1. Kartoza Handbook

This site comprises the organisational and technical documentation for Kartoza.

This is where we highlight the procedures, principles, and processes related to Development, DevOps, and GIS, in line with the organisations best practices.

This is open content, available on GitHub and freely licensed as public domain content under the terms of CC0 1.0 Universal.

!!! danger "DISCLAIMER"

    This content is delivered without any warranty, express or implied. Use at own risk.

0.1. Purpose

The purpose and function of this collection of documents is to perform the following:

- Improve consistency in processes and products
- Improve efficiency and innovation
- Increase transparency and accountability
- Improve value for clients
- Provide a space for the dissemination and proliferation of ideas
- Promote a culture of openness and collaboration
- Provide a single source of truth for resources
- Promote personal growth and development
- Add value to the community

0.2. Scope

Kartoza is a company that specializes in Open Source Geospatial solutions. As a result, topics covered by this documentation will be limited to categories relevant to the operations of Kartoza. These categories are outlined as follows:

- GIS: Resources for Geographic Information Systems and data
- Development: Software development processes, tools, and conventions
- DevOps: Developer operations and system administration
- Resources: Cheatsheets, links, media, and other resources

This repository is limited to these categories, with some slight overlap in domain verticals.

In the majority of instances, where overlaps with adjacent fields of interest, such as Data Science, "Big Data", or to some extent even earth observation and remote sensing, these elements should be primarily remanded to external references in the resources section.

Whilst some resources (such as tutorials on Semi-Automated Classification with QGIS) may be considered valuable additions, the priority of this content is to remain a concise collection of resources directly related to the operations and key competencies of Kartoza staff.

0.3. Limitations

This collection of documents does not constitute a replacement for Standard Operating Procedures (SOPs) and company policy. Due to the rate at which modern technology develops, opinions change, project needs or priorities are adjusted, and the operational requirements of the organisation evolve, this collection is somewhat ephemeral and should be considered a dynamic "living document" which is subject to constant change and iteration.

Processes and documentation from this collection are developed in conjunction with the broader community, independant contractors, temporary staff, juniors, and interns. As

such they are not guaranteed to reflect the views of Kartoza, and are not intended to be a substitute for official policy.

As resources and processes mature, they may be incorporated into official SOPs as required.

1. About Kartoza

Kartoza is a South Africa-based Free and Open Source GIS (FOSSGIS) service provider. We use GIS software to solve complex location-related problems for individuals, businesses and governments around the world.

Kartoza was formed as a merger between Linfiniti and Afrispatial.

KARTOZA
OPEN SOURCE GEOSPATIAL SOLUTIONS

Learn more at https://kartoza.com

1. Contributing

Although Kartoza is a privately held Open Source development and consulting company, the organisation deeply values transparency, delivery of value to the broader community, and continuous engagement with all stakeholders.

Community contributions to this documentation site and associated resources are welcome. Contributions are expected to adhere to the QGIS.org Diversity Statement and Code of Conduct.

!!! info

If you have any queries or feedback, please contact us at info@kartoza.com

0.1. Conventions

The following conventions outline expectations for contributions to this documentation project:

- Use grammar checking tools where available, such as grammarly or spell checking extensions for your IDE
- Request a review for internal changes before they are merged into the main repository
- Default to British English spellings rather than American English
- Do not commit sensitive information or links to non-public resources. This includes internal unlisted youtube channels, cloud storage repositories such as nextcloud, or personal details
- Due to the nature of the contents in this repository, when making large edits that do not create new content, communicate with team members to prevent collisions
- When producing assets such as images, ensure they are the minimum viable size and do not commit large resources to git

- Assets and media elements such as images are best left out of the source control where possible. Use an external storage system (e.g. minio/ s3), or paste an image into an image to get a GitHub reference to the media item rather than committing to git. This includes screenshots etc. that are likely to change or be updated over time. Use the assets directory to store assets that are not likely to change such as logos
- When using assets, upload them to a suitable file path according to their primary usage location, e.g. `assets/images/resources/cheatsheets/postgresql/joins.png`
- Due to the depth and breadth of these resources, it is necessary to manually index new pages in various subcategories to ensure access and discoverability
- It makes sense to use a consistent legend of emoji for tagging project and documentation items. Although it may have a steep initial learning curve or implementation strategy, using emoji and unicode symbols to tag elements is a fun and intuitive way to attach metadata to elements which makes visually scanning over documents and commit histories much more effective in multiple languages. Please see the polyglot document for more information.
- Build and check your changes locally to catch any errors before committing them to the main repository
- TODO: come up with some formatting guideline (e.g. max line length etc)

