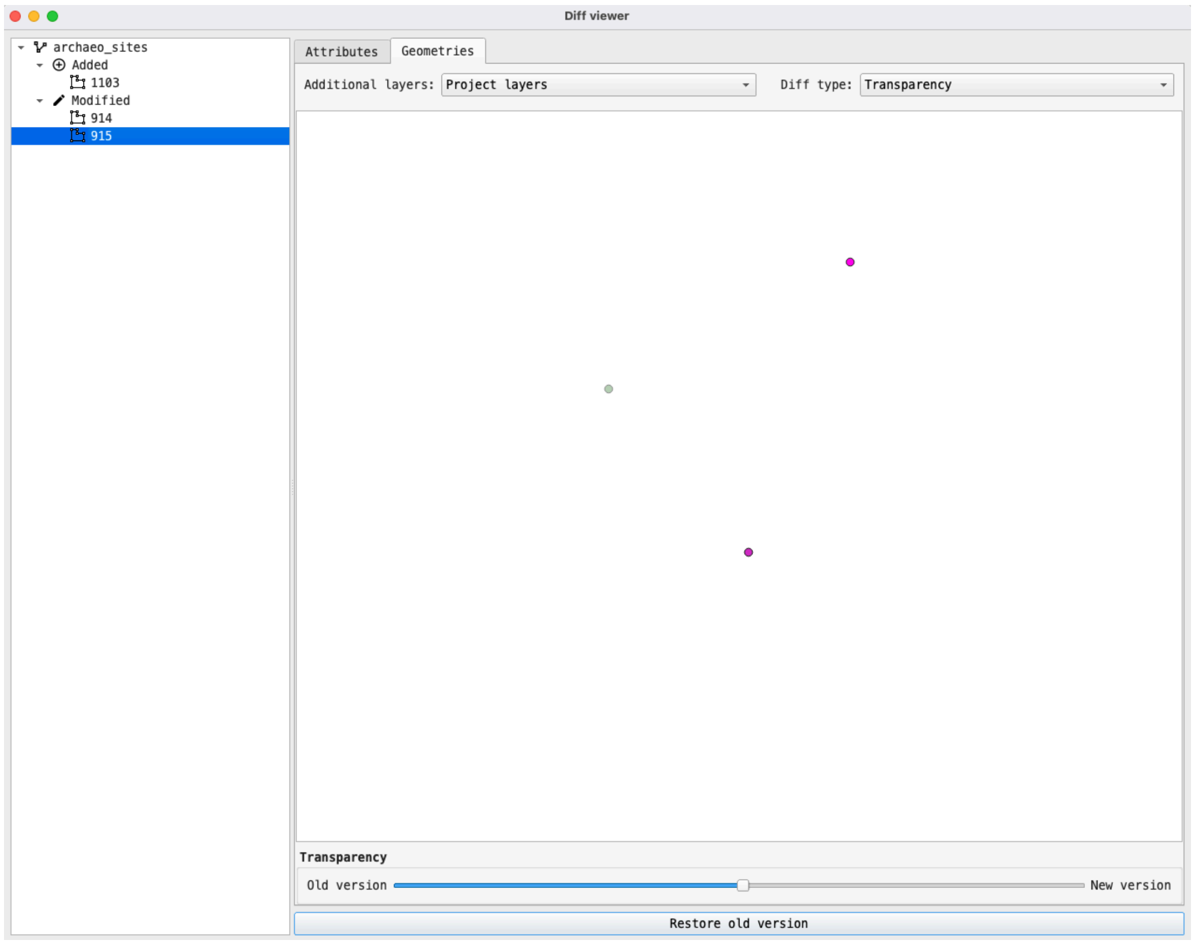
The visual diff viewer panel will open. On the left we can see (in a tree-like structure) the layer that was edited (archaeo_sites, there would be more if we had edited more than one layer), the type of change (Added, Modified)m and the primary key of the added/modified entry. On the right, there are two types of diffs available, the Attributes table, and the Geometries tab. For the Added feature, the attribute table will be all green (as these are all new entries), and the geometry tab will not be much useful. However, if we switch to the *Modified* features, we can see that kart highlights the rows that have been modified in a convenient table with Old/New value pairs.



	Old value	New value	Change type
NameAlt			Unchanged
AltSizeHa			Unchanged
Pages	196	196	Unchanged
SourceID	84	84	Unchanged
AltSourceID			Unchanged
StartDate	-63	-332	Modified
AltSizeQual	A	A	Unchanged
Name	Elevation Point 714	Elevation Point 714	Unchanged
SizeQual	C	C	Unchanged
AltSource			Unchanged
AltSourcePages			Unchanged
Subtype	{"structure"}	{"structure"}	Unchanged
Type	Settlement	Fortified Settlement	Modified
Easting			Unchanged
SiteHeight			Unchanged
ArchaeoStatus	Surveyed	Surveyed	Unchanged
Northing			Unchanged
AncientName			Unchanged
Latitude			Unchanged
EndDate	638	-63	Modified
Source	Bar-Zertal 2022	Bar-Zertal 2022	Unchanged
Longitude			Unchanged
fid	914	914	Unchanged
SizeHa	0.5	0.5	Unchanged
Morphology	Flat site on natural mound	Mound/Tell	Modified
LocQual	A	A	Unchanged
Period	Roman-Byzantine	Hellenistic	Modified
Notes			Unchanged
geometry	Point	Point	Unchanged

If we switch to the Geometries tab instead, and select the point that we have moved from the list on the left, then we will see how Kart highlights with transparency the previous location of the geometry/nodes and in a darker color the new location.

On the top of the Geometries tab, there are also two dropdown dialogs, one (*Additional Layers*) related to the layer we want to show in the Diff Viewer in addition to the modified geometries (options are: Project Layer, OSM Basemap, No additional layers), and another (*Diff type*) related to how we want to visualize the Diff (options are: Transparency, Swipe, Per-vertex diff). We found both transparency and per-vertex diff to be particularly useful in case of polygon geometries.



Note that the kart CLI also offers a way to inspect changes to the working copy, through the **`kart status`** and **`kart diff`** commands. For example, for the edits that we just made the output would be something similar:

