

# The Golem Project

Crowdfunding Whitepaper

golem

**DRAFT v0.9**

**October 2016**

# Table of contents

## [Overview of the Golem Project](#)

[Grand vision and core features](#)

[Golem as an Ecosystem](#)

[Supply of Infrastructure](#)

[Demand for Computing Resources](#)

[Software & Microservices](#)

[The first use case: CGI rendering](#)

[Long term vision: Golem as a building block of Web 3.0](#)

[Golem Network Token \(GNT\)](#)

[Application Registry](#)

[Transaction Framework](#)

## [Roadmap](#)

[Brass Golem](#)

[Clay Golem](#)

[Stone Golem](#)

[Iron Golem](#)

[Future integrations](#)

## [Crowdfunding](#)

[Crowdfunding summary](#)

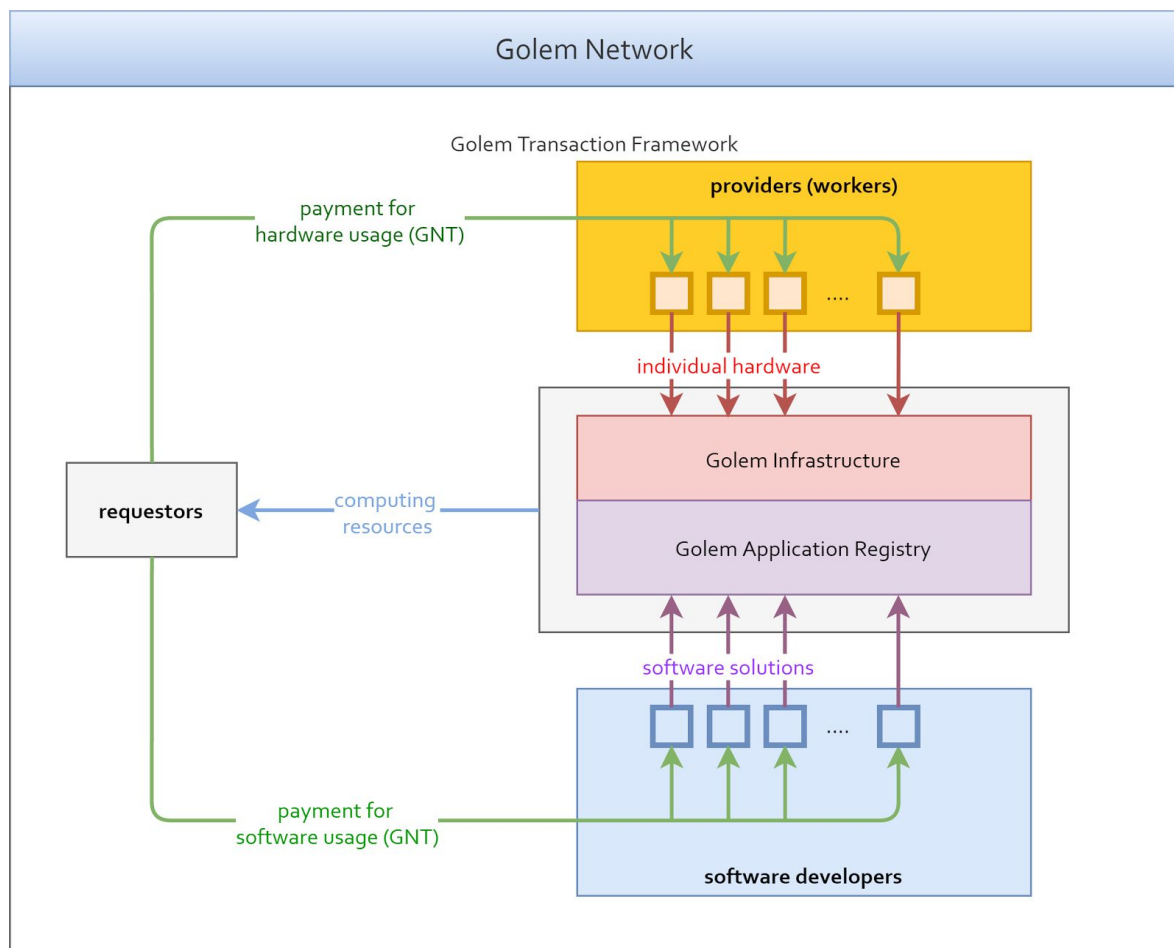
[Budget and levels of funding](#)