0.2. Tags, Badges, and Shields

Tagging elements with emoji is useful for visual identification and search of various elements in broader categories, but sometimes more explicit metadata is required to be attached to something to indicate whether it constitutes a general resource, opinion, community standard, or whether something is a known reference item connected to an official SOP. One method of identifying such features may be using shields.io, for example:

- 

`![Best Practice](https://img.shields.io/badge/kartoza-best--practice-blue)`
- 

`![Community](https://img.shields.io/badge/community-standard-brightgreen)`

-  `![Industry](https://img.shields.io/badge/industry-standard-yellowgreen)`

0.3. Translations

Due to the scope and intension for frequent updates to this documentation, additional languages will not be supported at this time.

Translations and i18n are handled by the documentation framework, as outlined in the mkdocs and mkdocs-material documentation.

0.4. Framework

This documentation uses the mkdocs-material framework, and site configuration is specified in the mkdocs.yaml file. Various extensions are supported to improve usability, such as pymdown, which may be enabled via pull requests. Please note that only extensions which provide relevant value will be considered for integration, and extensions with significant learning curves or duplication should be avoided. Extensions which provide accessibility improvements are welcome.

0.5. Building

The online documentation is built using github actions and published to the gh-pages branch.

To build the documentation locally, use the docker command `docker run --rm -it -v ${PWD}:/docs squidfunk/mkdocs-material build` to populate the site directory with the static content. To serve the data for testing, a simple solution is to use a python webserver to serve the data at `127.0.0.1:9101` using the command `cd site && python -m http.server --bind 127.0.0.1 9101`.

Note that the generated site data and assets are explicitly excluded from git.

1. Polyglot: The Emoji Map

Although it may have a steep initial learning curve or implementation strategy, using emoji and unicode symbols to tag elements is a fun and intuitive way to attach metadata to elements which makes visually scanning over documents and commit histories much more effective in multiple languages.

In order for this to be effective, a consistent method of referencing the emoji meanings is required. This is challenging because the utilisation of emoji are typically context specific, which requires a mapping of emoji meanings for various contexts.

> Kartoza values inclusion and diversity. Please contribute to ensure that the items represented here remain inclusive and fair wherever possible.

0.1. gitmoji

Tagging commit messages in git is a useful tool for visually assessing the issues addressed by a particular commit.

A community standard has already been developed, available at https://gitmoji.dev/

1.1. kitmoji (gitmoji-)

gitmoji is fairly comprehensive collection, and used quite widely in the tech community (relative to similar projects). The downside to this is that it is rather verbose and becomes cumbersome to learn and use.

The simple solution is to select a subset (kit) of the gitmoji icons and use them as broader higher level categories. This keeps things a bit more consistent and allows gradual adoption of the wider collection.

id	Icon	Reference	Function
1	✨	:sparkles:	New features
2	🐛	:bug:	Bugfix
3	♻️	:recycle:	Refactoring/ Comments
4	📝	:memo:	Documentation
5	💄	:lipstick:	UI
6	⚡	:zap:	Performance
7	🚀	:rocket:	CI/ CD/ Deployment
8	✅	:white_check_mark:	Testing
9	🔒	:lock:	Security
10	🔥	:fire:	Remove data
11	⏪	:rewind:	Revert changes
12	⚗️ / 💩	:alembic: / :poop:	Experiments/ PoC/ bad code

Not every commit has to have a gitmoji, but it's useful for common cases.

0.2. GeoMoji

As with gitmoji, we need a consistent way to reference Geographic Information Elements. This could be related to data sources, licenses, tools, or standard metadata categories. Some may confuse the concept of geomoji with generic symbology and

signage, but in this instance the reference is to a series of common Emoji characters which can be used as concise, visually effective hashtags that can be used across across applications, search tools, git messages, documentation, or social media. It would be great to extend this concept to a collection of map symbols (e.g. using emoji instead of a font library).

A alpha-state concept project is in development at https://github.com/zacharlie/geomoji.