Expand

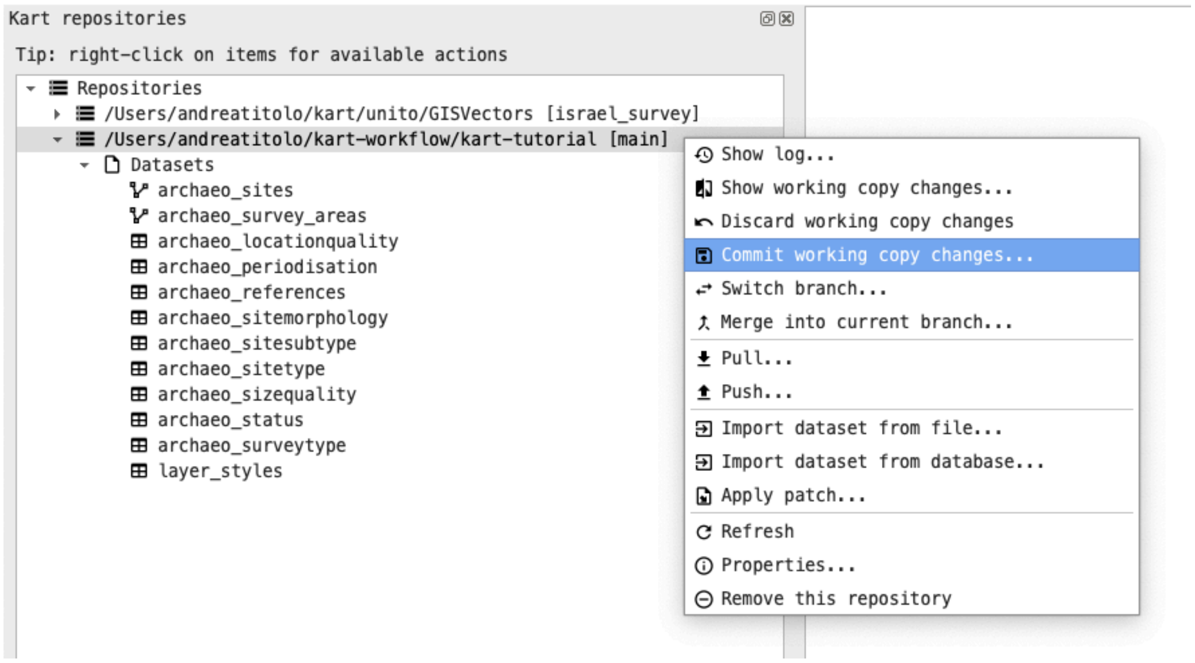
```
kart status
On branch main

Changes in working copy:
  (use "kart commit" to commit)
  (use "kart restore" to discard changes)

  archaeo_sites:
    feature:
      1 inserts
      1 updates

kart diff
--- archaeo_sites:feature:914
+++ archaeo_sites:feature:914
-                                     Type = Settlement
+                                     Type = Fortified Settlement
-                               Morphology = Flat site on natural mound
+                               Morphology = Mound/Tell
-                                   Period = Roman-Byzantine
+                                   Period = Hellenistic
-                                StartDate = -63
+                                StartDate = -332
-                                  EndDate = 638
+                                  EndDate = -63
+++ archaeo_sites:feature:1103
+                                      fid = 1103
+                                 geometry = POINT(...)
+                                     Name = A new point
+                                  NameAlt =
+                              AncientName =
+                                     Type = Settlement
+                               Morphology = Flat site
+                                   Period = Hellenistic
+                                StartDate = -332
+                                  EndDate = -63
+                            ArchaeoStatus = Surveyed
+                                 Latitude =
+                                Longitude =
+                                 Northing =
+                                  Easting =
+                                  LocQual = A
+                                   SizeHa =
+                                 SizeQual = C
+                               SiteHeight =
+                                   Source = Archaeological Survey of Israel
+                                    Pages =
+                                    Notes =
+                                  Subtype = {"hamlet"}
+                                AltSource =
+                           AltSourcePages =
+                              AltSourceID =
+                                AltSizeHa =
```

3.5.8 Commit working copy changes



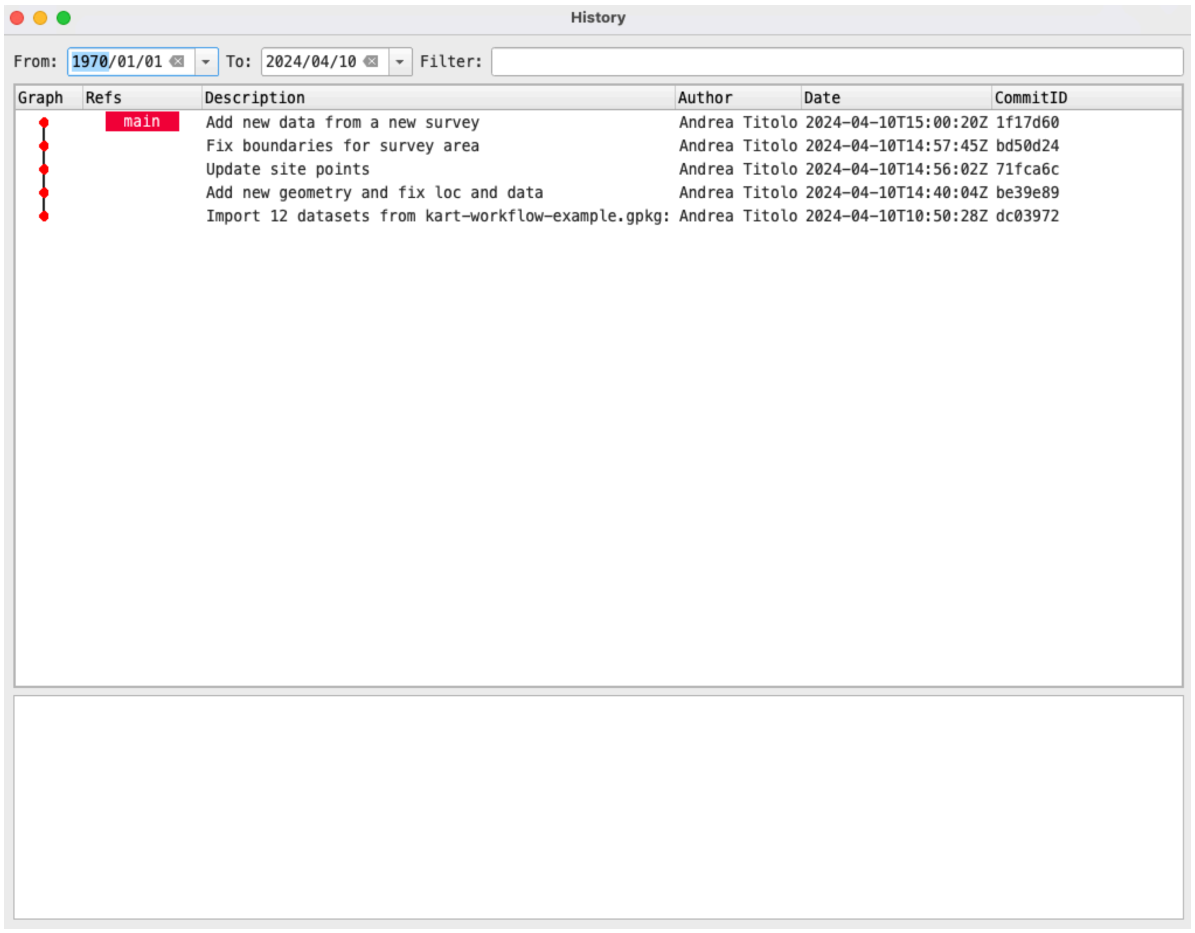
relevant documentation

```
kart commit -m 'YOUR COMMIT MESSAGE'
[main be39e89] your commit message
  archaeo_sites:
    feature:
      1 inserts
      2 updates
  Date: Wed Apr 10 16:40:04 2024 +0200
```

Before proceeding

Add a couple more points or modify existing ones, and make at least other two-three commits, by repeating the same steps above. You can also edit different layers, such has the table layers, or the survey_areas layers.