## [Golem Team](#)

# Overview of the Golem Project

## Grand vision and core features

- Golem is the first truly *decentralized* supercomputer, creating a global market for computing power. Combined with flexible tools to aid developers in securely distributing and monetizing their software, Golem altogether changes the way compute tasks are organized and executed. By powering decentralized microservices and asynchronous task execution, Golem is set to become a key building block for future Internet service providers and software development. And, by substantially lowering the price of computations, complex applications such as CGI rendering, scientific calculation, and machine learning become more accessible to everyone.
- Golem connects computers in a peer-to-peer network, enabling both application owners and individual users ("requestors") to rent resources of other users' ("providers") machines. These resources can be used to complete tasks requiring any amount of computation time and capacity. Today, such resources are supplied by centralized cloud providers which, are constrained by closed networks, proprietary payment systems, and hard-coded provisioning operations. Also core to Golem's built-in feature set is a dedicated Ethereum-based transaction system, which enables direct payments between requestors, providers, and software developers.
- The function of Golem as the backbone of a decentralized market for computing power can be considered both Infrastructure-as-a-Service (IaaS), as well as Platform-as-a-Service (PaaS). However, Golem reveals its true potential by adding dedicated software integrations to the equation. Any interested party is free to create and deploy software to the Golem network by publishing it to the Application Registry. Together with the Transaction Framework, developers can also extend and customize the payment mechanism resulting in unique mechanisms for monetizing software.



## Golem as an Ecosystem

Golem's business case boils down to the fact that, due to relatively recent technological advances, the market for computing resources can be organized according to entirely new principles. In contrast, the compute market today is dominated by heavyweight players such as Amazon, Google, Microsoft and IBM, who leverage their market power and assets to ensure hefty margins, resulting in inefficiently priced compute services. Luckily, the market is not doomed to function this way forever. With Golem the supply of computing resources is based on contributions of individual and professional providers, combined with an array of dedicated software solutions via Golem's Application Registry – itself operating on a single and competitive market with nearly [complete information](#) market.

Scaling the compute market enabled by Golem requires onboarding three groups: suppliers of computing resources ("providers"), task creators ("requestors") who submit their tasks to be computed by the network, and of course, software developers. These three groups comprise Golem's unique, interdependent ecosystem.

Group	Golem features	Incentive to participate
<b>Requestors</b>	Golem offers tools to execute compute-intensive tasks.	Requestors get access to affordable and scalable solutions, which combine hardware and software.
<b>Providers</b>	Golem combines and utilizes (almost) any kind of existing computing hardware.	Hardware providers get paid for renting out their hardware.
<b>Software Developers</b>	Golem is a flexible platform to deploy and monetize software.	Software developers use Golem as a distribution channel, associated with access to hardware.

## Supply of Infrastructure

The supply of computing power to the network comes from providers. This could be anyone, from an individual user renting out idle CPU cycles of a gaming PC, to a large datacenter contributing their entire capacity. Providers have the incentive to join Golem because they receive payments from requestors for the completed tasks. Of course, Golem's user interface will be easy to use, giving providers a clear way to set prices and decide what fraction of their own idle resources they are willing to rent out.

## Demand for Computing Resources

In order to reward providers for contributing their resources, Golem needs to attract requestors seeking additional computing resources. The market Golem creates will be highly competitive due to nearly complete information, and the ease of deploying tasks on any hardware. This will not only make using Golem simple, which will attract requestors - highly competitive setup will also increase efficiency of the market, very likely resulting in much more comprehensive and advantageous pricing compared to the existing cloud computing platforms.

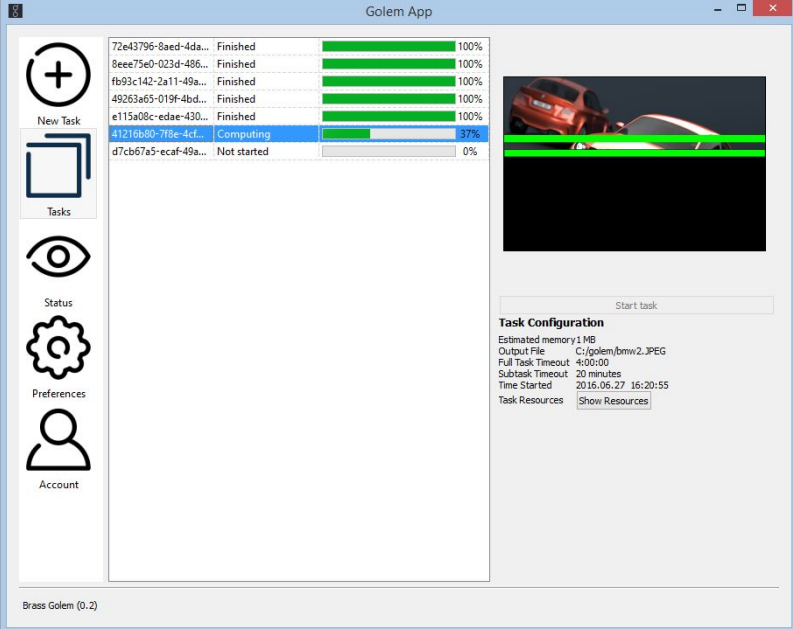
One important feature is that Golem's marketplace will enable requestors to become providers: because most of them will need additional resources only occasionally. They will be able to rent out their own hardware and earn extra fees. In addition, financial aspects are not going to be the sole incentive to use Golem: one of its key features will be the availability of diverse software running on the Golem network, accessible from the Application Registry.

