Scailable platform tiral terms & conditions

> Version 0.1.12, 03-03-2021.

Preliminaries

This agreement concerns the **trial** usage of our platform. You receive trial access after signing up for at https://www.scailable.net.

As stated, our platform is available for a free trial for startups and students; for others, the **Scailable platform license agreement** applies. If you are using our the licensed version of our platform and do not have access to our license agreement, please do contact us at go@scailable.net.

Terms of usage of the trial agreement

In this document, we describe the agreement we (Scailable BV, Eindhoven, the Netherlands) and you (the trial user of our product and services) enter into when you enroll in our trial; when you sign up for a Scailable student or startup account, you accept the applicability of the terms and conditions described in this document.

We first describe our product in more detail to introduce all the terms necessary to specify our agreement. Next, we describe both our responsibilities under this agreement (and the limits thereof), and yours. Finally, we discuss potential termination of this agreement.

If anything in this agreement is unclear to you, or if you seek an alternative agreement, do contact us. You can reach our support team at go@scailable.net.

> **NOTE:** At the bottom of this document you will find a list of terms used.

Our product

The Scailable **beta** program consists of several products / services (jointly called *services* in this document) that interact. Before we can clarify our commitments during your trial period, we need to have a shared understanding regarding the services Scailable provides. Note that you can start using our services after successfully applying for a Scailable trial account.

Functionally, the Scailable platform allows you to *transpile* (i.e., convert) an *algorithm* (i.e., a list of operations a computer carries out on some *input* providing some desired functionality) described in a specific language (e.g., `Python`, `R`, `C`, `ONNX`) to a *WebAssembly binary*. The transpiling is carried out by our *toolchains*. You can submit an algorithm in a pre-specified format to our toolchain(s) using standard REST requests (see our API docs), although we also provide user-friendly *packages* (such as the sclblpy `Python` package and the sclbl CLI tools) that allow you to update a (specific subset of) algorithms directly from, for instance, your `Python` workspace. Note that the set of algorithms our toolchains can transpile is currently limited---if you attempt to transpile an algorithm that is outside our capabilities, the toolchain will return an error.

The WebAssembly binary we create is an alternative specification of the same algorithm (i.e., offering the same functionality) that is (almost always) smaller in size (i.e., uses fewer bits to specify the functionality), is fully portable, and can (generally) be executed faster, with less computational overhead.

After transpiling, we will make this WebAssembly binary available for you to deploy to our *micro-containers*, or to execute on our servers using our *REST endpoint*s. We will refer to the execution of a WebAssembly binary (i.e., the optimized version of the user-specified algorithm) on a specific input as a *task*. The result of the task is called the *output*. Note that next to using our toolchains to create a WebAssembly binary, you can also create such an executable yourself. As long as this executable adheres to our specifications, it can be uploaded to our platform and executed in our micro-containers or using our REST APIs.

The two paragraphs above describe the core services of Scailable: we allow you to *a)* automatically generate efficient versions of your algorithms and *b)* subsequently deploy this efficient version of the alogrithm, securely and modularly, to our micro-containers. Additionally, during the trial period, it is possible to use our admin interface to administer (i.e., edit or delete) your algorithms and REST endpoints and assign your algorithms to specific micro-containers.

A practical example

To further clarify our services, we provide an applied example. Suppose you train a machine learning model to detect objects in images; a common Machine Learning task. You can now use Scailable *services* to deploy your model by following the following steps:

1. You can use our [sclblpy] *package* to upload your fitted model (the *algorithm*) to our *toolchain*.
2. Our *toolchain* will *transpile* your fitted model to a *WebAssembly* binary.
3. Next, you install the Scailable *micro-container* on the desired device on which you would like to run the algorithm.
4. Finally, you can use the admin interface to deploy the binary to the device and consume your tasks on the device.

> **NOTE:** The fitted model itself *is* an algorithm in the sense that it provides a list of operations that turn the input (the image) into the desired output (the detected object).

Our commitment

At Scailable, we are committed to delivering high quality and reliable products and services. However, during your trial period, it is understood by both parties entering into this agreement that we (Scailable) are making our services available for *temporary use that is solely aimed at evaluating whether or not you would like to enter into a license agreement*. Hence, you can try out our services to *evaluate* them; if you want to use our services in any production-level application it is understood that, before doing so, you will enter into a license agreement.

Note that although we are doing all we can to assure the quality and consistency of our services even fro trial users, we can not and do not provide *any guarantees* during the trial period. Explicitly, this means that:

1. We cannot and do not guarantee that the functionality provided by the WebAssembly binaries we generate is exactly the same as the functionality provided in your initial algorithm. Although the underlying process is deterministic, and we are extensively testing our toolchains, unexpected things can (and thus eventually will) happen: we encourage you to test the output generated by executing our WebAssembly binaries and we do not accept any liability for possible erroneous outputs.
2. We cannot guarantee that your task is executed within the micro-container (or, alternatively, on our servers) and thus that you receive your desired output on time. When you use one of our REST endpoints, we do not provide any performance guarantees (i.e., the time it takes to execute your task), nor can we guarantee that our servers are always available to run your task. If you use, in the trial stage, our micro-containers, we cannot and no not guarantee their functionality. We strive for as much uptime as possible, and we try to carry out your task as fast as we can. However, during your trial period, our services might be interrupted, or tasks might not get executed properly: we strongly encourage you to ensure that fallbacks are in place on your end to deal with a potential outage or failure to execute your task timely. We do not accept any liability for delayed or unfinished tasks.
3. We currently cannot guarantee that your algorithm and input data are secure and stored according to all rules and regulations that your algorithm and input data are susceptible to. Since effectively, we do not know the nature of your algorithm and input data, we do not know whether these are (e.g.,) subject to GDPR. Thus, it is your responsibility to comply to any regulations that apply during the trial period.
4. We can and do guarantee that we do not use your algorithm and input data for anything other than running your task / providing you with a sample of our functionality. Your input data is (temporarily -- these are purged every 30 days) stored in our server logs. We try to secure our services as much as we can, but we cannot guarantee this security is never breached. Your algorithm, i.e., the non-WebAssembly specification of your functionality is stored on our servers for 24hours and subsequently removed. We do not accept any liability for security breaches of your input data or algorithm.
5. We cannot guarantee that our services will be available in the future in the same form as they were during your trial period. If any of our services become (temporarily or indefinitely) unavailable we will try to give as much prior notice as possible, and we will try to allow you to download the WebAssembly binaries we created for you. However, note that effectively our services could stop functioning at any time. We do not accept any liability for the termination of our services in the future.

To summarize: We seek to deliver an excellent service, and additionally will respond as quickly as possible to any issues/bug reports/feature requests, etcetera. However, during your trial period, we are unable to provide any guarantees.

> **NOTE:** If you need performance / legal / functional guarantees please contact us at go@scailable.net. We are happy to discuss such guarantees outside of the trial program when we enter into a license agreement.

Your commitment

We ask a few things from you as a member in your trial period:

1. Make sure you guard yourself (i.e., your applications and services that use Scailable) against potential downtime or erroneous output on our side.
2. Make sure you understand the regulations surrounding your algorithm and input data that apply wherever you are based. When you intend to use "sensitive" input data and need guarantees regarding the processing of this data, contact us.
3. If you find bugs, mistakes, etcetera, please let us know; we will try to fix them asap.
4. Feel free to test the limits of our services, but be friendly; if you find yourself able to hack into any of our services, please let us know. We will be proud of you, applaud your efforts, and subsequently fix the issue. If you intend to stress-test our services, let us know in advance, and we will monitor the performance together with you.
5. We expect "fair" usage of our services; if you understand (or become aware of) the fact that your actions (e.g., your uploads or task consumption) are slowing down our systems let us know; we will look for a solution together.
6. We expect you to enter into a license agreement at the moment you are using our services commercially (i.e., when your usage extends beyond a mere trial).

We can and will interrupt the services we deliver at any time if you violate our conditions as implied by the above. We seek to provide prior notice if we take such an action, but in the case of a severe breach of our agreement, we reserve the right to terminate your trial account immediately and indefinitely.

License breach

If you do end up using your trial account to provide otherwise licensed services (i.e., by using our platform, toolchains, webassembly binaries, or micro-containers for commercial use) we reserve the right to, even after the fact, charge you with our common license fees. If you are in doubt, please contact us.

Termination of this agreement

You can terminate this agreement at any time by simply refraining from using our services. Additionally, you can delete your account on the settings page in the admin interface. If you do so, we will contact you at most once using your provided email address to (e.g.) ask why you have stopped using our services, after which we will permanently delete your records (or within 90 days, whichever comes first).

We reserve the right to terminate this agreement, and hence our services, at any time. However, we will make an effort to

1. Provide ten days prior notice when services are interrupted or changed (e.g., when our pricing is changed).
2. Allow you to download your WebAssembly executables and your account information if our service is terminated.

Terms used:

Here we explain the terms used in this document.

- *algorithm*: A list of operations to carry out on some input data (possibly void) producing some output data (possibly void) independent of the language this list of operations has been specified in.
- *trial account*: The user account we create for you, identified by your email address, to use our services.
- *trial program*: The finite time period our services are available to those possessing a trial account under the terms and conditions specified in this document.
- *input*: The data (bits) that are operated upon by an algorithm.
- *output*: The data (bits) that are the result of the algorithm operating on the input.
- *packages*: Scailable packages are pieces of code (often open-source) that can be downloaded by our users to simplify the process of uploading an algorithm to our toolchain(s).
- *REST endpoint*: An `URL`, in our context one that starts with `https://taskmanager.sclbl.net:8080/task/`, that you can call to execute a task (click here for more info about REST).
- *micro-containers*: The software, which is part of our services, which you can install on a targeted edge device to deploy WebAssembly binaries and execute Scailable tasks on the edge. Note, micro-containers are proprietary and protected software created by Scailable.
- *services*: In this document all the products and functionality provided by Scailable during the trial program.
- *task*: A combination of an algorithm and a specific input.
- *toolchain*: A (proprietary) Scailable software application that takes an algorithm and transpiles it to a WebAssembly executable.
- *transpile*: Informally this would mean to "rewrite into another language". Thus, take an algorithm specified in some language (`C`, `Python`, `ONNX`, etc.) and express that same algorithm in a different language.
- *WebAssembly binary* An algorithm expressed in the WebAssembly language that can be executed by a runtime. In this context, we use the term to refer to algorithms that have start and endpoints specific to Scailable; for details, see our Scailable tutorial on WebAssembly.