Codabench Wiki

Codabench Wiki PDF

Codabench Team

Table of contents

1. Home

1.1 Documentation

Welcome to the Codabench wiki!

Codabench is a platform allowing you to flexibly specify a benchmark. First you define tasks, e.g. datasets and metrics of success, then you specify the API for submissions of code (algorithms), add some documentation pages, and "CLICK!" your benchmark is created, ready to accept submissions of new algorithms. Participant results get appended to an ever-growing leaderboard.

You may also create inverted benchmarks in which the role of datasets and algorithms are swapped. You specify reference algorithms and your participants submit datasets.

Here are some links to get you started:

Getting Started with Codabench
Basic Installation Guide
Compute Worker Setup
Administrative Procedures

Use the top bar or the search functionality to navigate the wiki!

1.2 Useful links

Governance Document
Privacy and Terms of Use
About

2. Participants

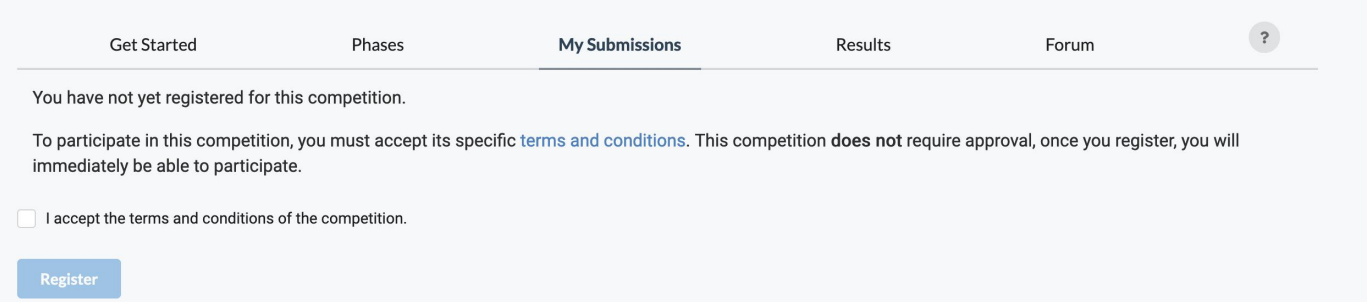
2.1 Participating in a Competition

Signing up and updating your settings

When you sign up, you will have to provide your name and a valid email.

Registering for a Benchmark

To make an entry in a benchmark click on the "My Submissions" tab, you will then be prompted to accept the rules to register to that benchmark. When registering, a request may be sent to the benchmark organizer. You will be notified when the benchmark organizer has approved your registration request. Follow the instructions of the organizers.

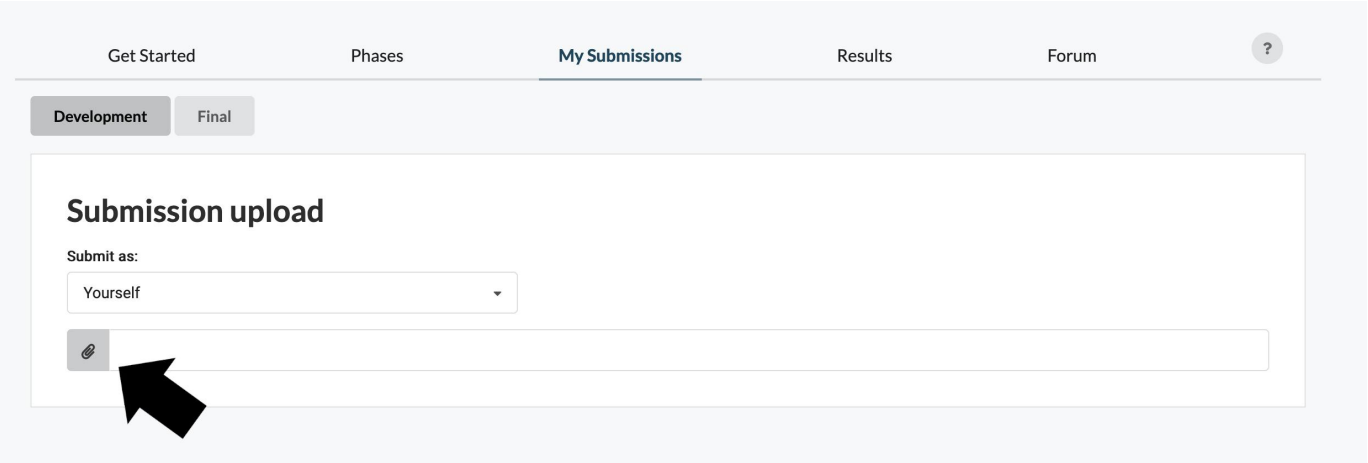


Making a Submission

Making a submission to a benchmark involves uploading a bundle (.zip archive) containing files with your answer, in the format that has been specified by the benchmark organizer. There are two types of submissions: - ***Code*** submissions contain a metadata file specifying the command to execute - ***Results*** submission contain the solution to the problem (no code executed on the platform)

To make a submission

1. Sign in to Codabench. If you do not have an account, you will need to create one.
2. Select the benchmark you want to work with.
3. Click the **My Submissions** tab. Here, you can access the data that has been provided by the benchmark organizer.
4. Click on the **paper clip logo** and select your zip file.

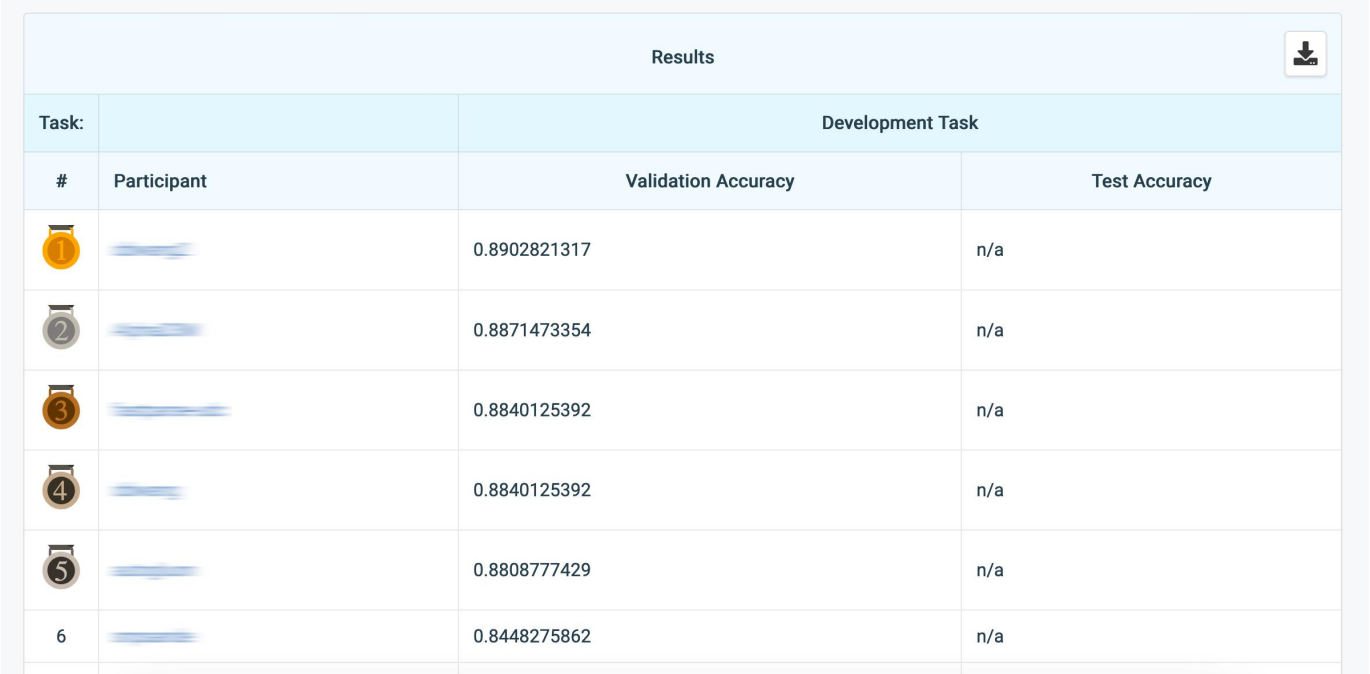


On this page, you can make new submissions, and see previous submissions for each phase in the competition.

You can also view all your submissions in the Resources Interface.

Viewing Benchmark Results

You can keep up with the progress of benchmarks you are participating in by clicking on the **Results** tab. This will display the leaderboard.



Results

Task:		Development Task	
#	Participant	Validation Accuracy	Test Accuracy
1	[redacted]	0.8902821317	n/a
2	[redacted]	0.8871473354	n/a
3	[redacted]	0.8840125392	n/a
4	[redacted]	0.8840125392	n/a
5	[redacted]	0.8808777429	n/a
6	[redacted]	0.8448275862	n/a

3. Organizers

3.1 Benchmark Creation

3.1.1 Getting Started Tutorial

Codabench is an upgraded version of the CodaLab Competitions platform, allowing you to create either **competitions** or **benchmarks**. A benchmark is essentially an **ever-lasting competition** with **multiple tasks**, for which a participant can make **multiple entries** in the result table.

This getting started tutorial shows a **simple example** of how to create a competition. Advanced users should check fancier examples and the full documentation. If you simply wish to participate in a benchmark or competition, go to Participating in a benchmark.

Getting ready

- Create a Codabench account (if not done yet)
- Download the sample competition bundle and the sample submission.
- Do not unzip them.

Create a competition

- From the front page of Codabench, in the top menu, go to the **Benchmark > Management**
- Click the green **Upload** button at the top right.
- Upload the (zipped) sample competition bundle => this will create your competition.

Make changes

- Click on the **Edit** gray button at the top to enter the editor.
- Make small changes
  - Change the logo in the **Details** tab to another png of jpg file.
  - Change the end date in the **Phases** tab: if the competition is terminated, you will not be able to make submissions.
- Save your changes and verify that they have become effective.

Make a submission

- In your competition page, go to the tab **My submissions**
- Submit the (zipped) sample submission bundle you downloaded.
- When your submission finishes, go to the **Result tab** to check if it shows up on the leaderboard.

> ⚠ **In some competitions, submissions successfully processed do not automatically get pushed to the leaderboard**

Publish your competition

- If you want to make your competition visible to all, you must publish it by checking the **Publish** box at the very bottom of the editor **Details** page.
- After you publish your competition, it should now be visible when you are not logged in, at the URL of your competition page.

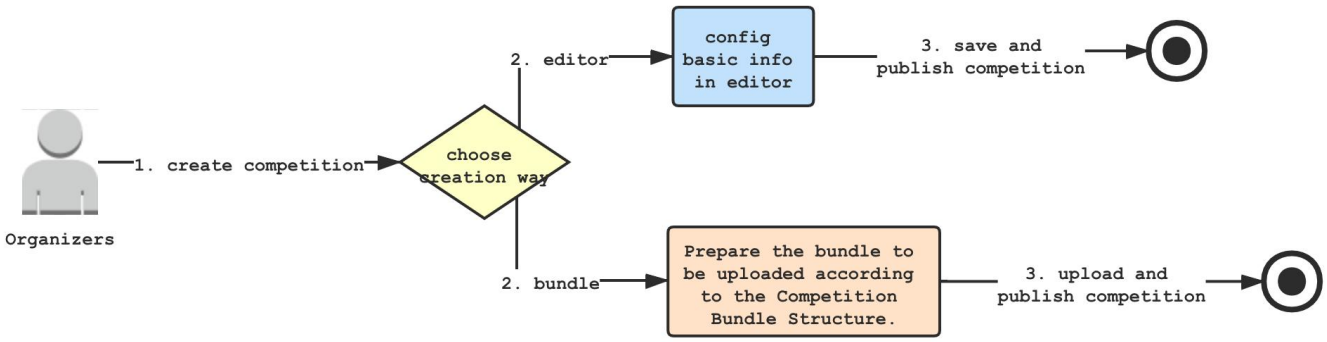
You are done with this simple tutorial. Next, check the more advanced tutorial.

You can also check out this blog post: How to create your first benchmark on Codabench.

3.1.2 Advanced Tutorial

Here is an advanced tutorial. If you are new to CodaBench, please refer to get started tutorial first. In this article, you'll learn how to use more advanced features and how to create benchmarks using either the editor or bundles. Before proceeding to our tutorial, make sure you have registered for an account on the Codabench website.

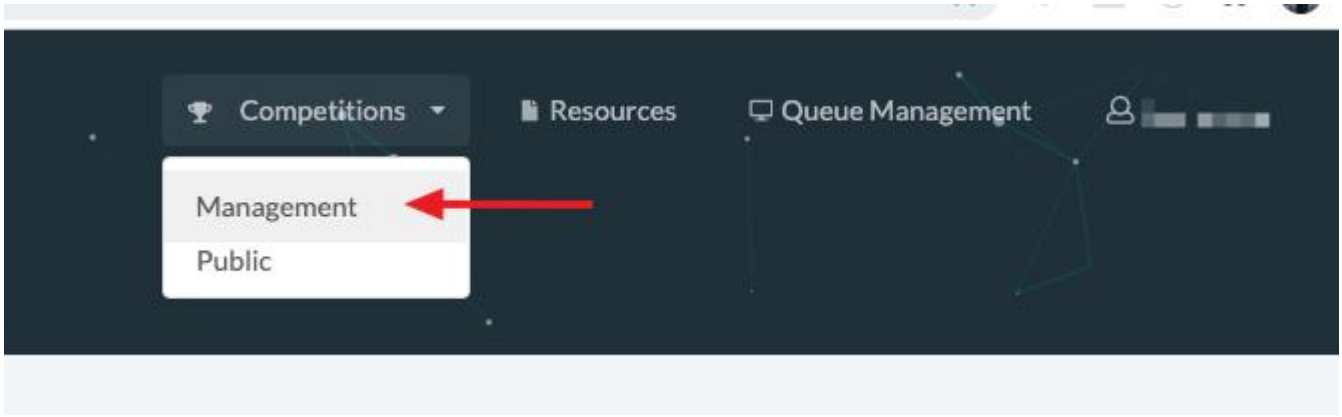
The image below is an overview of the benchmark creation process



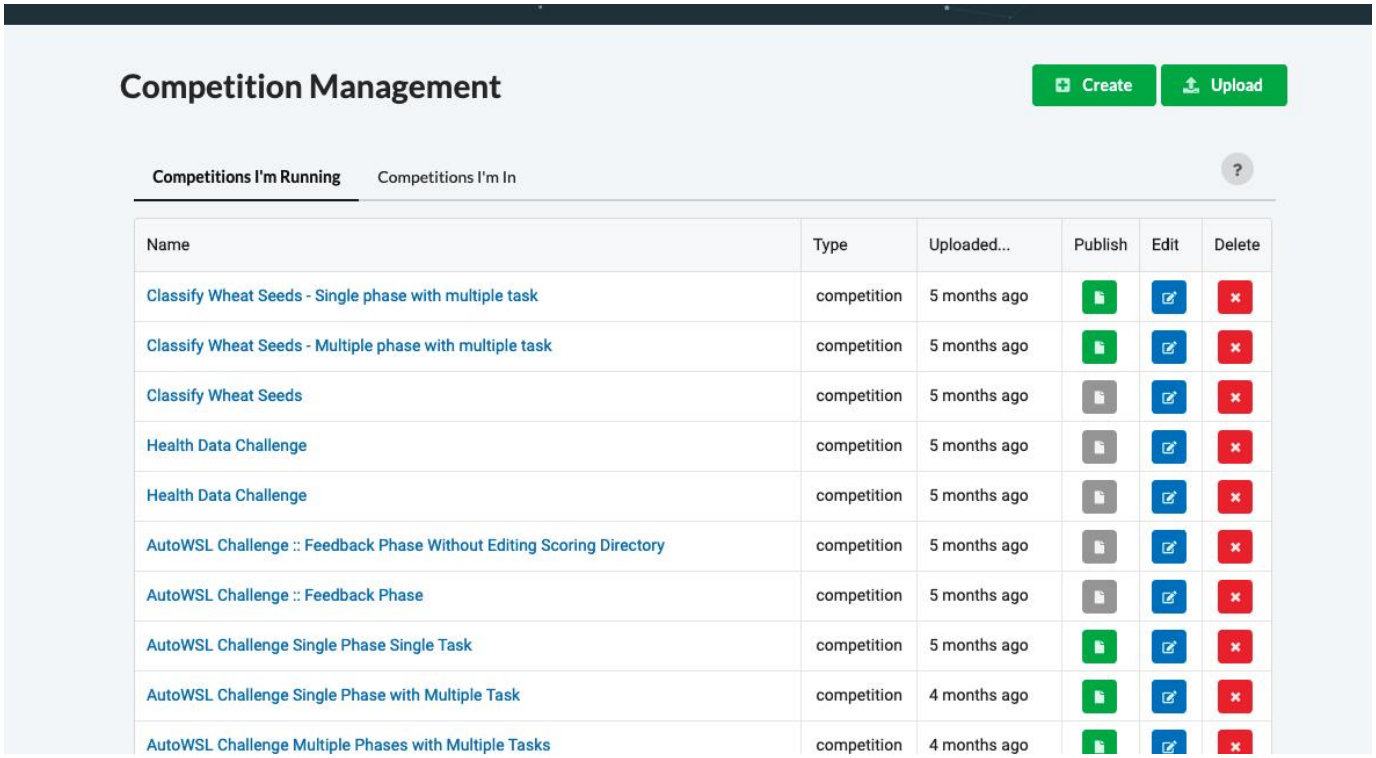
Creating a Benchmark by Editor

In this chapter, I'll take you step by step through the Editor's approach to creating benchmark, including algorithm type and dataset type.

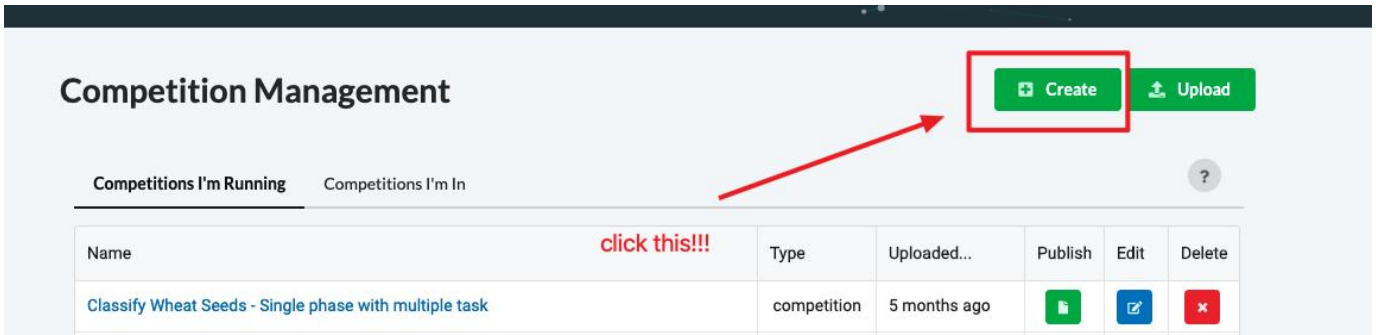
STEP 1: CLICK ON MANAGEMENT IN THE TOP RIGHT CORNER OF CODABENCH'S HOME PAGE UNDER COMPETITIONS.



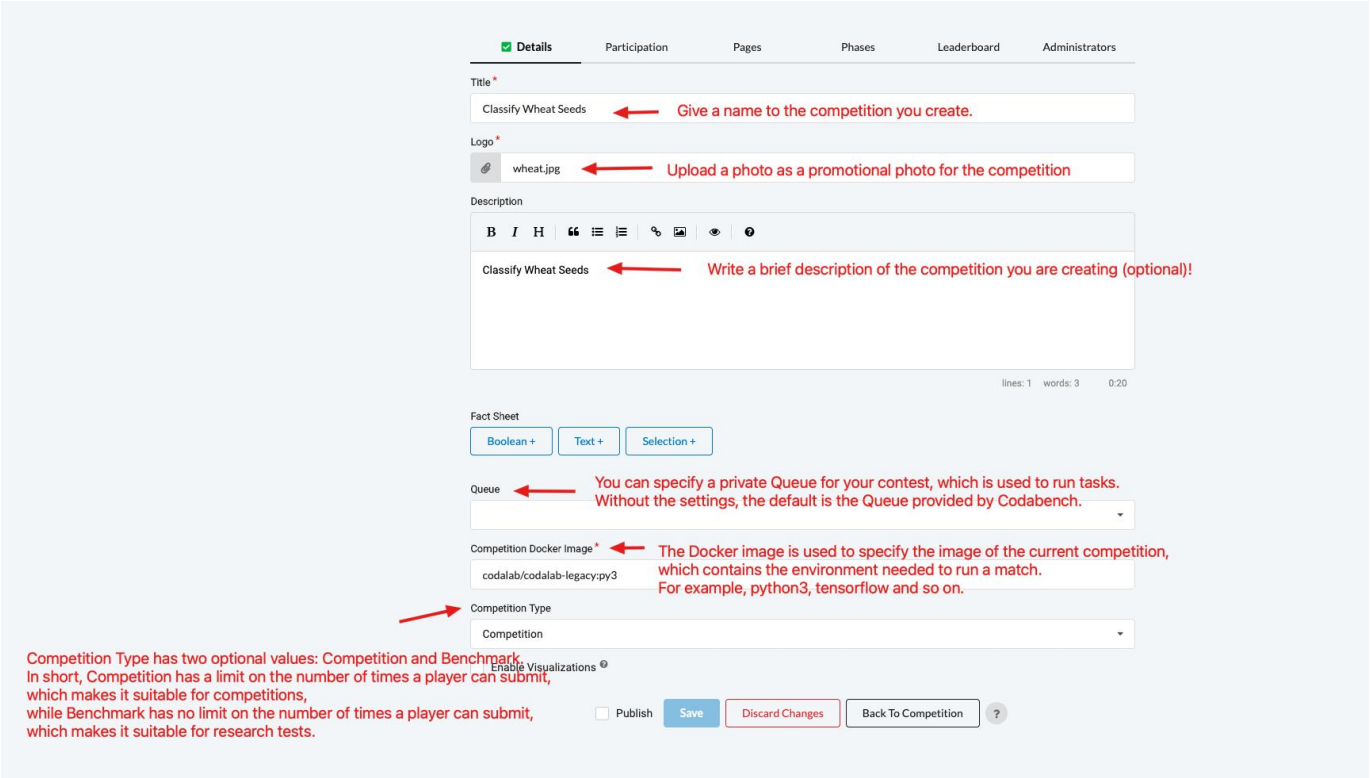
When you click on it, you will see the screen as shown in the screenshot below.



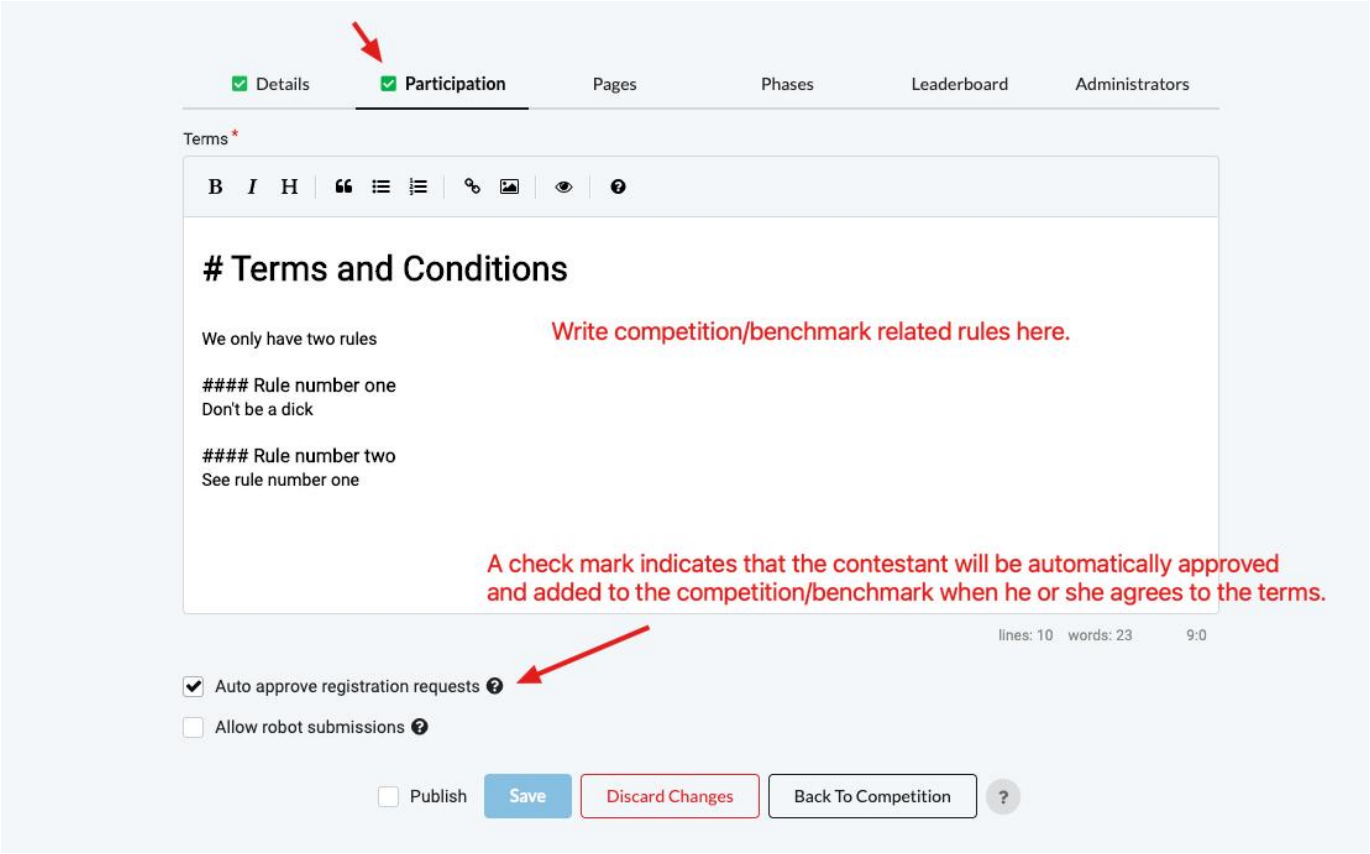
STEP 2: CLICK ON THE CREATE BUTTON IN THE TOP RIGHT CORNER OF COMPETITION MANAGEMENT.



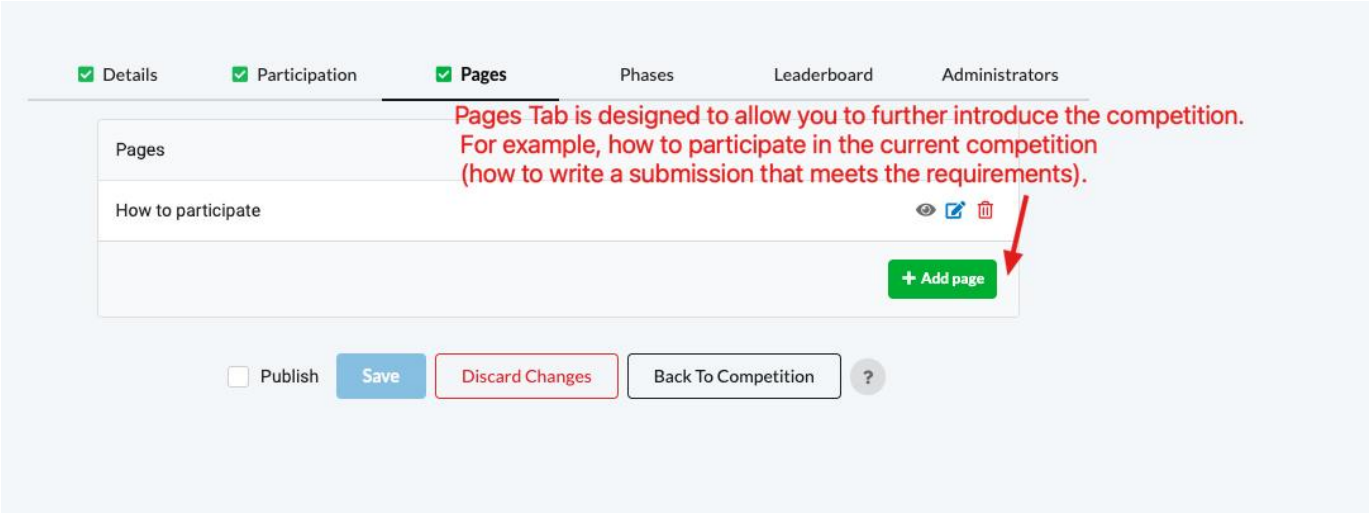
STEP 3: FILL IN THE DETAILS TAB CONTENT.



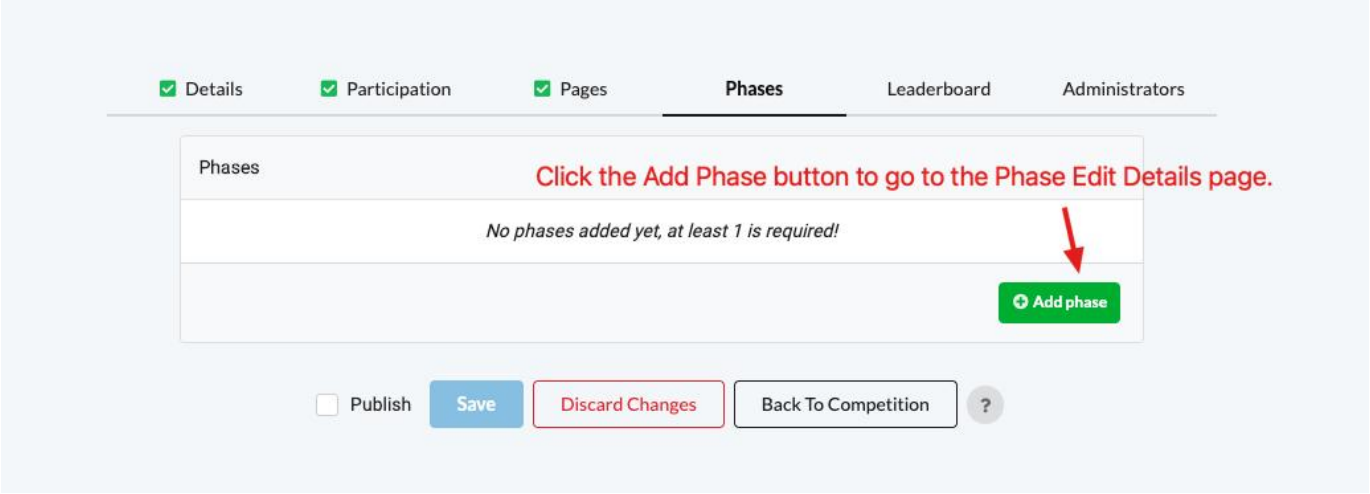
STEP 4: FILL IN THE PARTICIPANT TAB.

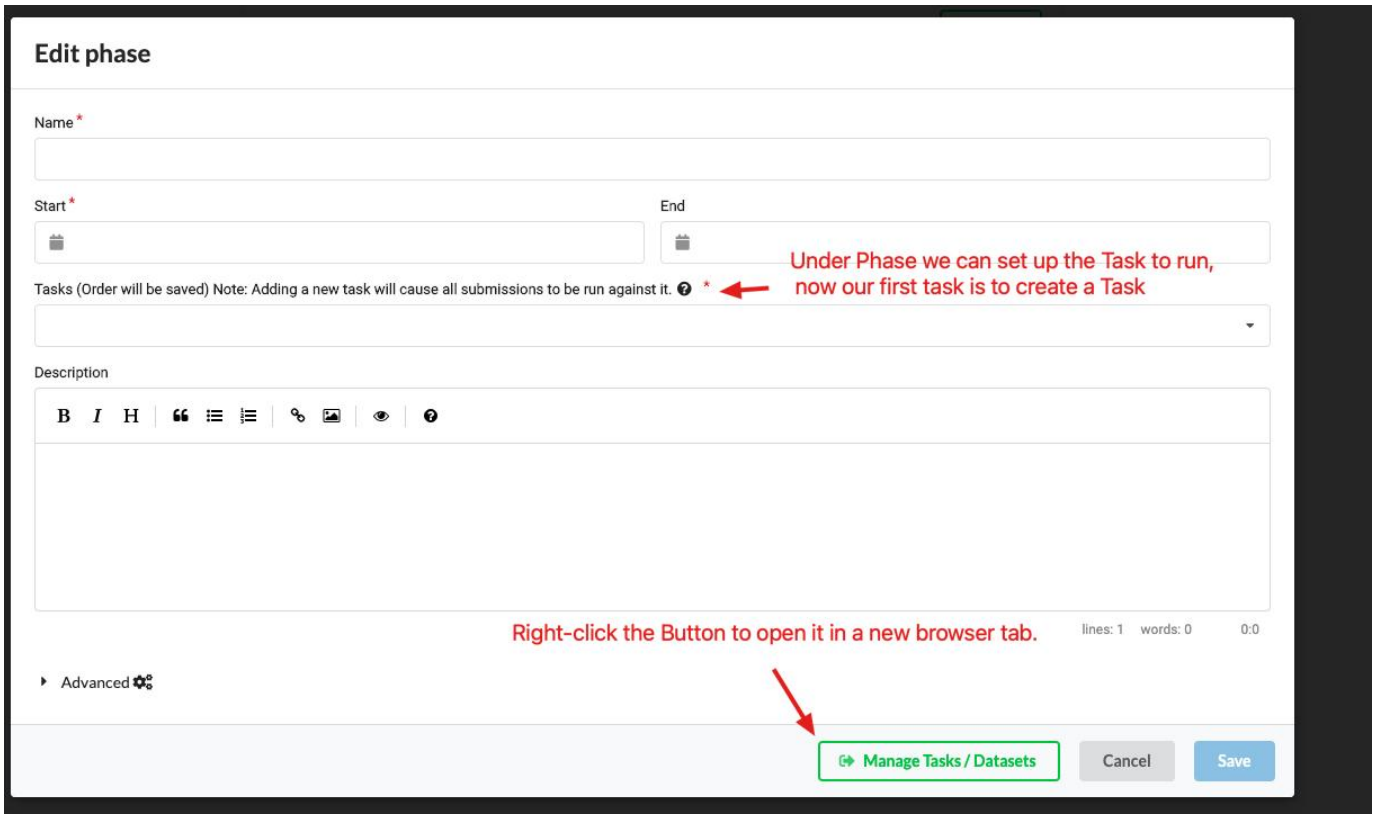


STEP 5: FILL IN THE PAGES TAB.