## Software & Microservices

Although some initial use cases (such as CGI rendering) are going to be developed and implemented by the Golem team, it is of course essential to engage other software developers to come up with their own ideas for Golem applications. The number and quality of such applications is one of the key factors in Golem's future success. For this reason, the

Application Registry and Transaction Framework are among the most important features of the entire ecosystem, and will be front and central to the development process. Once introduced, they will provide developers with flexible and efficient tools to deploy, distribute, and monetize software running on Golem. This is going to be a perfect solution for microservices and some of the forthcoming decentralized applications (DApps), but could also become an interesting way to distribute existing proprietary and open source software.

## The first use case: CGI rendering



The screenshot shows the Golem App interface. On the left is a sidebar with icons for 'New Task', 'Tasks', 'Status', 'Preferences', and 'Account'. The main area displays a list of tasks with their IDs, status, and progress bars. One task is currently 'Computing' at 37% progress. Below the task list is a 'Task Configuration' window with the following details:

Task Configuration	
Estimated memory	1 MB
Output File	C:/golem/bmw2_JPEG
Full Task Timeout	4:00:00
Subtask Timeout	20 minutes
Time Started	2016.06.27 16:20:55
Task Resources	<a href="#">Show Resources</a>

Below the screenshot, the text reads: **Golem Alpha release: CGI rendering using Blender** It is public, follow [the link](#) to test Golem.

CGI rendering is the first and very illustrative case of real Golem usage. Rather than using costly cloud-based services or waiting ages for one's own machine to complete the task, CGI artists can now rent compute resources from other users to render an image quickly. The payment from a requestor (in this case, a CGI artist) is sent directly to providers who made their resources available. In addition, when the artist's machine is idle, it can itself accept tasks from other users.

## Long term vision: Golem as a building block of Web 3.0

We believe that the future Internet will be a truly decentralized network, enabling users to securely and directly exchange content, without sharing it with corporations or other middlemen. Accordingly, Golem will be used not only to compute specific tasks, but also to bulk-rent machines in order to perform operations within a self-organizing network. Of course, this will require the simultaneous development of other technologies, many of which have gained significant traction in recent years.

Better data-sharing technologies are necessary, but taking into account the ongoing development of IPFS/Filecoin and Swarm, the appropriate solutions seem to be within reach. Eventually, the Ethereum network will become more scalable, more efficient, and include a fully functional network of micropayment channels. Once these technologies become available, it is easy to imagine Golem primarily as a platform for microservices, allowing users to run both small (e.g. a note-taking app) and large (e.g. a streaming service) applications in a

completely decentralized way. Although ambitious, this vision seems to be the ultimate argument for Golem's long-term potential.

## Golem Network Token (GNT)

The Golem Network Token ("GNT") account is a core component of Golem and is designed to ensure flexibility and control over the future evolution of the project. GNT is created during the crowdfunding period (described in this whitepaper) and, following the first major release of Golem, GNT will be attributed a variety of functions in the Golem network.

- Payments from requestors to providers for resource usage, and remuneration for software developers is going to be exclusively conducted in GNT.
- Once the Application Registry and Transaction Framework are implemented, GNT will be necessary for other interactions with Golem, such as submitting deposits by providers and software developers or participation in the process of software validation and certification (as described in the Application Registry section).
- The general conditions for using GNT will be set in the Transaction Framework, but specific parameters of these interactions will be possible to define within each software integration.

**The supply of GNT will be limited to the pool of tokens created during crowdfunding period.**

### Creation of the GNT and initial GNT account functionalities

- The Golem Network Token is a token on Ethereum platform. Its design follows widely adopted token implementation standards. This makes it easy to manage using existing solutions including Ethereum Wallet.
- Maximum number of tokens created during crowdfunding period:
  - Total: 1 000 000 000 (100%)
  - Crowdfunding participants: 820 000 000 (82%)
  - Golem Team 60 000 000 (6%)
  - Golem Factory GmbH 120 000 000 (12%)
- Sending 1 ether to the GNT account will create 1 000 GNT
- No token creation, minting or mining after the crowdfunding period.
- Tokens will be transferable once the crowdfunding is successfully completed.

*Go to the [crowdfunding section](#) to learn the details and see how to support the Golem Project via crowdfunding.*

## Application Registry

The Application Registry is an Ethereum smart contract, to which anyone can publish their own applications that are ready to run on Golem network. The goal of the Application Registry is to:

- Give developers a way to publish their integrations and reach out to users in a decentralized manner;
- Give requestors a place to look for specific tools fitting their needs;
- Give providers full control over the code they run because of the security concerns.

Since the Golem network is fully decentralized, we also want the Application Registry to be driven by the community.

Golem allows requestors to execute the code of an application on someone else's computer. This code is sandboxed and executed with the minimal required privileges. But software bugs are everywhere, and once in awhile people defeat sandboxes, manage to execute malicious code on a host machine, and sometimes even take it over. That's why we can't rely only on sandboxing. We could try to automatically evaluate whether or not the code is safe, but this is literally impossible (*vide* halting problem). The process of code review and validation cannot be fully automated and left to the autonomous network. On the other hand, it is impossible to assume that no one will ever publish malicious software to run on top of Golem network.