1. GIS

Resources, information, and processes related to Geographic Information Systems

???+ tip "Security"

    Note that all users, regardless of role, should understand and review the [security](../devops/security) section.

GIS is a core competency at Kartoza. All technical staff should endeavour to ensure that they have or develop some level of proficiency in GIS.

- Cartography
- Technologies
- Resources

1. Cartography

Guidelines on how to create effective cartography.

Information on the development of cartographic style guides.

1. Technologies

Outline of Key GIS Technologies that all GIS practitioners should be familiar with.

Outline GIS technologies and concepts for developers to get up to speed quickly.

1. GIS Resources

Where to find internal resource collections (projects, templates, data, assets etc)

External resource links

1. Development

???+ tip "Security"

    Note that all users, regardless of role, should understand and review the [security](../devops/security) section.

Kartoza is a consulting and development firm. In many instances, projects require developers to leverage existing tools and codebases, and to work with other organisations in a way that is consistent in their own conventions.

This documentation seeks to outline processes and practices so that internal project development and practices within the team remain consistent. Where it is noted that beneficial conventions listed here are not implemented in client projects, recommendations may be made that clients adopt such standards.

For the most part, however, it is less a concern of how a particular outcome is achieved, but rather that the result is consistent with the appropriate conventions.

- Conventions
- Technologies
- Environments

1. Conventions

https://kartoza.com/en/coding-standards/

- IDEs
- Processes
- Git

0.1. Project Conventions

SDLC, agile, scrumboards: i.e. project processes

0.2. Language Conventions

A foolish consistency...

2.1. Python

python specific content

2.2. JavaScript

js specific content

2.3. C++

c++ specific content

1. Technologies

It makes sense to attempt some level of standardisation across development practices, including technologies and frameworks in use, so that the barrier for entry to projects is lowered and it becomes simple to maintain consistent practices across the organisation.

- Languages
- Frameworks

0.1. Evaluating

It may be helpful to include some "smell tests" for evaluating good projects or technologies

1. Environments

Development environments configurations and conventions (conda/ venv/ poetry/ git codespaces/ docker workspaces etc)

Links and external resources

1. DevOps

Resources, information, and processes related to DevOps and system administration.

???+ tip "Security"

    Note that all users, regardless of role, should understand and review the [security](./security) section.

Note that this is not limited to DevOps team members and may include conventions and configuration for the management of workstations, servers, and other systems, which is relevant to multiple roles.

- Security
- Procedures
- Infrastructure

1. Security

- Cyber Hygiene Handbook
- Security Guidelines/ Rules
- Development Processes
- External Resources

0.1. Cyber Hygiene Handbook

'twould be handy to collect resources and advice on training users in best practices for cyber-hygiene (applies in broad strokes to staff, trainees, community, and more). Compiled into a nice concise resource with pretty pictures and minimal complexity++

Topics and structure could be outlined as follows:

Users:

- Identifying phishing attacks
- Credential management

Admins:

- Identifying whats on the network
- Account management
- Staying up to date

Developers:

- Authentication management
- Dependency management
- Scanning
- SSH, SSL, SSO (lots of SS), Cryptography, Hashes, Salts

1. DevOps Procedures

DevOps tasks are handled by the DevOps board

1. Infrastructure

- Development Infrastructure
- Personal Infrastructure
- Containers
- Kubernetes
- Rancher
- Rancher Desktop
- Development
- Staging
- Production
- CI/CD

0.1. Rancher Topics

Using Rancher, K3S, and Longhorn for single node Kubernetes deployment

1. Resources

- Cheatsheets
- External Resources
- Kartoza Media Center
- Tutorials

1. Cheatsheets

??? danger "Work in Progress"

    This article is under heavy development and is not considered production ready

Cheatsheets and snippets for important and common operations.

- PostgreSQL
- BASH
- Kubectl
- OSGeo4W

1. Tutorials

Tutorials.

- General: General tutorials
- QGIS: Tutorials related to QGIS
- Links: Links to third party and external tutorials and learning resources

1. Kartoza Media Center

Welcome to the Kartoza Media Center.

Here you can find links to media releases, social media feeds, and notices from Kartoza (Pty) Ltd

- Kartoza.com
- GitHub
- Twitter
- DockerHub
- Newsletters

1. Links

Links to interesting external resources