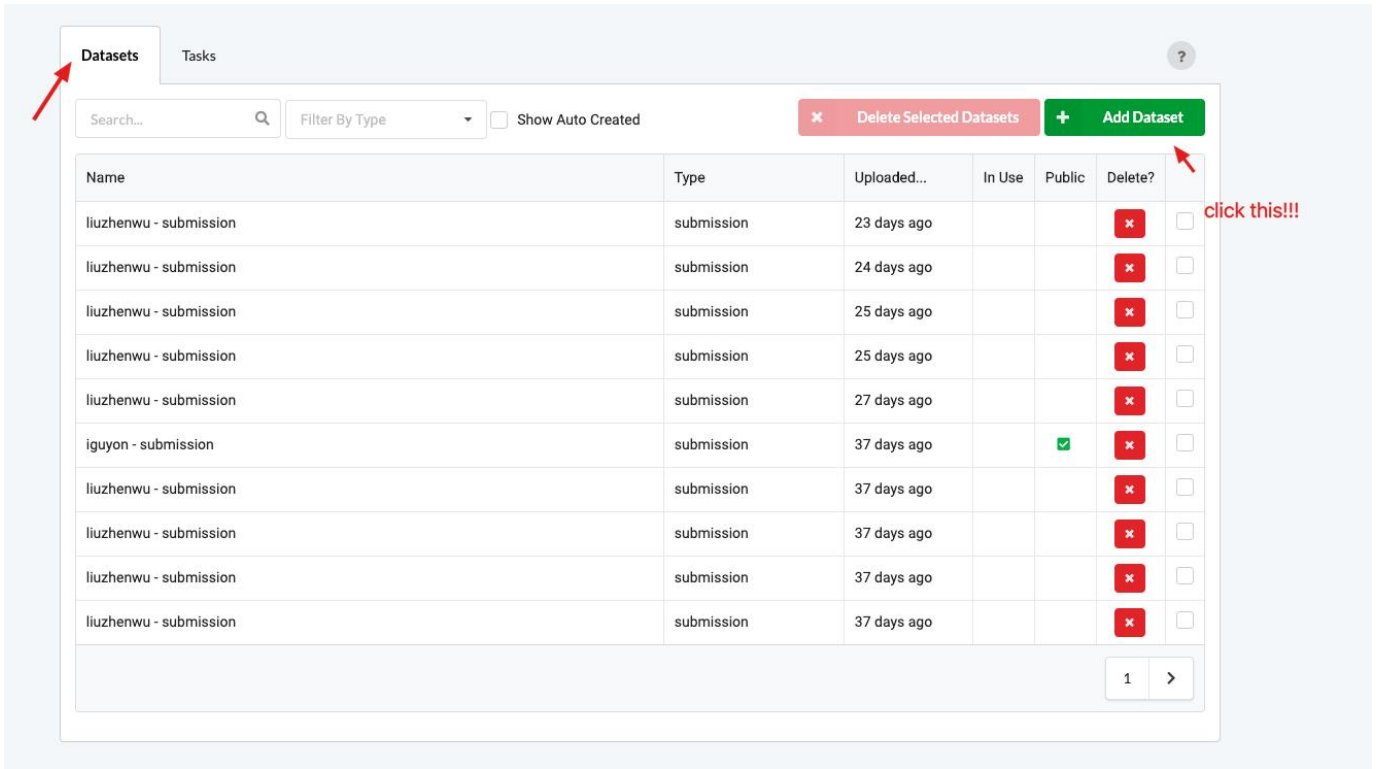
STEP 6: FILL IN THE PHASES TAB.





When you click on the Manage Tasks/Datasets button, you will see the screenshot shown below

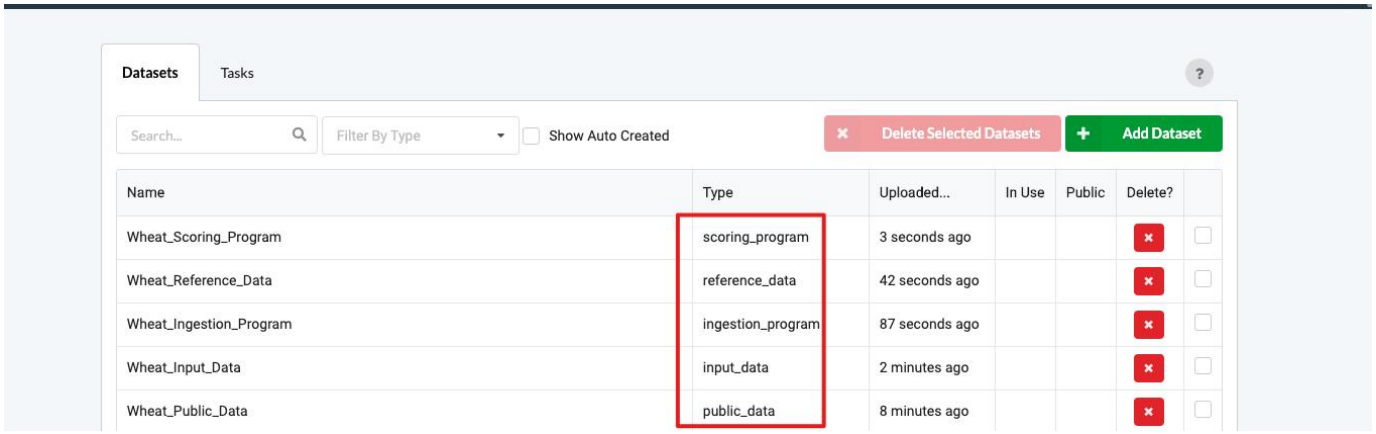
Click the Add Dataset button in the diagram to upload the resource files needed to create the competition.

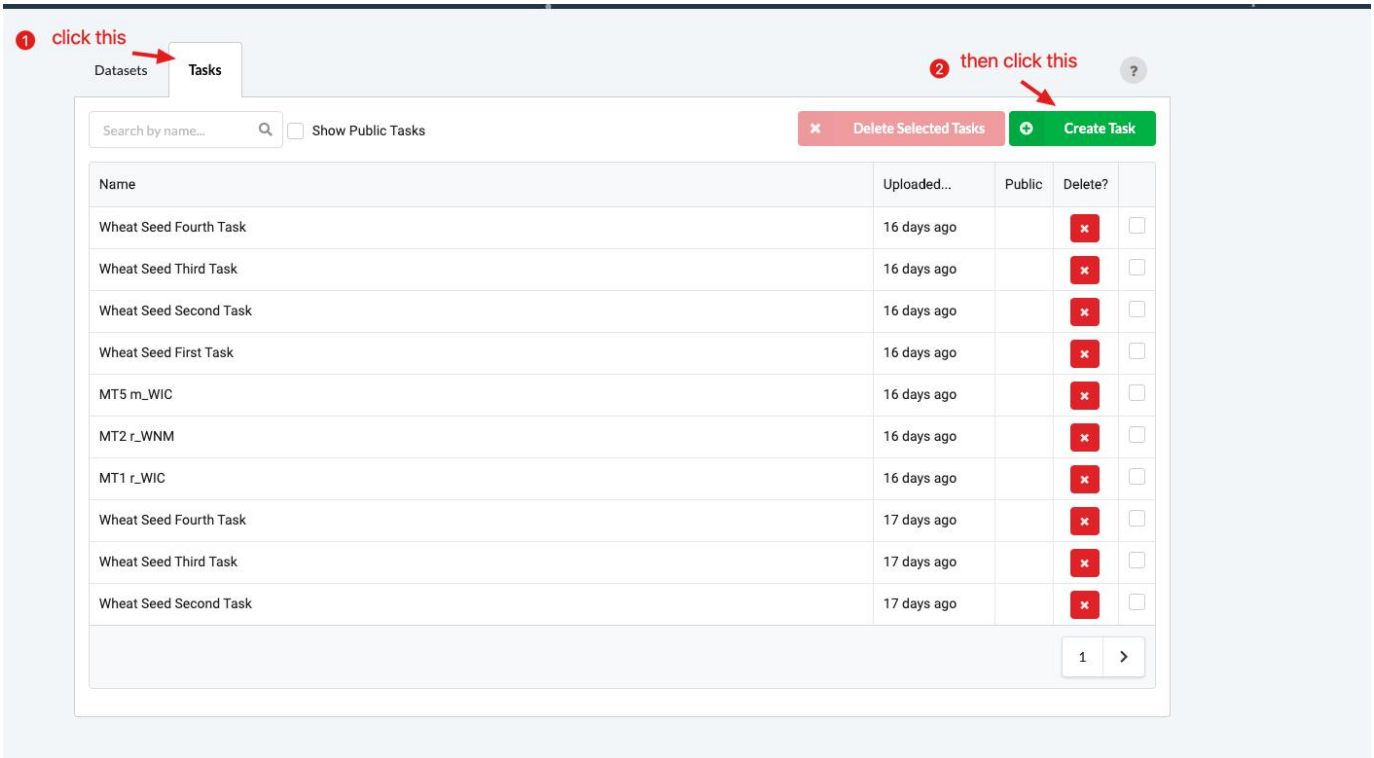


Creating a phase will require a bundle of the following types (.zip format), I'll give you more details on how to write these bundle later.

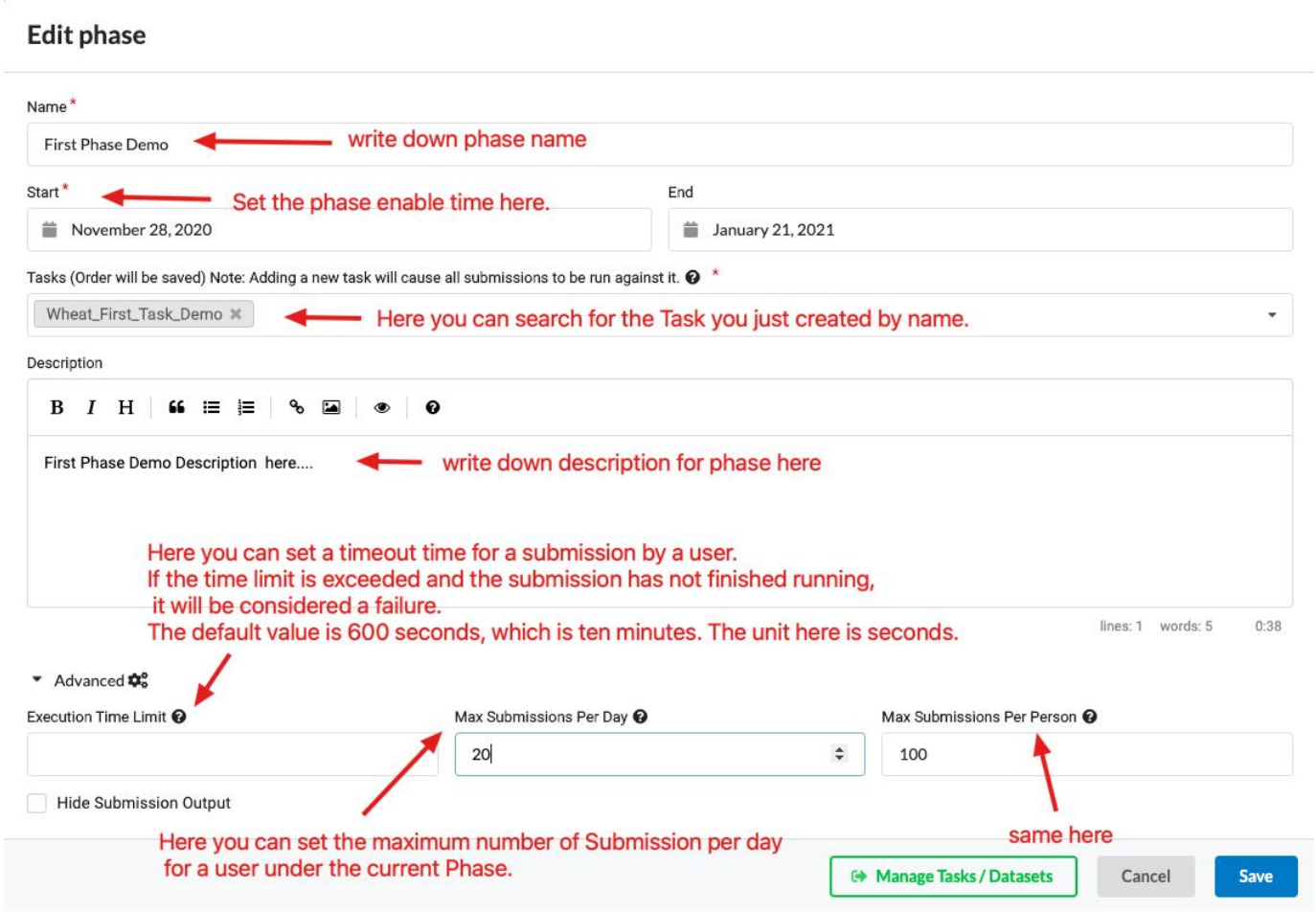
Now you just need to have this concept in your mind

Here are the screenshots of the 5 types of bundles after they were uploaded

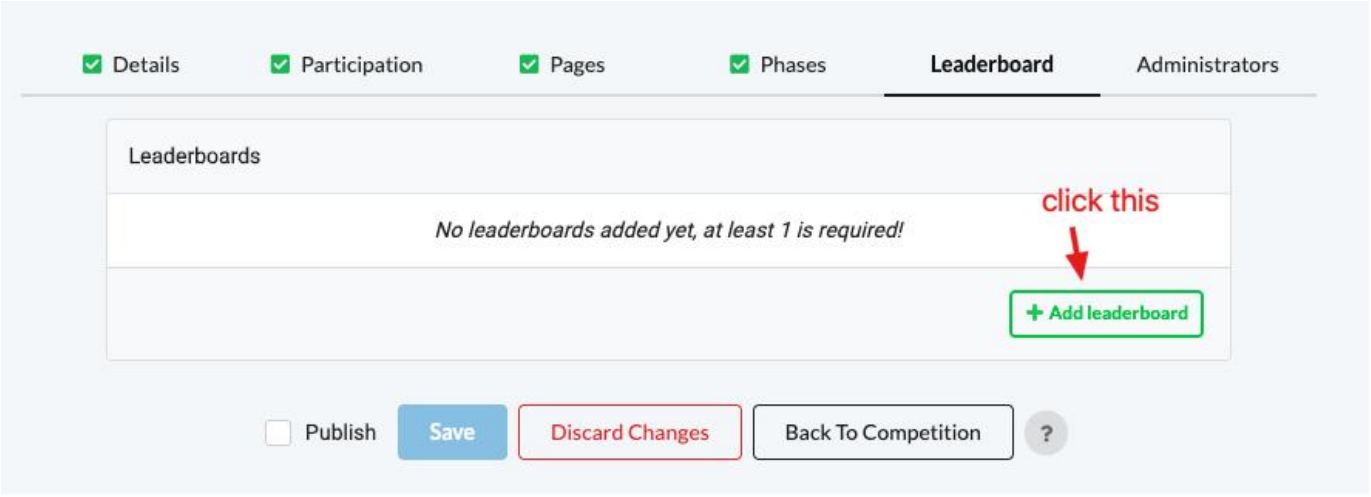


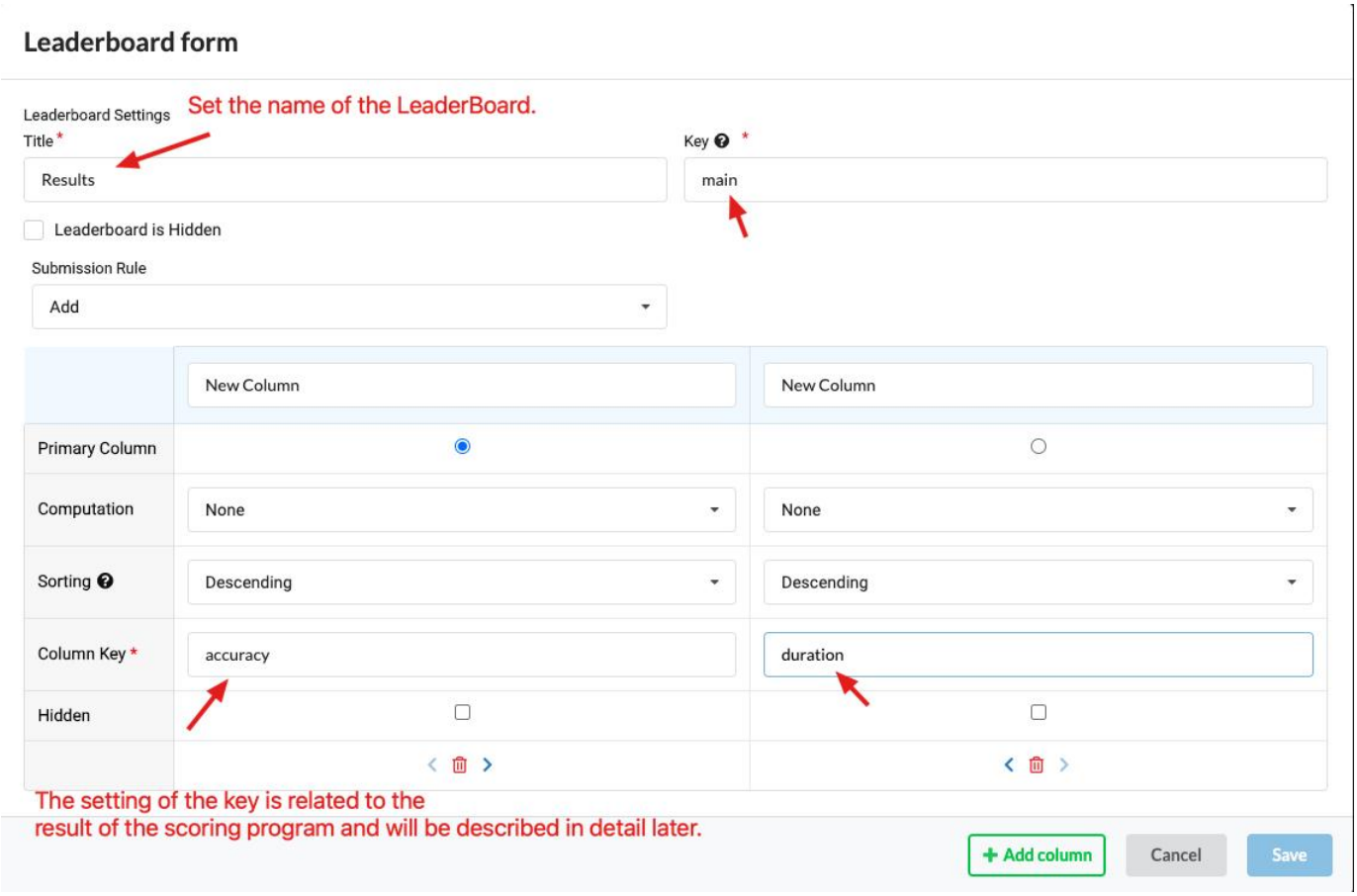

1 click this
Datasets
Tasks
2 then click this
Search by name...
Show Public Tasks
Delete Selected Tasks
Create Task
Name
Uploaded...
Public
Delete?
Wheat Seed Fourth Task 16 days ago
Wheat Seed Third Task 16 days ago
Wheat Seed Second Task 16 days ago
Wheat Seed First Task 16 days ago
MT5 m_WIC 16 days ago
MT2 r_WNM 16 days ago
MT1 r_WIC 16 days ago
Wheat Seed Fourth Task 17 days ago
Wheat Seed Third Task 17 days ago
Wheat Seed Second Task 17 days ago
1


Create Task
Details
Datasets
Name *
Wheat_First_Task_Demo
Description *
Wheat_First_Task_Demo
Create
Cancel

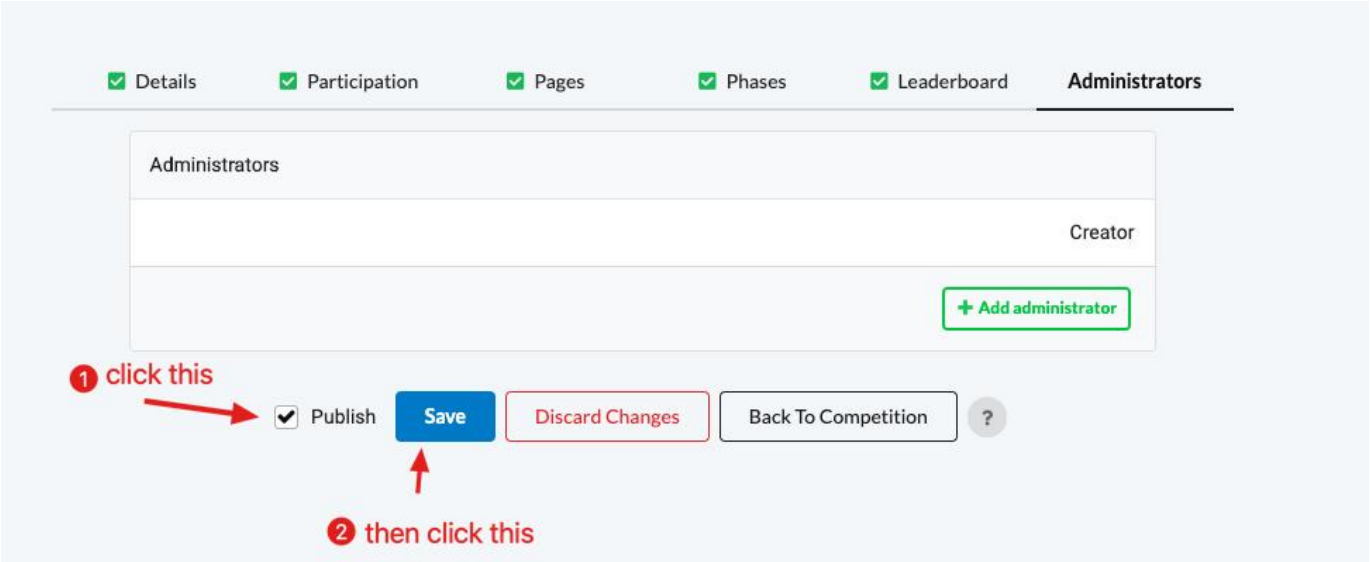


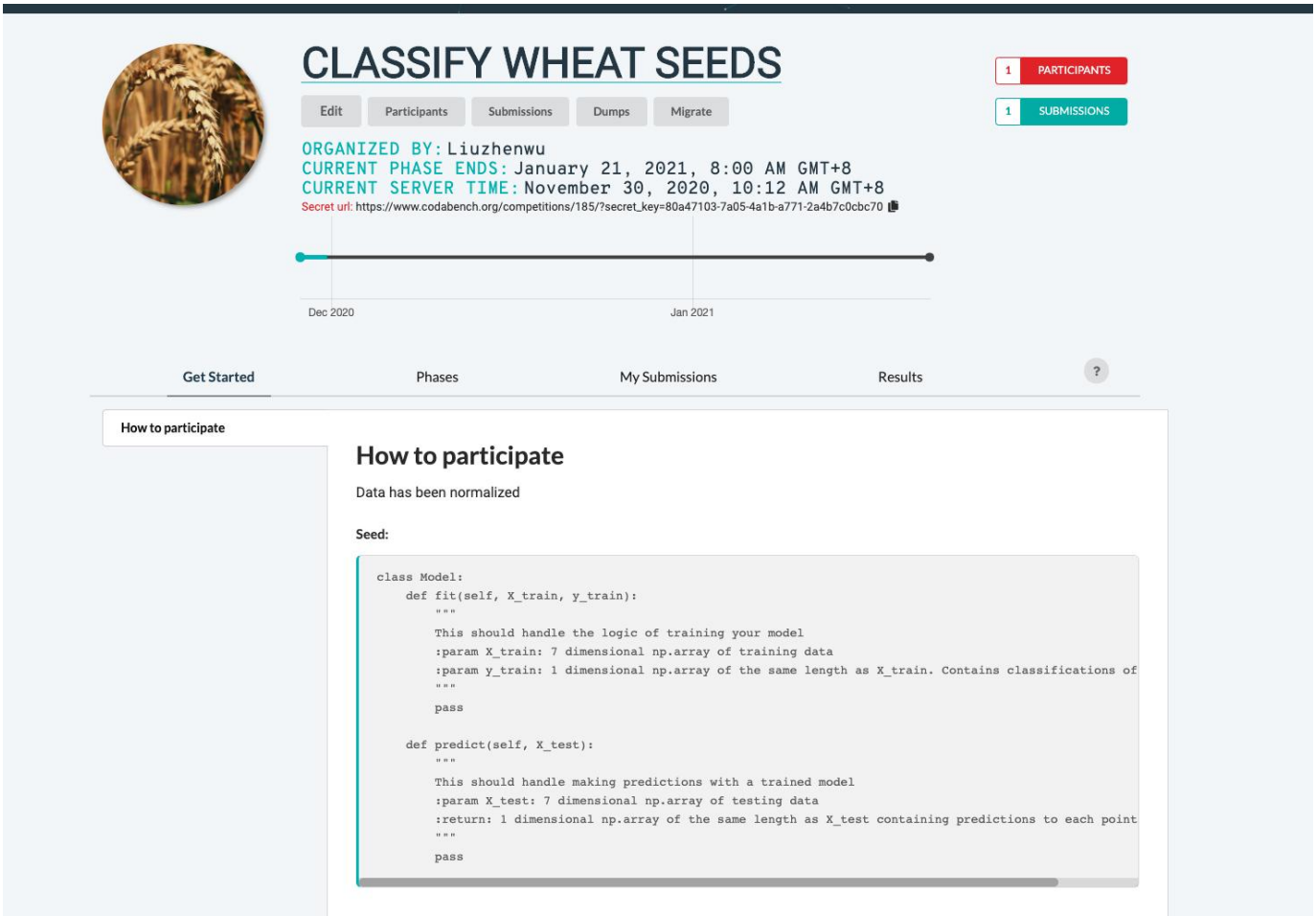
STEP 7: FILL IN THE LEADERBOARD TAB.





STEP 8: SAVE AND PUBLISH THE BENCHMARK





Creating a Benchmark by Bundle

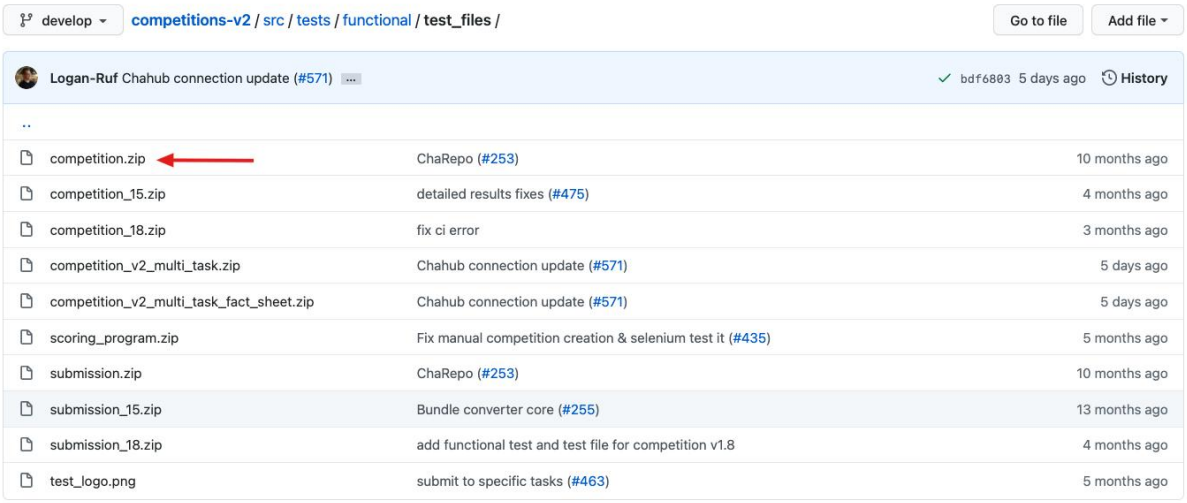
Creating a benchmark through bundles is a much more efficient way than using editors.

SIMPLE VERSION EXAMPLE: CLASSIFY WHEAT SEEDS

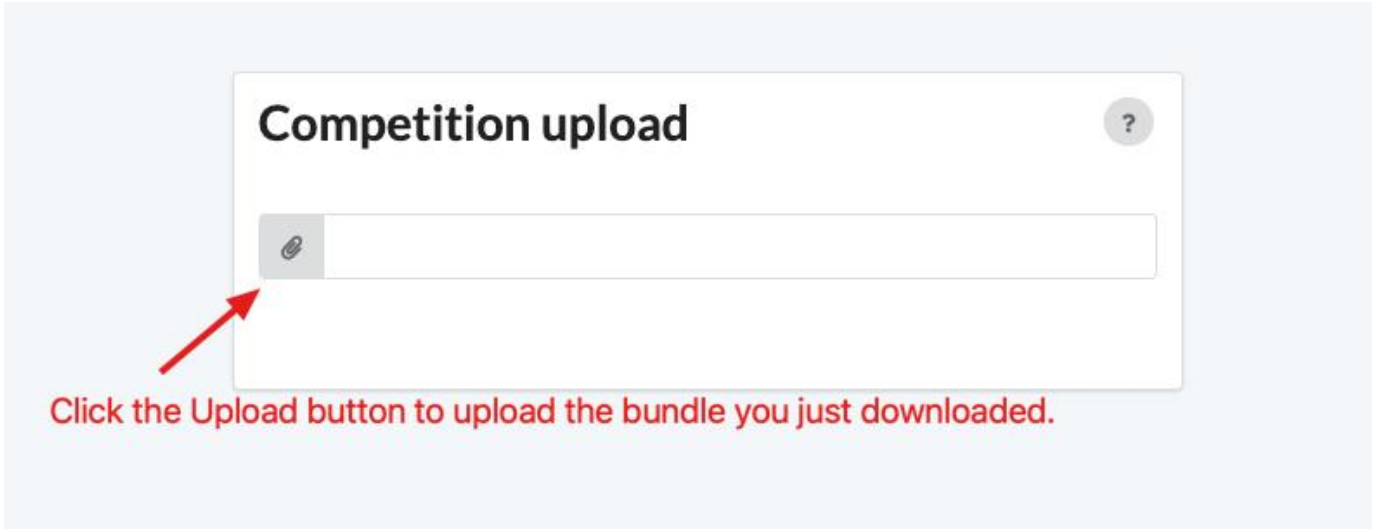
STEP 1: DOWNLOAD BUNDLE

https://github.com/codalab/competition-examples/blob/master/codabench/wheat_seeds/code_submission_bundle.zip

Click on the link above to download the bundle in the screenshot.