3.5.9 Inspect the commit tree

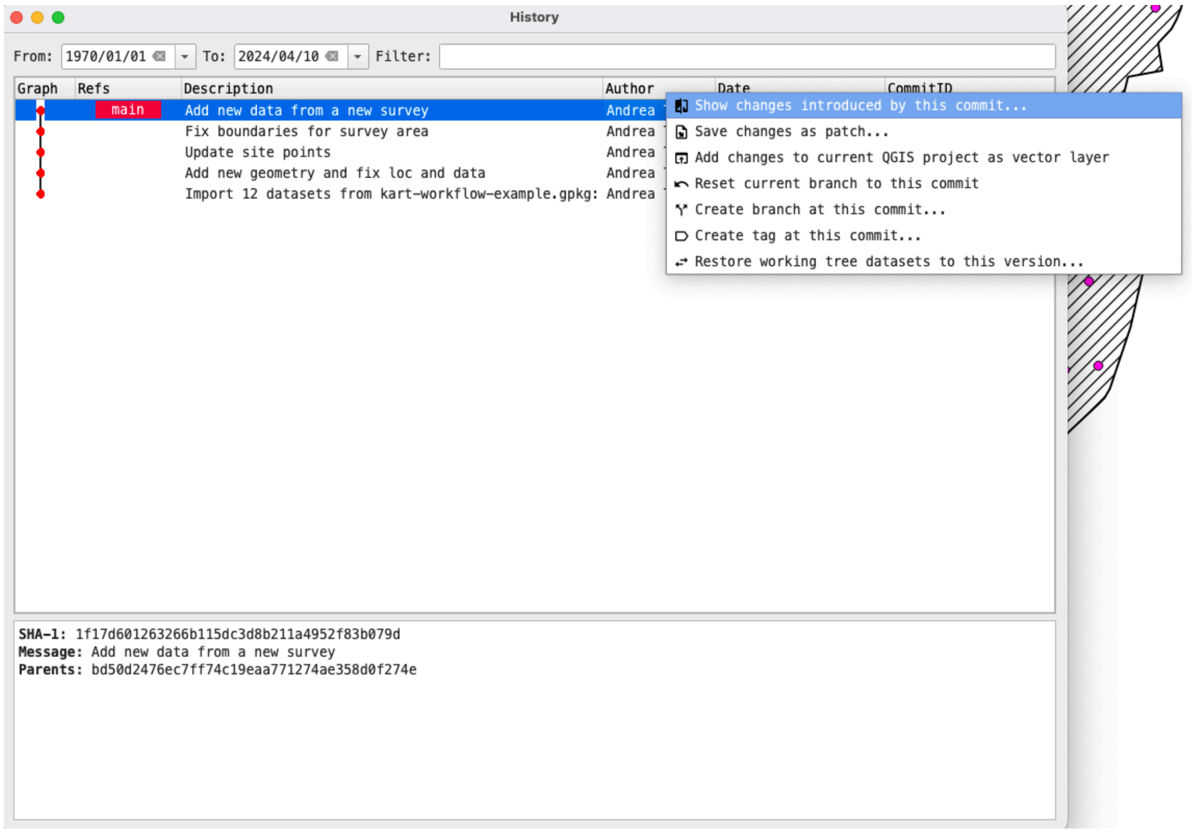


History

From: 1970/01/01 To: 2024/04/10 Filter:

Graph	Refs	Description	Author	Date	CommitID
●	main	Add new data from a new survey	Andrea Titolo	2024-04-10T15:00:20Z	1f17d60
●		Fix boundaries for survey area	Andrea Titolo	2024-04-10T14:57:45Z	bd50d24
●		Update site points	Andrea Titolo	2024-04-10T14:56:02Z	71fca6c
●		Add new geometry and fix loc and data	Andrea Titolo	2024-04-10T14:40:04Z	be39e89
●		Import 12 datasets from kart-workflow-example.gpkg:	Andrea Titolo	2024-04-10T10:50:28Z	dc03972

3.5.10 Inspect changes introduced in a specific commit (and other options)


History
From: 1970/01/01
To: 2024/04/10
Filter:
Show changes introduced by this commit...
Save changes as patch...
Add changes to current QGIS project as vector layer
Reset current branch to this commit
Create branch at this commit...
Create tag at this commit...
Restore working tree datasets to this version...

Graph	Refs	Description	Author	Date	CommitID
	main	Add new data from a new survey	Andrea		
		Fix boundaries for survey area	Andrea		
		Update site points	Andrea		
		Add new geometry and fix loc and data	Andrea		
		Import 12 datasets from kart-workflow-example.gpkg:	Andrea		

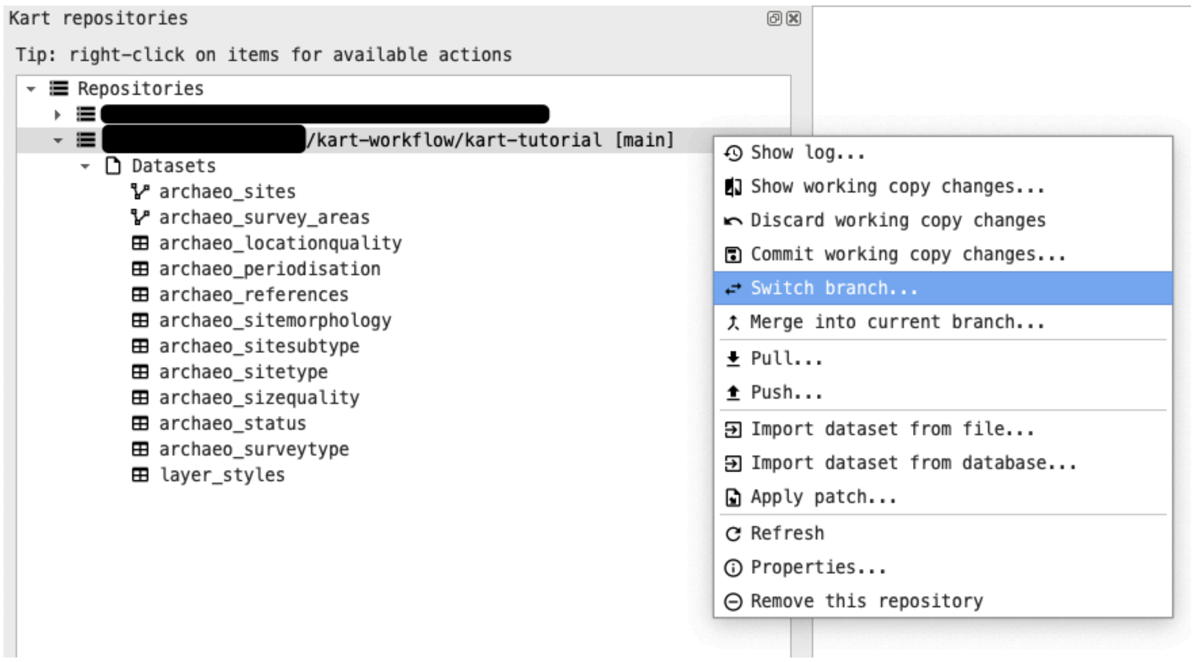
```
SHA-1: 1f17d601263266b115dc3d8b211a4952f83b079d
Message: Add new data from a new survey
Parents: bd50d2476ec7ff74c19eaa771274ae358d0f274e
```

3.6 Collaborating with Kart

Warning

Here we are using the `main` branch as an example to avoid too much redundancy, but this is usually **not a good practice**, as usually is best to have everyone working on secondary branches. As suggested in this comment, it is recommended that one user will do the merge, push the changes, and any other user will use the merged branch as starting point for any new edits in order to avoid further conflicts. This means that after the merge into main is completed and the updated main has been pushed to remote, new edits **should be done on new branches created from the updated main branch** (after updating the local main branch with `kart pull`).

