
Bringing Limitless Computational Engineering to Wind Industry

# Deliverable 4.1 - Technical Note on the ZephyCloud Open-API

ZephyOpen

Co-funded by the Horizon 2020 programme of the European Union

| | |
|---|---|
| Project no. | 783913 |
| Project acronym | ZephyCloud-2 |
| Project title | Bringing Limitless Computational Engineering to Wind Industry |
| Starting date | 01.10.2017 |
| Duration | 21 months |
| Deliverable title | Technical Note on the ZephyCloud Open-API |
| Due date of deliverable | 30.06.2018 |
| Work Package | WP4 |
| Task(s) involved | T4.1 |
| Type | R |
| Dissemination level | PU |
| Prepared by | Zephy-Science |

# Introduction

## Context

After a phase one project successfully delivered in 2016, Zephy-Science has been awarded in 2017 by the European Commission under the framework of Horizon 2020 Program with an additional phase two grants for a two-year R&D program. One of the main objectives of this program is to develop open-API partnerships with wind energy players from academic and business organizations who are developing modeling chains based on opensource solvers which are running on local servers, thus constrained in accuracy. Indeed, cloud bursting can solve a critical problem in the wind industry which is the compromise between accuracy and speed:

- accuracy being a growing function of time with traditional on-premise calculations, accuracy is limited due to slow simulations and tight project deadlines;
- accuracy being a growing function of cost with cloud bursting, accuracy is unlimited and only cost-effectiveness (value for money) should be addressed in defining R&D strategies.

This program, called ZephyOPEN, has been launched in June 2018.

## Summary

In a first section, the concept of ZephyOpen is described.
Secondly, its usage and pricing strategy are discussed in more detail.
Eventually, information related to ZephyOpen technical design are addressed in a last section.

# 1 ZephyOpen - the concept

**Opening our advanced cloud calculation engine to developers through Open-API**

A few definitions first:
ZephyCloud Open-API consists of an **Application Programing Interface** (API) specifically developed for making the ZephyCloud-2 calculation power available to third-party developers and letting them build and sell new modeling chains while getting access to ZephyCloud's competitive advantage.

**ZephyCloud** is the back-end service used to perform burst computing from software applications through Extern Computing Services, charged to the end-user in **ZephyCoins**, a virtual currency whose price depends on the user's subscription level. The cloud computing servers are provided by Amazon Web Services.

The modelling chains developed by external developers, either independently or in a partnership with Zephy-Science, will be hosted on ZephyFARM. **ZephyFARM** is the public front-end web-browser client application, powered by ZephyCloud, which will act as the platform offering both pre- and post-construction modelling solutions for both offshore and onshore wind farms.

This Open-API platform is aligned with our ambition to foster collective intelligence while opening up new client channels for our Cloud service.

## 1.1 Value

We will develop a specific API of the ZephyCloud-2 calculation engine addressed to third-party developers who want to leverage the competitive advantage of ZephyCloud-2' s computational power in their own simulation products **without having to deal with complex cloud implementations headaches**.

**Opensource-based modeling chains can now generate new and automated revenues through ZephyFARM platform.**

This allows to commercialize ZephyCloud-2 under Platform-as-a-Service (PaaS) model.

### 1.2 Criteria for Success

Each of the following points is considered as a criterion of success.

The open-API service is designed to:

- ensure security of the platform;
- ensure the scalability of the computational fleet (allow bursting);
- ensure the scalability of the computational power (allow building clusters of instances);
- be flexible to allow updates without breaking third-party applications' functionalities.

The new modeling chain included within ZephyOPEN should:

- always be available and ready to be run 24/7;
- be accessible either from ZephyFARM web-platform or from direct request to the API;
- be available globally (including China).

### 1.3 Partner Adoption

ZephyOpen is dedicated to opensource-based modeling chains solving problems related to the wind industry.

Modeling chain developers (**ZephyOpen Partners**) may want to implement their work within ZephyOpen, in order to...

- contribute in enhancing wind industry state of the art;
- generate new sources of revenues;
- solve queuing issues when the computing charge exceeds their own hardware capacities.

It should thus be mainly oriented towards:

- Universities and research institutes;
- Entrepreneurs;
- Major wind industry players having developed an internal opensourced-based workflow.