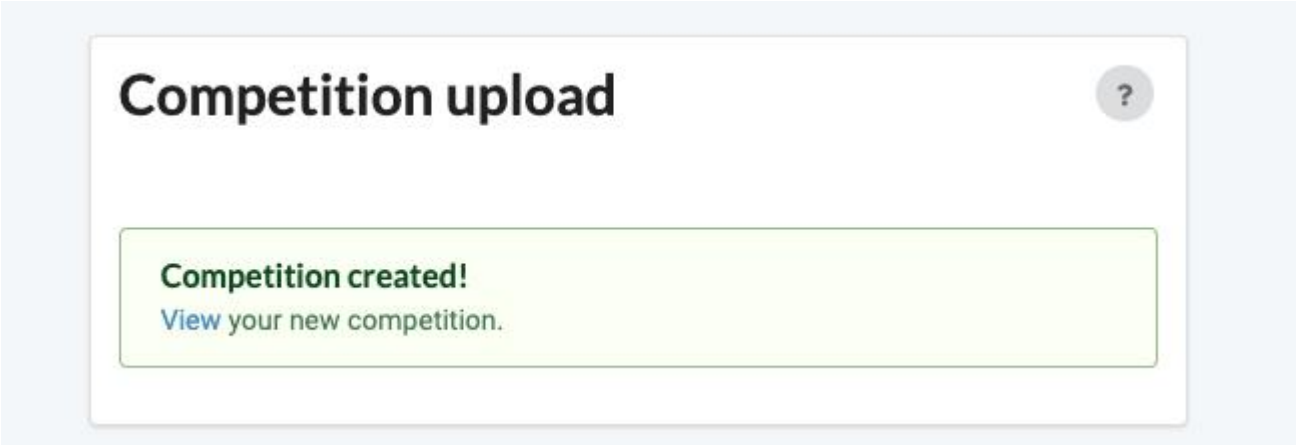


STEP 2: GO TO THE BENCHMARK UPLOAD PAGE

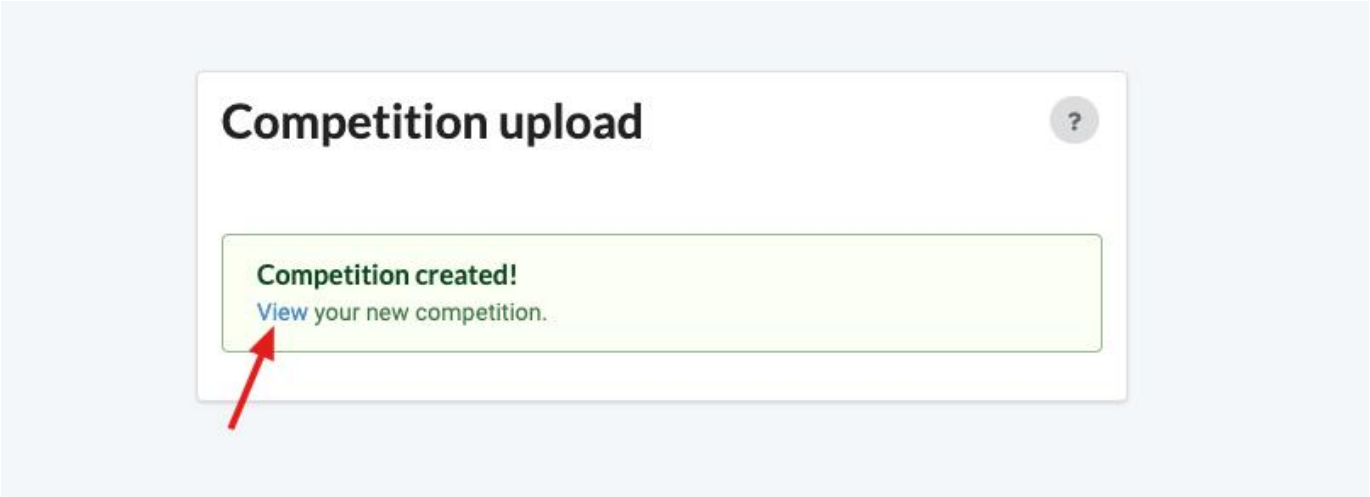


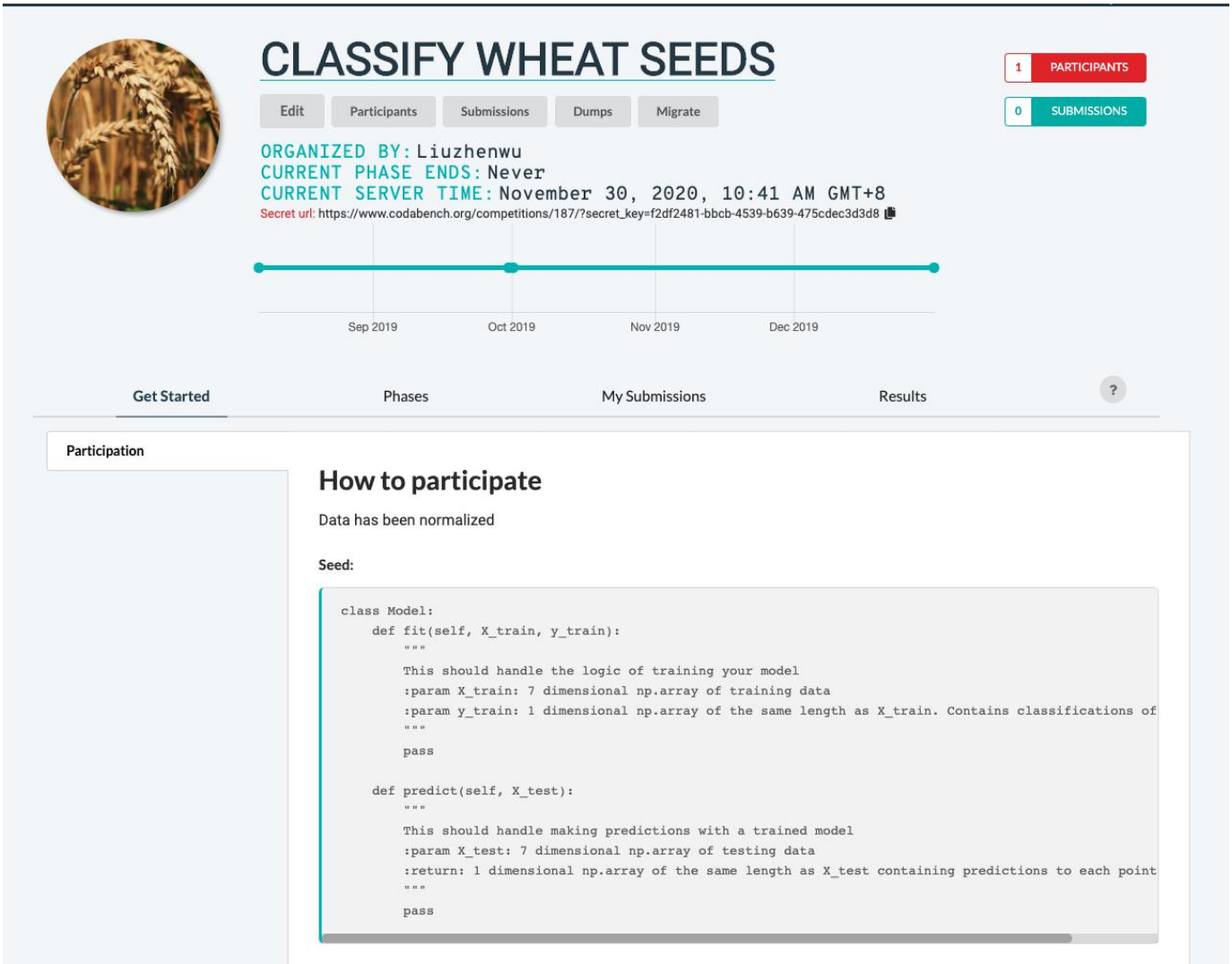
STEP 3: UPLOAD THE BUNDLE

After the bundle has been uploaded, you will see the screenshot shown below.



STEP 4: VIEW YOUR NEW BENCHMARK





Benchmark Examples

Example bundles for code & dataset competition can be found here:

https://github.com/codalab/competition-examples/tree/master/codabench

IRIS

Iris Codabench Bundle is a simple benchmark involving two phases, code submission and results submission.

AUTOWSL

Two versions of the Automated Weakly Supervised Learning Benchmark: - Code submission benchmark - Dataset submission benchmark

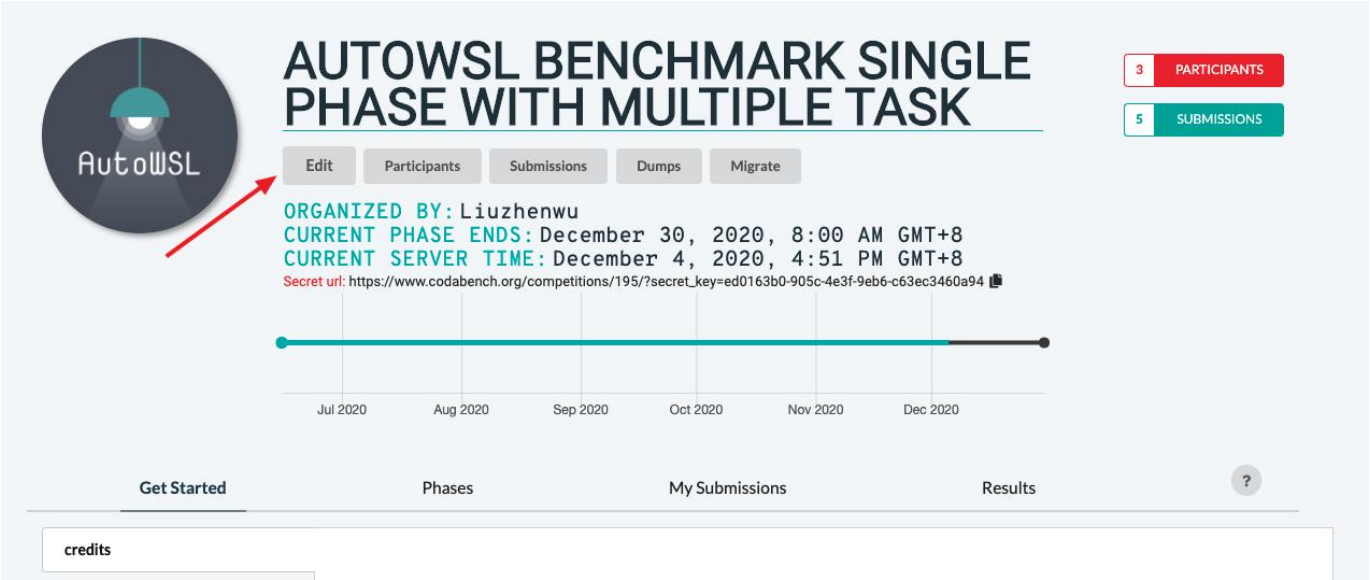
MINI-AUTOML

Mini-AutoML Bundle is a benchmark template for Codabench, featuring code submission to multiple datasets (tasks).

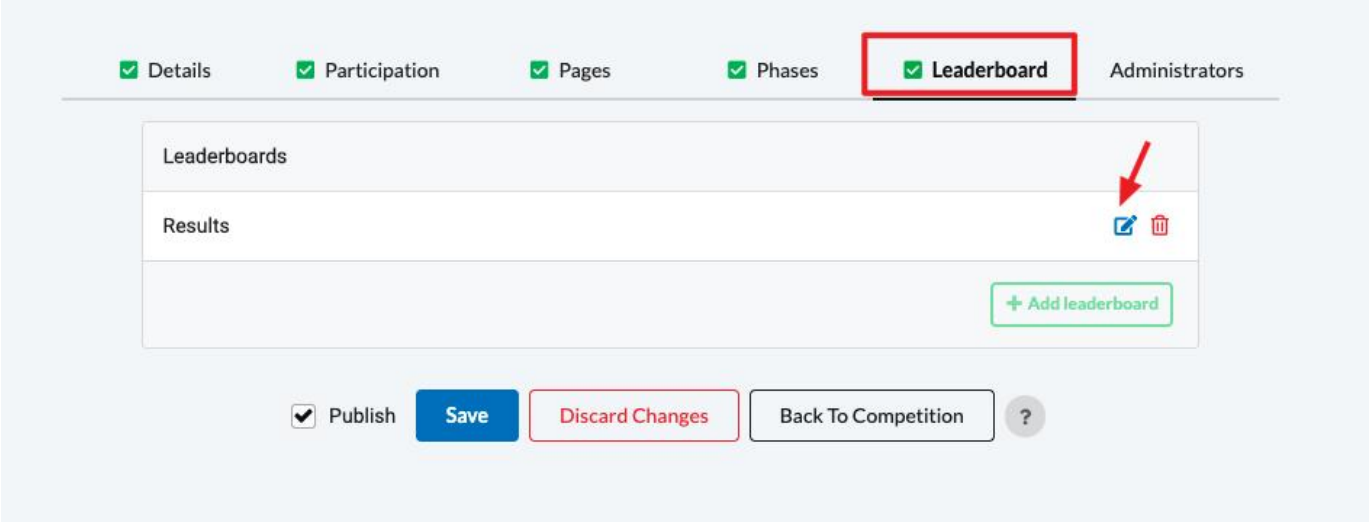
How do I set up submission comments for multiple submissions?

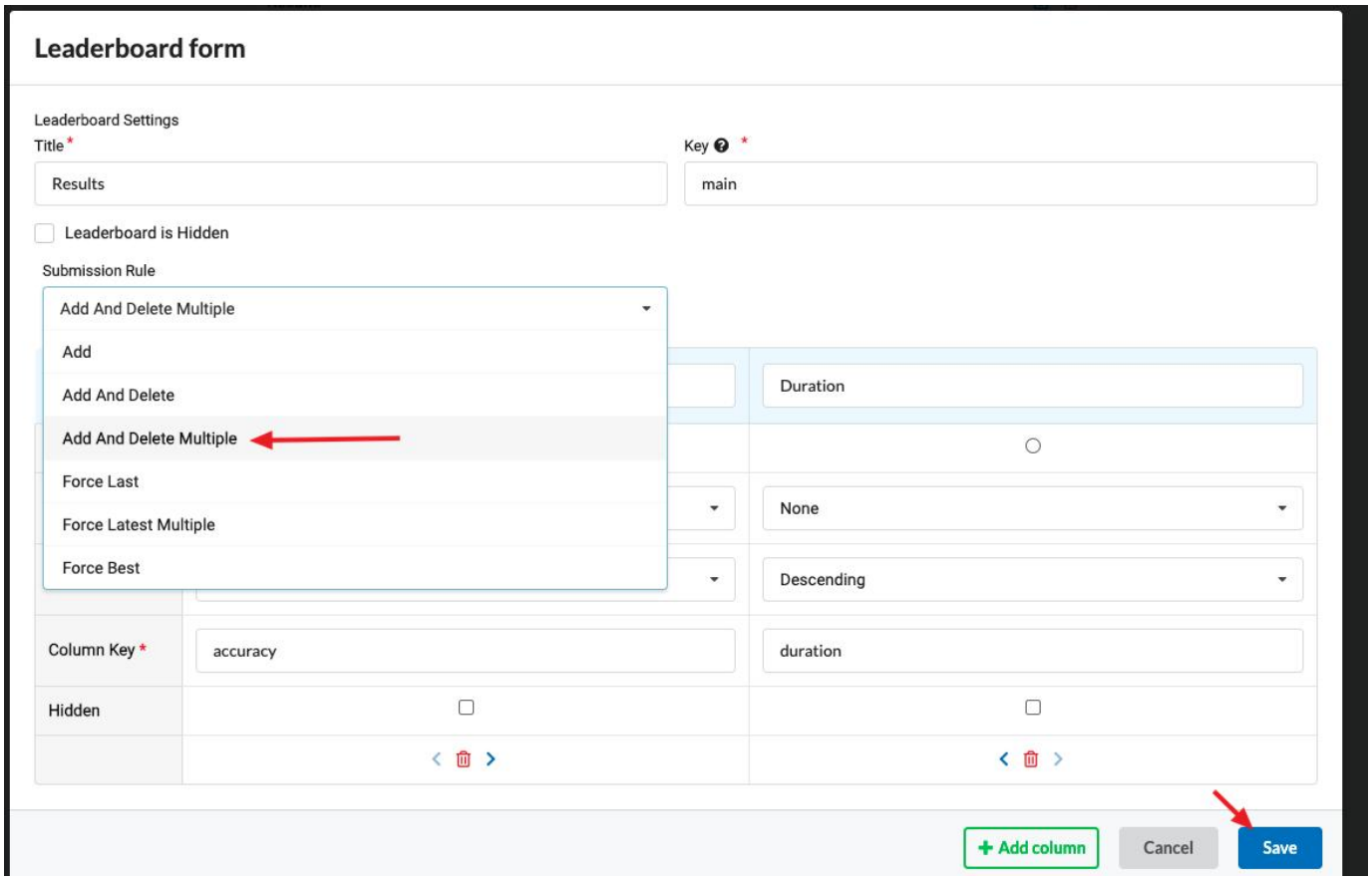
STEPS

Step 1: Click the edit button

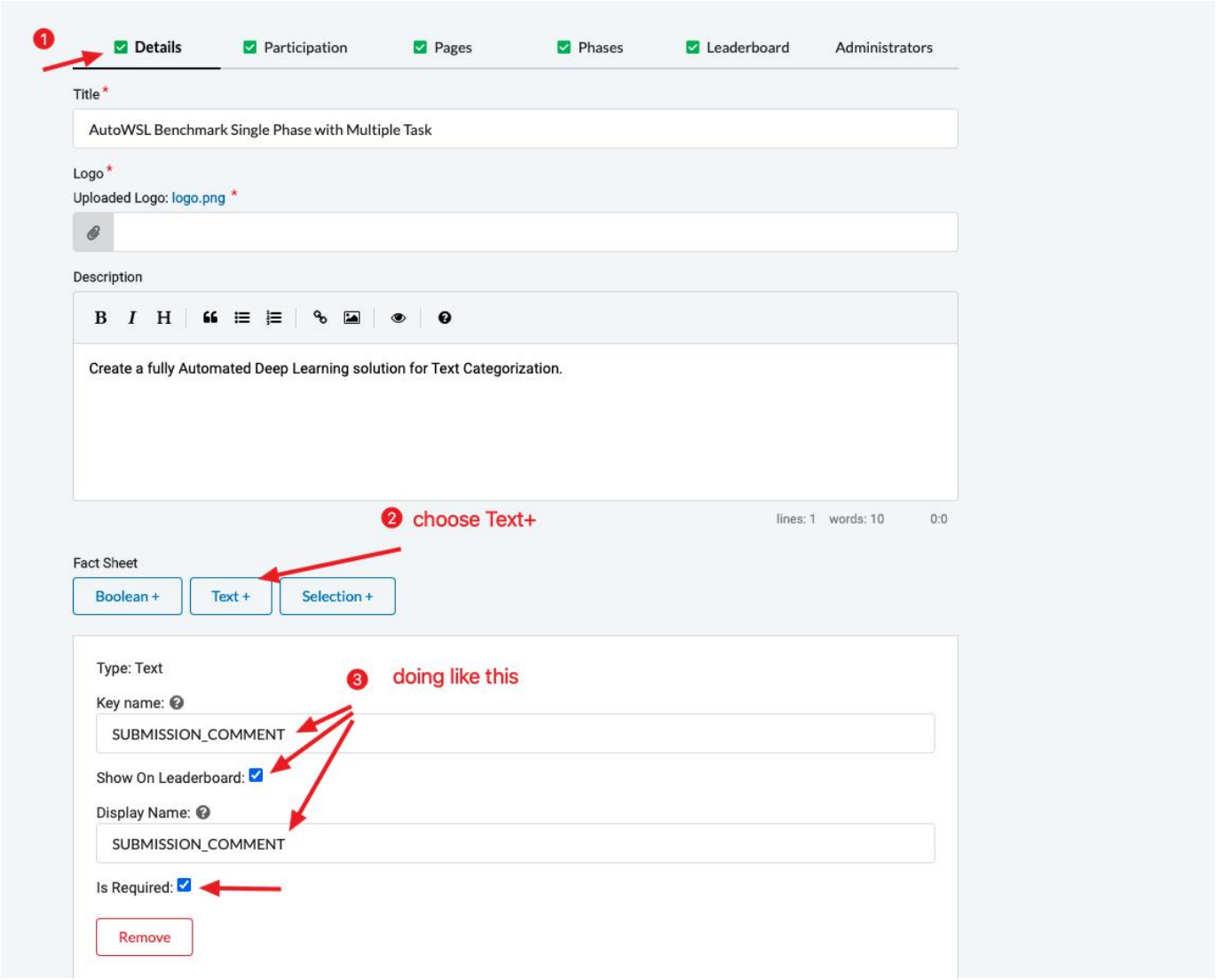


Step 2: Enable multiple submissions on leaderboard



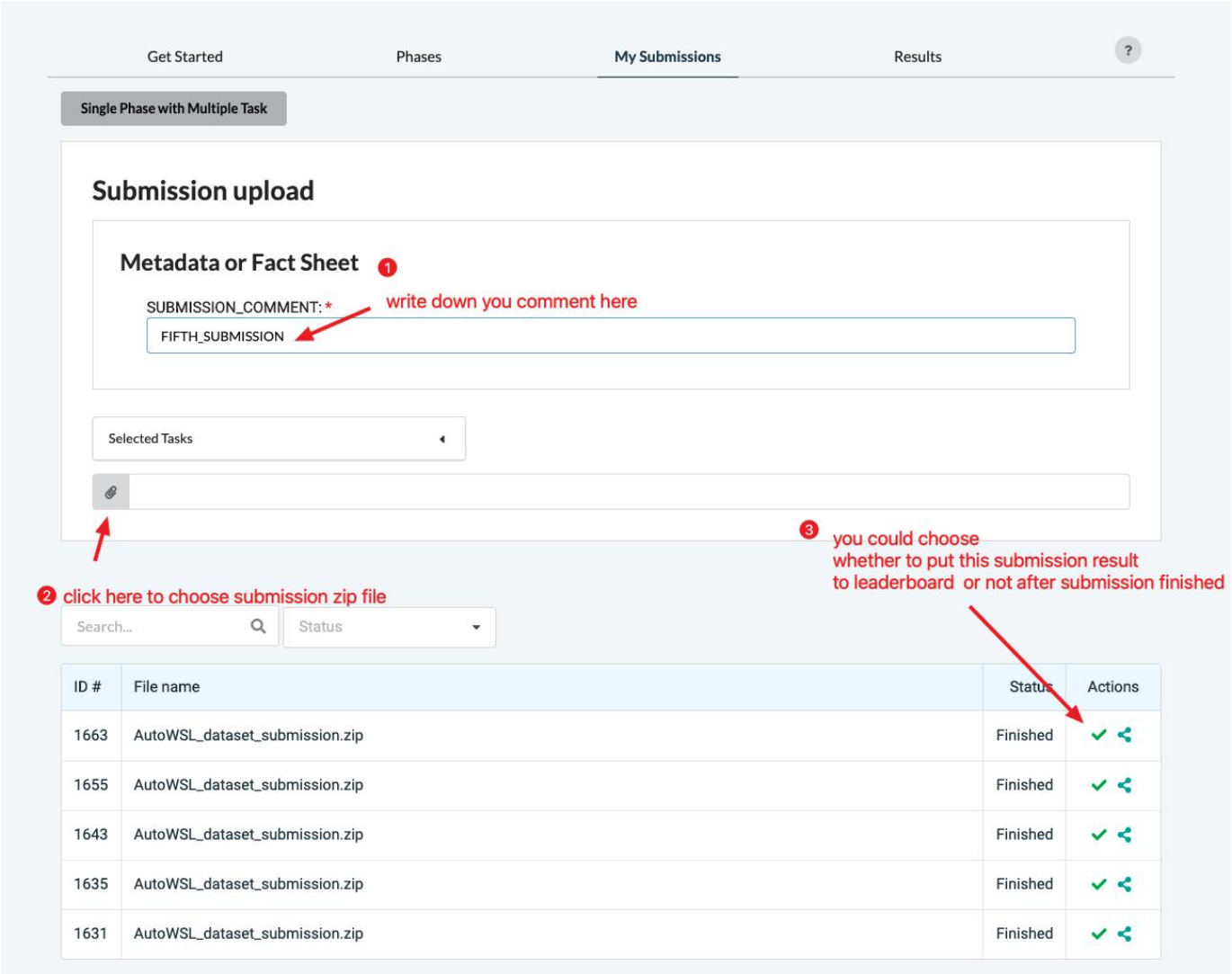

Leaderboard form
Leaderboard Settings
Title *
Results
Key *
main
Leaderboard is Hidden
Submission Rule
Add And Delete Multiple
Add
Add And Delete
Add And Delete Multiple
Force Last
Force Latest Multiple
Force Best
Duration
None
Descending
Column Key *
accuracy
duration
Hidden
+ Add column
Cancel
Save

Step 3: Set up submission comment

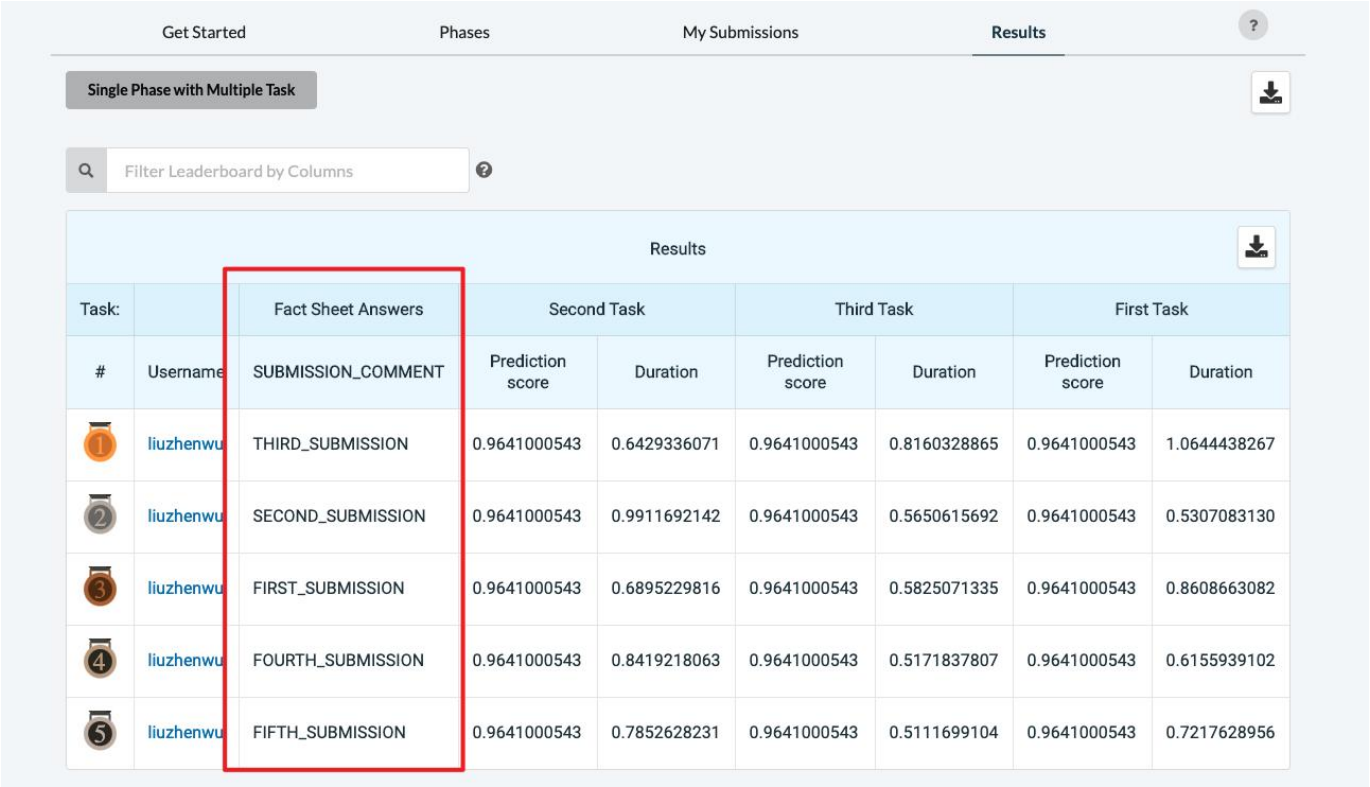


Step 4: Save all changes

Step 5: Leave a comment before making submission



Step 6: Check out the leaderboard

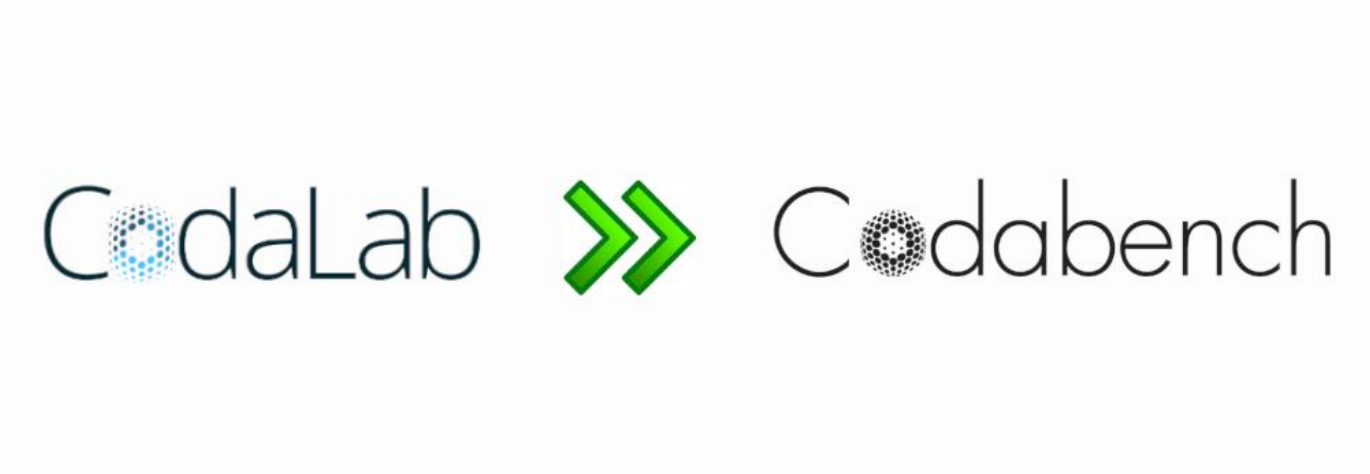


Results

Task:		Fact Sheet Answers	Second Task		Third Task		First Task	
#	Username	SUBMISSION_COMMENT	Prediction score	Duration	Prediction score	Duration	Prediction score	Duration
1	liuzhenwu	THIRD_SUBMISSION	0.9641000543	0.6429336071	0.9641000543	0.8160328865	0.9641000543	1.0644438267
2	liuzhenwu	SECOND_SUBMISSION	0.9641000543	0.9911692142	0.9641000543	0.5650615692	0.9641000543	0.5307083130
3	liuzhenwu	FIRST_SUBMISSION	0.9641000543	0.6895229816	0.9641000543	0.5825071335	0.9641000543	0.8608663082
4	liuzhenwu	FOURTH_SUBMISSION	0.9641000543	0.8419218063	0.9641000543	0.5171837807	0.9641000543	0.6155939102
5	liuzhenwu	FIFTH_SUBMISSION	0.9641000543	0.7852628231	0.9641000543	0.5111699104	0.9641000543	0.7217628956

3.1.3 How to Transition from Codalab to Codabench?

Codabench serves as a modern, faster, and more reliable upgrade to CodaLab, designed to supersede it. This quick guide is meant for CodaLab users wondering how to successfully adopt this new challenge platform.



What's new in Codabench?

Codabench includes all features from CodaLab Competitions, and proposes a faster and more intuitive interface. It also has new features such as:

- Live logs during submission processes
- Storage quotas
- Computation servers management for all users.

It also emphasizes on benchmarking, allowing dataset submissions and multiple leaderboard rows per user. Finally, future project development and maintenance will be focused on Codabench.

Do I need to create a new account?

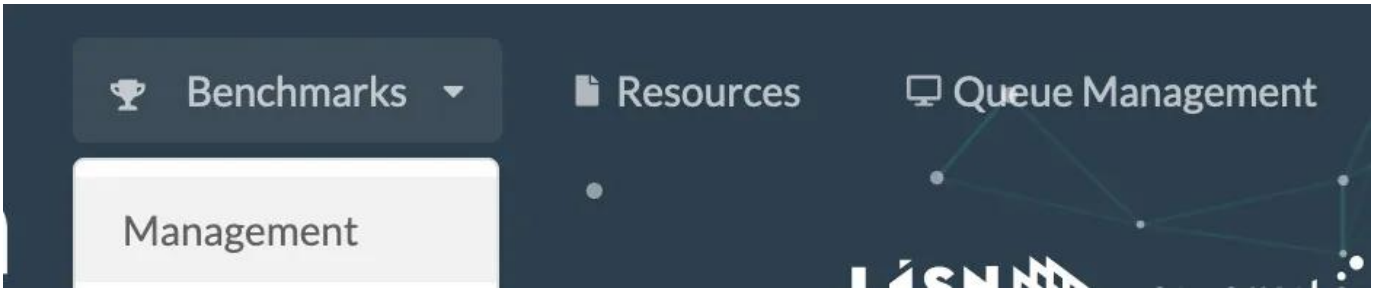
Yes, even if you previously had a CodaLab account, you need to create a new account on Codabench. Sign up is quick and free. From there, as a competition participant, you are all set.

The next questions concern competition organizers.

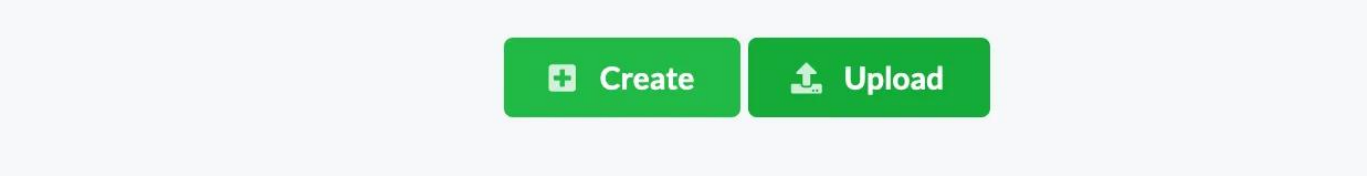
Can I upload my old competition bundles to Codabench?

Yes! That is the good news: competition bundles are back-compatible. This means that you can upload your CodaLab competition bundles into Codabench without any modifications and have them working just fine.

Simply go to "*Benchmarks > Management*", then click on "*Upload*" and select your competition bundle.



Go to "Benchmarks > Management".

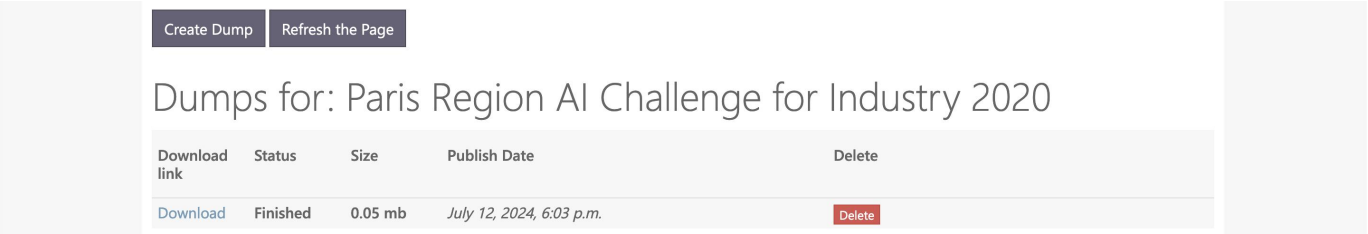


Click on "Upload" and select your competition bundle.

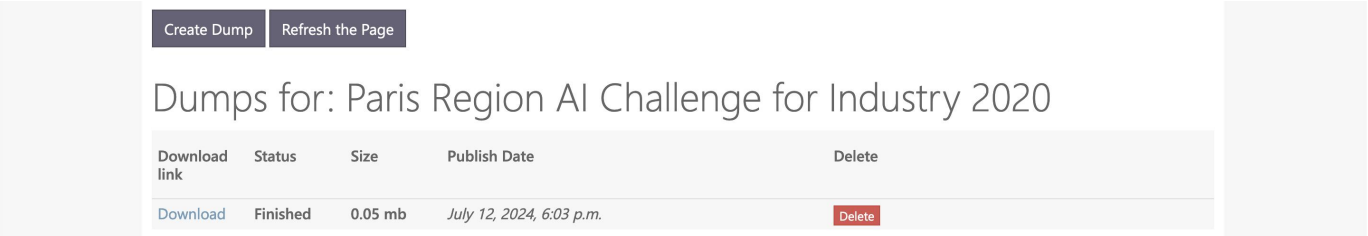
That's it! Your competition is ready to receive submissions.