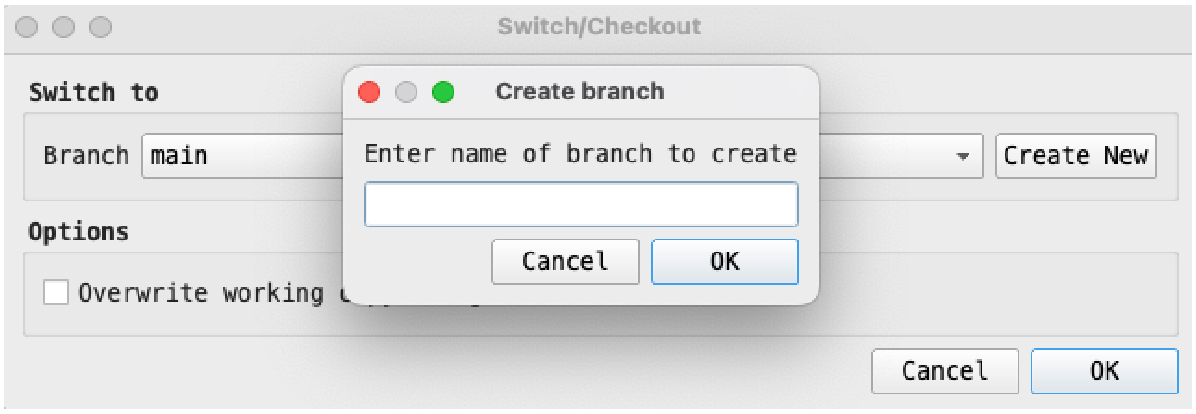
3.6.1 Branching

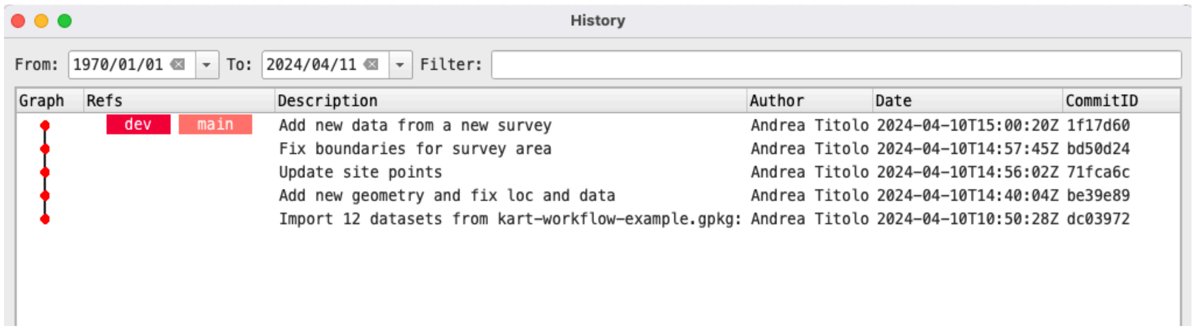

Kart repositories

Tip: right-click on items for available actions

Repositories
[redacted]
[redacted]/kart-workflow/kart-tutorial [main]
Datasets
archaeo_sites
archaeo_survey_areas
archaeo_locationquality
archaeo_periodisation
archaeo_references
archaeo_sitemorphology
archaeo_sitesubtype
archaeo_sitetype
archaeo_sizequality
archaeo_status
archaeo_surveytype
layer_styles

Show log...
Show working copy changes...
Discard working copy changes
Commit working copy changes...
Switch branch...
Merge into current branch...
Pull...
Push...
Import dataset from file...
Import dataset from database...
Apply patch...
Refresh
Properties...
Remove this repository





From: 1970/01/01 To: 2024/04/11 Filter:

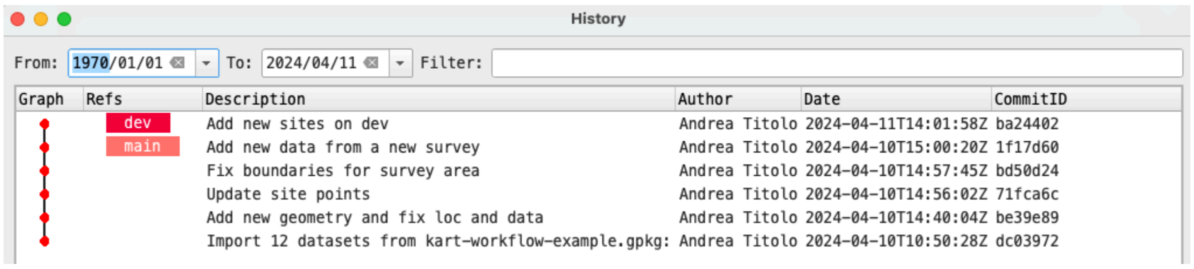
Graph	Refs	Description	Author	Date	CommitID
●	dev main	Add new data from a new survey	Andrea Titolo	2024-04-10T15:00:20Z	1f17d60
●		Fix boundaries for survey area	Andrea Titolo	2024-04-10T14:57:45Z	bd50d24
●		Update site points	Andrea Titolo	2024-04-10T14:56:02Z	71fca6c
●		Add new geometry and fix loc and data	Andrea Titolo	2024-04-10T14:40:04Z	be39e89
●		Import 12 datasets from kart-workflow-example.gpkg:	Andrea Titolo	2024-04-10T10:50:28Z	dc03972

Kart QGIS Plugin docs

Kart docs command reference

3.6.1.1 Add features on the new branch

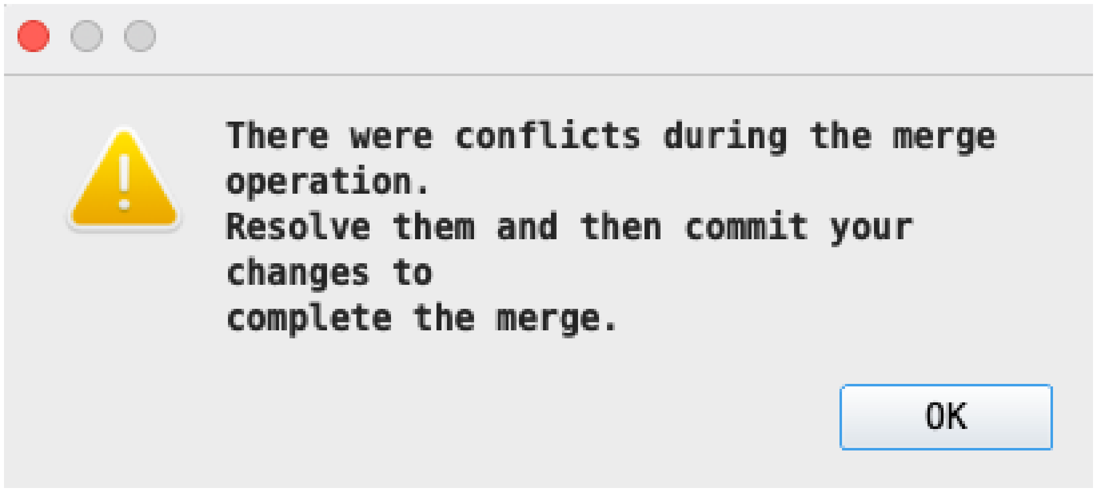
3.5.5



Graph	Refs	Description	Author	Date	CommitID
●	dev	Add new sites on dev	Andrea Titolo	2024-04-11T14:01:58Z	ba24402
●	main	Add new data from a new survey	Andrea Titolo	2024-04-10T15:00:20Z	1f17d60
●		Fix boundaries for survey area	Andrea Titolo	2024-04-10T14:57:45Z	bd50d24
●		Update site points	Andrea Titolo	2024-04-10T14:56:02Z	71fca6c
●		Add new geometry and fix loc and data	Andrea Titolo	2024-04-10T14:40:04Z	be39e89
●		Import 12 datasets from kart-workflow-example.gpkg:	Andrea Titolo	2024-04-10T10:50:28Z	dc03972

