



Sakuli

What is it? And why do I want to use it?

*Simon Hofmann
Software Engineer*

04.06.2019

Agenda

- Sakuli: What is it?
 - Tech Stack
 - Architecture
 - Sakuli Lifecycle
 - Plugin System
- Sakuli: Why to use it?
 - Web Testing
 - Native Testing
 - Encryption



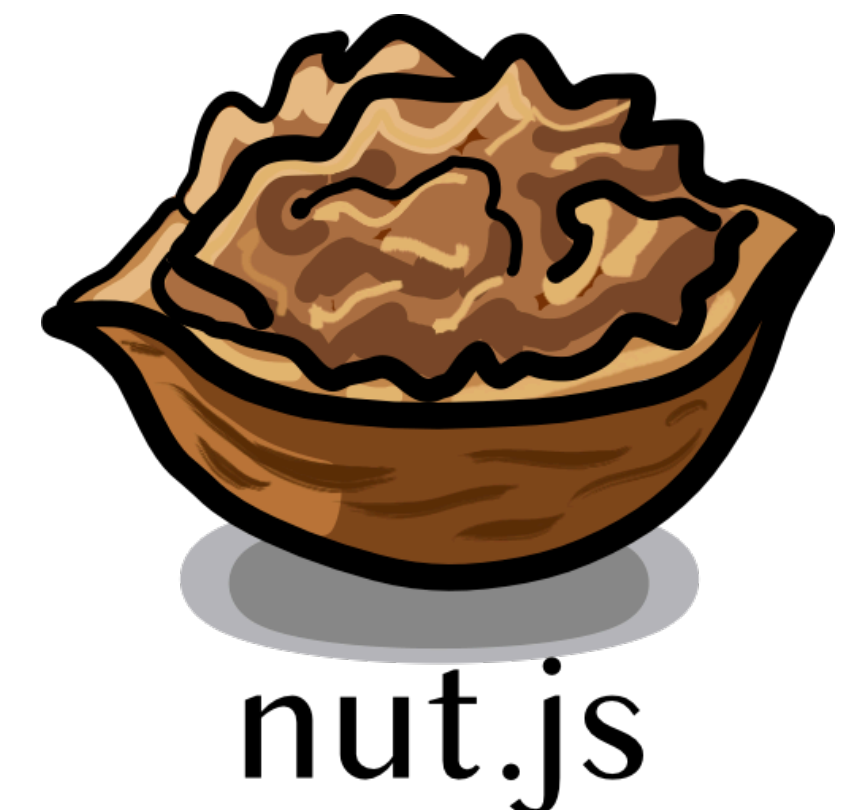


Sakuli: What is it?