How to move a competition from CodaLab to Codabench?

If you competitition is already live on CodaLab, that is fine too. You simply need to create a dump, download it and re-upload it on Codabench.



Go to "Dumps" organizer interface.



Click on "Create Dump" then "Download". You'll obtain a competition bundle that you'll be able to re-upload on Codabench, following the instructions of the previous section.

Leaderboard results won't be transferred. For that, you'll need to re-submit the submissions.

How to create a competition from scratch on Codabench?

If you don't have any previous competition, and want to learn how to create one from scratch, please refer to the Getting started guide.

Concluding remarks

Codabench, the new version of the competition and benchmark platform CodaLab, was launched on August 2023 and is already receiving great attention. For users accustomed to CodaLab, the transition to Codabench is quick and easy. Indeed, competition bundles are back-compatible, and all that is required is to create an account on Codabench. To go further, you can refer to Codabench's Wiki.

3.1.4 Competition Creation

Competition creation can be done two ways. Through the online form on Codalab, or by uploading a competition bundle to Codalab.

Bundle Upload

For more information on Bundle Upload see here:

Competition Creation: Bundle

For more information on Competition Bundle Structure, see here:

Competition Bundle Structure

GUI creation

For more information on GUI creation see here:

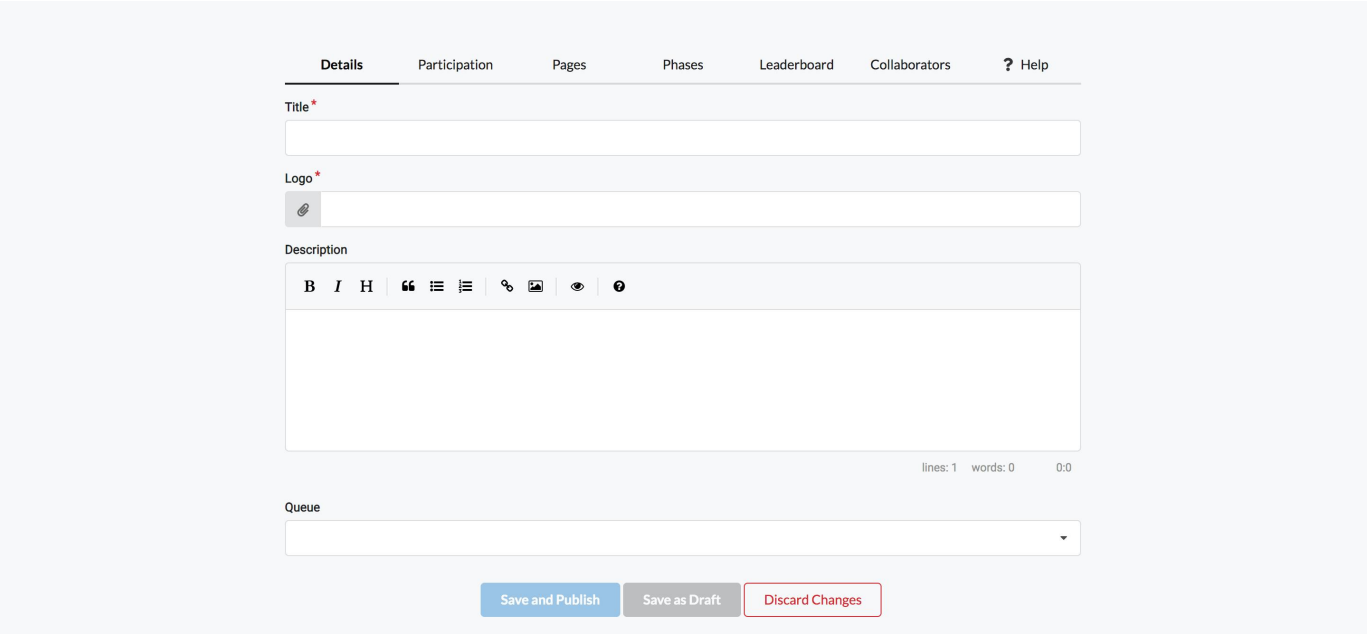
Competition Creation: Form

3.1.5 Competition Creation Form

Competitions can now be created through a wizard/form. This page will cover each different tab of the competition form correlating to a section, and the fields for each section.

Details

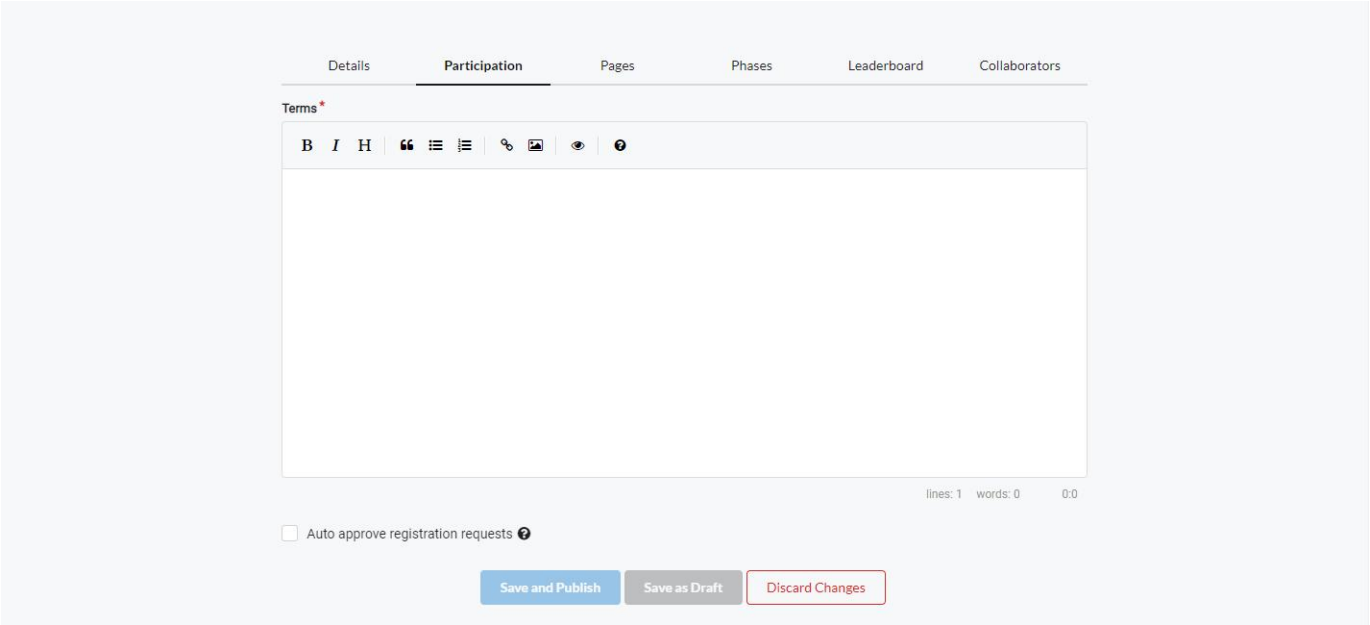
The details tab covers all basic competition info, such as title, logo, description, and the queue used.



- Title: The title of your competition.
- Logo: The logo corresponding to your competition
- Description: The description of your competition
- Queue: If you've previously created a queue through queue management, you can assign it to your competition here.

Participation

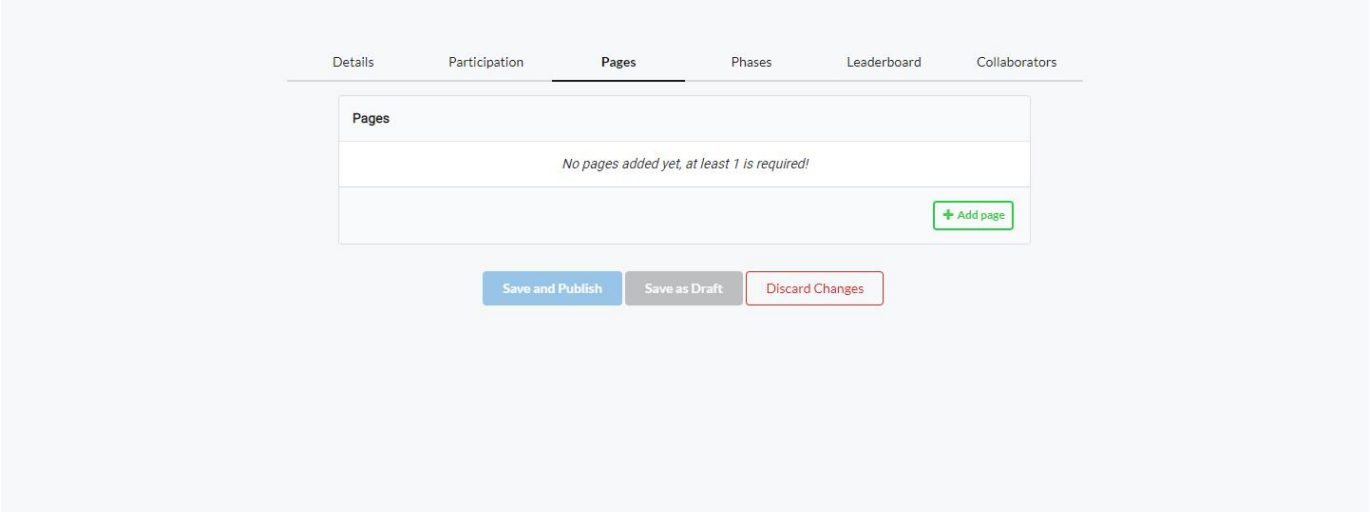
The participation tab covers your terms and conditions for the competition, and settings for auto-approval of participants.



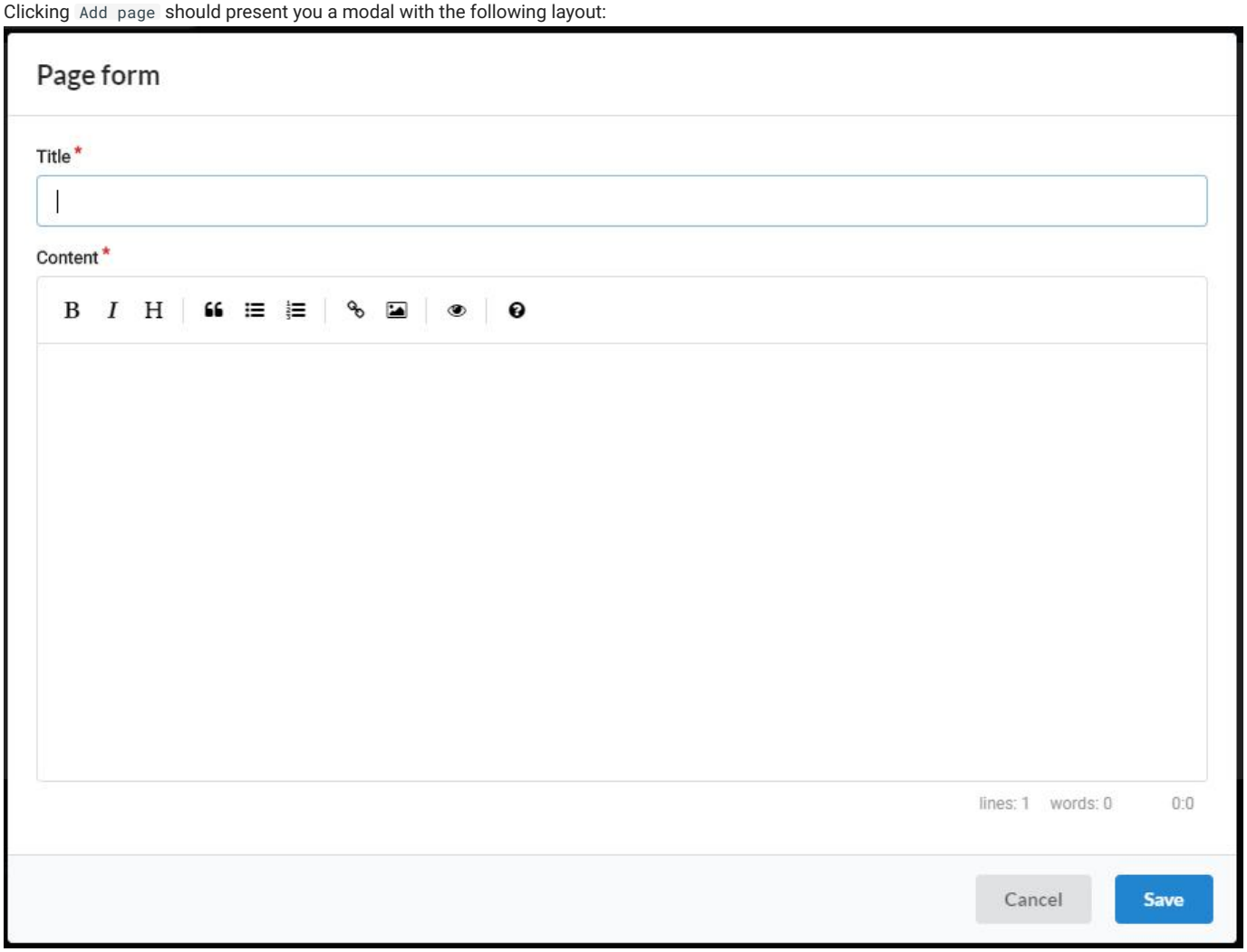
- Terms: Your terms and conditions for the competition.
- Auto Approve Registration: If checked, the organizer is not required to approve new participants.

Pages

In the pages section, you can add any additional content you would like to display to competition participants as `pages`. (Tabs on the competition detail page).



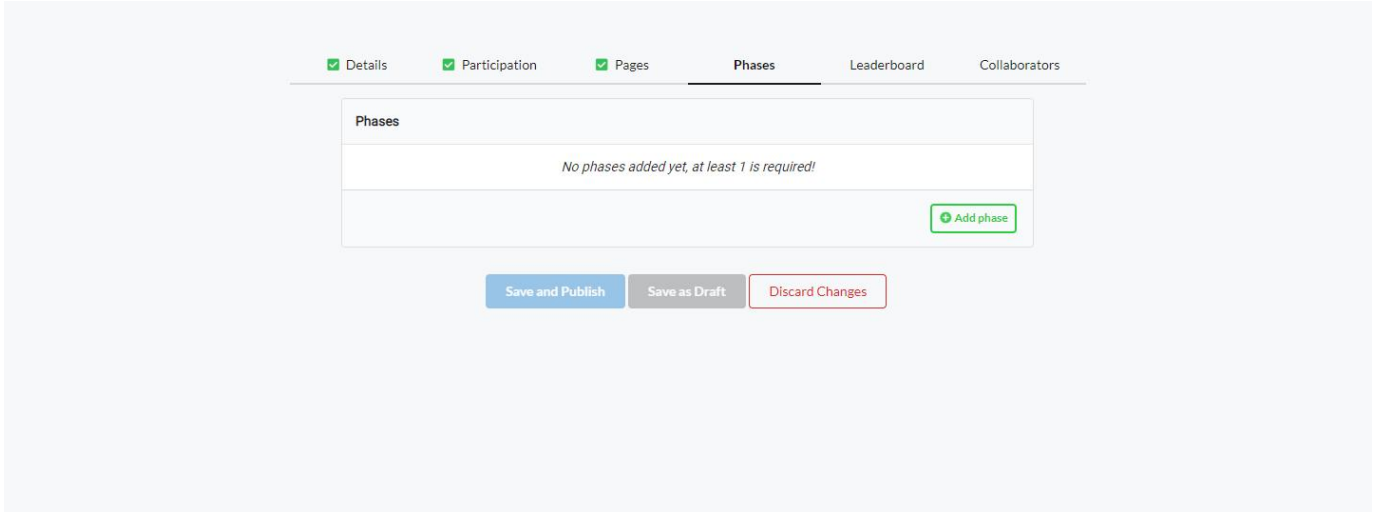
Clicking `Add page` should present you a modal with the following layout:



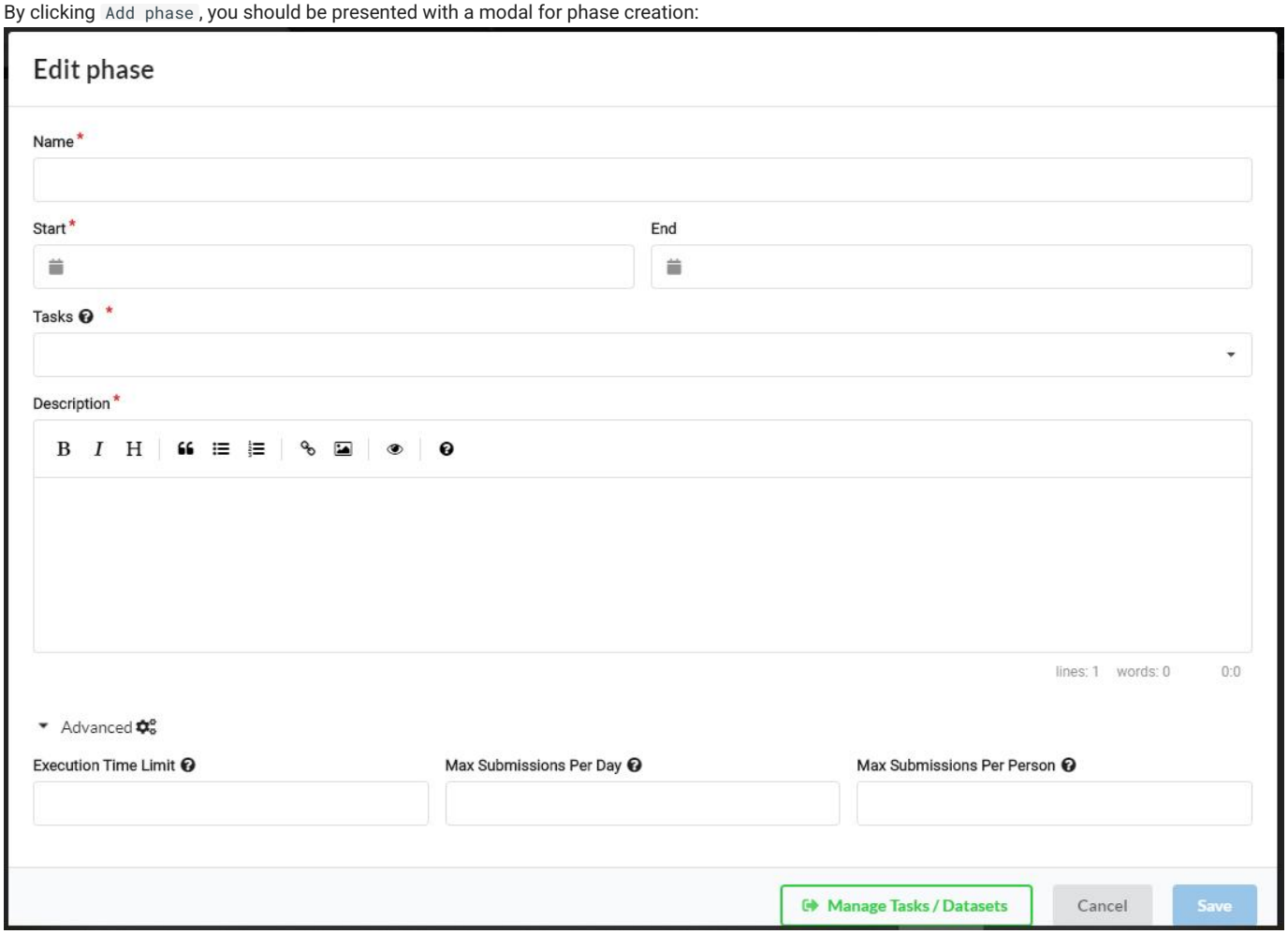
- Title: The title of the page you are adding
- Content: The content of your page formatted as Markdown.

Phases

The phases section allows you to define your phases and their attached tasks.



By clicking `Add phase`, you should be presented with a modal for phase creation:



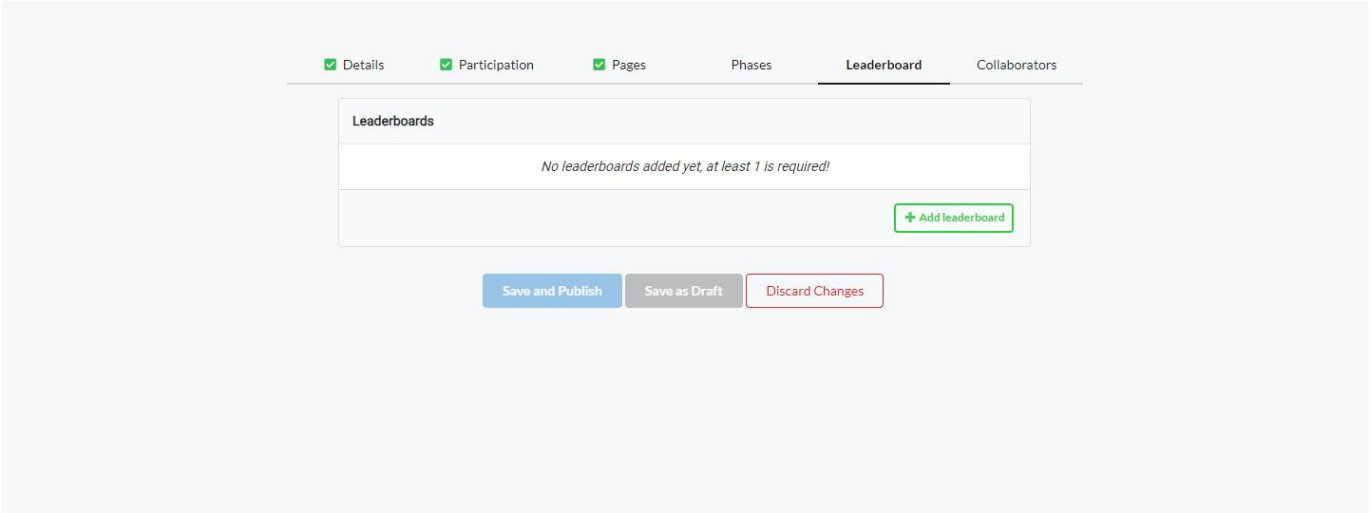
- Name: The name of your phase
- Start: The start day of your phase
- End: The end day of your phase
- Tasks: Here you can assign one or multiple task objects to your phase. Tasks are problems that the submission should be solving. For more information, see the explanation on competition structure here: If you don't have any tasks created yet, click the green button at the bottom of the new phase modal titled `Manage Tasks/Datasets`
- Description: The description of your phase

Advanced

- Execution Time Limit: The time limit for submission execution time measured in seconds (Currently the label says this is measured in MS, but this seems to be false)
- Max submissions per day: The max submissions allowed for the phase. The time period is from midnight UTC to midnight the next day UTC.
- Max submissions per person: The absolute max amount of submissions a participant can make on this phase.

Leaderboards

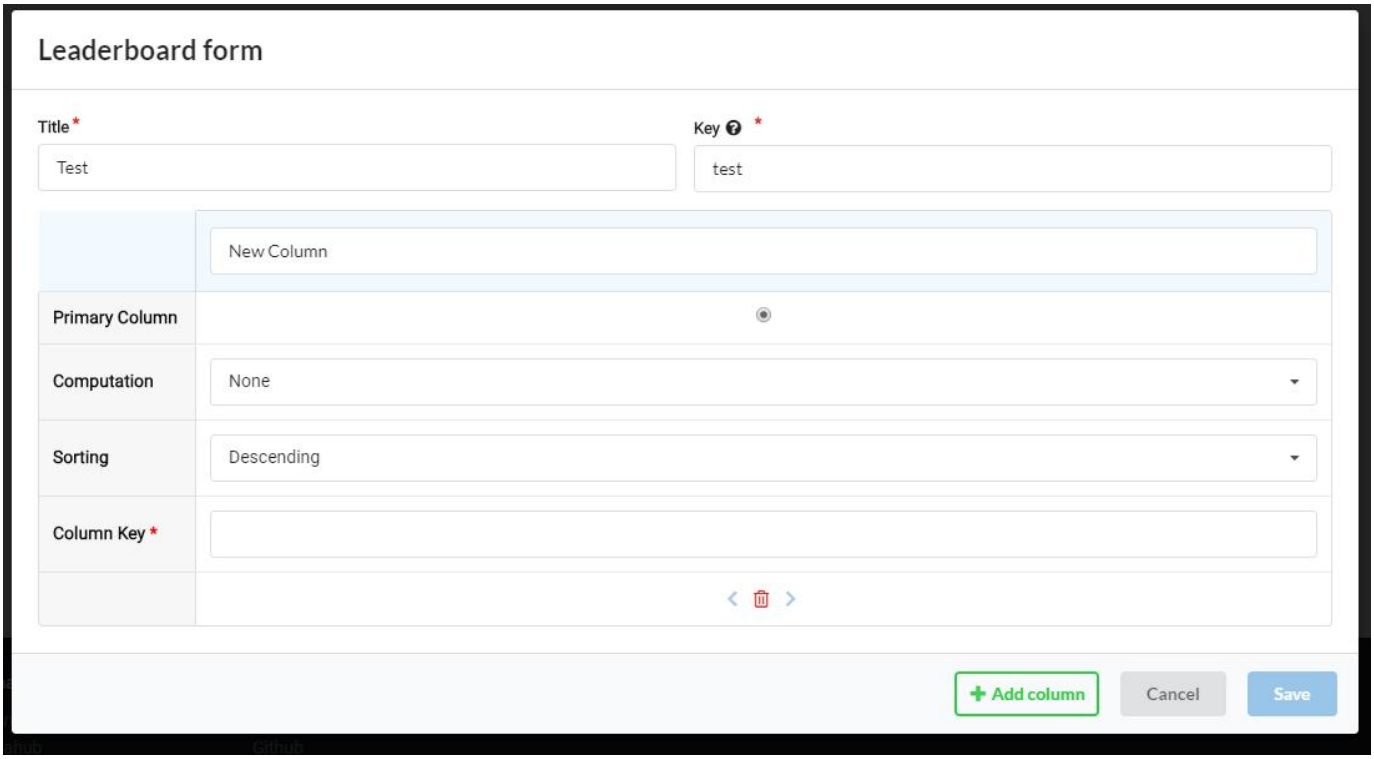
The leaderboards section allows you to define leaderboards which determine how submissions are scored.



Clicking add leaderboard should bring up this modal:

- Title: The title of your leaderboard
- Key: A unique name to refer to this leaderboard by (Preferably lowercase)

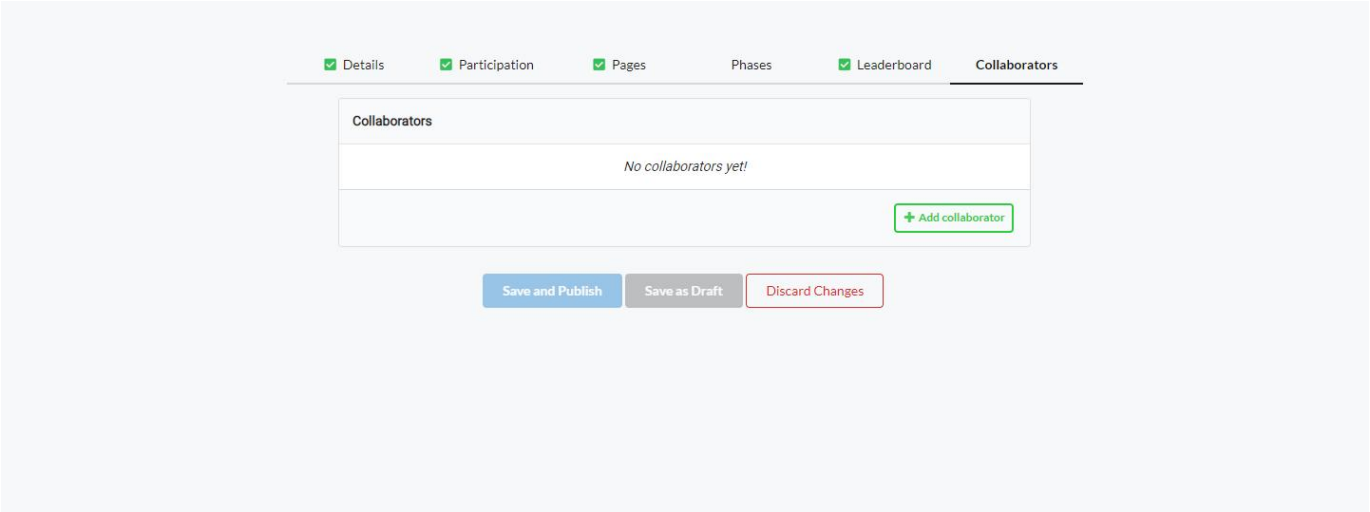
Adding a column will add some fields for the values of the column:



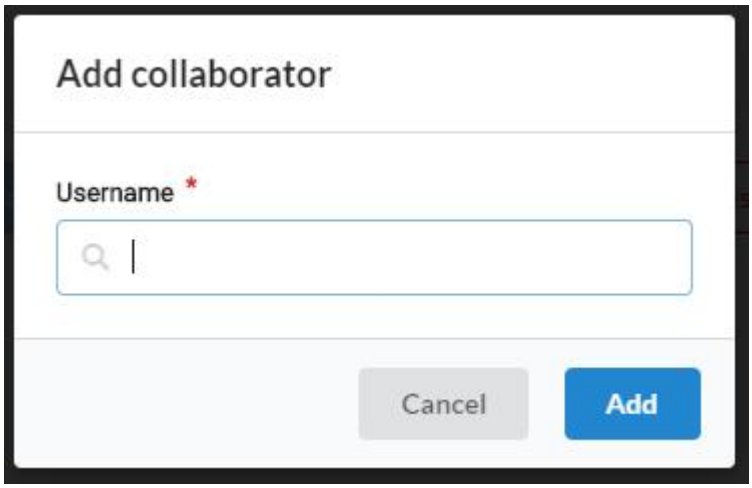
- Title (Unlabeled top input): The title of your column
- Primary Column: Whether this is the main column in the leaderboard (I.E: Is the sum/average of other columns)
- Computation (None/Average): If average is selected, this column should not have a score submitted to it, and will be a computation of the average of all the other columns.
- Sorting: Determine which way scores are sorted for this column.
- Column Key: A unique key to refer to this column by.

Collaborators

Here you can add other users to your competition as administrators.

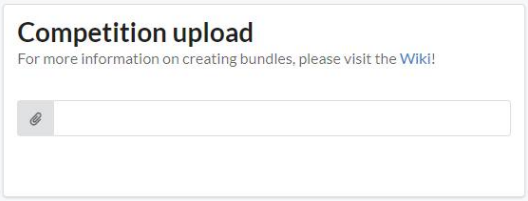


As per the other pages, clicking `Add collaborator` should bring up a modal. The search text can be the user's email if you know it, or their username.

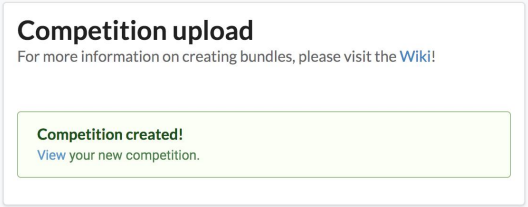


3.1.6 Competition Creation Bundle

This page is relatively simple. It's where you submit a completed competition bundle to Codabench, in order for it to be processed into a competition instance. For more information on competition bundles, see this link here: Competition Bundle Structure.



To begin, just click the paper clip icon, or the bar next to it. It should open a file select dialogue. From here, you select your competition bundle, and click upload. Once Codabench is done processing and unpacking your competition, you should be greeted with a success message and a link to your new competition.



Backward compatibility

If you previously used CodaLab Competitions, note that Codabench is compatible with CodaLab bundles.

3.1.7 Competition YAML Structure

A competition bundle is simply a zip file containing the `competition.yaml` which defines different aspects and attributes of your competition such as the logo, the html/markdown pages documenting your competition, and the data associated with your competition.

What is a Competition?

A competition is composed of a phase or many phases defining the active times of the competition, along with some other settings such as execution time limit. Each phase can have one or more tasks.

A task is the problem the submission should be solving, therefore submissions that solve a task can be thought of as a solution. A task consists of reference data, input data, scoring program, and an ingestion program.

For starting kits in v2, they should be solutions included with the competition bundle. See the example `competiton.yaml` or the section Competition YAML below for a link with more info. For more information on the different types of data, see the lower section of this page.

Example Competition Bundle Layout:

Some competition bundle examples!

> **Note**
>
> Files can be under a directory, they just have to be referenced by their full path in the YAML. See the Competition YAML section below.

```
--\ example_competition.zip
  |
  |- competition.yaml
  |- logo.png
  |- example_reference_data.zip
  |- example_scoring_program.zip
  |- example_solution.zip
  |- overview.md
  |- evaluation.md
  |- terms_and_conditions.md
  |- data.md
```

Example competition.yaml:

competition.yaml

```yaml
title: Example Competition Submit Scores
description: An example competition where submissions should output the score they want
image: logo.jpg
terms: terms.md
pages:
    - title: overview
      file: overview.md
    - title: evaluation
      file: evaluation.md
    - title: terms
      file: terms_and_conditions.md
    - title: data
      file: data.md
phases:
    - index: 0
      name: First phase
      description: An example phase
      start: 2018-03-01
      end: 2027-03-01
      tasks:
        - 0
tasks:
    - index: 0
      name: First Phase Task
      description: Task for the first phase
      scoring_program: example_scoring_program.zip
      reference_data: example_reference_data.zip
solutions:
    - index: 0
      path: example_solution.zip
      tasks:
        - 0
```

```
leaderboard:
  - title: Results
    key: main
    columns:
      - title: score
        key: score
        index: 0
        sorting: desc
```

Competition YAML

The `competition.yaml` file is the most important file in the bundle. It's what Codabench looks for to figure out the structure and layout of your competition, along with additional details. For more information on setting up a `competition.yaml` see the wiki page here: Competition YAML

Data Types And Their Role:

REFERENCE DATA:

Reference data is typically the truth data that your participant's predictions are compared against.