For this purpose, several **Open Innovation Programs** have been designed:

- ZephyACADEMIA, designed for wind energy research institutes and universities;
- ZephySTART, designed for wind energy consulting entrepreneurs;
- ZephyEXPERT, designed for wind energy leading consulting firms.

### 1.4 User Adoption

The new public modeling chain included within ZephyOpen are highlighted and promoted within ZephyFARM so that each user connected to ZephyFARM should know about the value of the introduced sub-module.

## 2 ZephyOpen Usage

### 2.1 Pricing strategy

The agreement between Zephy-Science and the ZephyOpen Partner around pricing strategy should define, for each of the Gold and Free status:

- the end-user **fixed prices**
- the **margin factors** to be applied on the service.

The financial structure as defined in an Agreement and accepted by both Parties is the following.

- Each end-user will be using the service by spending a given amount of ZephyCoins, referred as the end-user's price $P_{USER}$.

It should be noticed that two types of users are now possible: **Free** and **Gold** users (depending on the chosen ZephyCloud subscription).
**Free users are charged 5 times more than the Gold ones**.

- Each time the service is launched, the corresponding amount of ZephyCoins is directly deduced from the end-user's credits.
- Each time the service is launched, the ZephyOpen Partner is notified about the application request.
- Each time the service is ended, end-user and the ZephyOPEN partner are notified about the result availability.

$P_{USER}$ is divided between a cloud cost part $P_{CLOUD}$ and a service part $P_{SERV}$, both in ZephyCoins.

$P_{USER} = P_{CLOUD} + P_{SERV}$

- $P_{CLOUD}$ is the cloud computing price; it should be transparently defined by ZephyScience and fully agreed with the ZephyOPEN partner (refer to **"Project Technical Specifications"**). $P_{CLOUD}$ can follow a pricing grid depending on the accuracy requirements of the user, however $P_{CLOUD}$ should aim to keep a profit as close as possible as the on-demand cloud calculations, keeping a descent marge to cover the most expensive calculations for a given category. It should be noticed that 5% of the profit generated by $P_{FLOW}$ will fund the OpenFoam fundation.
- $P_{SERV}$ is the service price; it should be defined by the ZephyOPEN partner.

$P_{SERV}$ is eventually divided between ZephyScience $P_{ZS}$ and the ZephyOPEN partner $P_{ZOP}$, both in ZephyCoins. Note that the figures given below are only indicative and should be redefined specifically for each project.

$P_{SERV} = P_{ZS} + P_{ZOP}$

$P_{ZS} = max(P_{ZSmin}, ROY_{ZS} * P_{ZOP})$

$P_{ZSmin}$: majored processing cost $P_{ZSmin} = 0.2Zc$

$ROY_{ZS}$: royalty on service execution $ROY_{ZS} = 5\%$

Eventually, a **recurrent price** is charged on a weekly basis to cover for the storage costs of the data managed by the tool-chain.

## 2.2 Partner Usage

ZephyOpen partner usage is described in Figure 1. In this Figure, different possible situations are covered:

- Public or private modeling chain;
- Services requiring a graphical client or not.

ZephyOpen toolchain creation is described in Figure 2.

## 2.3 End-user Usage

ZephyOpen user usage is described in Figure 3.