Sakuli - Tech Stack



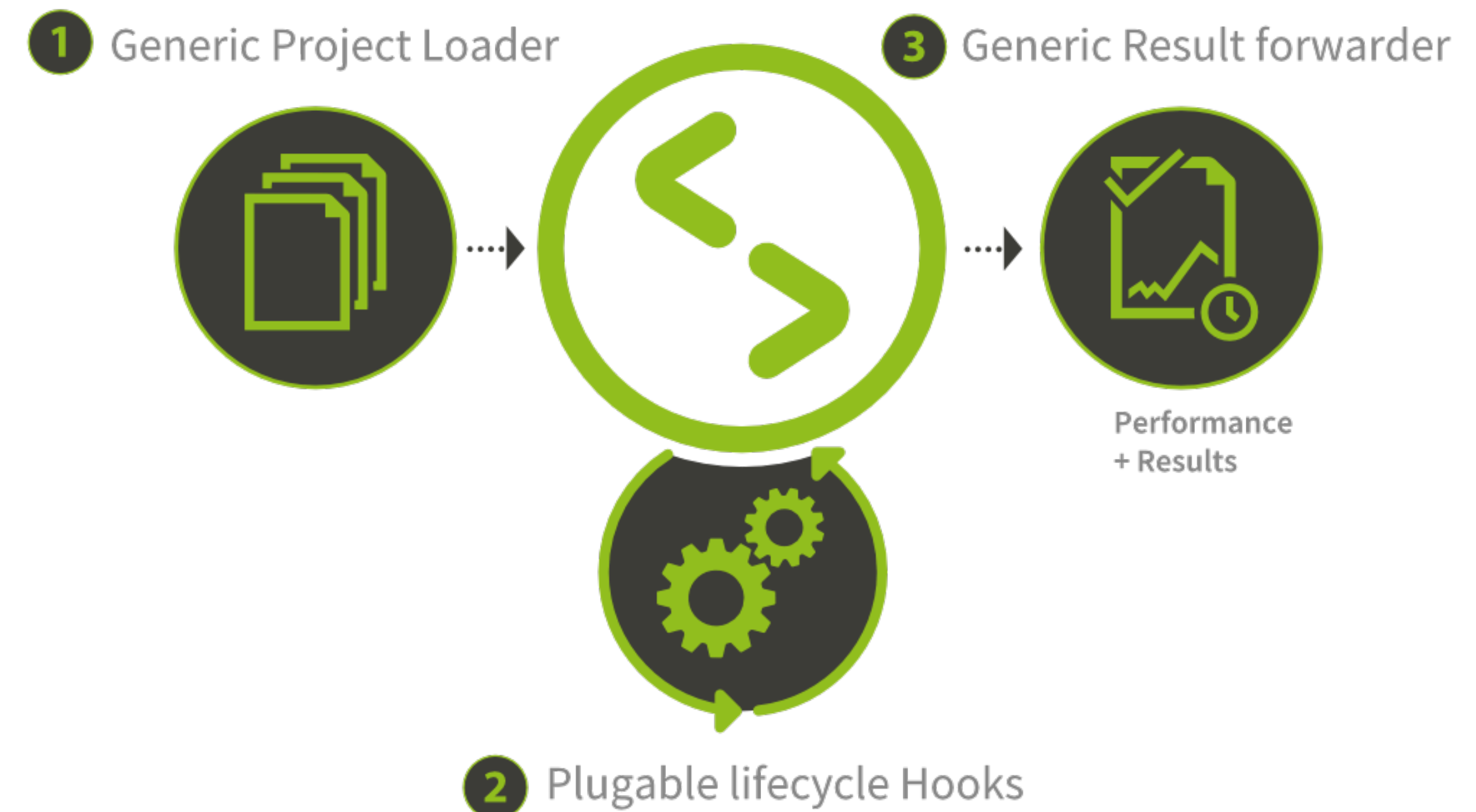
- Runtime: Node.js
- Web Testing: Selenium / Webdriver
- Native Testing: nut.js
- Programming Language
 - Framework: JavaScript (TypeScript)
 - Tests: JavaScript
- Installation: npm



Sakuli - Architecture



- Sakuli focuses on flexibility and extensibility
- At its core, Sakuli is a generic testrunner
- Processes combinations of
 - Project Loader
 - Context Provider
 - Result Forwarder
- Offers flexible configuration and own extensions



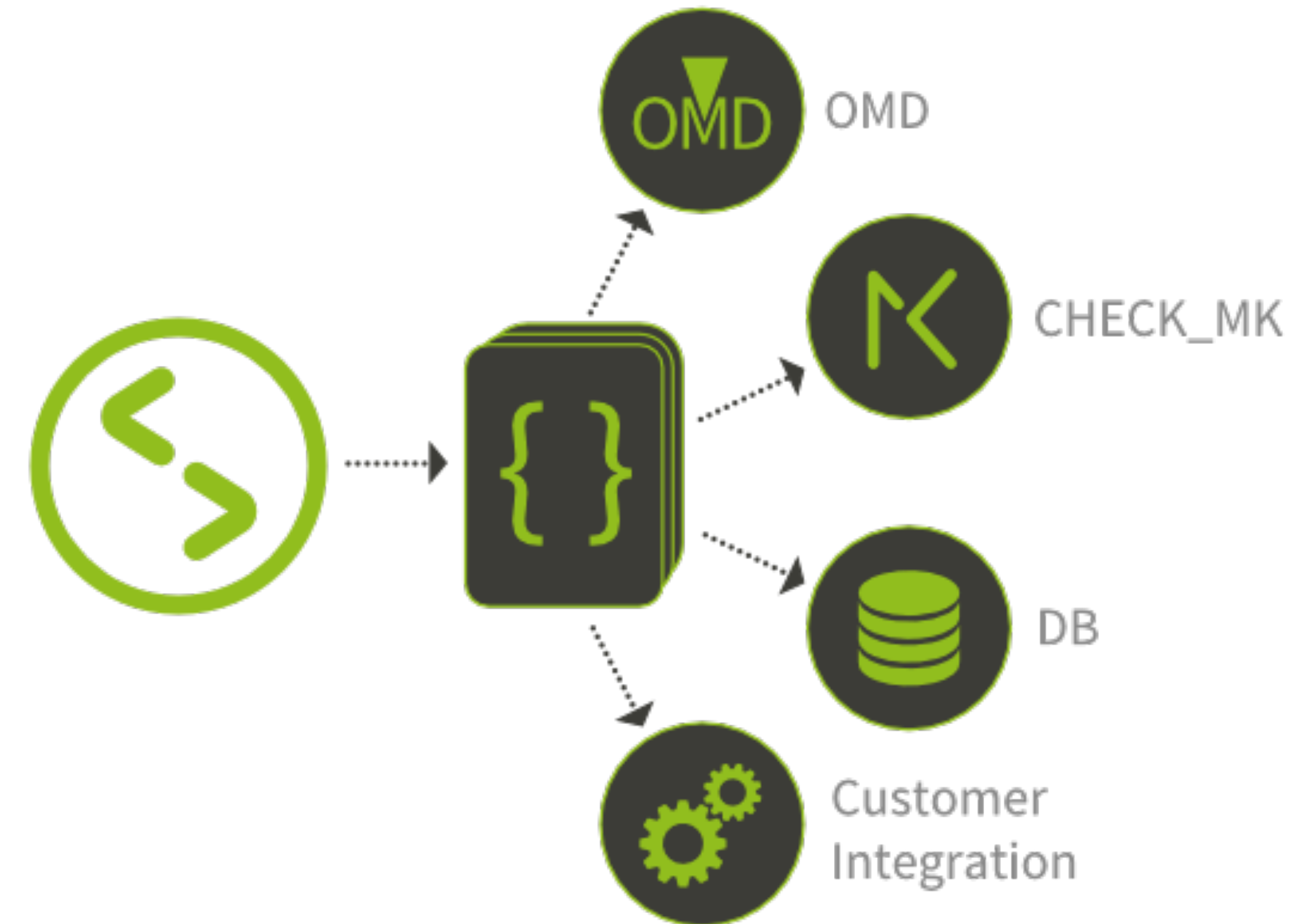
- Project loader provide abstract “Projects”
- Define how
 - Testfiles are read
 - Properties are loaded
- Own implementations of project loaders allow for an easy integration of Sakuli with 3rd-party systems