3.6.1.2 Add features to the main branch

3.6.2 Merging


There were conflicts during the merge
operation.
Resolve them and then commit your
changes to
complete the merge.

```
kart merge dev

Merging branch "dev" into main
Conflicts found:

archaeo_sites:
    archaeo_sites:feature: 2 conflicts

Repository is now in "merging" state.
View conflicts with `kart conflicts` and resolve them with `kart resolve`.
Once no conflicts remain, complete this merge with `kart merge --continue`.
Or use `kart merge --abort` to return to the previous state.
```

3.6.3 Dealing with conflicts

1.
2.
3.

3.6.3.1 Viewing conflicts

Note

Currently we are unable to access the graphical tool due to a possible bug in kart, thus we will continue with the CLI only. We will update this tutorial with the relevant info once we get past this issue. If you don't want to use the CLI, you can try following the kart plugin docs about solving conflicts and then continue on to the next section.

Expand

```
kart conflicts
archaeo_sites:
    archaeo_sites:feature:
        archaeo_sites:feature:1106:
            archaeo_sites:feature:1106:ours:
                                     fid = 1106
                                geometry = POINT(...)
                                    Name = A site on main branch
                                 NameAlt =
                             AncientName =
                                    Type = Building
                              Morphology = Flat site
```

3.6.3.2 Solving conflicts

`kart revolve --with=(ancestors|ours|theirs|delete)`

pull request

0.12.3

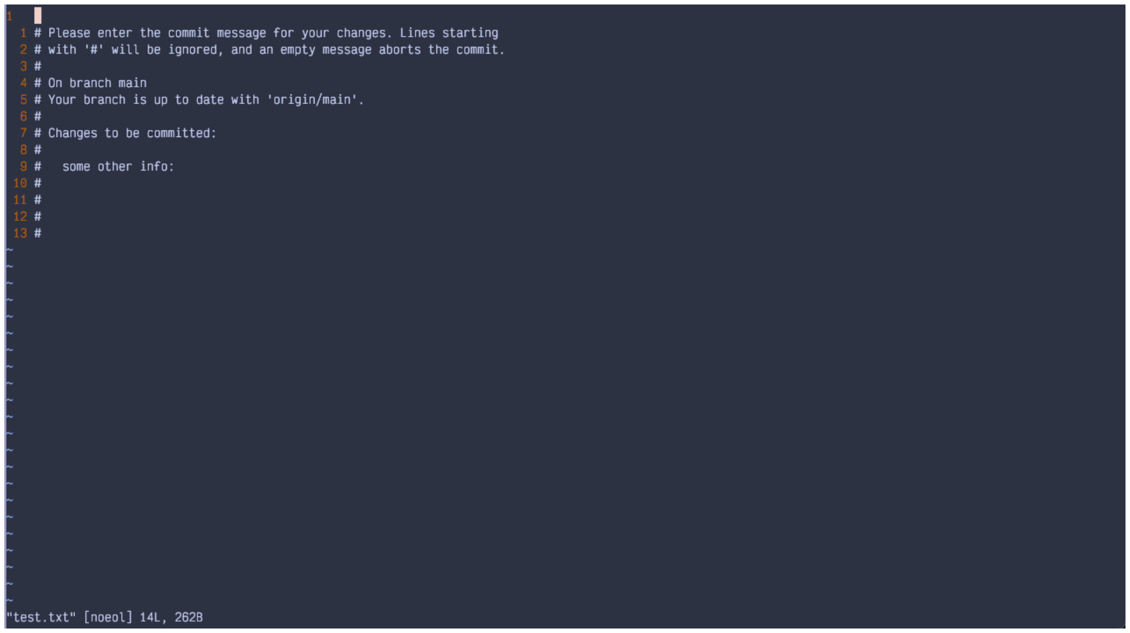
! Caution

If there is only a partial overlapping of primary keys, kart will merge those keys that can be merged, while the other features will wait for the conflict resolution to be renumbered. This means that if you have some keys in the `yourbranchname` branch that do not overlap with primary keys in your `main` branch, and you use `theirs` as an option for renumbering, these primary keys will be added first after the keys already present in `main` (as mentioned here). This will break the sequential order of features from the secondary branch, however, it is more of a visual issue than a real one, unless you are relying on the order of those primary keys for other operations.

```
kart resolve --renumber=theirs
Resolved 2 conflicts. 0 conflicts to go.
Use `kart merge --continue` to complete the merge
```

3.6.3.2.1 A note about terminal text editors

With some possible differences, the interface will be like this for vi/vim

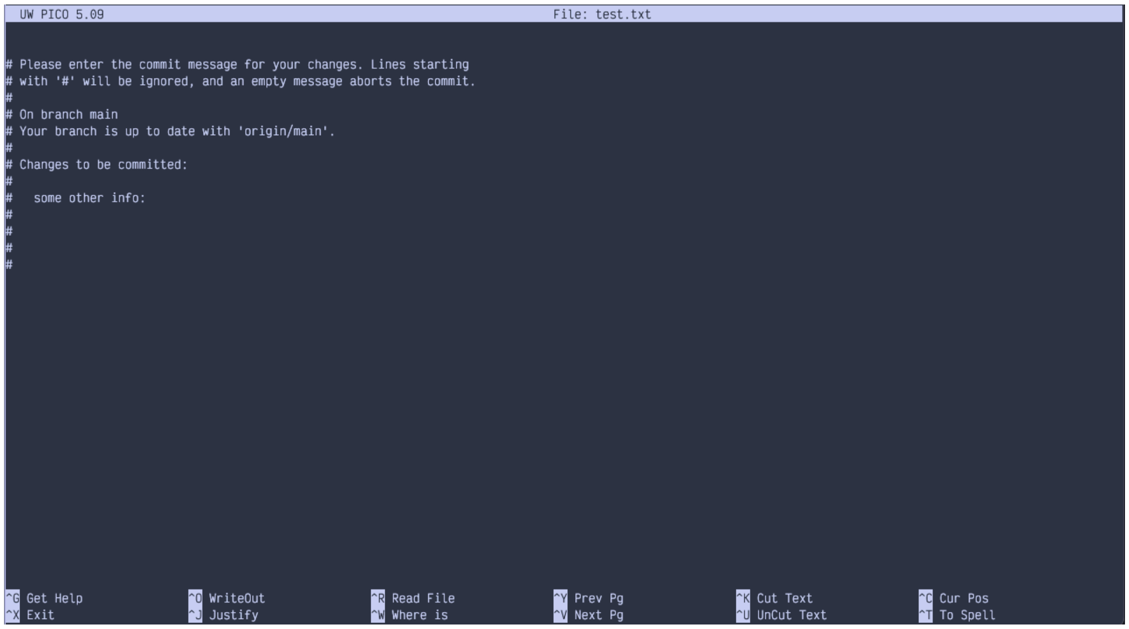


```

# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#
#   some other info:
#
#
#
#
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"test.txt" [noeol] 14L, 262B
```