We solve these problems by splitting Application Registry users into three categories: authors, validators and providers. Authors publish applications, validators review and certify applications as safe and trustworthy by adding them to their own whitelist. Validators may also mark applications as malicious by adding them to their own blacklists. Providers are also given the right to choose whom to trust by selecting validators whose lists are used by the particular instance of Golem running on their nodes. Apart from that, providers may maintain their own whitelists or blacklists. This gives each provider a lot of flexibility in deciding exactly what software to run, and what amount of work to put into software curation. What is more, this system does not exclude any party, and there is always room for new validators to emerge.

By default Golem runs using a whitelist of trusted applications. Since an empty whitelist is a problem for someone just trying Golem out for the first time, we will add a number of verified entries to the whitelist as a part of the initial distribution. A provider can take advantage of this mechanism, managing her own whitelist, or simply using whitelists of validators she trusts.

On the other hand, a provider running a computing farm may wish to rely entirely on blacklists. This is an option tailored for administrators of dedicated machines, who want to maximize their profits and who are willing to ride the bleeding-edge. In this scenario, a blacklist is used to banish any known troublemaking applications. Again, the provider can maintain her own blacklist, or use the blacklists of validators she trusts.



## Transaction Framework

When creating something new and exciting, it's hard if not impossible to predict all the opportunities which the new artifact will suddenly make possible. Golem is a generalized global supercomputer, and as such, it will no doubt find its niche with vastly varied applications. They might need very diverse remuneration models. We are not able to design a one-size-fits-all payment system for Golem, nor will we attempt to force one upon application authors.

When a developer integrates her application with Golem, she has the freedom to decide which transaction model she implements, as long as it is compliant with Golem's Transaction Framework. The Transaction Framework will take the form of a set of requirements to follow; basic requirements may include:

- Entry in the Application Registry;
- Use of open source and/or deterministic environment, such as EVM;
- Community approval or rating of transaction model;
- Use of GNT for remunerating software and resource providers.

We are building the transaction framework on top of Ethereum. Ethereum gives us expressive power, which is much-needed when implementing advanced, trustless schemes. This includes components which are extremely difficult to do well in P2P networks, such as evaluating reputation.

Example transaction framework components:

- Diverse payout schemes such as [nanopayments](#), [batching](#)
- Off-chain payment channels
- Custom receipts
- Payment to software developer
- Per-unit use of software (per-node, per-hour, etc.)

*In the future, this might evolve into community-reviewed template code to be used in the implementation of custom transaction models.*

It is also possible to introduce more sophisticated components into the transaction model design, to meet specific goals not related to payments. For example:

- Requestor escrow for tasks where a higher level of commitment is required (high price because of specialized hardware or long running subtasks); the requestor may create a two-party escrow account and require providers to take part in it.
- Provider deposit: the requestor may require to be in control of some amount of timelocked GNT.

- Requestor deposit: the provider may accept tasks *only* from requestors who are in control of some amount of timelocked GNT.
- Registration of a task as an anchor for a TrueBit-style conflict resolution fallback mechanism.

## Roadmap

In this section we present planned milestones for Golem development. You will find a nontechnical description of Golem's architecture [in this post](#) on our blog, some thoughts about challenges ahead [here](#) and of course you can examine the code on [GitHub](#).

The successive versions of Golem's software are split into milestones. This plan should be considered preliminary, as Golem is using bleeding-edge technologies, and is itself a very complex project.

Each milestone should be preceded with research, with results described in technical whitepapers. There are however two very important things to note:

1. At every stage new functionalities are added.
2. The scope of deliverables at every milestone depends on the level of funding raised. In the description of milestones presented below, these functionalities have been assigned to four indicative funding scenarios.
3. Functionalities marked with '+/++/+++' will be implemented if the adequate [level of funding](#) is reached.

We have codenamed concurrent versions of Golem using descriptions from [the long list of D&D's golems](#). The analogs might not be perfect, but these are e-golems after all.

## Brass Golem

*...these are created to fulfill one goal, set at the time of their creation, and wait with absolute patience until activated to perform this task.*

Brass Golem is where we are at the moment with our proof-of-concept, in alpha testing now. This current version of Golem is only focused on rendering in Blender and LuxRender, and although it will be useful to CGI artists, we consider CGI rendering to be a proof of concept primarily, and also a training ground. Brass Golem should be frozen within 6 months after end of crowdfunding period and a full battery of tests. Even though we do not expect that Blender CGI rendering will create enough turnover to justify all the work we have put into the project, this will be the first decentralised compute market.