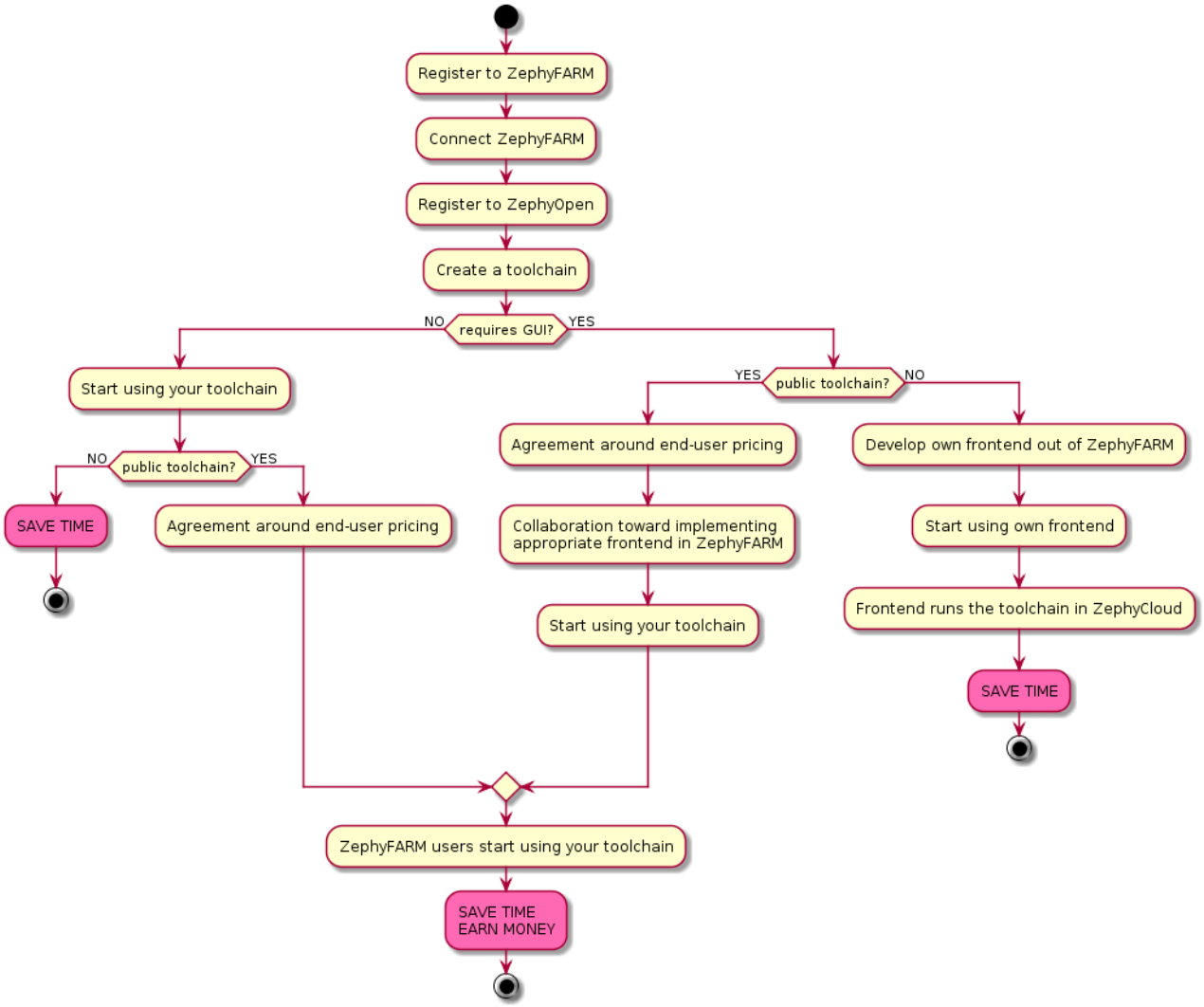


Figure 1: Partner usage: Overview

# 3 Technical Design

## 3.1 Architecture Overview

An overview of the architecture is given in Figure 4

## 3.2 Security

The security is designed to work against three kind of potential attackers:

- external attackers
- malicious/buggy end users
- malicious/buggy partners' tool-chains

### 3.2.1 General security against attackers

We are using multiple preventive measures.

- Monitoring tools, based on *Elasticsearch*, *Fluentd* and *Kibana*.
- Alerting system, based on *Prometheus*.
- Supervision, based on *Grafana*.
- Internal network security, based on key-based authentication, ssh tunneling, and firewalls.