SCORING PROGRAM

The scoring program is the file that gets ran to determine the scores of the submission, typically either based on the submission's prediction outputs, or the results from the submission itself when compared with the reference data. Usually this should be a script like a python file, but it can generally be anything.

This is also paired with a `metadata.yaml` file with a key `command` that correlates to the command used to run your scoring program. There are also special directories available for use.

Example: Here's what a `metadata.yaml` might look like:

metadata.yaml

```
command: python3 /app/program/scoring.py /app/input/ /app/output/
```

This specifies that python3 is going to run the scoring program (which is going to be located in /app/program/scoring.py). It also gives, as `args`: - The input folder, which contains either the user's own results or predictions from ingestion - The output folder where the score is placed.

The arguments are optional, but passing them may be more convenient.

The scoring program outputs a `scores.json` file containing the results for each column of the leaderboard. After computing a submission, this file should look like something like this:

scores.json

```
{"accuracy": 0.886, "duration": 42.4}
```

The keys should match the leaderboard columns keys defined in the `competition.yaml` file.

The scoring program can also output detailed results as an HTML file for each submission. Click here for more information.

INGESTION PROGRAM

The ingestion program is a file that gets ran to generate the predictions from the submissions if necessary. This is usually a python script or a script in another language, but it can generally be anything.

The ingestion program is also paired with a `metadata.yaml` that specifies how to run it. It should have a key `command` that is the command used to run your ingestion program. The same special directories should be available to your ingestion program.

Example: Here's what an ingestion `metdata.yaml` might look like this:

metadata.yaml

```
command: python3 /app/program/ingestion.py /app/input_data/ /app/output/ /app/program /app/ingested_program
```

Just like the example above, this specifies we're using python to run our ingestion program. Please note that it is not necessary to pass these directories as arguments to the programs, but it can be convenient. More information about the folder layout here.

INPUT DATA

This is usually the test data used to generate predictions from a user's code submission when paired with an ingestion program.

3.1.8 YAML Structure

This page describes all the attributes in the Codabench competition definition language, using YAML. This is used to create configuration files in Codabench competition bundles.

Iris competition YAML file example!

Versioning

A version for the YAML is required as this platform can support multiple versions of a `competition.yaml` file. For examples of v1.5 bundles, look here.

For all v2 style competition bundles, be sure to add `version: 2` to the top of the competition.yml file.

Note: Not all features of v1.5 competitions are currently supported in v2.

Competition Properties

REQUIRED

- **title:** Title of the competition
- **image:** File path of competition logo, relative to `competition.yaml`
- **terms:** File path to a markdown or HTML page containing the terms of participation participants must agree to before joining a competition

OPTIONAL

- **description:** A brief description of the competition.
- **registration_auto_approve:** True/False. If True, participation requests will not require manual approval by competition administrators. Defaults to False
- **docker_image:** Can specify a specific docker image for the competition to use. Defaults to `codalab/codalab-legacy:py3`. More information here.
- **make_programs_available:** Can specify whether to share the ingestion and scoring program with participants or not. Always available to competition organizer.
- **make_input_data_available:** Can specify whether to share the input data with participants or not. Always available to competition organizer.
- **queue:** Queue submissions are sent to. Can be used to specify competition specific compute workers. Defaults to the standard queue shared by all competitions. The queue should be referenced by its **Vhost**, not by its name. You can find the Vhost in `Queue Management` by clicking the eye button `View Queue Detail`.
- **enable_detailed_results:** True/False. If True, competition will watch for a `detailed_results.html` file and send its contents to storage. More information here.
- **show_detailed_results_in_submission_panel:** a boolean (default: `True`) If set to `True`, participants can see detailed results in the submission panel
- **show_detailed_results_in_leaderboard:** a boolean (default: `True`) If set to `True`, participants can see detailed results in the leaderboard
- **contact_email:** a valid contact email to reach the organizers.
- **reward:** a string to show the reward of the competition e.g. "$1000" for competition.
- **auto_run_submissions:** a boolean (default: `True`) if set to `False`, organizers have to manually run the submissions of each participant
- **can_participants_make_submissions_public:** a boolean (default: `True`) if set to `False`, participants cannot make their submissions public from submissions panel.
- **forum_enabled:** a boolean (default: `True`) if set to `False`, organizers and participants cannot see or interact with competition forum.

```yaml
# Required
version: 2
title: Compute Pi
image: images/pi.png
terms: pages/terms.md

# Optional
description: Calculate pi to as many digits as possible, as quick as you can.
registration_auto_approve: True
```

```yaml
docker_image: codalab/codalab-legacy:py37 # default docker image
make_programs_available: True
make_input_data_available: False
enable_detailed_results: True
show_detailed_results_in_submission_panel: True
show_detailed_results_in_leaderboard: True
contact_email: organizer_email@example.com
reward: $1000 prize pool
auto_run_submissions: True
can_participants_make_submissions_public: False
forum_enabled: True
```

Pages

REQUIRED

- **title:** String that will be displayed in the competition detail page as the title of the page
- **file:** File path to a markdown or HTML page relative to competition.yaml containing the desired content of the page.

```yaml
pages:
  - title: Welcome
    file: welcome.md
  - title: Getting started
    file: pages/getting_started.html
```

Phases

REQUIRED

- **name:** Name of the phase
- **start:** Datetime string for the start of the competition. ISO format strings are recommended. Use `YYYY-MM-DD HH:MM:SS` date-time format. (Example date-time: 2024-12-31 14:30:00)
- **end:** Datetime string for the end of the phase (optional for *last phase only.* If not supplied for the final phase, that phase continues indefinitely). Use `YYYY-MM-DD HH:MM:SS` date-time format. (Example date-time: 2024-12-31 14:30:00)
- **tasks:** An array of numbers pointing to the index of any defined tasks relevant to this phase (see tasks for more information)

OPTIONAL

- **index:** Integer for noting the order of phases, Phases *must* be sequential, without any overlap. If indexes are not supplied, ordering will be assumed by declaration order.
- **max_submissions:** Total submissions allowed per participant for the entire phase
- **max_submissions_per_day:** Submission limit for each participant for a given day
- **auto_migrate_to_this_phase:** Cannot be set on the first phase of the competition. This will re-submit all successful submissions from the previous phase to this phase at the time the phase starts.
- **execution_time_limit:** Execution time limit for submissions, given in seconds. Default is 600.
- **hide_output:** True/False. If True, stdout/stderr for all submissions to this phase will be hidden from users who are not competition administrators or collaborators.
- **hide_prediction_output:** True/False. If True, participants won't be able to download the "Output from prediction step".
- **hide_score_output:** True/False. If True, participants won't be able to download the "Output from scoring step" containing the `scores.txt` file.
- **starting_kit:** path to the starting kit, a folder that participants will be able to download. Put there any useful files to help participants (example submissions, notebooks, documentation).
- **public_data:** path to public data, that participants will be able to download.
- **accepts_only_result_submissions**(default=False): When set to True, the phase is expected to accept only result submissions.

```yaml
phases:
  - index: 0
    name: Development Phase
    description: Tune your models
    start: 2019-12-12 13:30:00  # Time in UTC+0 and 24-hour format
    end: 2020-02-01 00:00:00  # Time in UTC+0 and 24-hour format
    execution_time_limit: 1200
    starting_kit: starting_kit
    public_data: public_data
    accepts_only_result_submissions: True
```

```
    tasks:
      - 0
  - index: 1
    name: Final Phase
    description: Final testing of your models
    start: 2020-02-02 00:00:00 # Time in UTC+0 and 24-hour format
    auto_migrate_to_this_phase: True
    accepts_only_result_submissions: False
    tasks:
      - 1
```

Tasks

REQUIRED

- **index:** Number used for internal reference of the task, pointed to by solutions (below) and phases (above)
- **name:** Name of the Task
- **scoring_program:** File path relative to `competition.yaml` pointing to a `.zip` file or an unzipped directory, containing the scoring program

OR

- **key:** UUID of a task already in the database. **If key is provided, all fields other than index will be ignored**

OPTIONAL

- **description:** Brief description of the task
- **input_data:** File path to the data to be provided during the prediction step
- **reference_data:** File path to the data to be provided to the scoring program
- **ingestion_program:** File path to the ingestion program files
- **ingestion_only_during_scoring:** True/False. If true, the ingestion program will be run in parallel with the scoring program, and can communicate w/ the scoring program via a shared directory

```
tasks:
  - index: 0
    name: Compute Pi Developement Task
    description: Compute Pi, focusing on accuracy
    input_data: dev_phase/input_data/
    reference_data: dev_phase/reference_data/
    ingestion_program: ingestion_program.zip
    scoring_program: scoring_program.zip
  - index: 1
    name: Compute Pi Final Task
    description: Compute Pi, speed and accuracy matter
    input_data: final_phase/input_data/
    reference_data: final_phase/reference_data/
    ingestion_program: ingestion_program.zip
    scoring_program: scoring_program.zip
```

Solutions

REQUIRED

- **index:** Index number of solution
- **tasks:** Array of the tasks (referenced internally) for which this solution applies.
- **path:** File path to `.zip` or directory containing the solution data.

```
solutions:
  - index: 0
    path: solutions/solution1.zip
    tasks:
    - 0
    - 1
  - index: 1
    path: solutions/solution2/
    tasks:
    - 0
```

Fact Sheet

OPTIONAL

JSON for asking metadata questions about each submission when they are submitted - **KEY:** Programmatic name for a response. Should not contain any whitespace. - **QUESTION TYPE:** - **"checkbox":** Prompts the user with a checkbox for a yes/no or true/false type question - ***Required*** **SELECTION**: [true, false] - **"text":**: Prompts the user with a text box to write a response. - ***Required*** **SELECTION**: "" - `"is_required": "false"` will allow the user not to submit a response. Otherwise, the user will have to type something. - **"select":** Gives the user a dropdown to select a value from. - **SELECTION**: Give an array of comma separated values that the user can select from: ["Value1","Value2","Value3",...,"ValueN"] - TIP: If you want this selection to be optional you can add "" as an option. ex. ["", "Value1", ...] and set `"is_required": "false"` - `is_on_leaderboard:` setting this to "true" will show this response on the leaderboard along with their submission.

STRUCTURE

```
fact_sheet: {
    "[KEY]": {
        "key": "[KEY]",
        "type": "[QUESTION TYPE]",
        "title": "[DISPLAY NAME]",
        "selection": [SELECTION],
        "is_required": ["true" OR "false"],
        "is_on_leaderboard": ["true" OR "false"]
}
```

```
fact_sheet: {
    "bool_question": {
        "key": "bool_question",
        "type": "checkbox",
        "title": "boolean",
        "selection": [True, False],
        "is_required": "false",
        "is_on_leaderboard": "false"
    },
    "text_question": {
        "key": "text_question",
        "type": "text",
        "title": "text",
        "selection": "",
        "is_required": "false",
        "is_on_leaderboard": "false"
    },
    "text_required": {
        "key": "text_required",
        "type": "text",
        "title": "text",
        "selection": "",
        "is_required": "true",
        "is_on_leaderboard": "false"
    },
    "selection": {
        "key": "selection",
        "type": "select",
        "title": "selection",
        "selection": ["", "v1", "v2", "v3"],
        "is_required": "false",
        "is_on_leaderboard": "true"
    }
}
```

Leaderboards

LEADERBOARD DETAILS

REQUIRED

- **title:** Title of leaderboard
- **key:** Key for scoring program to write to
- **columns:** An array of columns (see column layout below)

OPTIONAL

- **submission_rule:** "Add", "Add_And_Delete", "Add_And_Delete_Multiple", "Force_Last", "Force_Latest_Multiple" or "Force_Best". It sets the behavior of the leaderboard regarding new submissions. See Leaderboard Functionality for more details.
- **hidden:** True/False. If True, the contents of this leaderboard will be hidden to all users who are not competition administrators or collaborators.

Column Details

REQUIRED

- **title:** Title of the column
- **key:** Key for the scoring program to write to. The keys must match the keys of the `scores.json` file returned by the scoring program, as explained with more details here.
- **index:** Number specifying the order the column should show up on the leaderboard

OPTIONAL

- **sorting:** sorting order for the column: Descending (desc) or Ascending (asc)
- Ascending: smaller scores are better
- Descending: larger scores are better
- **computation:** computation to be applied *must be accompanied by computation indexes*
- computation options: sum, avg, min, max
- **computation_indexes:** an array of indexes of the columns the computation should be applied to
- **precision:** (*integer, default=2*) to round the score to *precision* number of digits
- **hidden:** (*boolean, default=False*) to hide/unhide a column on leaderboard

```yaml
leaderboards:
  - title: Results
    key: main
    submission_rule: "Force_Last"
    columns:
      - title: Accuracy Score 1
        key: accuracy_1
        index: 0
        sorting: desc
        precision: 2
        hidden: False
      - title: Accuracy Score 2
        key: accuracy_2
        index: 1
        sorting: desc
        precision: 3
        hidden: False
      - title: Max Accuracy
        key: max_accuracy
        index: 2
        sorting: desc
        computation: max
        precision: 3
        hidden: False
        computation_indexes:
          - 0
          - 1
      - title: Duration
        key: duration
        index: 3
        sorting: asc
        precision: 2
        hidden: False
```

3.1.9 Competition Docker Image

The competition docker image defines the docker environment in which the submissions of the competitions or benchmarks are run. Each competition can have a different docker environment, referred by its DockerHub name and tag.

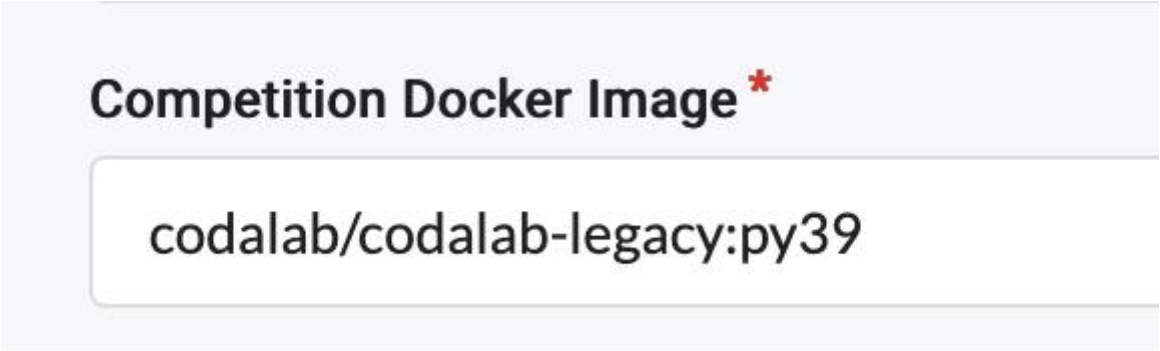
Default competition docker image

The default competition docker image is `codalab/codalab-legacy:py37`. More information here: https://github.com/codalab/codalab-dockers

Set up another image

You can select another docker image:

- In the `competition.yaml` file, using `docker_image: username/image:tag`
- In the editor field "Competition Docker image" as shown in the following screenshot:



Building an image

If the default image does not suit your needs (missing libraries, etc.), you can either:

- Select an existing image from DockerHub
- Create your own image from scratch
- Create a custom image based on the Codalab image. (more information below)

If you wish to create a custom image based on the Codalab image, you can follow the steps below:

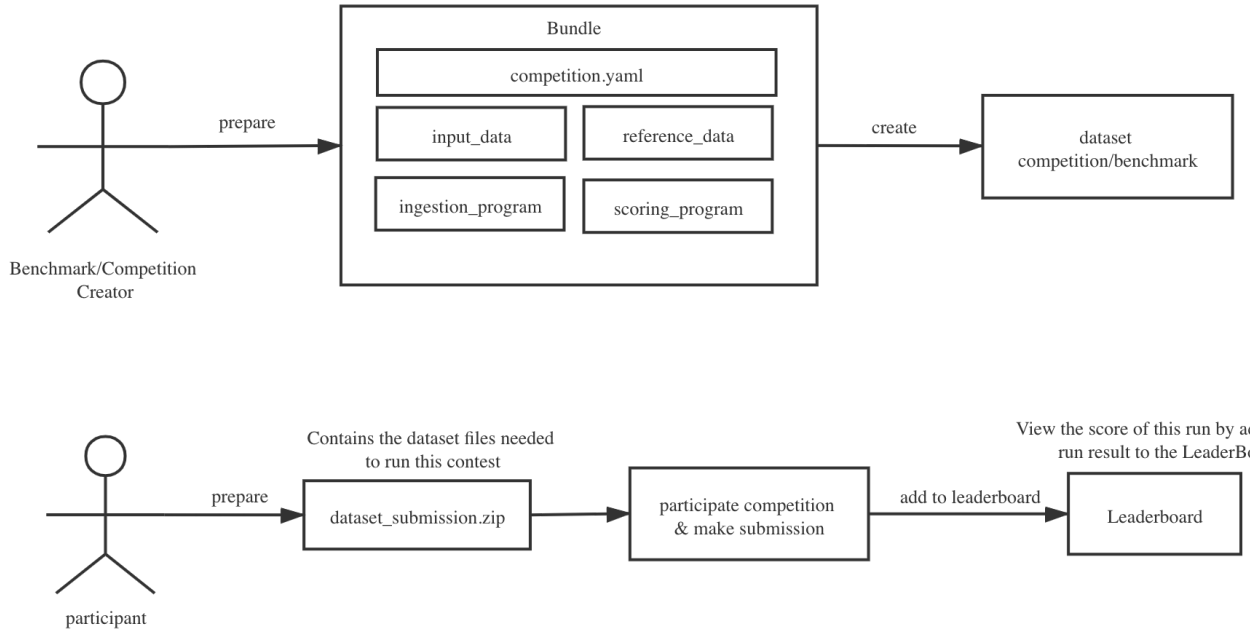
1) Install Docker
2) Sign up to DockerHub
3) `docker run -itd -u root codalab/codalab-legacy:py39 /bin/bash`
4) Use `docker ps` to find running container id
5) Now run `docker  exec -it -u root <CONTAINER ID> bash`
6) Install anything you want at the docker container shell (`apt-get install`, `pip install`, etc.)
7) Exit the shell with `exit`
8) `docker commit <CONTAINER ID> username/image:tag`
9) `docker login`
10) `docker push username/image:tag`

3.1.10 Dataset Competition Creation and participate instruction

This page focuses on how to create a dataset contest via bundle and make submission for dataset competition

Overall process

The brief process can be summarized in the following diagram



There are two main parts: - the contest organizer creates the dataset competition by uploading a bundle (For more information on how to create a contest via bundle, and the definition of bundle, you can refer to this link Competition-Creation:-Bundle)

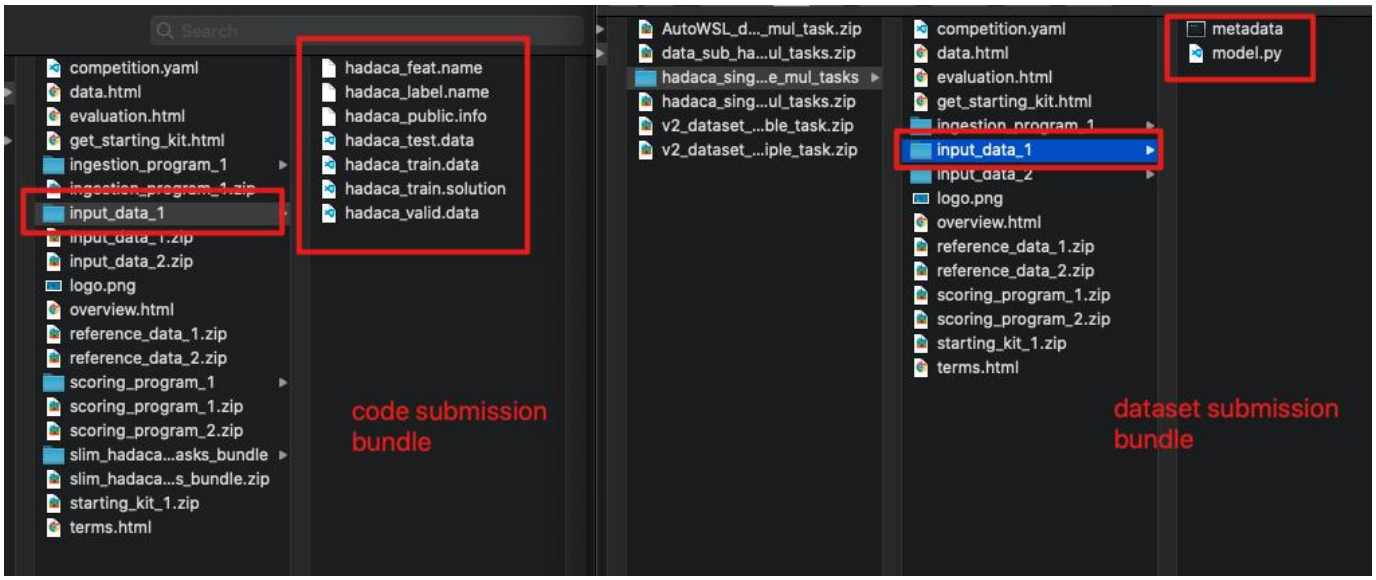
- Competition participant submission dataset

Differences from the code submission competition

FOR THE COMPETITION CREATOR

The main difference is the definition of the bundle, which differs from the code commit bundle in 2 ways

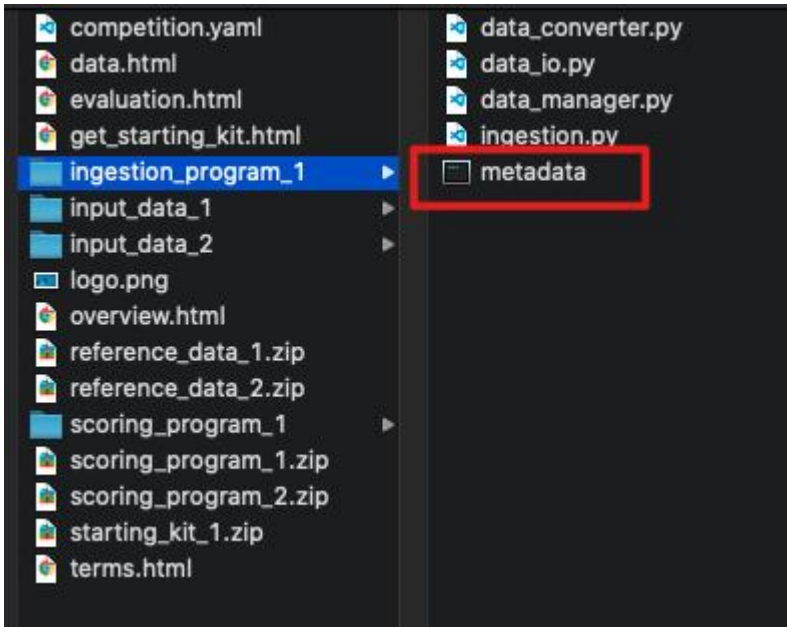
Input data

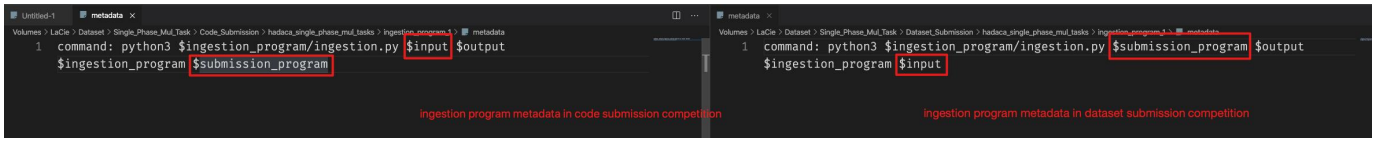


- In the code submission, input data folder is filled with the dataset files

  • In the dataset submission, input data folder is filled with the sample code submission files(NB: the sample code submission file is the algorithm file to be submitted by the participants in the code submission.)

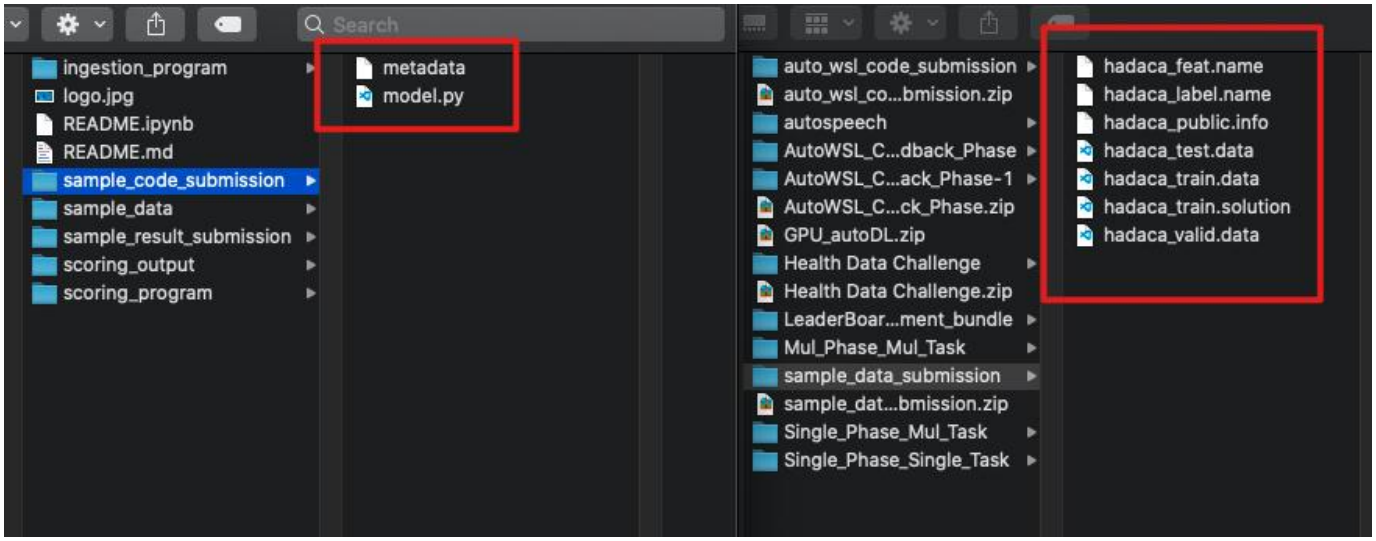
Ingestion program





- Unlike the code submission, we need to switch the position of the variables $input and $submission_program - In a dataset submission competition, the contents of $submission_program is the dataset submitted by the participant, and the contents of $input is the competition creator's built-in sample code submission.

For the competition participant



- The left-hand side of the image above shows the contents of the documents that competition participants need to prepare for the code submission competition

- the right-hand side of the image above shows the contents of the documents that competition participants need to prepare for the dataset submission competition
- The competition creator needs to define the bundle by specifying the content of the dataset file to be uploaded
- For example, the right-hand side of the picture shows the contents of the dataset file required for the HADACA competition to run.
- Therefore, when competition participants upload their dataset submission, the zip file must contain all the files shown on the right side of the picture above.

3.1.11 Leaderboard Features

For specific information on leaderboard and column fields, see the explanations in the YAML structure.

Writing scores

A leaderboard and column are written to via their keys. A leaderboard declaration like so

```yaml
leaderboard:
  - title: Results
    key: main
    submission_rule: "Force_Last"
    columns:
      - title: Accuracy Score 1
        key: accuracy_1
        index: 0
        sorting: desc
      - title: Accuracy Score 2
        key: accuracy_2
        index: 1
        sorting: desc
      - title: Max Accuracy
        key: max_accuracy
        index: 2
        sorting: desc
        computation: max
        computation_indexes:
          - 0
          - 1
      - title: Duration
        key: duration
        index: 3
        sorting: asc
```

would require, via the scoring program, the following `scores.json` file

scores.json

```
{"accuracy_1": 0.5, "accuracy_2": 0.75, "duration": 123.45}
```

This is the end result shown on the leaderboard:

```
| Accuracy Score 1 | Accuracy Score 2 | Max Accuracy | Duration |
|:----------------:|:----------------:|:------------:|:--------:|
|              0.5 |             0.75 |         0.75 |   123.45 |
```

Computation

Scores should not be written to computation columns, instead they will be calculated by the platform at the time scores are read from `scores.json`.

Computation options are: - sum - avg - min - max

These are applied across the columns specified as `computation_indexes`.

So in the example above, the computation option specified is `max` and the indexes are 0 and 1, meaning we will take the max score of columns at index 0 and 1 (i.e: .5 and .75) so .75 is returned in the computation.

Primary columns

Ranking is determined first by the primary column of the leaderboard. In the `competition.yaml`, this is the column at index 0. This option can be changed in the competition editor. After sorting scores by the primary column (asc or desc as specified on the column) sorting then continues from left to right. Final sorting is done by the `submitted_at` timestamp, so that if submissions have identical scores (as in the case of baselines), the earlier submissions will be ranked higher.

Example (with Max Accuracy set as the primary column):

```
| Rank | Accuracy Score 1 | Accuracy Score 2 | Max Accuracy | Duration |
|------|:----------------:|:----------------:|:------------:|:--------:|
|   1  |              0.5 |             0.75 |         0.75 |   123.45 |
|   2  |             0.43 |             0.75 |         0.75 |   123.45 |
```

```
|   3  |              0.6 |              0.6 |          0.6 |      100 |  # submitted at Jan 1, 2020
|   4  |              0.6 |              0.6 |          0.6 |      100 |  # submitted at Jan 2, 2020
```

So we sort the submissions by the primary column, (Max Accuracy) and then by columns from left to right, so accuracy 1, then accuracy 2, then duration, then by submission_at.

Submission rules

The submission rule set the behavior of the leaderboard regarding new submissions. Submissions can be forced to the leaderboard or manually selected, can be unique or multiple on the leaderboard, etc.

- **Add**: Only allow adding one submission
- **Add And Delete**: Allow users to add a single submission and remove that submission
- **Add And Delete Multiple**: Allow users to add multiple submissions and remove those submissions
- **Force Last**: Force only the last submission
- **Force Latest Multiple**: Force latest submission to be added to leaderboard (multiple)
- **Force Best**: Force only the best submission to the leaderboard

Here are the corresponding values for the YAML field `submission_rule`: "Add", "Add_And_Delete", "Add_And_Delete_Multiple", "Force_Last", "Force_Latest_Multiple" or "Force_Best".

Hidden Leaderboard

If a leaderboard is marked as hidden, it will not be visible to participants in the competition. It will only be visible to platform administrators, competition administrators, and competition collaborators.

Downloading Leaderboard Data

If an administrator, competition administrator, and competition collaborator would like to download the current leaderboard data, they will have access to a button labeled "CSV" on the leaderboard page. This creates a downloadable ZIP file. Each CSV file inside will be titled with the name of the leaderboard. The first row of the CSV is the title for each column, followed by all the submissions on the leaderboard. This can be access directly through the API by sending a GET request to `[HOSTNAME]/api/competitions/'ID'/get_csv` where 'ID' is the competition ID.

3.1.12 Example Cancer Benchmarks

This is our use case of cancer benchmarks. This document focuses on how to run the following three bundles in Codabench

- `CODABENCH CANCER HETEROGENEITY DT#1 TRANSCRIPTOME PANCREAS`
- `CODABENCH CANCER HETEROGENEITY DT#2 METHYLOME PANCREAS`
- `CODABENCH CANCER HETEROGENEITY DT#3 IMMUNE CELL TYPES`

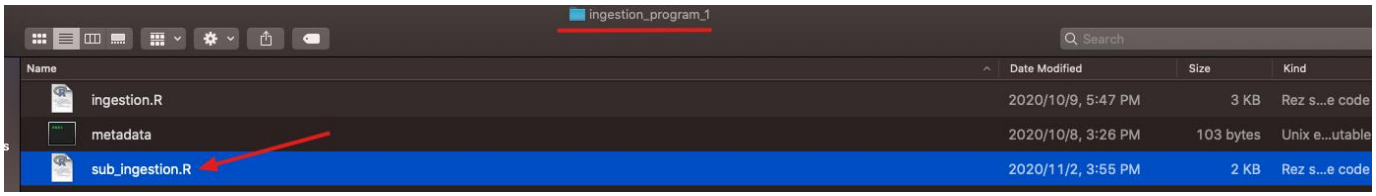
Steps

1. DECOMPRESSING THE ORIGINAL BUNDLE

Unzip the bundle from its original zip file format into a folder.