List of proposed functionalities:

- Basic Task Definition Scheme that allows to prepare first task definition;

- Basic Application Registry - first version of Ethereum-based Application Registry which allows to save tasks defined with basic task definition scheme;
- [IPFS](#) integration for coordinating task data and content delivery, e.g. deliver files needed to compute a task, deliver the results back to the requester;
- Docker environment with Golem-provided images for sandboxing the computations;
- Local verification: a probabilistic verification system based on the calculation of a fraction of the task on the requestor's machine;
- Basic UI and CLI;
- Basic reputation system;
- Implementation of [Blender](#) and [LuxRender](#) tasks.

## Clay Golem

*There is a chance (...) that a Clay Golem will be possessed by a chaotic evil spirit. If this happens control is lost and the Golem attacks the closest living creature.*

Clay Golem is a big leap from the Brass milestone. Clay milestone introduces the Task API and the Application Registry, which together are going to make Golem a multi-purpose, generalized distributed computation solution. Developers now have the means to integrate with Golem. This advance, however, may come at the cost of compromised stability and security, so this version should be considered an experiment for early adopters and tech enthusiasts. Prototype your new ideas and solutions on Clay.

List of proposed functionality:

- Basic Task API: an interface that allows a user to define simple tasks;
- Initial [Transaction Framework Model](#) with hard-coded payments schemes;
- Redundant verification: a verification scheme based on the comparison of redundant computation results;
- Basic subtask delegation: a mechanism for more fined grained subtasks distribution;
- (+) Support for virtual machines as a sandbox for computation;
- (+) Set of tutorials for developers explaining how to implement their own tasks for Golem network;
- (++) Example task implementations, eg. example machine learning and scientific tasks.

## Stone Golem

*Stone Golems do not revoke their creators control like (...) Clay Golems.*

Stone Golem will add more security and stability, but also enhance the functionalities implemented in Clay. An advanced version of the Task API will be introduced. The Application Registry will be complemented by the Certification Mechanism that will create a community-driven trust network for applications. Also, the Transaction Framework will create an environment that will allow Golem to be used in a SaaS model.

List of proposed functionalities:

- Full Task API: an interface that allows users to define tasks;
- [Application Registry](#): where developers publish applications ready to run on Golem;
- [Transaction Framework](#) that allows a choice of remuneration models for task templates;
- Basic Certification support for Software: A mechanism that allows users to whitelist and blacklist applications, building a decentralized trust network;
- Support for SaaS: the possibility to add support for proprietary software which can be used in tasks. Payments for task creators should also be implemented in the application;
- (+) SaaS tasks examples - example use cases that shows developers how to create tasks available in SaaS model;
- (++) Golem web client: a web interface for Golem nodes as an alternative to the native GUI / console interface;
- (+++) Provider dashboard - providing stats, graphs and more advance settings management for providers;

## Iron Golem

*Iron Golems are made of iron and are among the strongest type of Golem. They never revoke the control of the wizard that created them.*

Iron is a deeply tested Golem that gives more freedom to developers, allowing them to create applications that use an Internet connection or applications that run outside the sandbox. Of course, the decision to accept higher-risk applications will still belong to the providers renting their compute power. Iron Golem should be robust, highly resistant to attacks, stable and scalable. Iron will also introduce various tools for developers that will make application creation far easier. Finally, the Golem Standard Library will be implemented.

List of proposed functionality:

- External data link: enables Golem to use resources and interface with software outside of the Golem network;
- Host-direct mode: a trusted mode for explicitly whitelisted applications or invulnerable environments, where Golem runs computation outside the Docker/VM;
- Certification support for Environments;
- Network Status Dashboard - public website displaying basic stats about Golem Network;
- (+) Additional security mechanism - tasks that uses public data link or host-direct mode are particularly challenging for security. Additional means may be necessary to make running those tasks safer for providers (eg. central audit oracles, agreements contracts or code-execution observers may be implemented);
- (++) Golem Developer Toolkit: a set of diagnostic and test tools to make creation process of applications for Golem even easier;

- (++) Reputation-system: reputation protocol that allows the node to effectively supervise network behaviour;
- (++) Advanced transaction system: a system that automatically tries to match requestors with providers in a way that is most profitable to all participants;
- (+++) devp2p integration - changes in p2p and network protocols using new version of devp2p;
- (+++) MapReduce and topological sorting of tasks: add the next abstraction layer, allowing users to define more generic tasks that are interdependent;
- (+++) Golem Standard Library (Golem STD): language agnostic functionality providing access to the low level core components required to interact with Golem from within a programming language. Special attention will be paid to I/O functions exposed to tasks and subtasks related functionalities. Each supported programming language will have bindings to Golem STD. These bindings will serve as a means of extending the default standard library of the language in question (custom extensions provided by developers of programming languages will also be possible). With Golem STD an automatic task definition, independent from the operating system, will be possible. Golem STD will allow users to create Golem applications using different programming languages, which shall significantly increase the number of potential use cases and simplify task creation process.