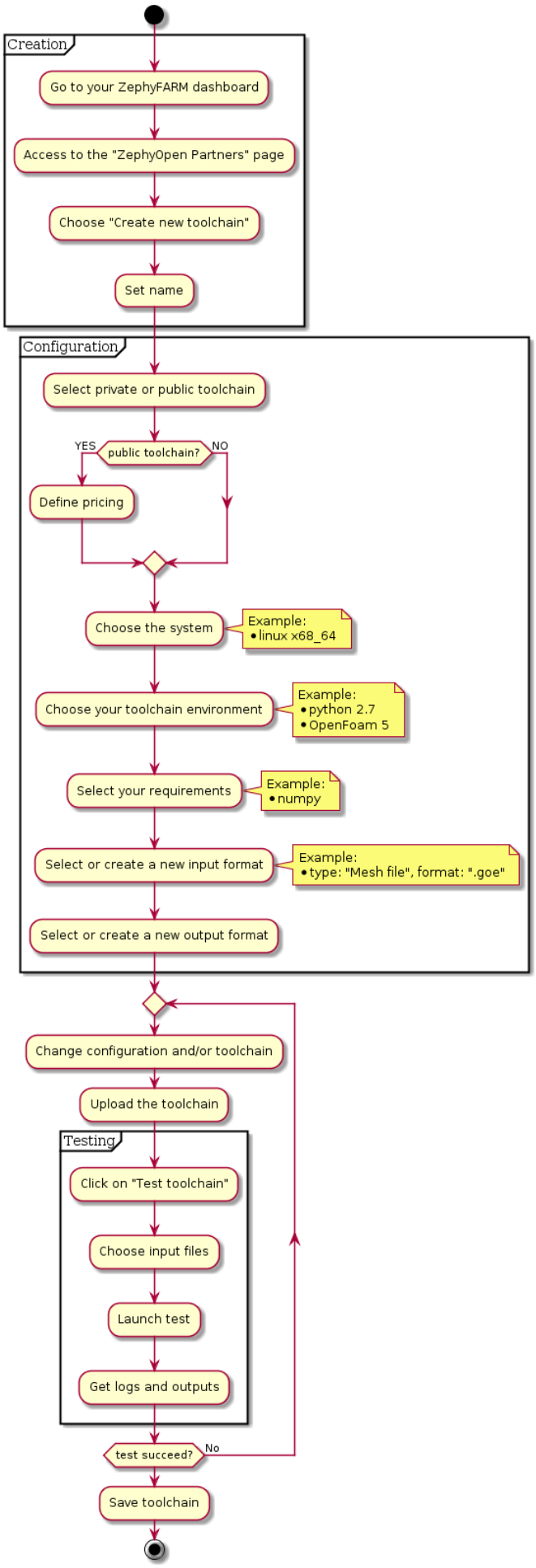


Figure 2: Partner usage: Create a tool-chain

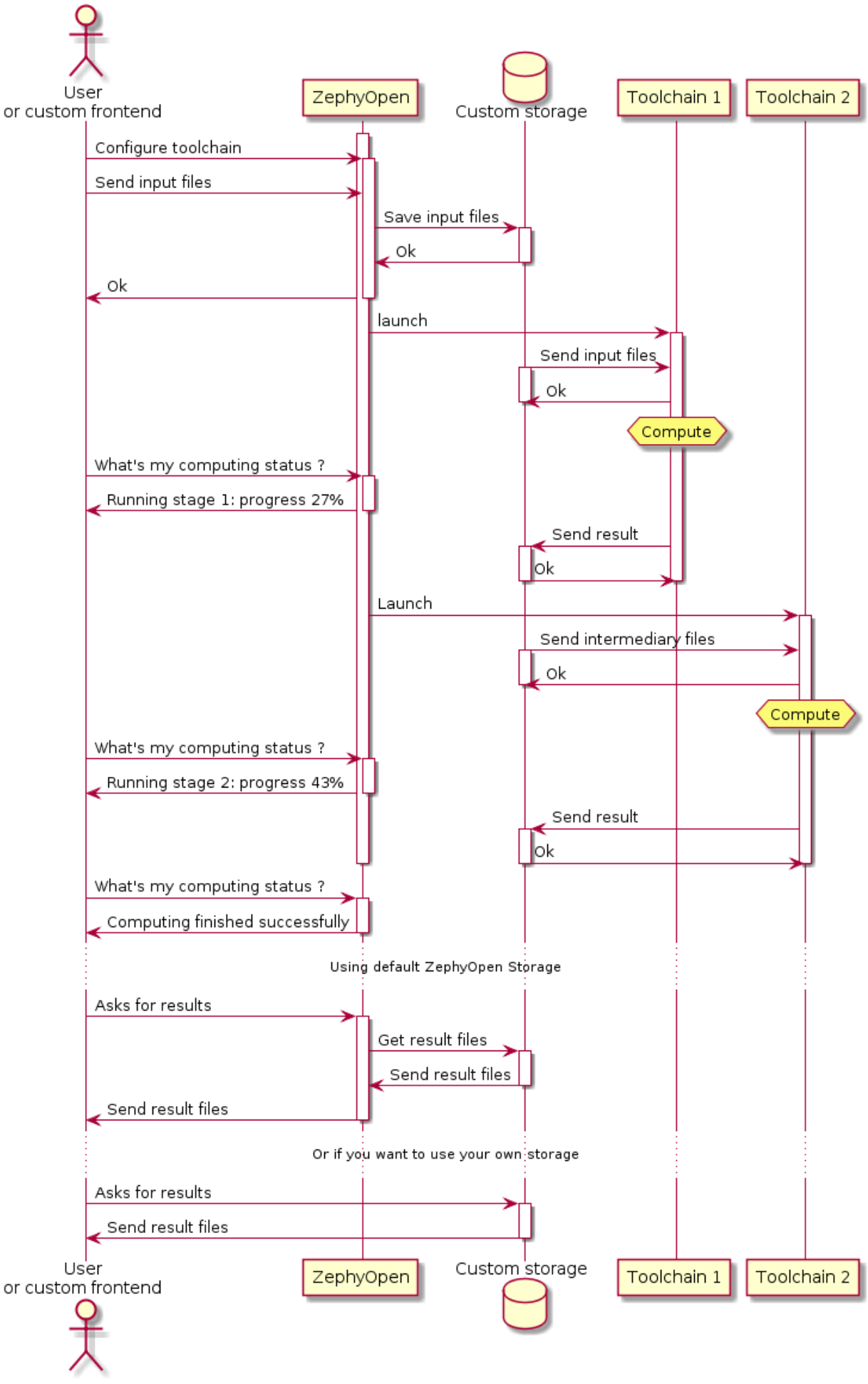


Figure 3: User usage

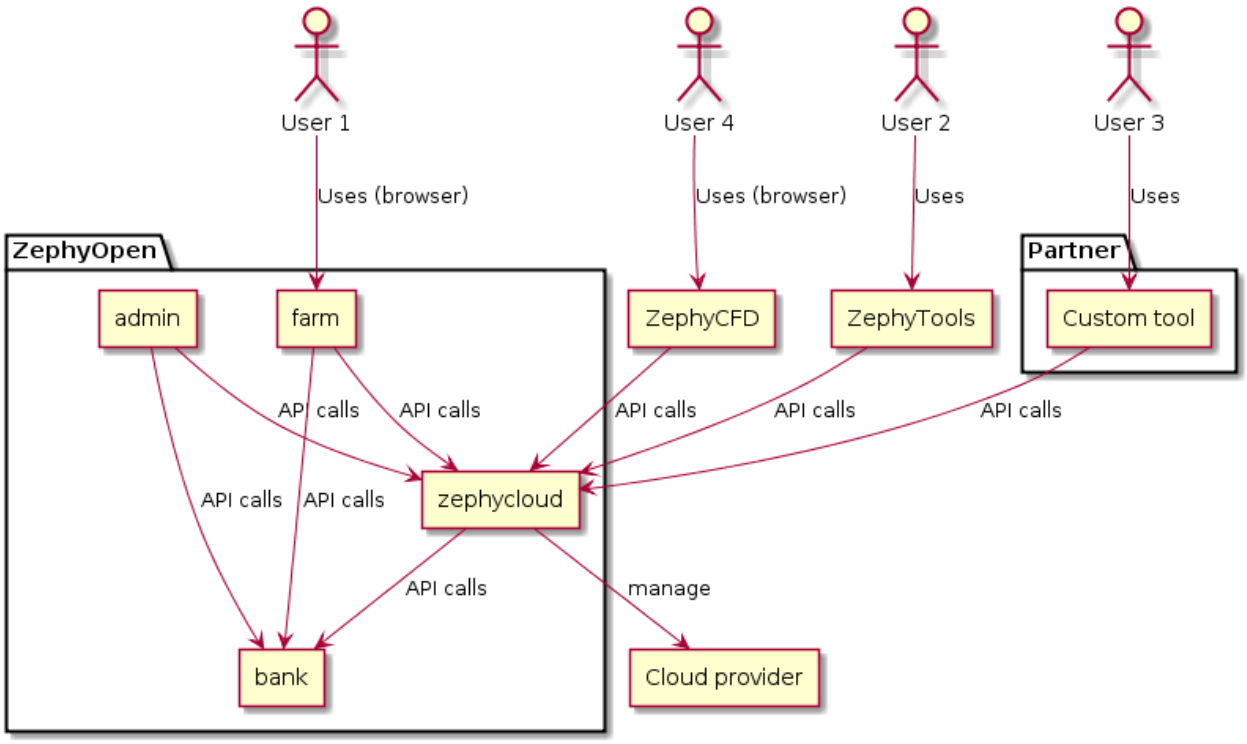


Figure 4: Architecture Overview

- Asymmetric encoding of company secrets based on GPG and custom tooling implementation.
- Automatic daily backup system.

Our websites, tools and API also use some conventional preventive technics:

- https everywhere; we refuse http connections.
- Systematic input sanitizations.
- SQL library escaping to prevent *SQL injections*.
- *Session-less Basic-auth* authentication for some components, to prevent *Session hijacking attack*.
- Systematic confirmation or check of referrer for significant actions, to prevent *Cross-Site Request Forgery*.

### 3.2.2 Security against external tool-chains

Our partners will run some of their code on our cloud accounts. Because some partner could be malicious or just write tool-chain with bugs, we have to use some specific technics:

- The tool-chain will run as an unprivileged user on the cloud virtual machines.
- If several tool-chains need to be run, they will all be launched in a separated virtual machine.
- The virtual machines running tool-chains will not be allowed to access the network (except for clustering) This rule is ensured twice:
  - by a virtual machine firewall;
  - by the cloud provider security policies.
- Virtual machines have no access to any of the data storages.
- Our tool-chain controller will run polling mechanisms to send inputs, check tool-chain status and get results. The virtual machines don't have any specific access nor credential.

## 3.3 API concerns

### 3.3.1 Overview

We splitted the platform into separated modules using the *Single Responsibility Principle*. The modules are separated as follows (refer to Figure 4):

- bank: API responsible for users information and ZephyCoins management
- zephycloud: API responsible for tool-chain execution
- farm: General web interface where users can buy credits, launch tools, etc...
- admin: Administrative web interface for Aziugo team
- ZephyCFD: An web interface dedicated to a specific toolchain

We planed to release several more tool-specific interfaces, but **farm** provides a generic interface for public tool-chains.
The specific interfaces could be either client-side or web interfaces.
They should communicate directly with our APIs (**zephycloud** and **bank**).

Web interfaces could be integrated directly into the **farm** web interface, but this integration will take place inside an *iframe*. Our partners may also develop specific interfaces for their own tool-chains. Our partners should host their specific web interfaces by themselves.