2. DECOMPRESSING INGESTION_PROGRAM_1.ZIP

3. MODIFY THE SUB_INGESTION.R FILE IN THE INGESTION_PROGRAM_1 FOLDER.



Add lines 19 and 20 of code, and replace the underlined variable in line 25 with submission_program_dir

Two new lines of code have been added to allow the v2 compute worker to find the user-submitted program (program.R). (Because the v2 compute worker does not support searching for user-submitted code in subfolders.)

```
child_dir <- list.files(path=submission_program)
submission_program_dir <- paste0(submission_program, .Platform$file.sep, tail(child_dir, n=1))
```

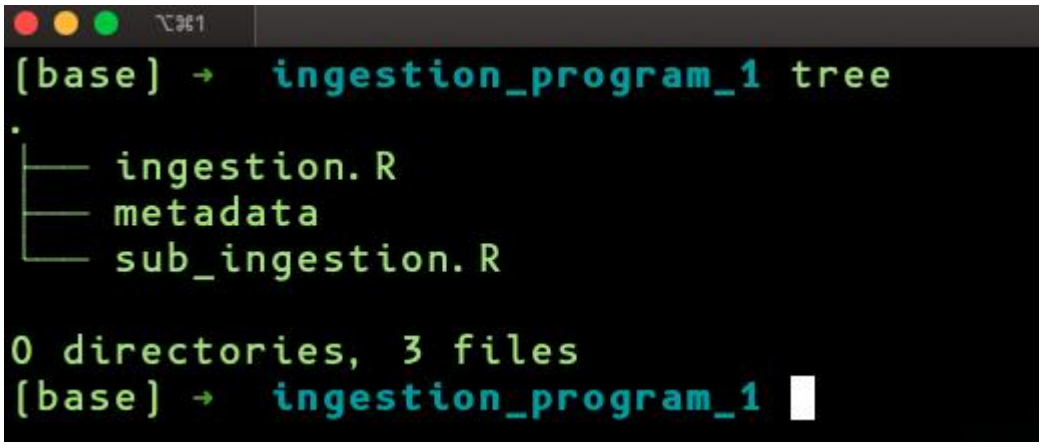
```
// read code submitted by the participants :
.tempEnv <- new.env( )
source(
    file  = paste0(submission_program_dir, .Platform$file.sep, "program.R")
  , local = .tempEnv
)
```

sub_ingestion.R

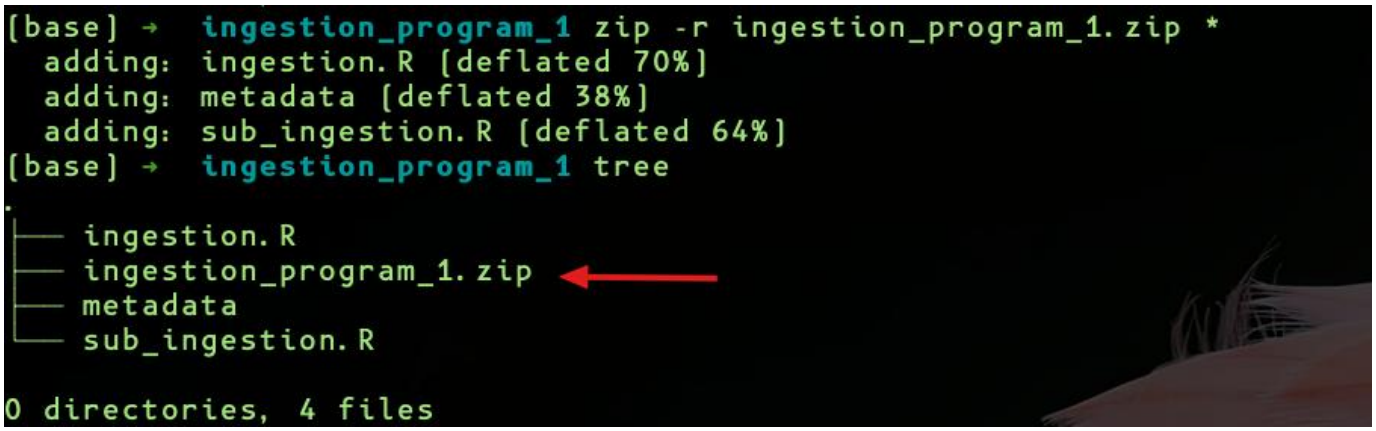
```r
args <- commandArgs(trailingOnly = TRUE)

## index of the input file :
i                 <- trimws(x = args[1] )
## input data directory :
input             <- trimws(x = args[2] )
## output file :
output_results    <- trimws(x = args[3] )
## output A :
output_A          <- trimws(x = args[4] )
## output T :
output_T          <- trimws(x = args[5] )
## directory of the code submitted by the participants :
submission_program <- trimws(x = args[6] )
## input k value
input_k_value     <- as.numeric(trimws(x = args[7] ))


child_dir <- list.files(path=submission_program)
submission_program_dir <- paste0(submission_program, .Platform$file.sep, tail(child_dir, n=1))

## read code submitted by the participants :
.tempEnv <- new.env( )
source(
    file  = paste0(submission_program_dir, .Platform$file.sep, "program.R")
  , local = .tempEnv
)

#source(
#  file  = "MT#0_starting_kit/program.R"
#  , local = .tempEnv
#)

## read input data :
D_matrix <- readRDS(file = paste0(input, .Platform$file.sep, "D_met", i, ".rds") )
#D_matrix <- readRDS(file = paste0(input, .Platform$file.sep, "D_rna", i, ".rds") )
# D_matrix = readRDS("public_data.RDS")


```

4.SAVE THE CHANGES AND RE-ZIP THE INGESTION_PROGRAM_1 FOLDER.

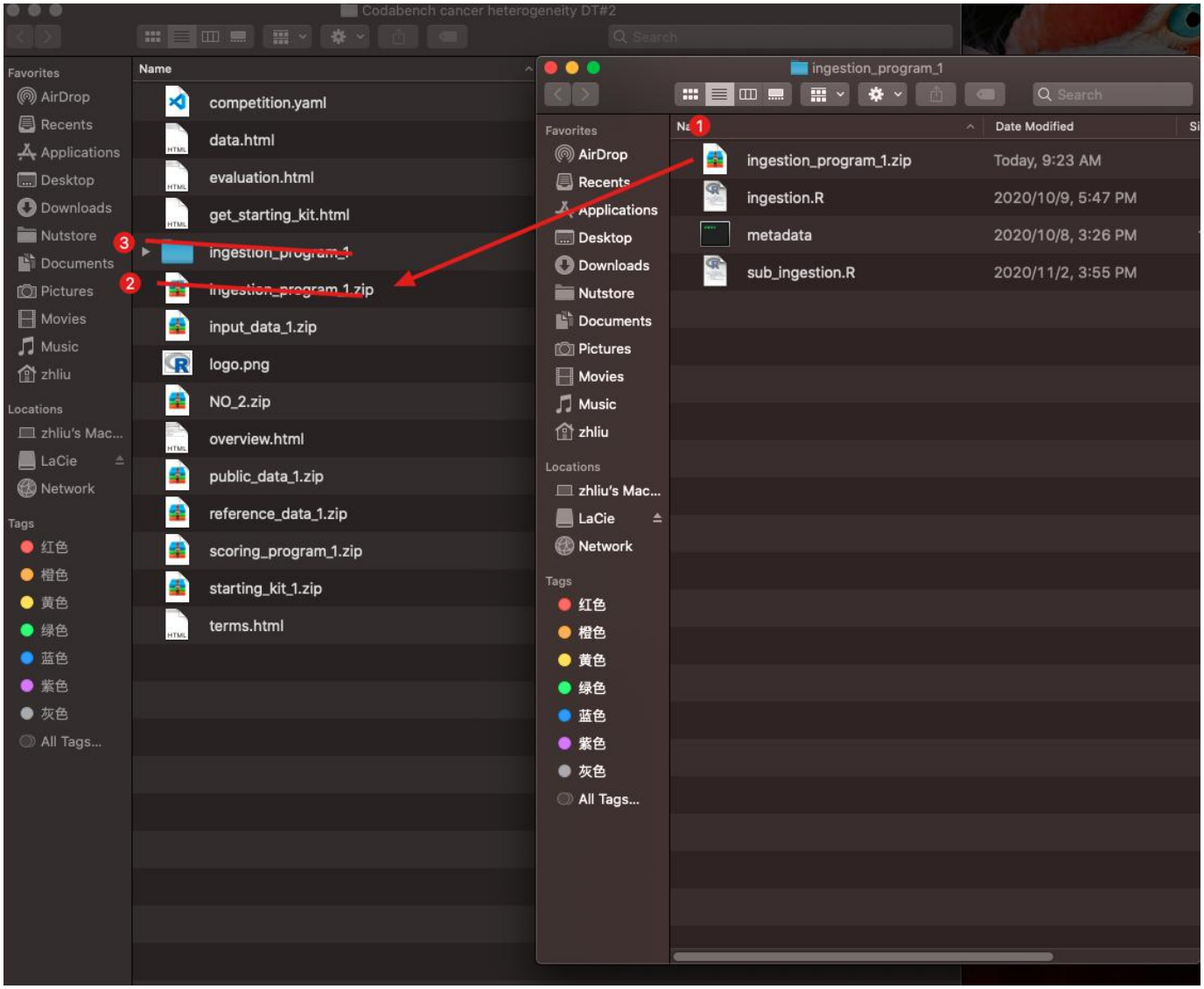
Open the command line and go to the ingestion_program_1 folder.



Use the following command to package the modified folder as a zip file `zip -r ingestion_program_1.zip *`

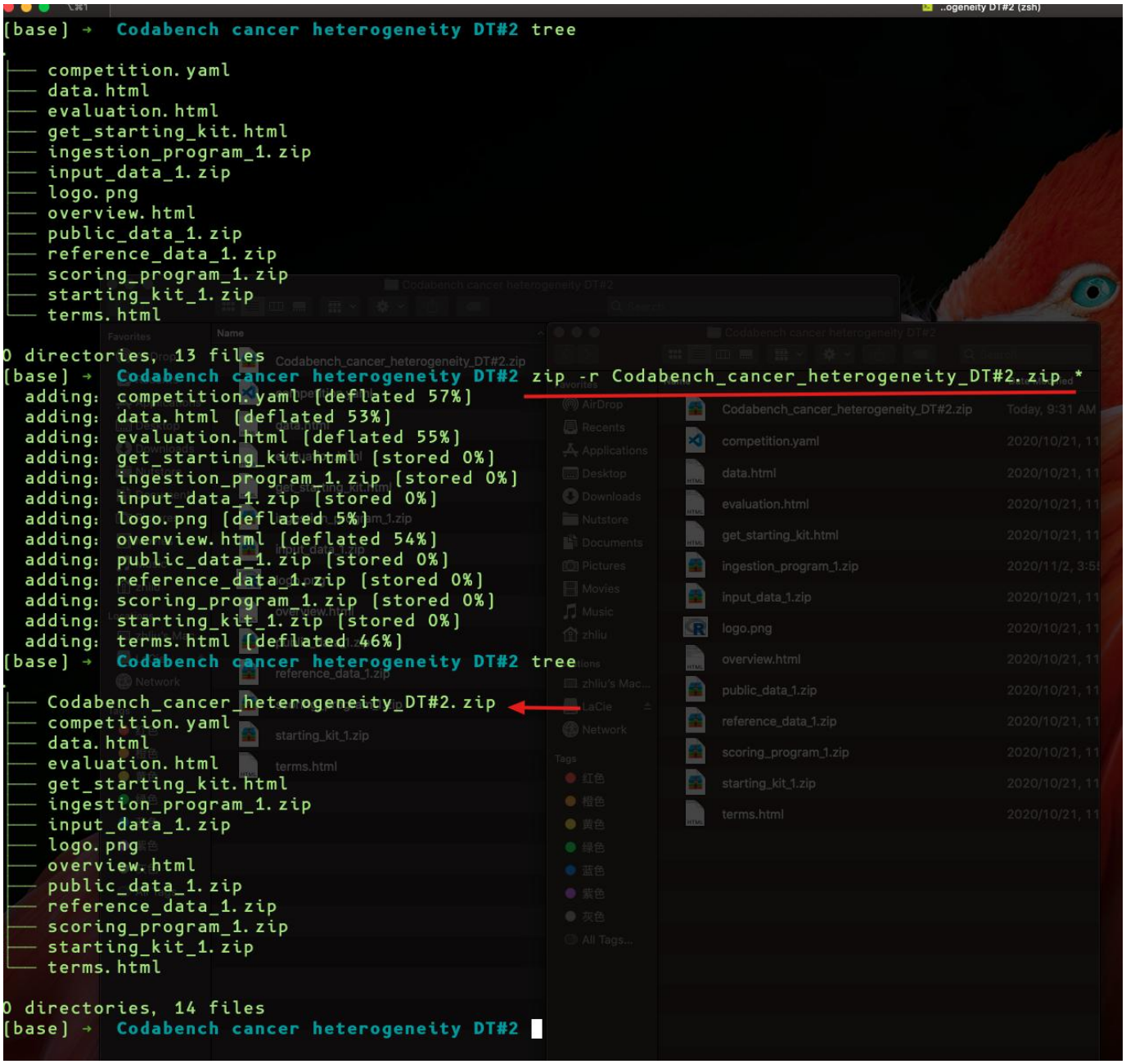


Replace the latest compressed ingestion_program_1.zip file with the previous ingestion_program_1.zip file, and delete the ingestion_program_1 folder.



5. RECOMPRESS THE MODIFIED ORIGINAL BUNDLE.

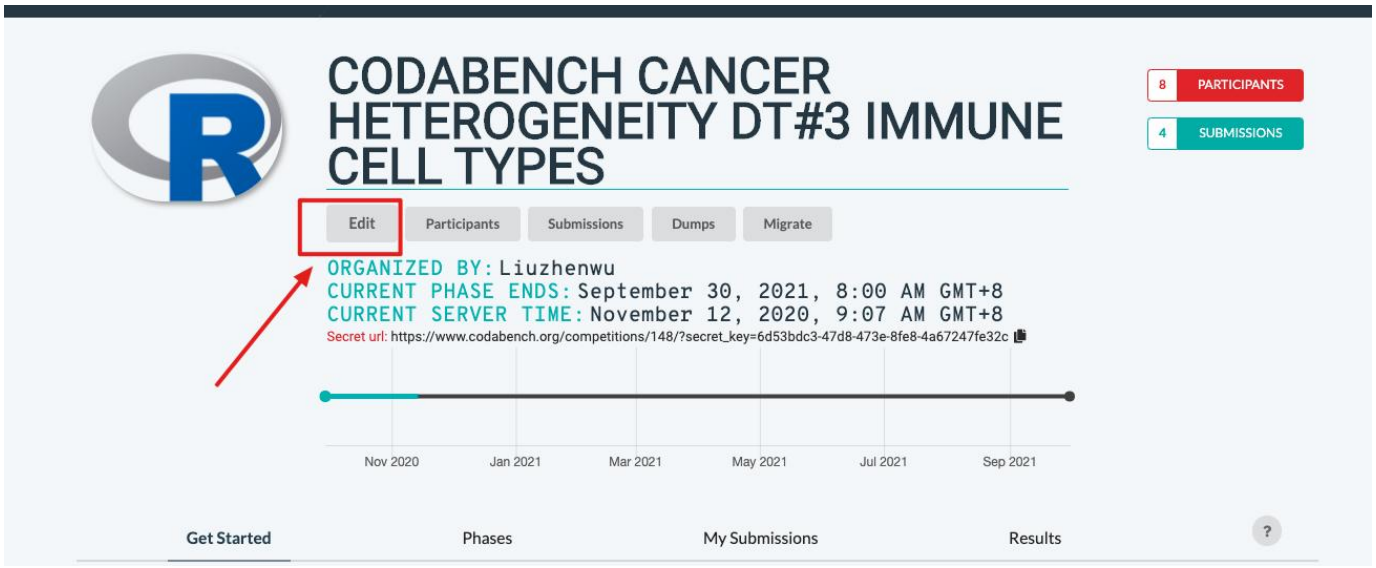
Go to the directory at the same level as `competition.yaml` and execute the following command to compress the file `zip -r Codabench_cancer_heterogeneity_DT#2.zip *`

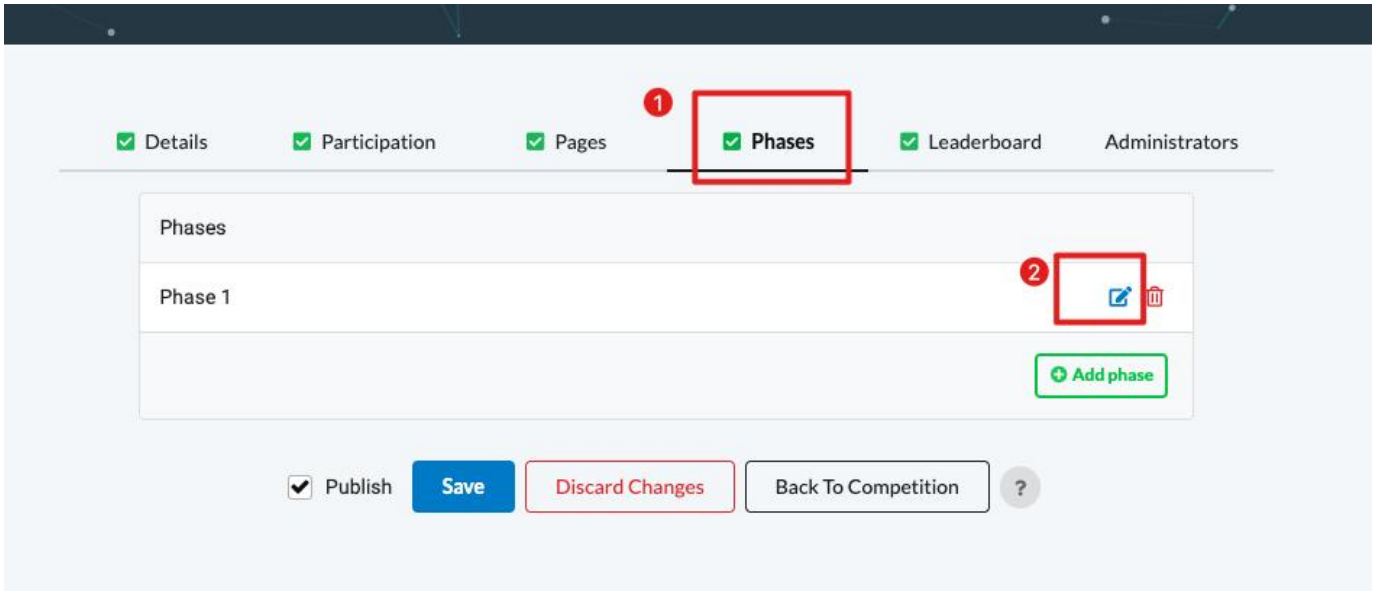


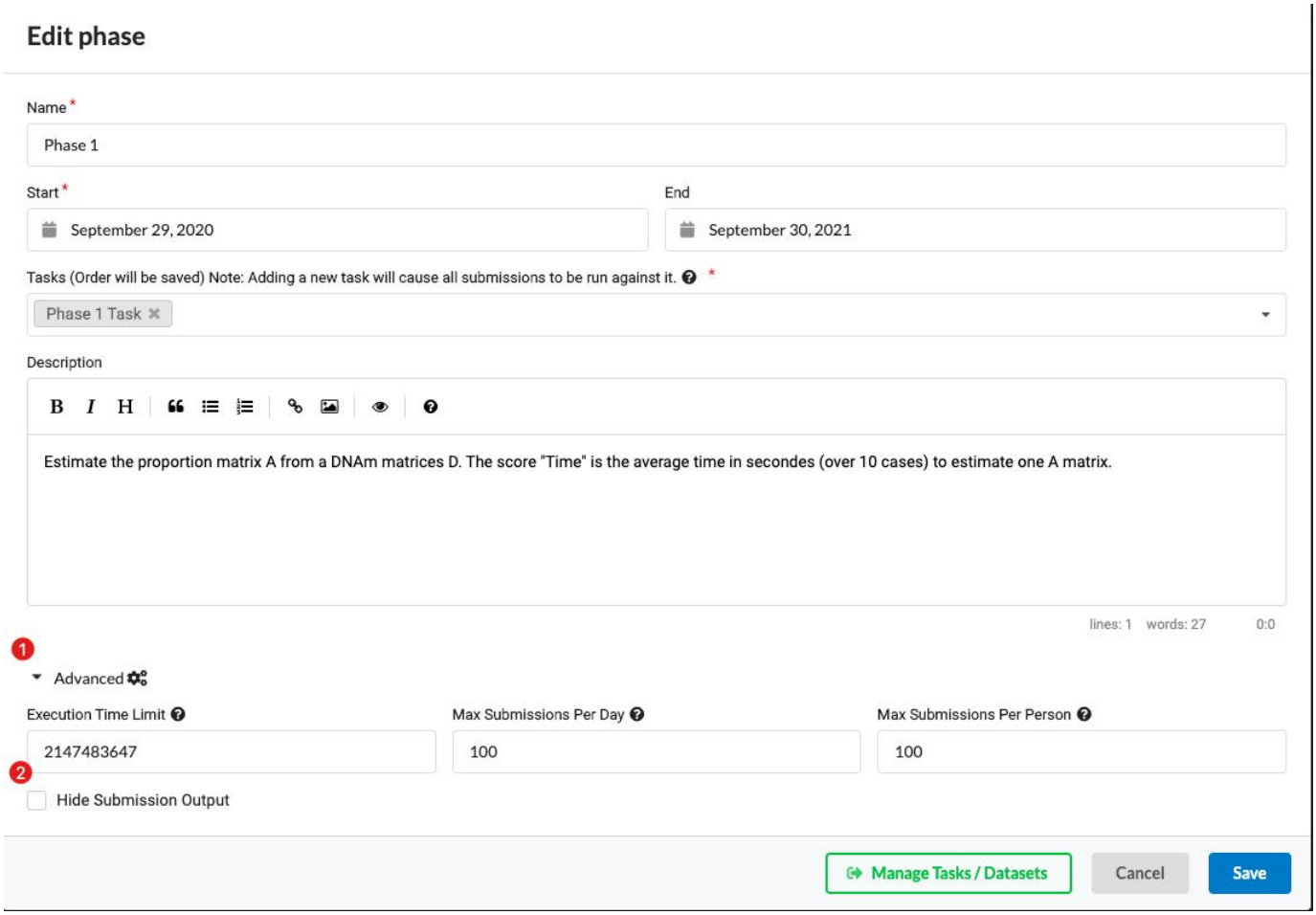
6. CREATING COMPETITION WITH COMPRESSED BUNDLES

7. MODIFY THE DEFAULT EXECUTION TIME

The default execution time is 10 minutes, but since these three bundles are time-consuming to execute, you have to turn it up.


CODABENCH CANCER HETEROGENEITY DT#3 IMMUNE CELL TYPES
8 PARTICIPANTS
4 SUBMISSIONS
Edit
Participants
Submissions
Dumps
Migrate
ORGANIZED BY: Liuzhenwu
CURRENT PHASE ENDS: September 30, 2021, 8:00 AM GMT+8
CURRENT SERVER TIME: November 12, 2020, 9:07 AM GMT+8
Secret url: https://www.codabench.org/competitions/148/?secret_key=6d53bdc3-47d8-473e-8fe8-4a67247fe32c
Nov 2020
Jan 2021
Mar 2021
May 2021
Jul 2021
Sep 2021
Get Started
Phases
My Submissions
Results


1
Details
Participation
Pages
Phases
Leaderboard
Administrators
Phases
Phase 1
2
Add phase
Publish
Save
Discard Changes
Back To Competition



We recommend that you adjust the time to the maximum value of `2147483647`, so that the task will not time out and be forced to terminate by the compute worker.

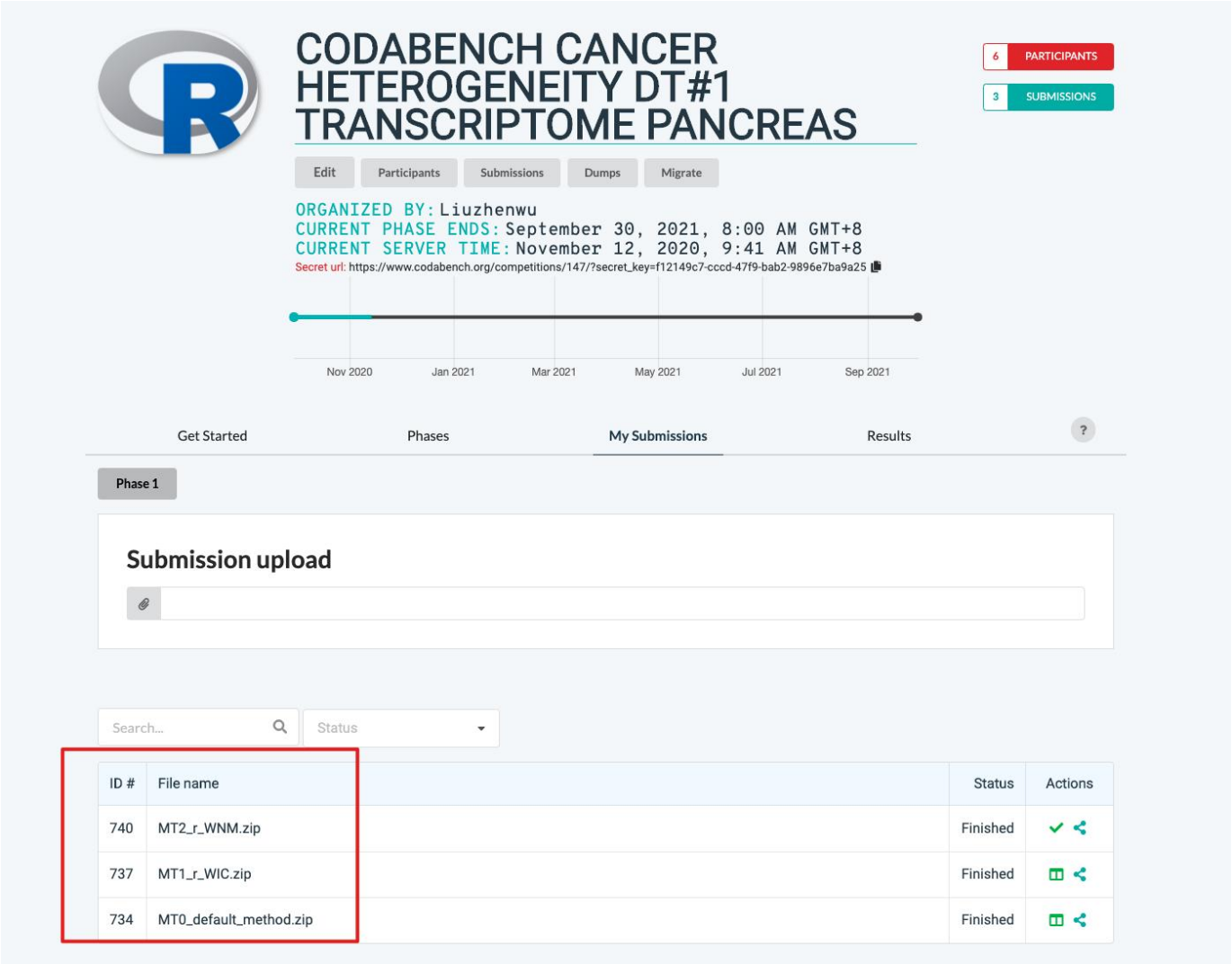
Summary

This paragraph summarizes the results of the execution of three bundles in codalab v2.

CODABENCH CANCER HETEROGENEITY DT#1 TRANSCRIPTOME PANCREAS

https://www.codabench.org/competitions/147/

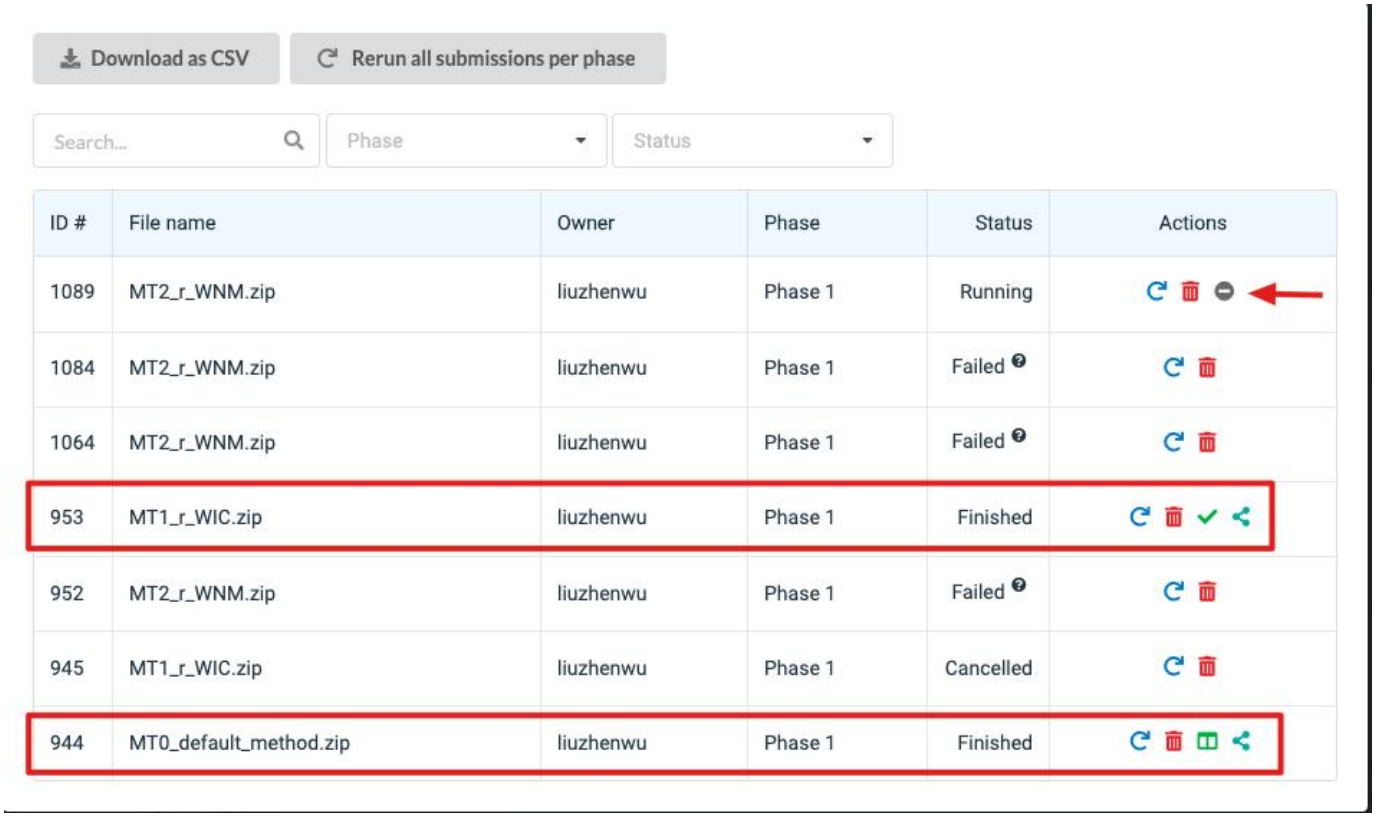
All three submissions were successful.



CODABENCH CANCER HETEROGENEITY DT#2 METHYLOME PANCREAS

https://www.codabench.org/competitions/174/

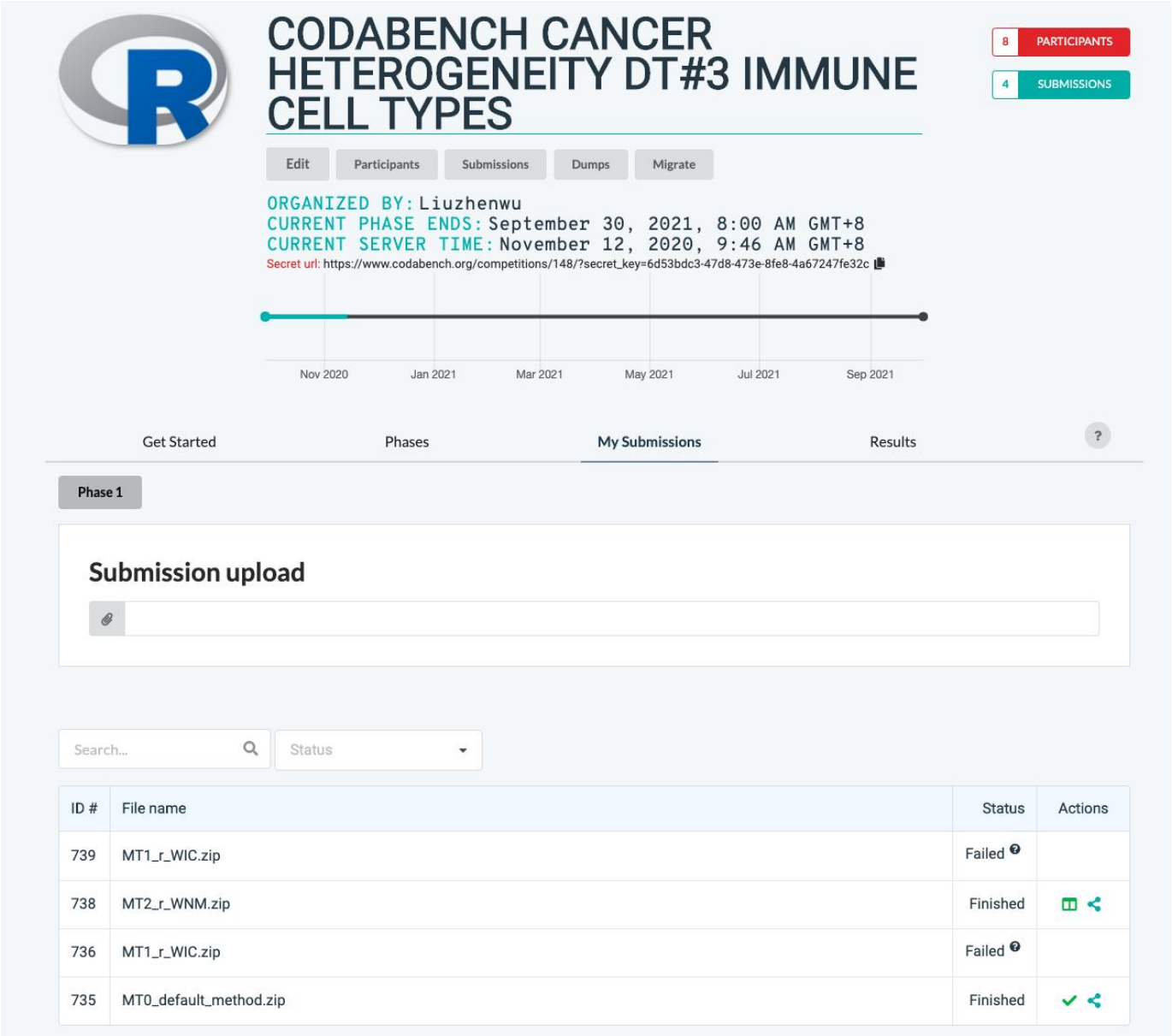
Two Submissions were successfully run, while the third failed due to insufficient execution time (We have now adjusted from the original 10,000 minute execution time limit to a maximum of 2,147483647.)