## Future integrations

There are numerous Ethereum dapps and future platforms currently under development or in alpha release. We see great opportunities in this environment, not to mention solutions that could potentially be used as a part of Golem's ecosystem, either directly or as extensions. The following systems will be considered for integration and their implementation will be dependent upon the release of production code and complexity of integration:

- Payment channel solutions based on P2P routing and transactions, eg. [Raiden](#) or [multi-party payment channels](#);
- External decentralized identity services, e.g. [uPort](#);
- External solutions for task verification or reputation, eg. [TrueBit](#);
- External solutions for storage, eg. FileCoin, Swarm.

# Crowdfunding

The crowdfunding of Golem and the corresponding token creation process are organised around smart contracts running on Ethereum. Participants willing to support development of the Golem Project can do so by sending ether to the designated address. By doing so they create Golem Network Tokens (GNT) at the rate of 1 000 GNT per 1 ETH.

A participant must send ether to the account after the start of the crowdfunding period (specified as the block number). Crowdfunding ends when the end block is created, or when the amount of ether sent to the account reaches the maximum.

## Crowdfunding summary

GNT created per 1 ether	1 000 GNT
Minimum ether*	150 000 ETH
Maximum ether	820 000 ETH
% of tokens generated to Golem Team	6%
% of tokens generated to Golem Factory GmbH	12%
Approximate date of start (StartBlock)	(to be announced)
Approximate date of end (EndBlock)	(to be announced) + 3 weeks
Maximum number of GNT generated	1 000 000 000 GNT
of which crowdfunding participants	820 000 000 GNT
of which Golem Team and Golem Factory GmbH	180 000 000 GNT

\*Minimum financing is not implemented in the crowdfunding contract. If minimum financing is not reached, refunds will be implemented by a separate contract.

The crowdfunding address will be announced at the crowdfunding start through the following channels:

- Project webpage: [golem.network](http://golem.network)
- Official Twitter: [twitter.com/golemproject](https://twitter.com/golemproject)
- Official Slack: [golemproject.slack.com](https://golemproject.slack.com) (you can join [here](#))
- Official Blog: [blog.golemproject.net](http://blog.golemproject.net)
- Reddit: [reddit.com/r/golemproject](https://reddit.com/r/golemproject)

Please, double-check the address before sending ETH. For security reasons, we advise to confirm the address using at least two different sources above.

On the project webpage, you will also find a detailed guide on how to participate in the crowdfunding using either the Ethereum Wallet or Parity.

Crowdfunding is implemented as a smart contract with a few simple parameters:

- Golem Factory GmbH: controls the contract and the address to which gathered ether will be sent (implemented as a multisig address);
- Percent of pre-allocated tokens is 18% (6% - Golem Team, 12% - Golem Factory GmbH);
- StartBlock, EndBlock: these block numbers indicate the start and the end of the crowdfunding process;
- maxCap: maximum cap for this crowdfunding, denominated in GNT;
- GNT creation rate, denominated in ETH.

The crowdfunding contract conforms to a few important rules:

- Before the crowdfunding starts, no ether can be sent to the crowdfunding contract;
- After the crowdfunding (either maxCap was reached or the crowdfunding deadline passed), no ether can be sent to the contract;
- During the crowdfunding, participants simply send ether to the crowdfunding contract which results in GNT creation;
- All created tokens are **locked** during the crowdfunding;
- Only after the crowdfunding period has ended:
  - Any user can initiate the transfer of ether to the specified address of Golem Factory GmbH;
  - The crowdfunding contract creates an 18% endowment of tokens such that crowdfunding participants' tokens constitute 82% of supply, regardless of the level of funding;
  - The crowdfunding contract finalizes funding which results in an allocation of founders' tokens and unlocking the created GNT.

There is no minimum financing specified in the contract code. If minimum financing is not reached, then after the crowdfunding period, ether is sent back to the participants from the

Golem Factory GmbH account. Because GNT is locked by default, if minimum financing is not reached then ether return procedure is simple and straightforward.

Crowdfunding process leads to creation of GNT, a backbone token for the Golem Network. GNT implementation follows widely adopted token implementation standards with two additional functionalities core to the crowdfunding process and future upgrades, namely, token creation and token migration:

- Create token - during the crowdfunding process, the crowdfunding contract can issue new GNT (based on the amount of sent ETH).
  - By default, created GNT is locked (nontransferable). Only when crowdfunding is finalized are tokens unlocked and participants able to transfer them.
  - The creation function is enabled only during the crowdfunding phase; it does not allow the creation of tokens after the crowdfunding phase is over (token supply is constant ever after).
- Migrate token - a function which implements GNT migration to another contract.
  - Does nothing by default, but if at some point a GNT upgrade is required, a separate migration contract can be specified and recommended by Golem Factory GmbH to be used to transfer tokens to the new contract.
  - Technically speaking, if a GNT upgrade is required, a contract implementing the *MigrationAgent* interface is created and set by Golem Factory GmbH in the GNT contract (for security reasons this can be done only once). Following that, each GNT holder can decide whether to call *MigrationAgent.migrateTokens* to transfer GNT to the new contract, or not.
  - *MigrationAgent* can be implemented only after the new token is implemented and deployed. That's why only an interface is provided right now.