### 3.3.2 Technologies

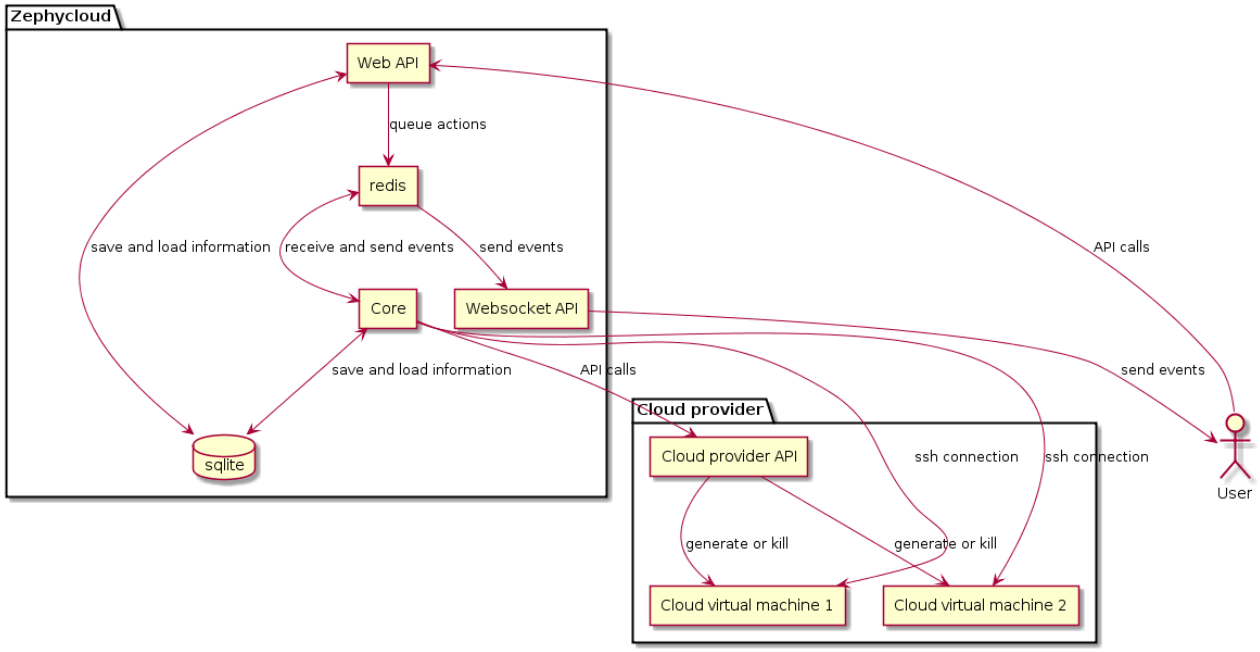


Figure 5: ZephyCloud structure

Because we provide really specialized scientific tools, we don't expect a significant traffic.
However most of the tools are really computing power intensive so that they will all run on cloud virtual machines, and thus will not have any impact on our modules.

Most modules have their own documentation. You will find here only zephycloud specific technical information. For the other modules, please refer to their own documentation.

**zephycloud** is implemented in *python 2.7*.
It uses a *SQLite* database, since most of the data are in a separated module (**bank**).
This database is responsible for tool-chain process information.

This module is also composed of separated modules, to ensure better security and availability:

- The web module: it is responsible for handling requests and queueing actions to execute.
- The core module: it runs the queued actions and manages tool-chain process.
- The websocket API: it dispatches information in real-time to the clients, allowing push notifications.

Communication between modules is based on a Redis server for message passing and the database for data integrity.

### The web module

It is a *json based http API* that allows to launch the toolchain.
The API is versioned via url route prefix.
For now, it uses *Basic-Auth authentication*, but it should use *OAuth* or similar technology for authentication.

It uses *nginx* as reverse proxy.
It uses *gunicorn* as *Web Server Gateway Interface*
It is based on the *Flask* micro framework.

All API calls are located on "https://api.zephy-science.com/v1/"

You can also ignore the "v1" folder until a new API version is released and set as default API.

Some generic rules about the requets:

- All requests should be done via https
- All requests should be json requests.
- All requests require the user to be logged in.
- All requests sending files should call the file argument files[]
- All requests should call a trailing slash url (like a folder address)

All responses should have the following structure, even in case of error:

```
{
    "success": 1,
    "error_msgs": ["List of error messages, or empty array in case of success"],
    "data": "Whatever data. Could be int, string, array or object"
}
```

### The core module

This module is implemented in python.
It consists of a main loop waiting for queued actions to be launched in the **redis** server.
For each tool-chain it should launch, it creates a separated process responsible for the tool-chain management.
This separated processes will:

- create the cloud virtual machine using cloud specific library or API
- push the input files using ssh connections
- launch and monitor the tool-chain on the virtual machine
- retrieve the tool-chain result

The separated processes are designed as exception-safe and ensure the cloud virtual machine to be released in any case. However the core module also has a virtual machine garbage collection loop.

### The websocket module

This module aims to provide real-time information to the client about the tool-chain process the user has sent.
It uses client-server encryption using TLS and the websocket API.
An optional goal of this module is to use the new W3C push API when this API will be stabilized enough and implemented across browsers and operating systems.
It uses internal authentication mechanism until the OAuth module is ready.
It is based on Python 2.7 and tornado library.
It sends and receives json messages.
It's protocol is also versionned and protocol version negotiation is done at connection.
It receives messages from the redis server and can read the database as read-only to limit security issues.