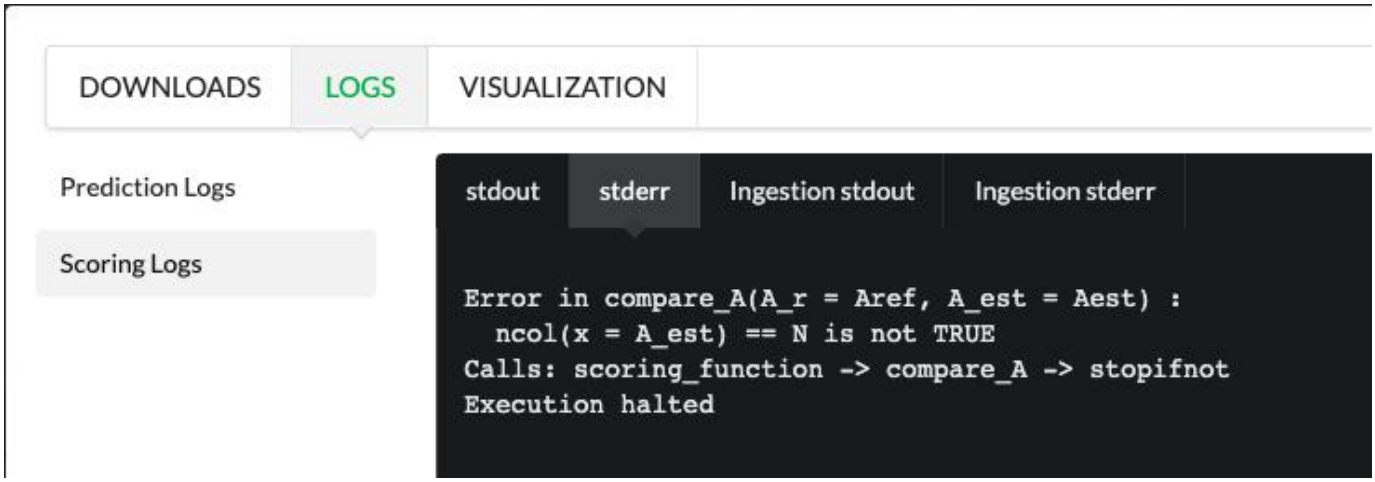
CODABENCH CANCER HETEROGENEITY DT#3 IMMUNE CELL TYPES

https://www.codabench.org/competitions/148/

2 Submissions run successfully, 1 execution fails (screenshot below)



Failed execution screenshot:



3.1.13 Public Tasks and Tasks Sharing

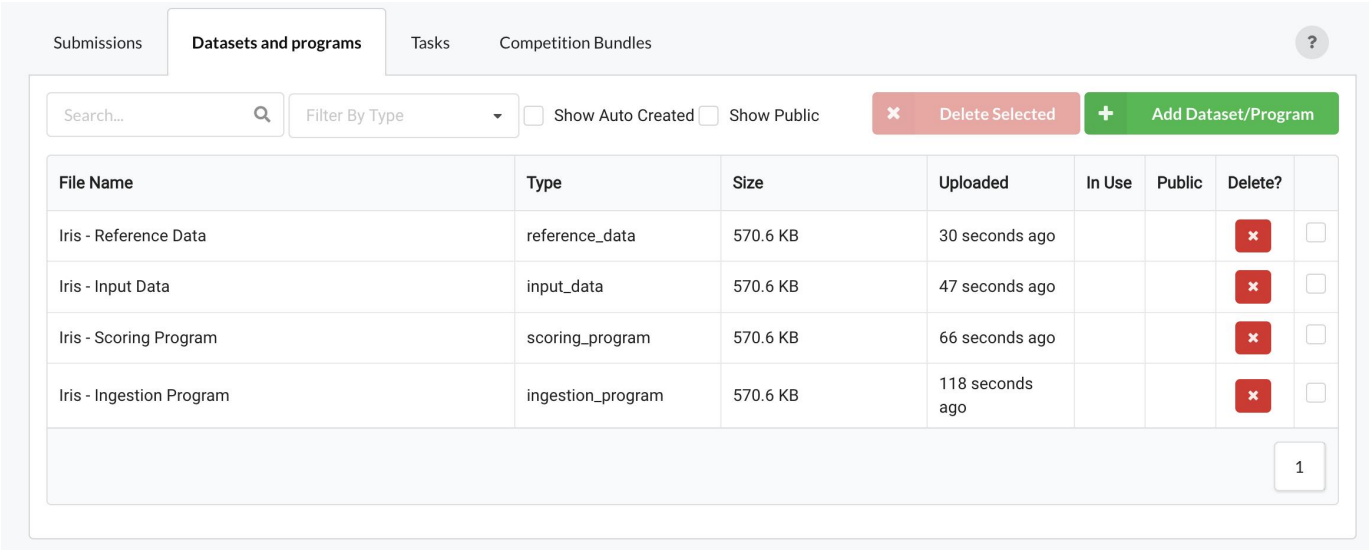
Codabench tasks are a combination of datasets and programs:

- Scoring program
- Ingestion program
- Input data
- Reference data

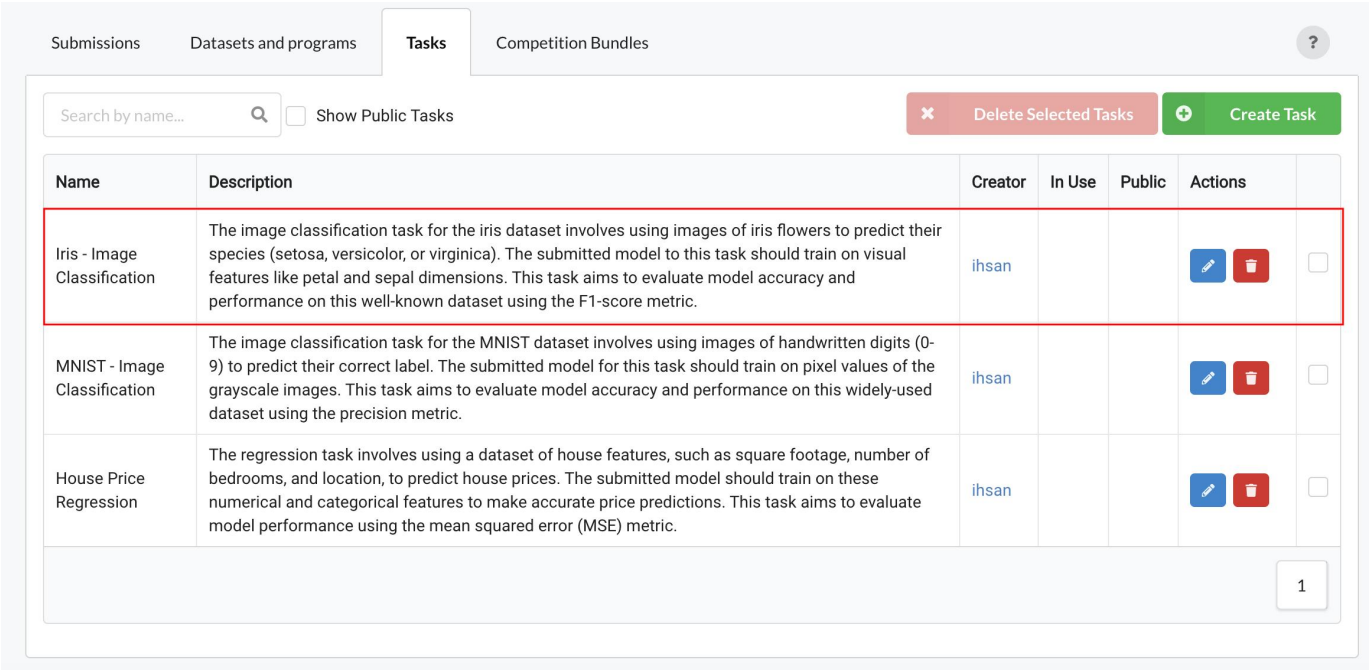
A scoring program is required while others are optional in a task.

In the Codabench Resources Interface you can upload datasets and programs in the `Datasets & Programs` tab and then create a task in the `Tasks` tab.

Example of uploaded datasets and programs:



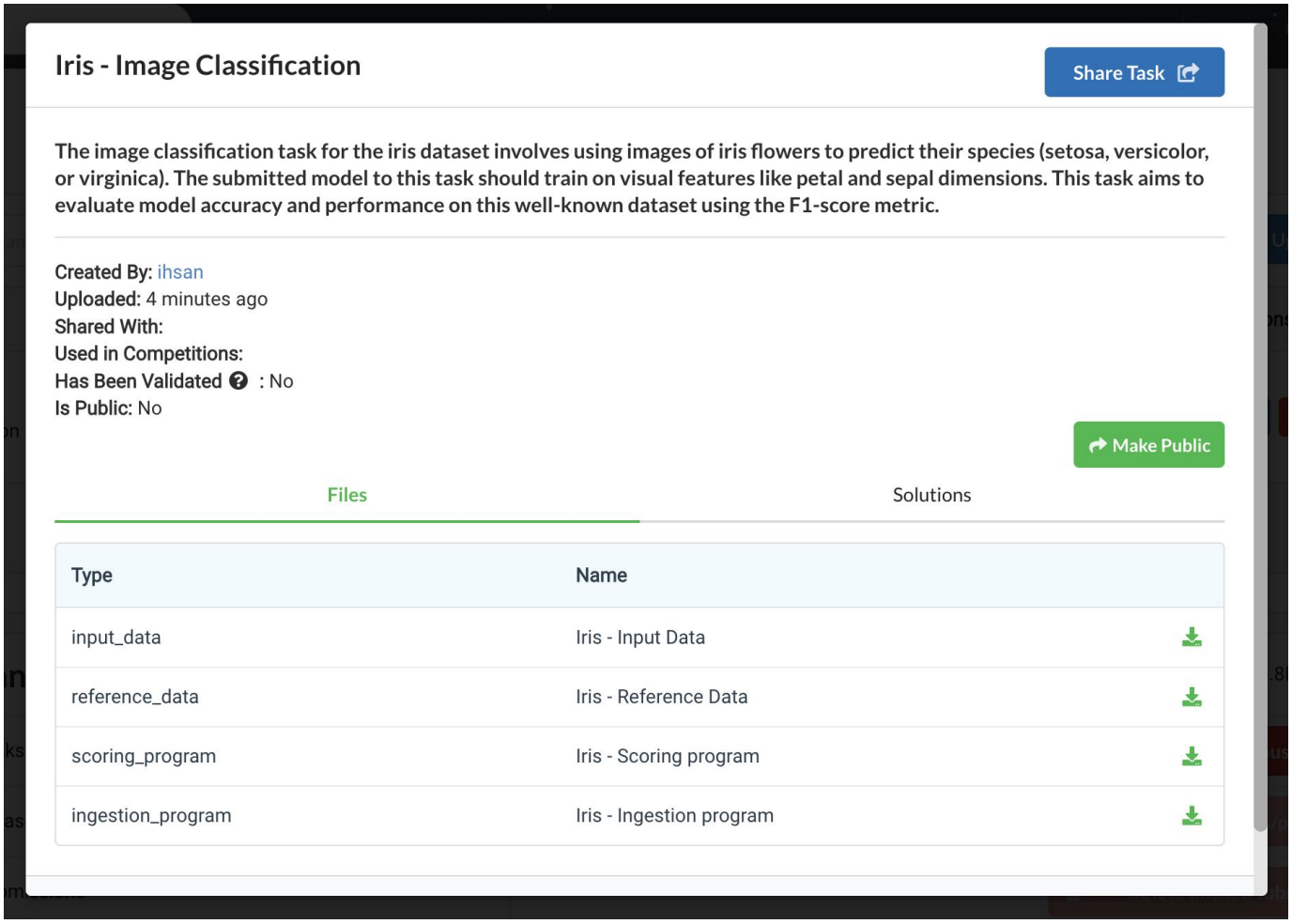
Example of a task created using the above datasets and programs:



Make a Task Public

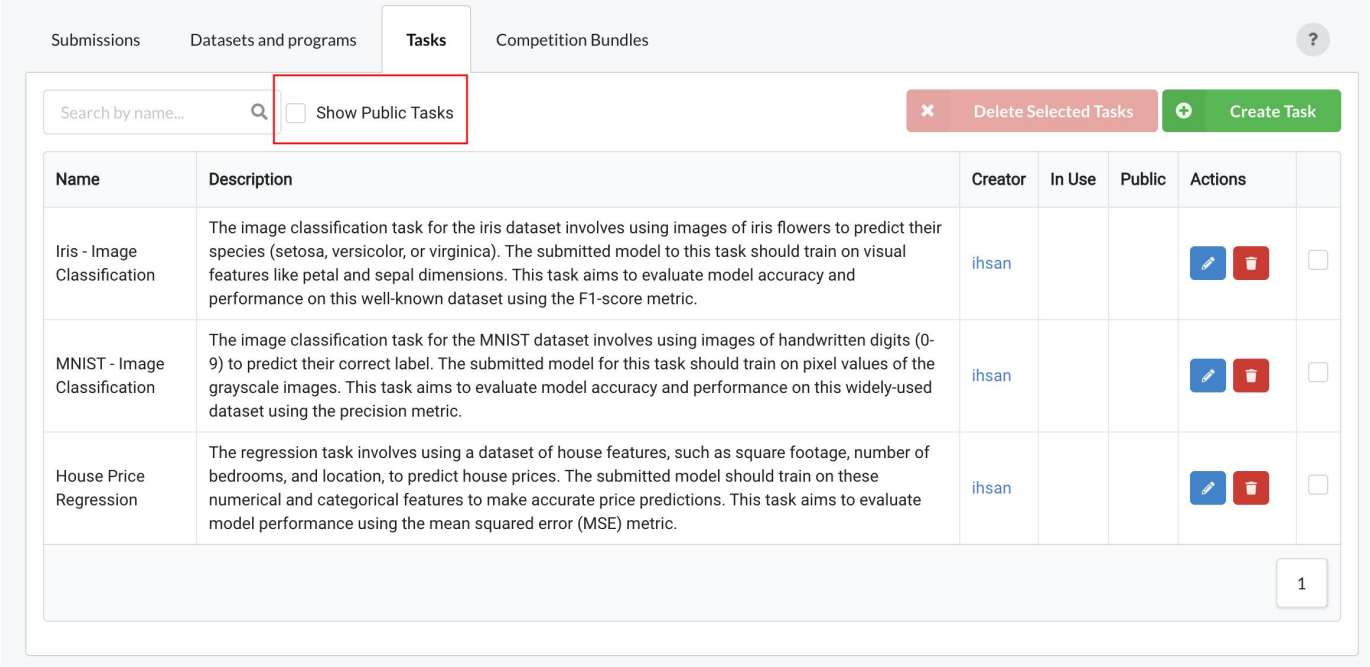
You can make a task public that you have created by clicking on the task name to show task details and then click the button `Make Public`

EXAMPLE OF TASK DETAILS:



Search Public Tasks

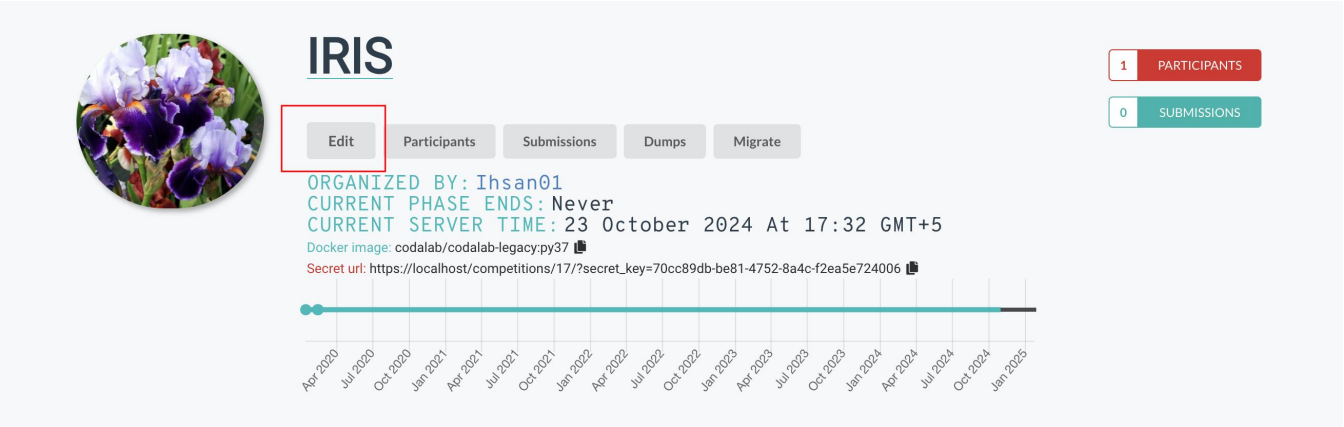
To search public tasks, you can check the `Show Public Tasks` to view public tasks from other users



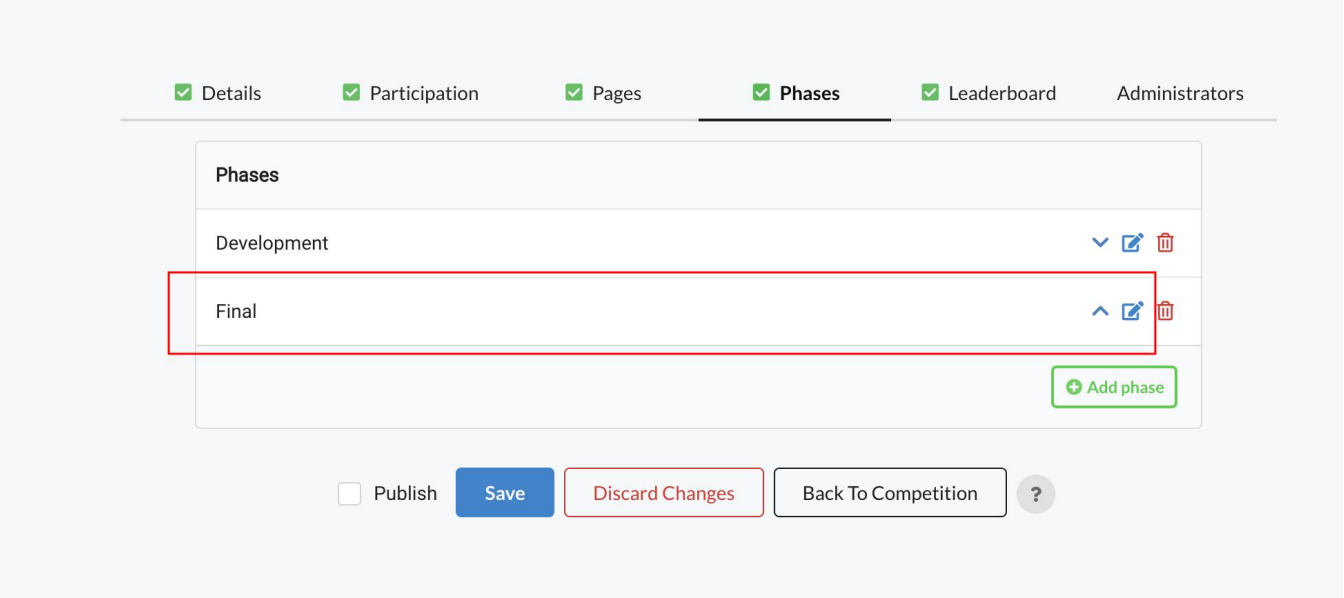
Use Public Tasks in Competitions

You can use public tasks created by other people in your competitions, to do this follow the steps below:

1. Open your competition and click `Edit` button



2. Click the `Phases` tab and click the edit button in front of the phase where you want to use a public task



3. Start writing the task name in the Tasks field and the matching task will show up. Click the task in the list to select it

3.1.14 Detailed Results and Visualization

Detailed results is a means of passing extra information from the scoring program to the frontend.

This is done via writing to a `detailed_results.html` file (OR any `.html` -- first by alphabetical order -- in the output folder), and setting `enable_detailed_results` to `True` in competition settings (via yaml or editor).

This file is watched for changes and updated on the frontend every time the file is updated, so users can get a live feed from the compute worker.

There is no limitation to the contents of this HTML file, and can thus be used to relay any information desired. Use case ideas:

- Plot data using a python plot library like matplotlib or seaborn.
- plot the learning curve over time of a reinforcement learning challenge
- plot the slope of a linear regression model
- plot the location of clusters in a classification challenge
- plot anything you can conceive of
- Run a profiler that outputs a network of method calls.
- Display any additional data about the submission file that can not be distilled down in to a score of some kind

How to include figures

Figures can be included directly inside the HTML code, by converting them in bytes format. An example is given in the scoring program of the Mini-AutoML bundle.

scoring.py

```
[...]

# Path
input_dir = '/app/input'    # Input from ingestion program
output_dir = '/app/output/' # To write the scores
reference_dir = os.path.join(input_dir, 'ref')  # Ground truth data
prediction_dir = os.path.join(input_dir, 'res') # Prediction made by the model
score_file = os.path.join(output_dir, 'scores.json')          # Scores
html_file = os.path.join(output_dir, 'detailed_results.html') # Detailed feedback

def write_file(file, content):
    """ Write content in file.
    """
    with open(file, 'a', encoding="utf-8") as f:
        f.write(content)

def make_figure(scores):
    x = get_dataset_names()
    y = [scores[dataset] for dataset in x]
    fig, ax = plt.subplots()
    ax.plot(x, y, 'bo')
    ax.set_ylabel('accuracy')
    ax.set_title('Submission results')
    return fig

def fig_to_b64(fig):
    buf = io.BytesIO()
    fig.savefig(buf, format='png')
    buf.seek(0)
    fig_b64 = base64.b64encode(buf.getvalue()).decode('ascii')
    return fig_b64

def main():
    # Initialized detailed results
    write_file(html_file, '<h1>Detailed results</h1>') # Create the file to give real-time feedback

    [...] # compute the scores

    # Create a figure for detailed results
    figure = fig_to_b64(make_figure(scores))
    write_file(html_file, f'<img src="{figure}">')
```

Example

When the visualization is enabled, a link to the detailed results can be found on the leaderboard for each submission:

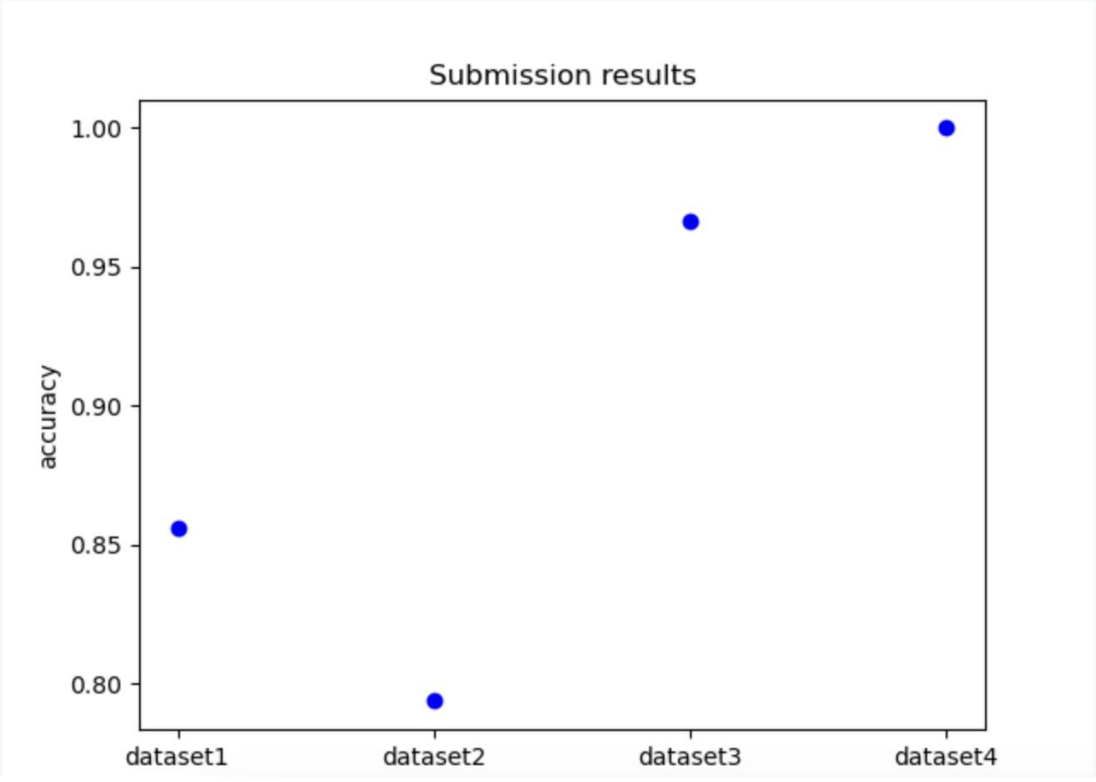
Results

	Fact Sheet Answers	Development Task						
y	Method name	Average Accuracy	Dataset 1	Dataset 2	Dataset 3	Dataset 4	Duration	Detailed Results
	Random Forest	0.9	0.86	0.79	0.97	1.0	3.08	Show detailed results
	K-Nearest Neighbors	0.87	0.77	0.76	0.96	1.0	1.4	Show detailed results
	Decision Tree	0.87	0.81	0.69	0.97	1.0	0.22	Show detailed results
	MultiLayer Perceptron	0.86	0.79	0.67	0.99	1.0	3.02	Show detailed results
	Gaussian NB	0.85	0.79	0.77	0.92	0.91	0.12	Show detailed results

The `detailed_results.html`, generated by the scoring program, is then shown:

Detailed Results

Detailed results

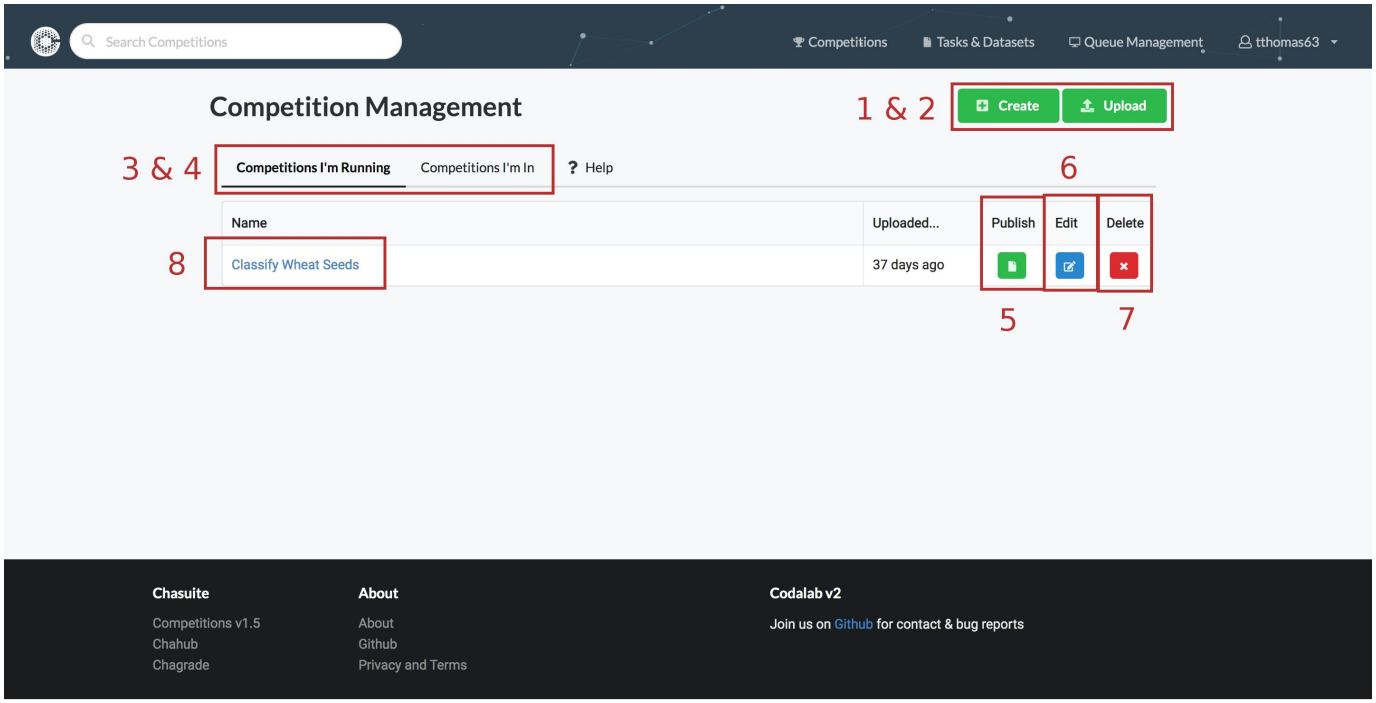


3.2 Running a Benchmarks

3.2.1 Benchmark Management & List Page

This page will show you how to create, manage, edit and delete your competitions.

It will also show you how to track the competitions you are currently in.



Competition create button (Form)

This button will take you to the wizard/form for creating competitions. This will allow you to walk through each step of creating a competition using our creation/edit form. For more information on this form/wizard, please see the following link: Competition Creation: Form

Competition create button (Upload)

This button will take you to the upload page for competition bundles. Here you will be able to upload a competition bundle, and if it is validated and processed successfully, you should see a link to your new competition. For more information on this page, please see the following link: Competition Creation: Bundle

Competitions I'm running tab

This should be the default selection for the tab navigation at the top. Having this selected will show you all the competitions you currently run/ manage, and the available actions for them.

Competitions I'm in tab

Clicking on this tab will change the main view of the page. You should now see a list of competitions you're competing in (Without any competition administrator options). Clicking any of these titles should bring you to the competition detail page of that competition.

Publish competition button

This button will publish your competition in order to make it publicly available. If your competition is already published, this button will appear green and be used to remove your competition from public availability (It will not be deleted). By default, if your competition is un-published, it appears grey.

Edit competition button

This button will take you to the wizard/form for editing competitions. For more information on the competition edit form, please see the link here

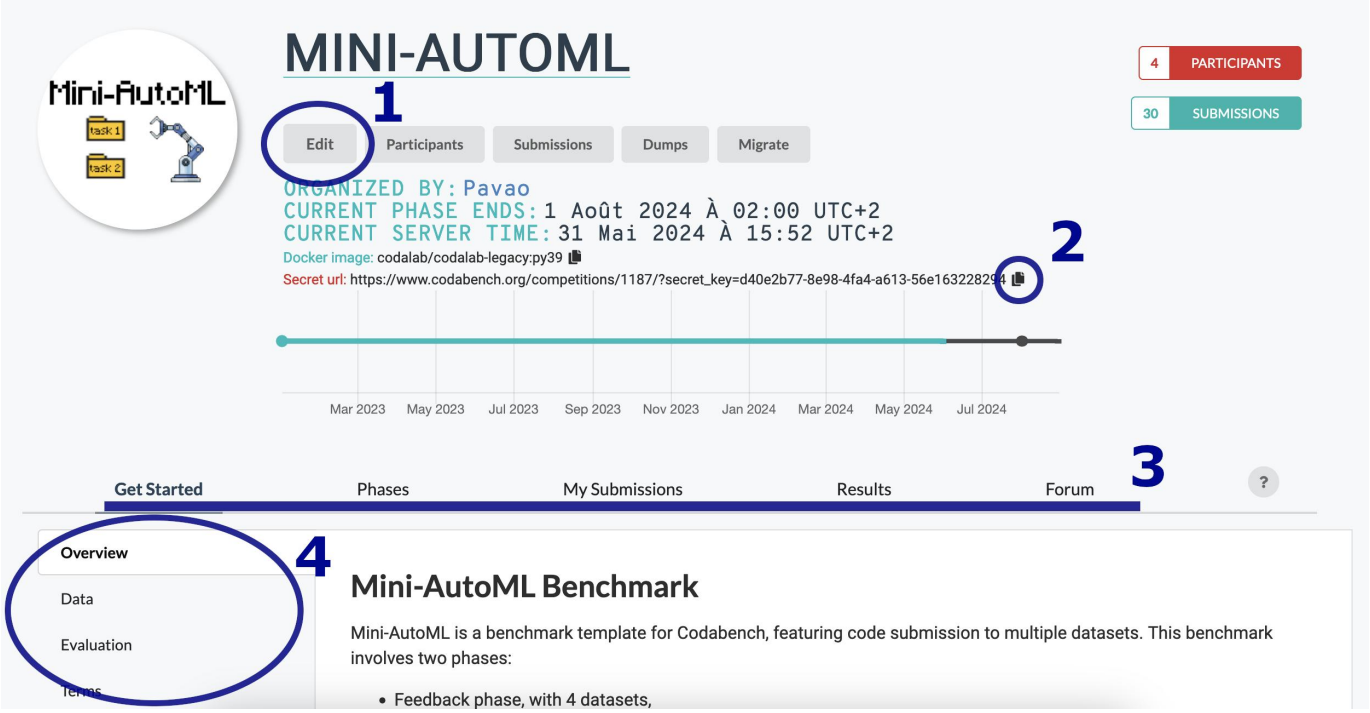
Delete competition button

Deletes your competition. There will be a confirmation dialogue before deletion. We cannot recover deleted competitions.

Competition link

A link to the competition's detail page where users can register, make submissions, view the leaderboard and terms, etc. For more information about the competition detail page, see the link here

The competition detail page is the main way to interact with competitions. This is where your participants (Or if you are a participant) will read the pages you uploaded, register, make submissions, and check results.