Migration is to be used if it turns out at some point that token upgrade is needed for whatever reason (e.g. changes in Ethereum, or changes in Golem's design). The upgrade will need action from token holders and cannot be imposed by Golem Factory GmbH.



## Budget and levels of funding

The ether raised during crowdfunding will be used by Golem Factory GmbH in accordance with the roadmap [presented above](#). Crowdfunding code implies that level of project financing might be anything between minimum financing and the maximum financing (cap). The roadmap is a full vision to be completed if the cap is reached.

Golem should be considered an R&D project involving bleeding-edge technologies. The progress we have already made while working on the Brass Golem alpha proves the validity of our general assumptions presented in this whitepaper, but we are also well aware of the huge amount of work ahead. The commitment of Golem team with respect to the technologies presented in this whitepaper is full, but still ultimately depends on the level of success of the crowdfunding.

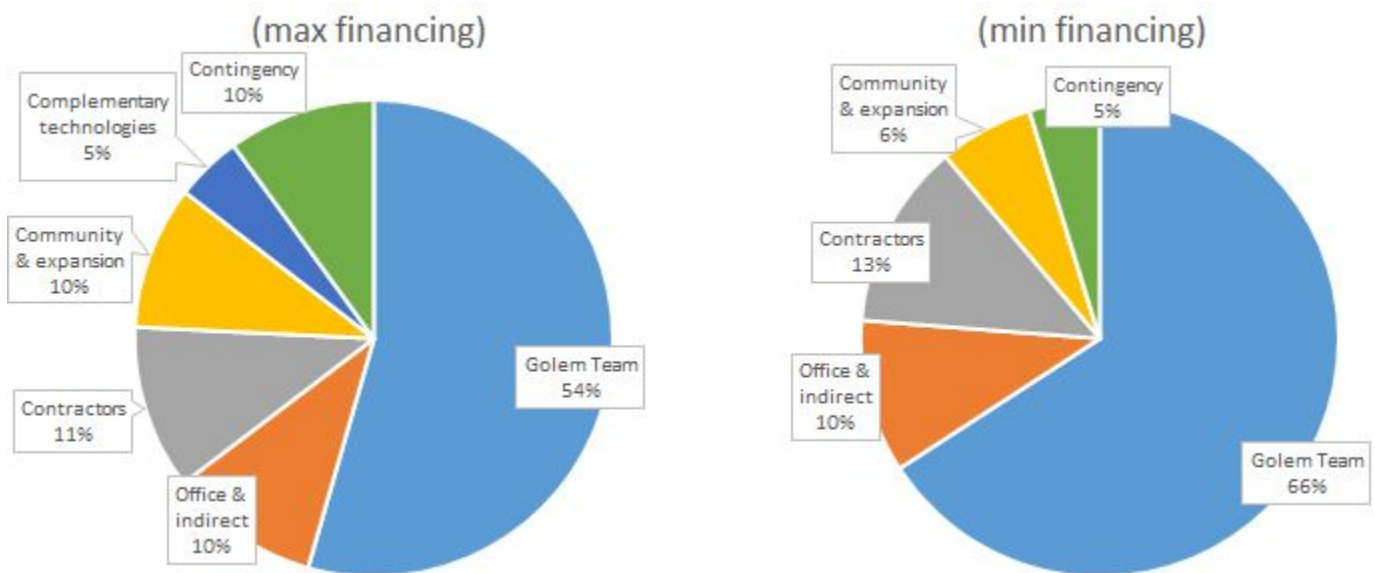
In the 'minimum financing' scenario, the ultimate deliverable is a working Iron Golem with functionality enabling the creation of a decentralized market for computing power, as well as a rudimentary toolbox for developers to integrate their own software with Golem. In particular, the minimum financing will be sufficient to introduce a basic version of both the Application Registry and the Transaction Framework.

In the 'maximum financing' scenario, we are making a commitment to deliver a much more advanced version of Iron Golem, which is not only gunning for total disruption of the market for computing power, but also dives head-first into the development of some important components of Web 3.0. In particular, this level of financing will make it possible to create a flexible platform to distribute and monetize innovative software solutions, notably dapps and microservices. Assuming the cap is reached in the crowdfunding, the Golem team commits to create an array of specific integrations useful to the entire community.