Figure 1: vi/vim interface

And like this for nano/emacs:

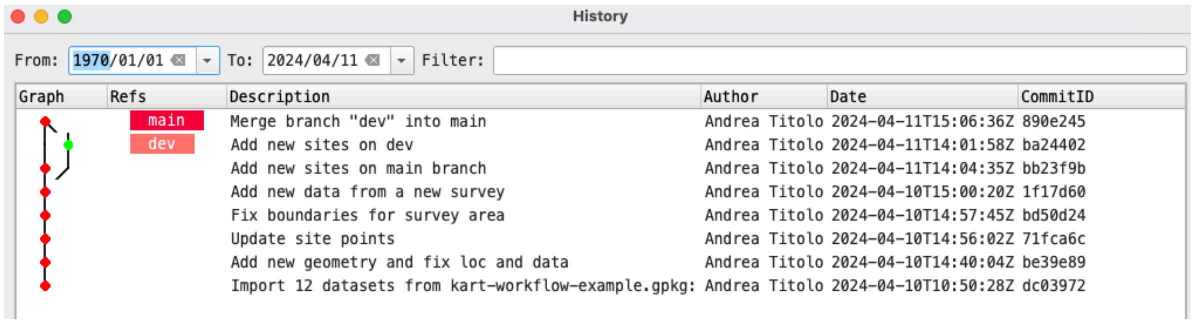


```
  UW PICO 5.09                                                        File: test.txt


# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch main
# Your branch is up to date with 'origin/main'.
#
# Changes to be committed:
#
#   some other info:
#
#
#
#
```

Figure 2: nano/emacs interface

```
Merging branch "dev" into main
No conflicts!
Merge committed as 890e2455fe81f0e58c5c5bed8cb7010e4fb174f8
Updating kart-tutorial.gpkg ..
```



Graph	Refs	Description	Author	Date	CommitID
	main	Merge branch "dev" into main	Andrea Titolo	2024-04-11T15:06:36Z	890e245
	dev	Add new sites on dev	Andrea Titolo	2024-04-11T14:01:58Z	ba24402
		Add new sites on main branch	Andrea Titolo	2024-04-11T14:04:35Z	bb23f9b
		Add new data from a new survey	Andrea Titolo	2024-04-10T15:00:20Z	1f17d60
		Fix boundaries for survey area	Andrea Titolo	2024-04-10T14:57:45Z	bd50d24
		Update site points	Andrea Titolo	2024-04-10T14:56:02Z	71fca6c
		Add new geometry and fix loc and data	Andrea Titolo	2024-04-10T14:40:04Z	be39e89
		Import 12 datasets from kart-workflow-example.gpkg:	Andrea Titolo	2024-04-10T10:50:28Z	dc03972

Kart QGIS plugin docs Kart docs Wiki
on the same topic github issue

3.6.3.3 Delete branches

Note

You cannot delete the branch you are currently on, so you will need to switch to another branch in order to delete it.

```
Deleted branch dev (was ba24402).
```

```
kart branch -a
* main
```

❗ Important

If you do not delete you development branch, remember to start new works by creating new branches from the `main` one, otherwise your might branch from outdated dataset!

3.6.4 Push changes to a remote repository

3.6.4.1 Connect to a remote repository

3.4

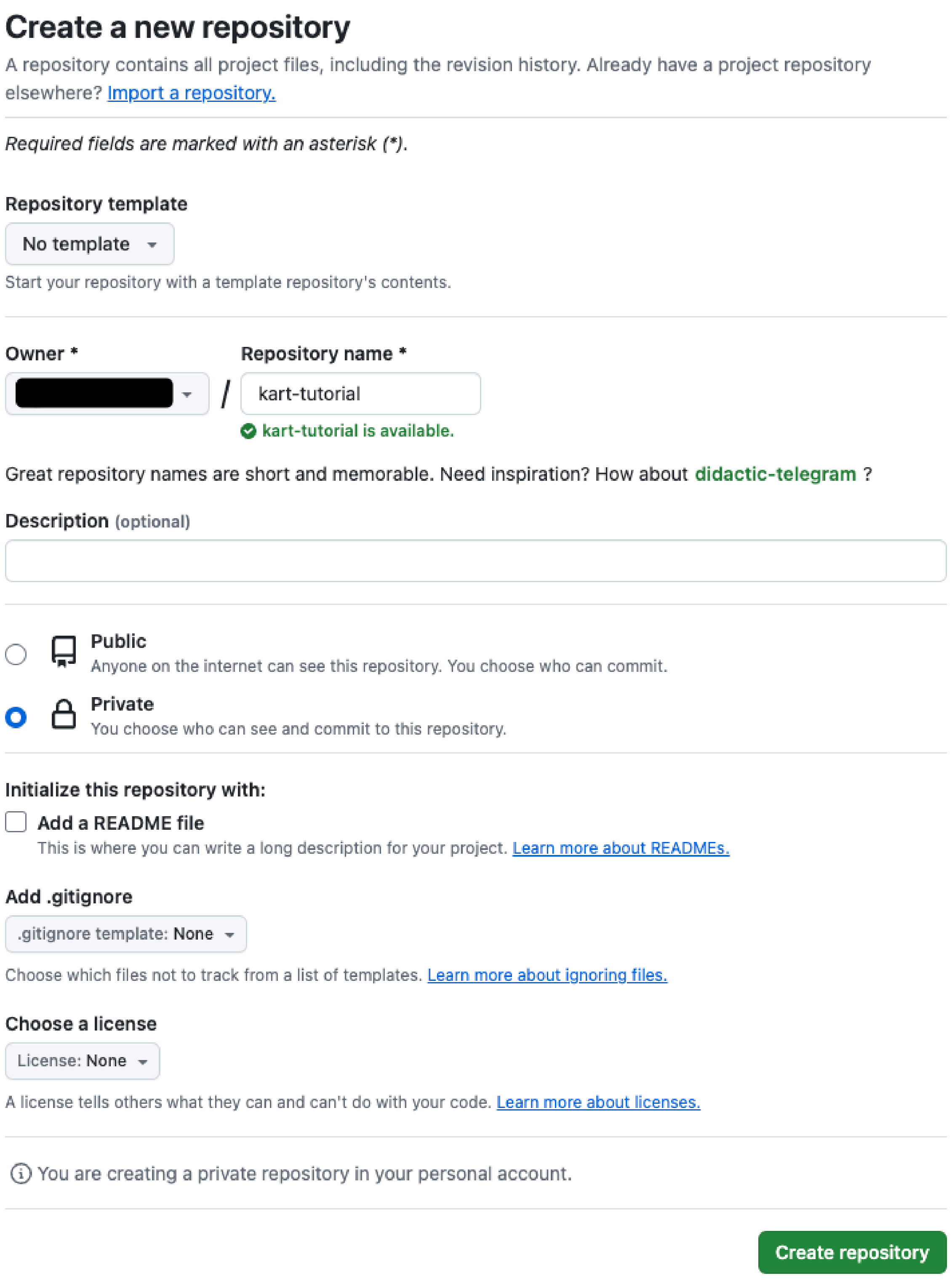
```
kart remote -v
origin  https://github.com/UnitoAssyrianGovernance/kart-tutorial.git (fetch)
origin  https://github.com/UnitoAssyrianGovernance/kart-tutorial.git (push)
```

3.6.4.1.1 Create a remote repository

instructions

⚠ Warning

As a general rule of thumb, if you are creating a repository from one online forges, remember to NOT include any README.md, .gitignore, or other files. Best-case scenario they will be ignored by kart, but worst-case they might break your workflow, as kart will recognize an out-of-sync remote with no real means of fixing it.


Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? Import a repository.

Required fields are marked with an asterisk (*).

Repository template

No template

Start your repository with a template repository's contents.

Owner * [redacted] / **Repository name *** kart-tutorial

kart-tutorial is available.

Great repository names are short and memorable. Need inspiration? How about **didactic-telegram** ?

Description (optional)

( ) **Public**
Anyone on the internet can see this repository. You choose who can commit.

(●) **Private**
You choose who can see and commit to this repository.

Initialize this repository with:

☐ **Add a README file**
This is where you can write a long description for your project. Learn more about READMEs.

Add .gitignore

.gitignore template: None

Choose which files not to track from a list of templates. Learn more about ignoring files.

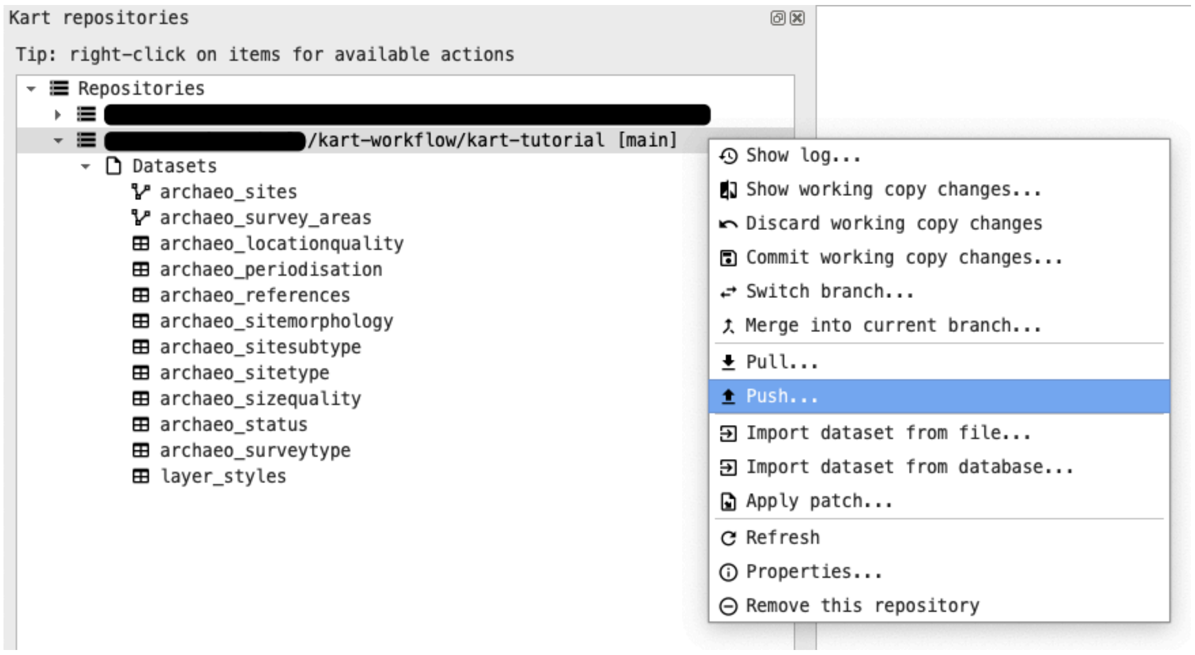
Choose a license

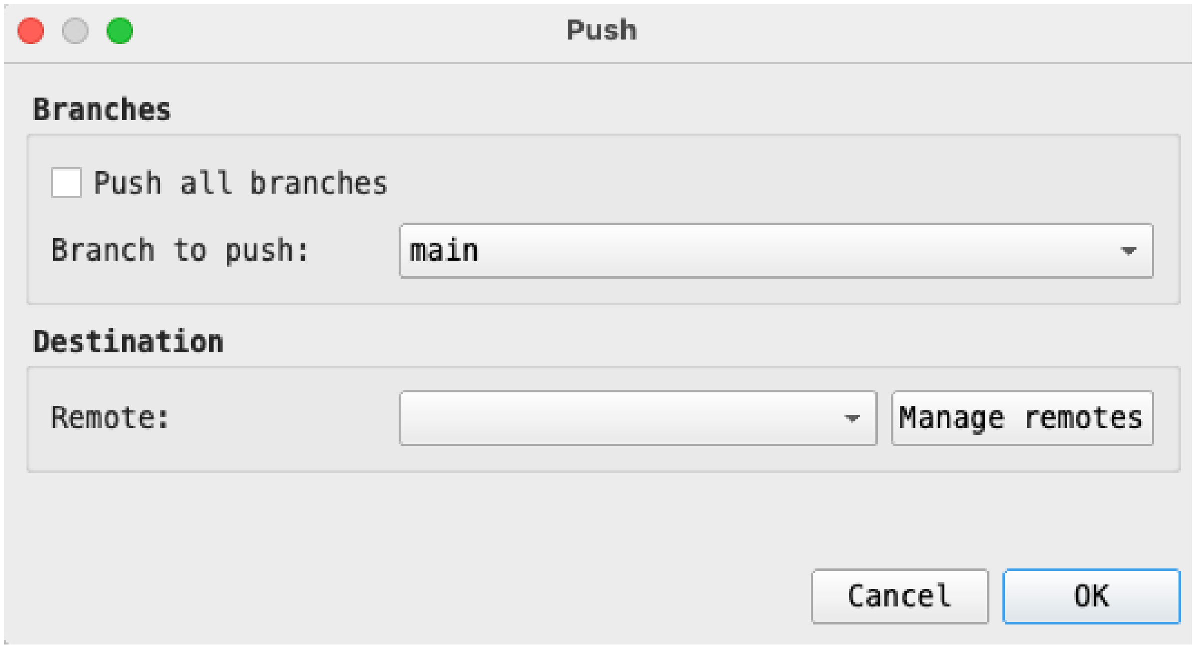
License: None

A license tells others what they can and can't do with your code. Learn more about licenses.

You are creating a private repository in your personal account.