- Context Providers provide the runtime environment for a test
- Defines, which components will be available at runtime
- Legacy Context provides e.g.:
 - Environment
 - Region
 - TestCase
 - etc.
- Able to react on lifecycle events

Sakuli - Result Forwarder



- Test results are provided as data object
- Plugins are able to consume this data object via interface
- Forwarders implement logic to process and forward test results to 3rd-party systems
- Forwarding will be done automatically during test execution



- Test execution follows a lifecycle
- Context Provider might react to certain lifecycle events
- Available “Lifecycle Hooks”:
 - ***onProject***: Project successfully loaded and available
 - ***beforeExecution***: Testsuite execution will start, but no test has been executed yet
 - ***beforeRunFile***: Testfile execution will start, but no test has been executed yet
 - ***readFileContent***: Testfile is read, input for test execution
 - ***afterRunFile***: Testcode in a file has been executed
 - ***afterExecution***: All tests of a project have been executed



- Sakuli plugins are installed like every other dependency (npm install ...)
- Plugins:
 - Register themselves at the Sakuli plugin registry
 - Will read their required properties
 - Are executed during the Sakuli lifecycle



- Professional support
- VNC Docker Images to effortlessly scale your E2E tests
- Monitoring Forwarder to continuously monitor your system from an end-user perspective
- Staggered offers:
 - # of instances
 - Support hours



Sakuli: Why to use it?

- Based on Selenium
- Configurable webdriver
- Takes care of annoying things:
 - Descriptive syntax
 - Advanced element querying
 - Automated retries
 - Waits on open network requests



But wait, there's more!

- Besides browser testing / automation, Sakuli goes one step further
 - Control your:
 - Mouse
 - Keyboard
 - Clipboard
 - Image based actions:
 - Click / Drag elements based on screenshot data
 - Cross platform, works on
 - macOS
 - Linux
 - Windows



One more thing




- Sakuli comes with a built-in encryption module
 - Create a masterkey
 - Encrypt your test secrets
 - Secrets will be decrypted right before using them in a test
- No secret management required
- No plain text secrets in your test files

 <https://sakuli.io>

 <https://labs.consol.de/sakuli/>

 <https://github.com/sakuli>

 https://twitter.com/sakuli_e2e



Any questions?



Thank you!



ConSol

Consulting & Solutions Software GmbH

St.-Cajetan-Straße 43

D-81669 München

Tel.: +49-89-45841-100

info@consol.de

www.consol.de

Twitter: [@consol_de](https://twitter.com/consol_de)