## Functionality vs funding

	Min financing	additional features
<b>Core Functionality</b>	<ul style="list-style-type: none"> <li>● [B] Basic Task Definition Scheme</li> <li>● [B] Basic Application Registry</li> <li>● [B] Local verification</li> <li>● [C] Basic Task API</li> <li>● [C] Redundant verification</li> <li>● [C] Subtask delegation</li> <li>● [S] Full Task API</li> <li>● [S] Application Registry</li> <li>● [S] Certification support for Software</li> <li>● [I] Certification support for Environments</li> <li>● [I] Public data-link</li> <li>● [I] Host-direct mode</li> </ul>	<ul style="list-style-type: none"> <li>● [I] MapReduce (+++)</li> <li>● [I] Topological sorting (+++)</li> <li>● [I] Additional security mechanism (++)</li> </ul>
<b>Reputation &amp; security</b>	<ul style="list-style-type: none"> <li>● [B] Basic reputation</li> </ul>	<ul style="list-style-type: none"> <li>● [I] Reputation system (++)</li> <li>● [I] Additional security mechanism for host-direct mode (+)</li> </ul>
<b>Transaction system</b>	<ul style="list-style-type: none"> <li>● [C] Initial Transaction Framework Model</li> <li>● [S] Transaction Framework</li> </ul>	<ul style="list-style-type: none"> <li>● [I] Advanced transaction system (++)</li> </ul>
<b>Integrations</b>	<ul style="list-style-type: none"> <li>● [B] IPFS</li> <li>● [B] Docker as sandbox</li> </ul>	<ul style="list-style-type: none"> <li>● [C] Virtual machine as sandbox (+)</li> <li>● [I] devp2p (+++)</li> </ul>
<b>UX/UI</b>	<ul style="list-style-type: none"> <li>● [B] Basic GUI and CLI</li> </ul>	<ul style="list-style-type: none"> <li>● [S] Web client (++)</li> <li>● [S] Provider dashboard (+++)</li> <li>● [I] Golem Developer Toolkit IDE</li> </ul>
<b>Use cases</b>	<ul style="list-style-type: none"> <li>● [B] CGI Rendering (Blender, LuxRender)</li> </ul>	<ul style="list-style-type: none"> <li>● [C] Machine-learning use case (++)</li> <li>● [C] Computational chemistry use case (++)</li> <li>● [S] Examples of SaaS integration (+)</li> </ul>
<b>Tools for developers</b>	<ul style="list-style-type: none"> <li>● [I] Basic Golem Developer Toolkit (GDT)</li> </ul>	<ul style="list-style-type: none"> <li>● [C] Tutorials for task developers (+)</li> <li>● [I] Advanced Golem Developer Toolkit (++)</li> <li>● [I] Golem Standard Library (Golem STD) (+++) <ul style="list-style-type: none"> <li>○ [I] Golem STD integration with GDT (+++)</li> <li>○ [I] Golem STD support for programming languages (+++)</li> </ul> </li> </ul>

<p>Additional features (indicative thresholds):</p> <ul style="list-style-type: none"> <li>● (+) 320k ETH</li> <li>● (++) 530k ETH</li> <li>● (+++) 820k ETH</li> </ul>	<p>Milestones:</p> <ul style="list-style-type: none"> <li>● [B] Brass Golem</li> <li>● [C] Clay Golem</li> <li>● [S] Stone Golem</li> <li>● [I] Iron Golem</li> </ul>
---	---



Budget structure for maximum and minimum financing

**Golem Team** consists solely of employment costs. We assume that with maximum financing we will be able to finance team of 20 people (most of them developers) for a period of 4 years.

**Office and indirect costs** includes costs of offices in both Zug and Warszawa, as well as other indirect, employment-related costs.

**Contractors** covers all third parties we are willing to work with. The number here is high largely because of security audits (four in case of maximum financing: one for every release). Legal and accounting services are also included in this category.

**Community animation and expansion activities** are strictly related to Golem's expansion plan. This includes both communication and marketing efforts to get new communities on board, as well as supporting (financing or co-financing) third party integrations with Golem. Activities here will be mostly requestor-oriented, to ensure that there are a growing number of use cases integrated, with users actively using them on the Golem network.

The **Complementary technologies** category covers expenditures on external technologies Golem is dependent on. This will most likely take the form of financing original efforts to introduce modifications needed by Golem.

**Contingency fund** is calculated as 10% of the total budget (5% for minimum financing).

# Golem Team



[Julian Zawistowski](#)  
CEO, founder



[Piotr 'Viggith' Janiuk](#)  
CTO, co-founder



[Andrzej Regulski](#)  
COO, co-founder



[Aleksandra Skrzypczak](#)  
Lead Software Engineer,  
co-founder



[Alex Leverington](#)  
P2P Network



[Paweł 'chfast' Bylica](#)  
Lead Ethereum Engineer



[Marek Franciszkiewicz](#)  
developer



Wendell Davis  
BDM



[Adam Banasiak](#)  
developer



[Paweł Peregud](#)  
developer



[Magdalena Stasiewicz](#)  
developer



[Radosław Zagórowicz](#)  
developer

Creating Golem is a tremendously challenging undertaking requiring a capable and dedicated team. Some of us - particularly, Julian, Andrzej, Piotr, Paweł, Wendell, and Radek - have worked together on projects such as Ethereum, Hydrachain, and the [Morfa programming language](#). Others joined Golem later on. Alex Leverington and Paweł Bylica have been core Ethereum developers for years, with Paweł still working on the Ethereum virtual machine (EVM).