3.6.4.1.2 Connect with the kart QGIS plugin




Push

Branches

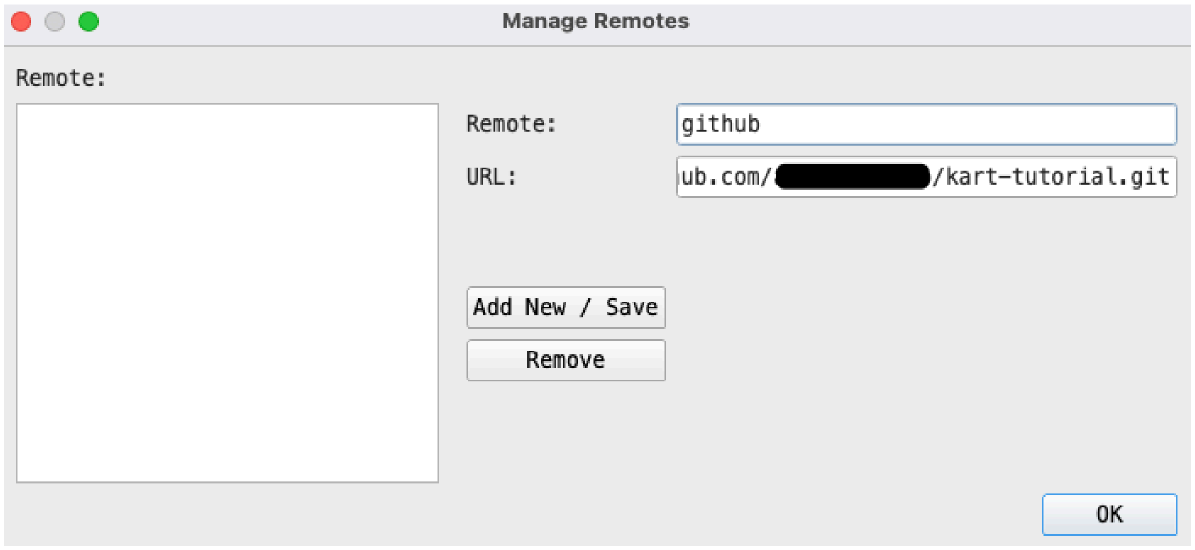
- [ ] Push all branches

Branch to push: main

Destination

Remote:

3.2


Manage Remotes

Remote:

Remote: github

URL: ub.com/[redacted]/kart-tutorial.git

3.6.4.1.3 Connect with the kart CLI

> **Note**
>
> `origin` is a convention, you can name your remote however you want, just make sure to use the name you choose in the next commands.

3.6.4.2 Push changes

1.

2.
3.

```
kart push -u origin main

Enumerating objects: 892, done.
Counting objects: 100% (892/892), done.
Writing objects: 100% (892/892), 226.96 KiB | 113.48 MiB/s, done.
Total 892 (delta 0), reused 892 (delta 0), pack-reused 0
To https://github.com/yourusername/kart-tutorial.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Note

In the same way, if you know that your remote has more recent changes that your local machine, you can use `kart pull` or the respective button in the kart plugin to pull changes from remote.
It is always best to pull changes if you are unsure someone made edits before your work!

3.6.4.3 Addendum: Interact with remote branches

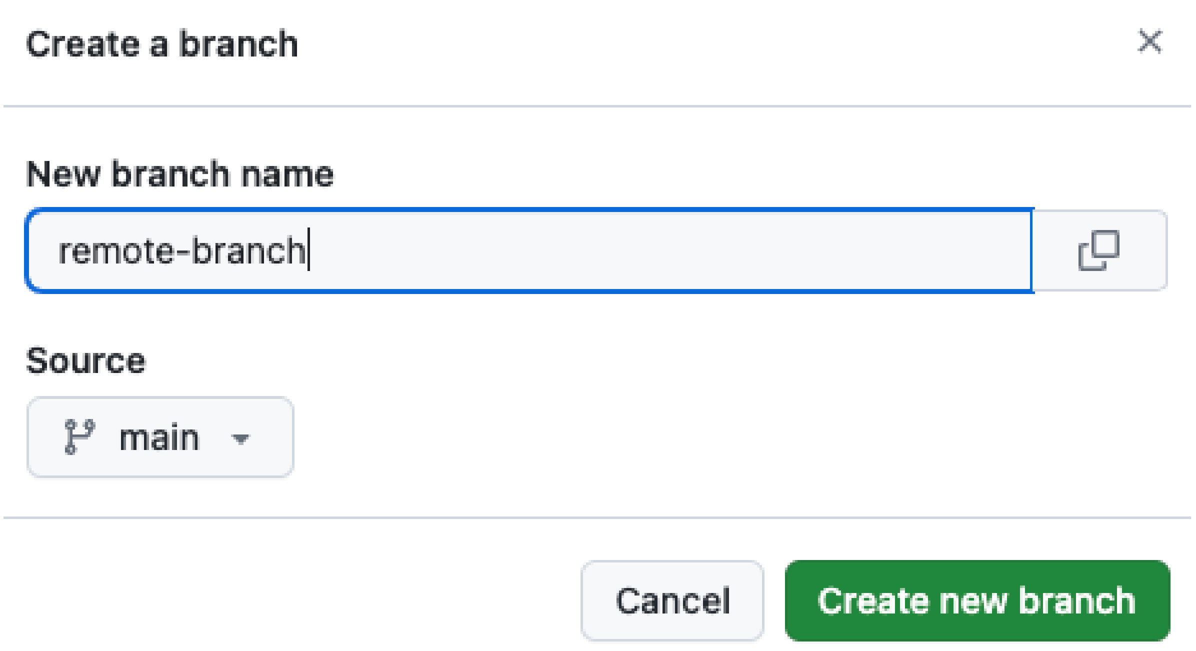
3.6.4.3.1 Push new branches to remote

```
kart push -u origin feature

Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'feature' on GitHub by visiting:
remote:      https://github.com/yourusername/kart-tutorial/pull/new/feature
remote:
To https://github.com/yourusername/kart-tutorial.git
 * [new branch]      feature -> feature
branch 'feature' set up to track 'origin/feature'.
```

3.6.4.3.2 Pull new branches from remote

docs for github



```
kart pull

 * [new branch]      remote-branch -> origin/remote-branch
Merging 890e245 into feature
Already up to date
```

```
kart branch -a

* feature
  main
  remotes/origin/feature
  remotes/origin/main
  remotes/origin/remote-branch
```

```
kart switch remote-branch

Creating new branch 'remote-branch' to track 'origin/remote-branch'...
```

3.6.4.3.3 Delete local and remote branches

1.

2.

3.

```
kart branch -d remote-branch

Deleted branch remote-branch (was 890e245).

kart push origin --delete remote-branch

To https://github.com/yourusername/kart-tutorial.git
 - [deleted]         remote-branch
```

4 Tips

4.1 Selective import tables

4.2 Rename data on import

4.3 Create data from QGIS and start version them with kart

github issue

1.

2.

3.

```
On branch main
Your branch is up to date with 'origin/main'.

Nothing to commit, working copy clean

Untracked tables:
  THE UNTRACKED LAYERS
```

4.

4.3.1 A note on the commit message when running `kart add-dataset`

3.6.3.2.1

4.3.2 Addendum - layer styles

Warning

Note that, at the time of writing (2023-11-15), due to a small mishandling of layer_styles schema by QGIS, if your layer name is longer than 30 characters and your style is named with the same name as the layer, you will incur into an error when trying to run `kart add-dataset layer_styles`. As explained in this issue, this is not a kart bug. We recommend keeping the layer style name (or the layer names) below 30 characters to avoid issues.

5 Useful Resources

- Kart website

5.1 Documentation

- Kart documentation
- Kart command reference
- Kart QGIS plugin documentation

5.2 Other resources

5.2.1 Presentations about kart

- Kart: An introduction to practical data versioning for geospatial
- Kart: A Practical Tool for Versioning Geospatial Data
- 2023 QGIS Data Versioning with Kart
- Spatial data versioning with the Kart QGIS Plugin

5.2.2 Guides and tutorials about Git

- Understanding Git: A Beginner's Guide to Version Control (With Visuals)
- Getting Git Right
- Git: the simple guide
- Git Official Documentation with useful resources
- Other tutorials from the Git website

5.2.3 QGIS

- QGIS Website
- QGIS User Guide
- QGIS Training Manual
- A Gentle Introduction to QGIS