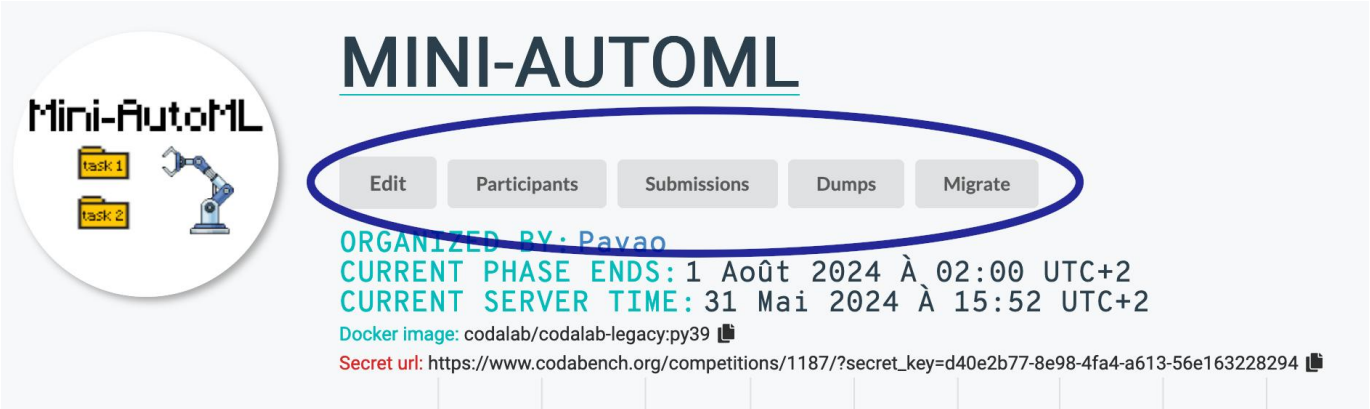
1) Editor (organizer feature)
2) Copy competition secret URL (Document icon, secret URL covered)
3) Competition Detail Tab Navigation

- Get Started
- Phases
- My Submissions
- Results
- Forum (if enabled)

4) Competition Detail Tab Pane Navigation

Competition organizer features

These features are only for competition organizers.

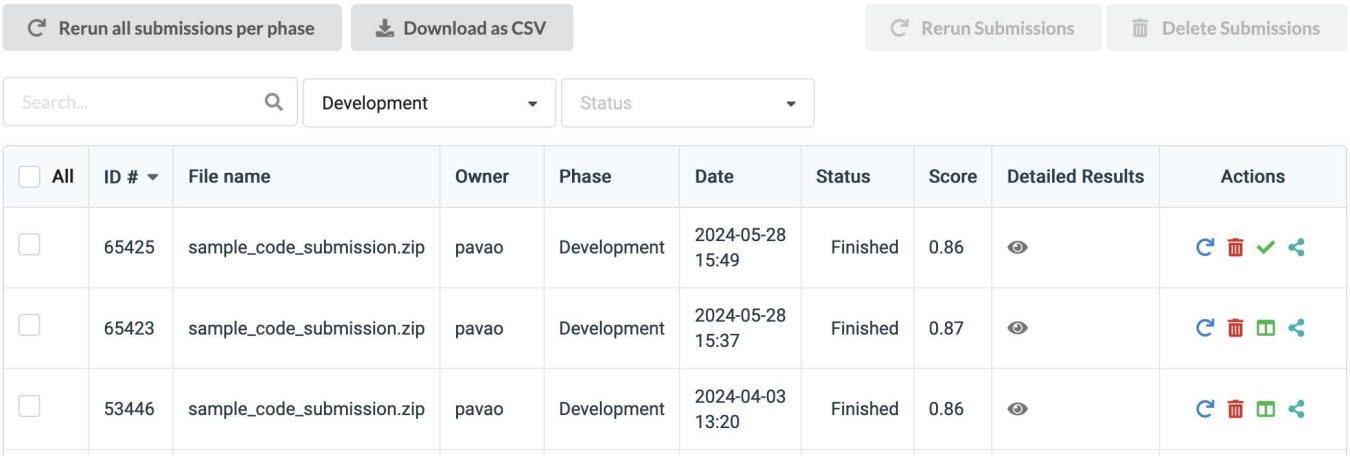


- Editor
- Manage participants
- Manage submissions
- Manage dumps (save the current state of the competition as a bundle)
- Migration

SUBMISSIONS

From here, you can: - Delete submissions - Re-run submissions - Set a submissions score - Force a submission to leaderboard

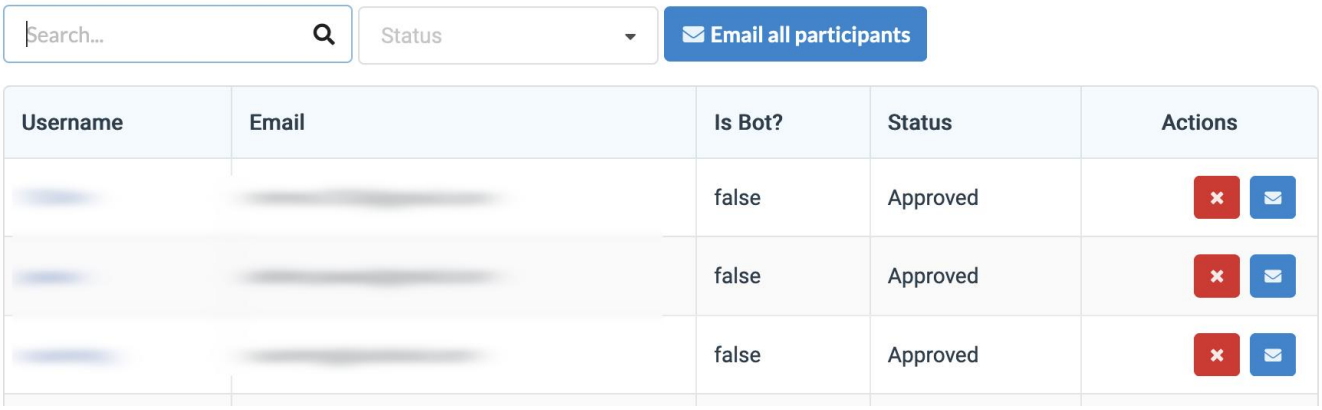
You can also view the logs, and all output associated with a submission.



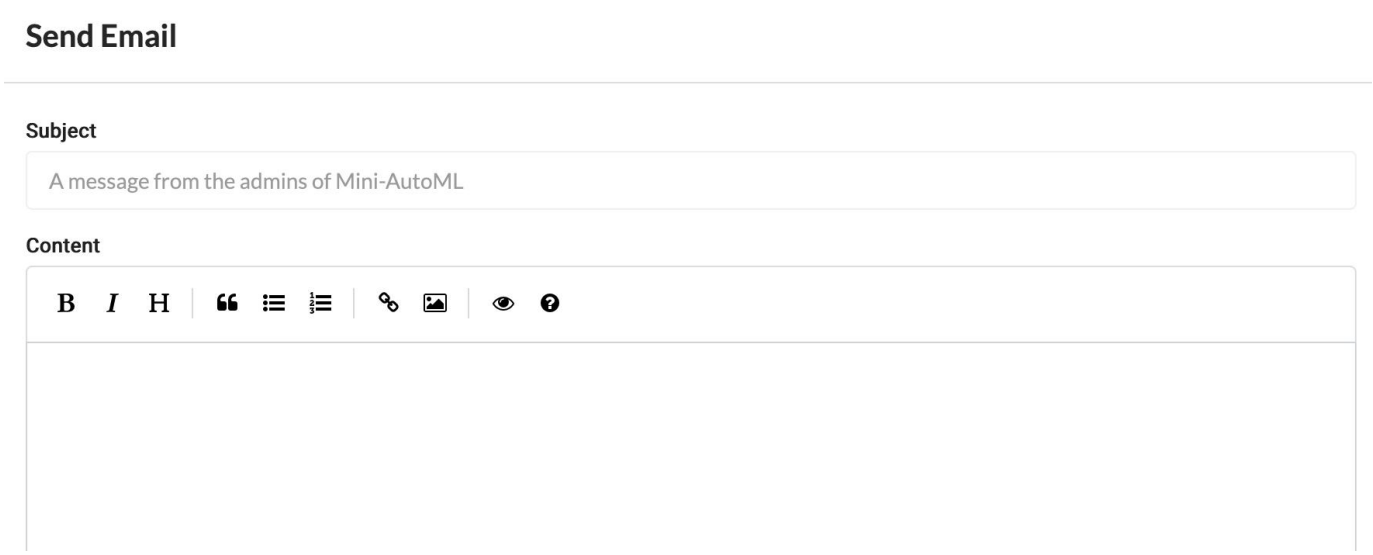
- Download CSV: Download a CSV file with all submission info
- Re-run all submissions per phase: Re-runs all submissions in a phase.
- Search: Used to search for a submission by file name
- Phase: Used to filter submissions by phase
- Status: Used to filter submissions by status
- Action Buttons:
- Blue Circular Arrow: Re-runs the submission
- Yellow Cross: Cancels the current submission if it is running
- Red Trash Can: Deletes the submission
- Green arrow: Puts this submission on the leaderboard

PARTICIPANTS

From here, you should be able to: - Email all participants - Approve/Deny participants - Revoke participants



- Search: Search for a participant by username or email address.
- Status: Filter participants by status
- Email Participants: Opens a modal to email all participants:
- Subject: The email subject
- Content: The email content



COPY COMPETITION SECRET URL

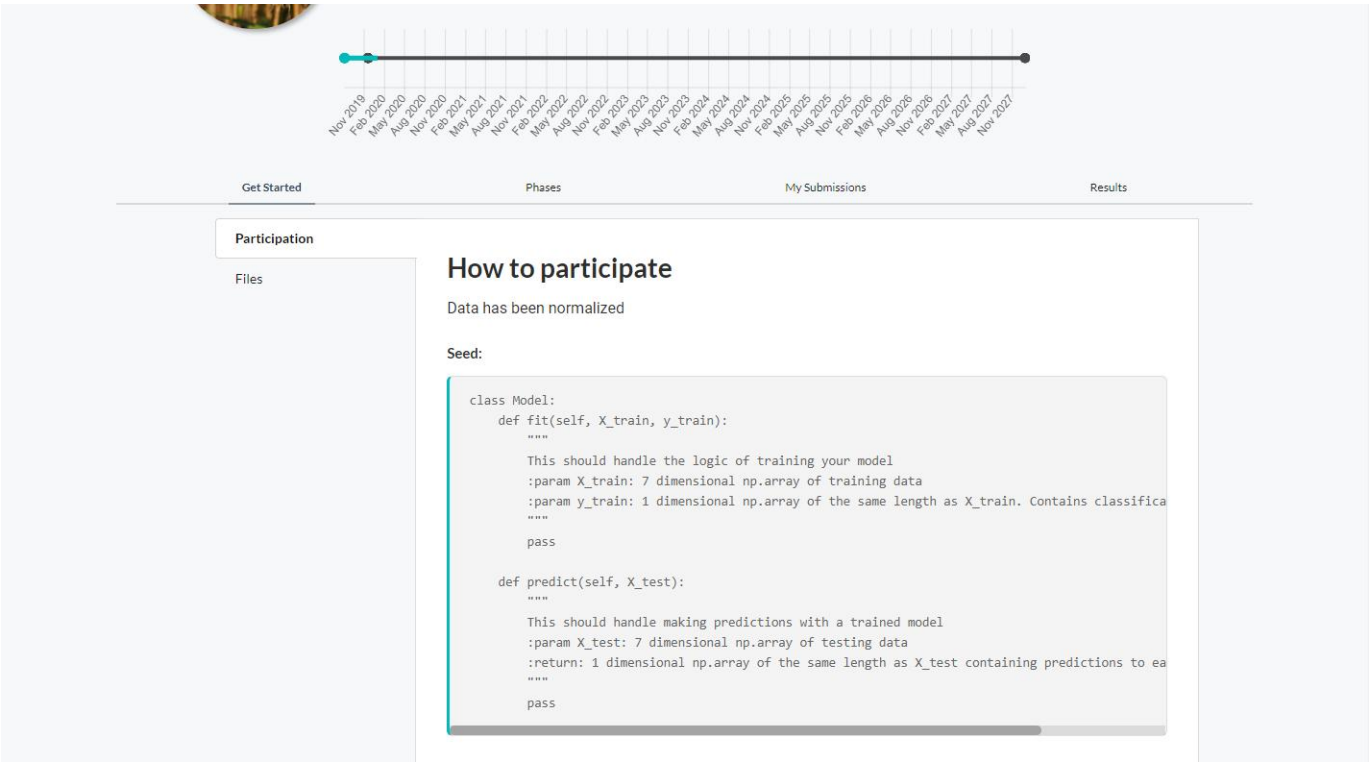
Clicking the document icon copies the competition secret URL to your clipboard.

3.2.2 Competition Detail Tab Navigation

Used to navigate between the different sections of the competition detail page.

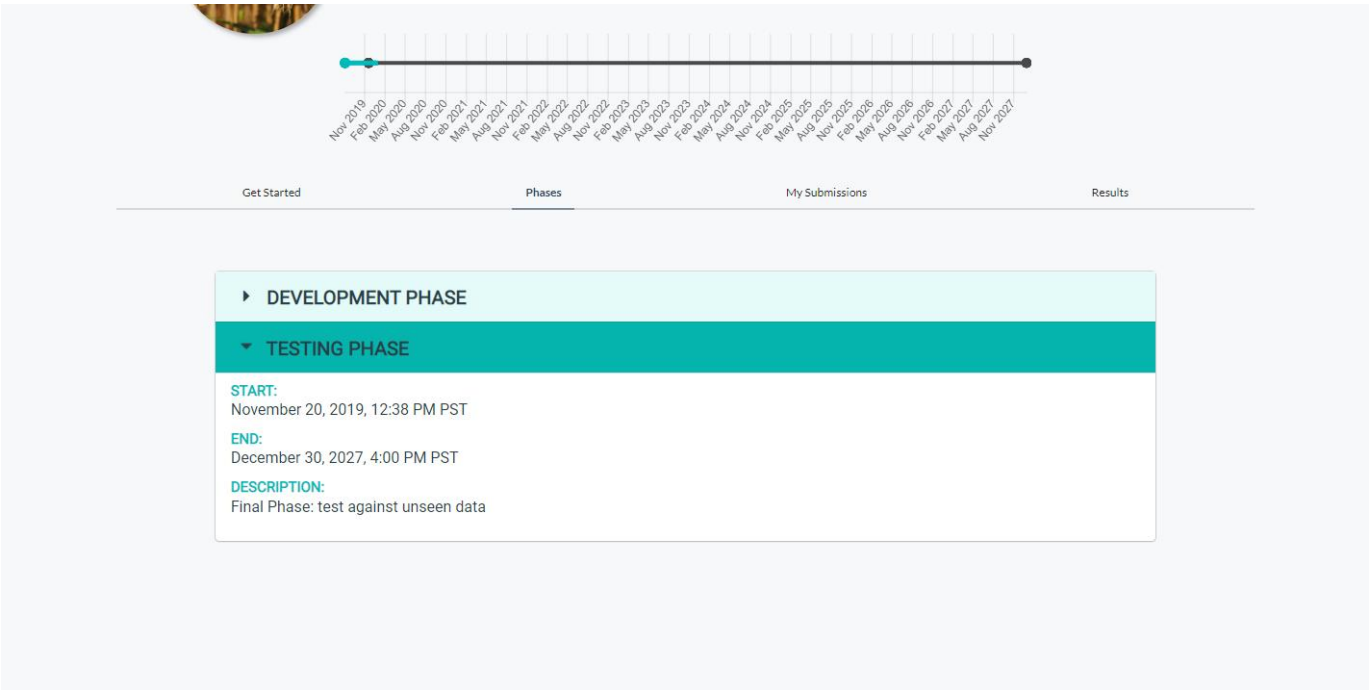
GET STARTED

Contains all the organizer made pages, and some defaults.



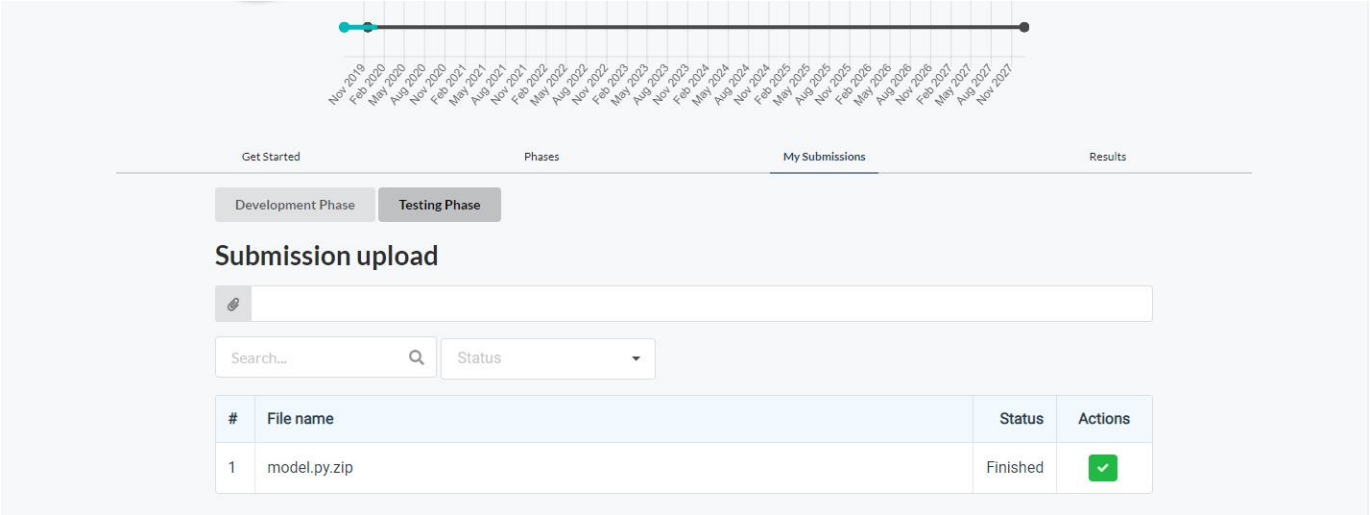
PHASES

Contains a diagram list with details on each phase in the order in which they're active.



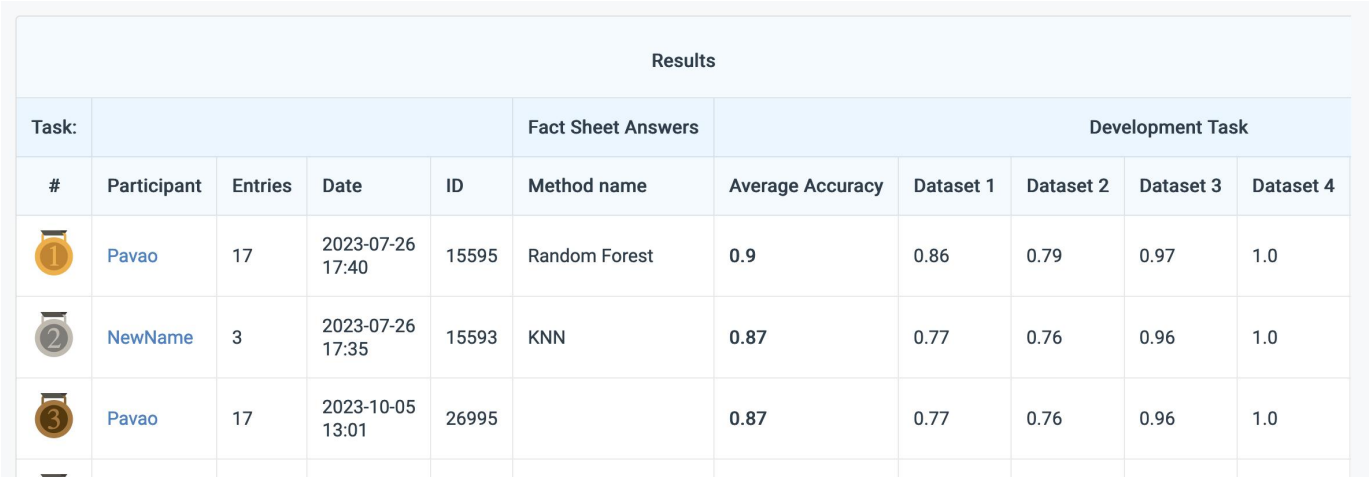
MY SUBMISSIONS

This view contains a table with all your submissions, and allows you to upload new ones.



RESULTS

This view contains the leaderboard results.



Results

Task:					Fact Sheet Answers	Development Task				
#	Participant	Entries	Date	ID	Method name	Average Accuracy	Dataset 1	Dataset 2	Dataset 3	Dataset 4
1	Pavao	17	2023-07-26 17:40	15595	Random Forest	**0.9**	0.86	0.79	0.97	1.0
2	NewName	3	2023-07-26 17:35	15593	KNN	**0.87**	0.77	0.76	0.96	1.0
3	Pavao	17	2023-10-05 13:01	26995		**0.87**	0.77	0.76	0.96	1.0

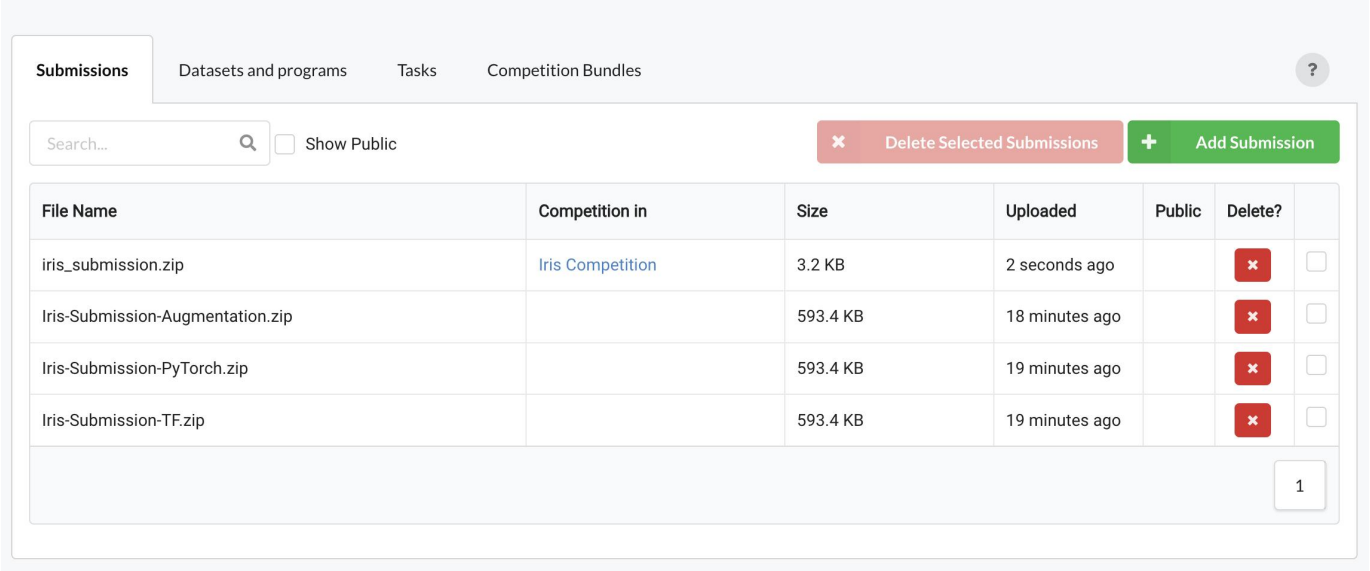
3.2.3 Ressource Management Submissions, Datasets/Programs, Tasks and Competition Bundles

This page is where you can manage your resources e.g. Submissions, Datasets, Programs, Tasks, and Competition Bundles. You can also view and manage your quota on this page.

You can access this interface by clicking on "**Resources**" in the main menu:

Submissions

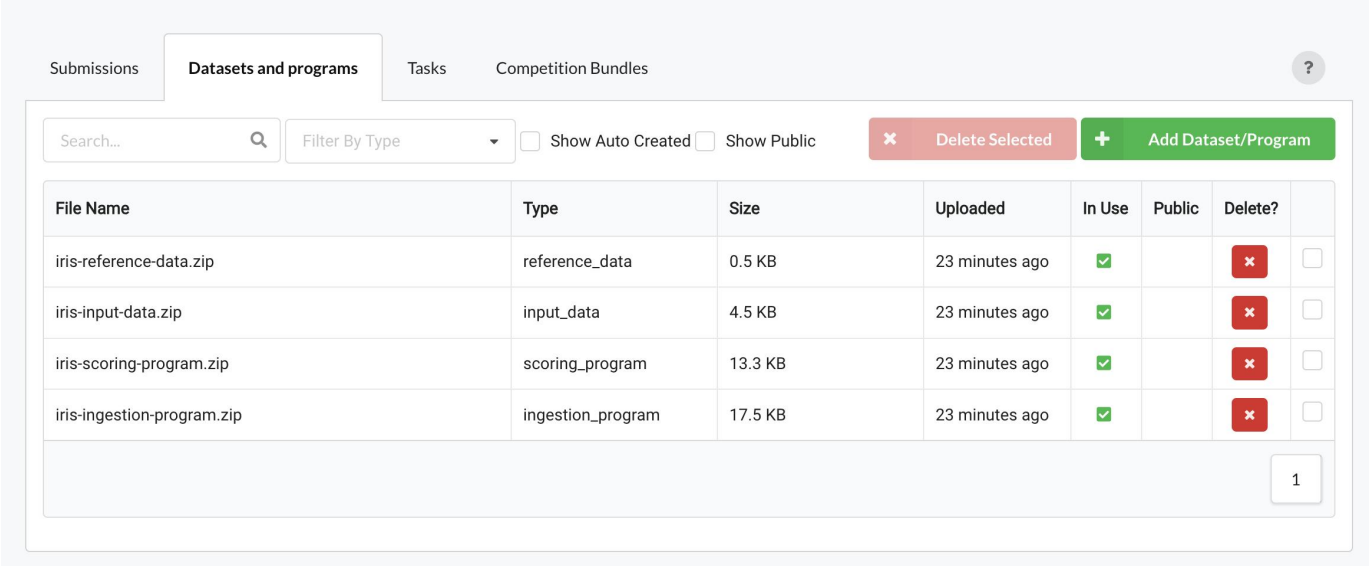
In this tab, you can view all your submissions either uploaded in this interface or submitted to a competition.



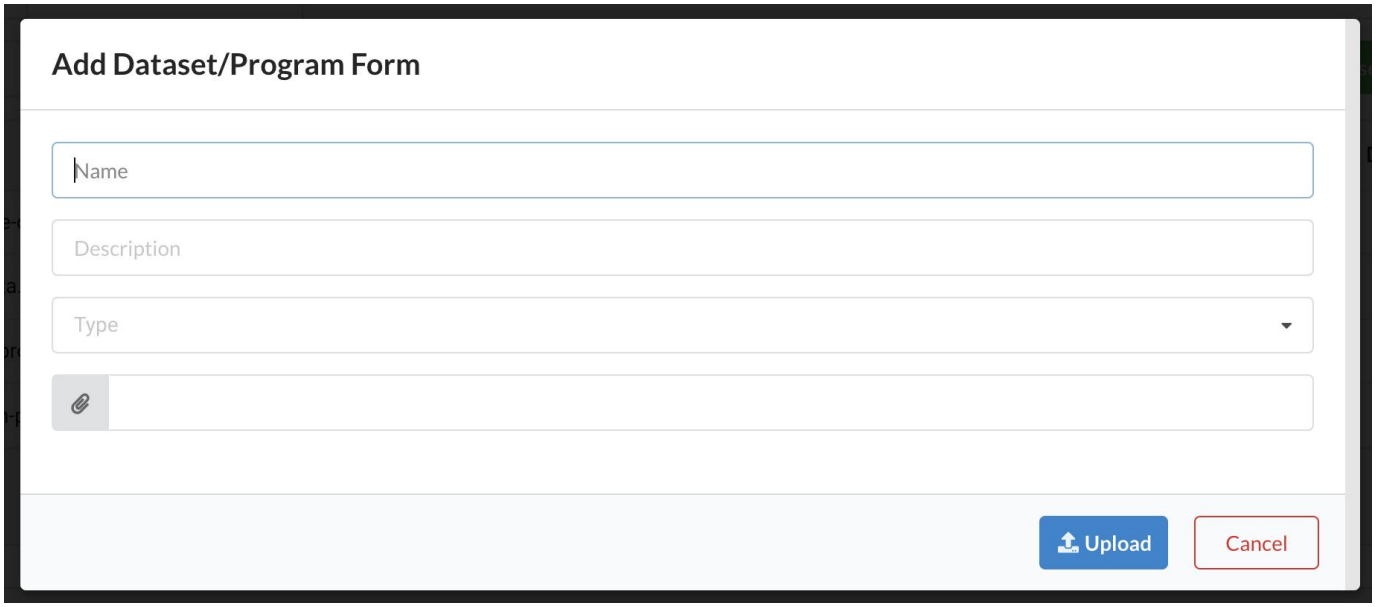
By clicking the `Add Submission` button you can fill a form and attach a submission file to upload a new submission. This is useful in different cases e.g. when you want to share a sample submission with the participants of a competition you are organizing.

Datasets/Programs

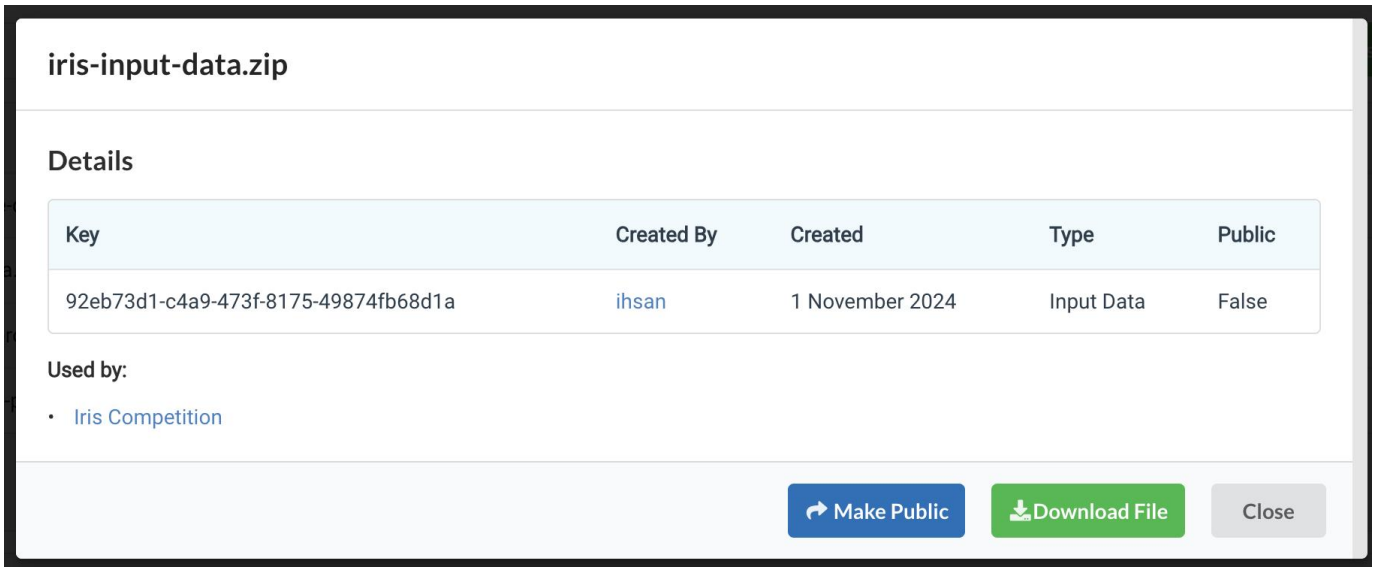
In this tab, you can view the datasets and programs that you have uploaded. You can also view auto-created and publicly available datasets/programs by checking the relevant checkboxes.



By clicking the `Add Dataset/Program` button you can fill a form and attach a dataset file to upload.



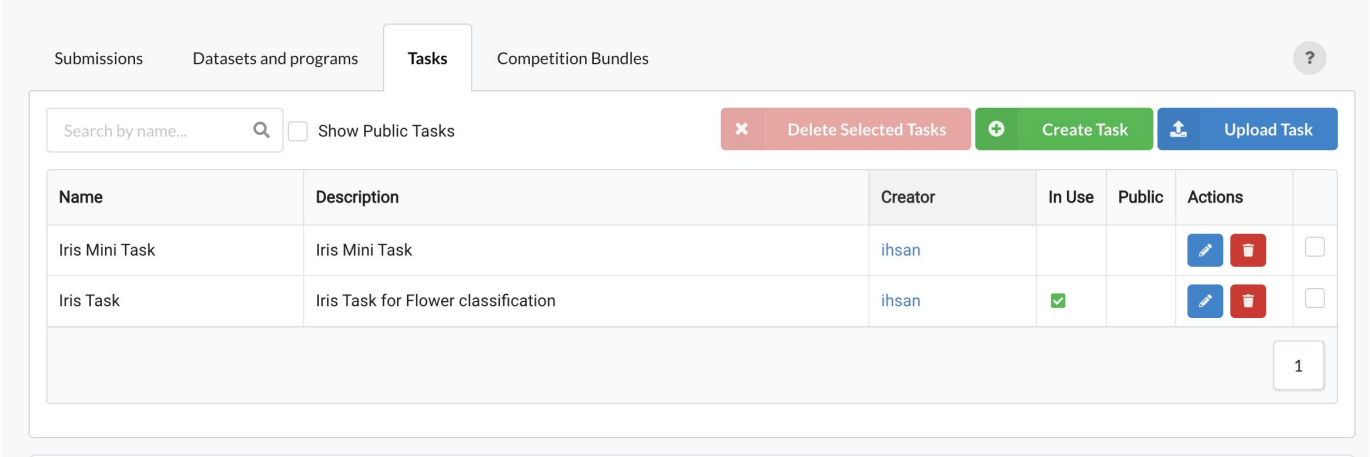
You can click on a dataset/program and make it public or private. This is useful when you want to share a dataset with participants so that hey can use it to prepare a submission for a competition.



For a general breakdown of the roles of different types of datasets, see this link: Competition Bundle Structure: Data types and their role.

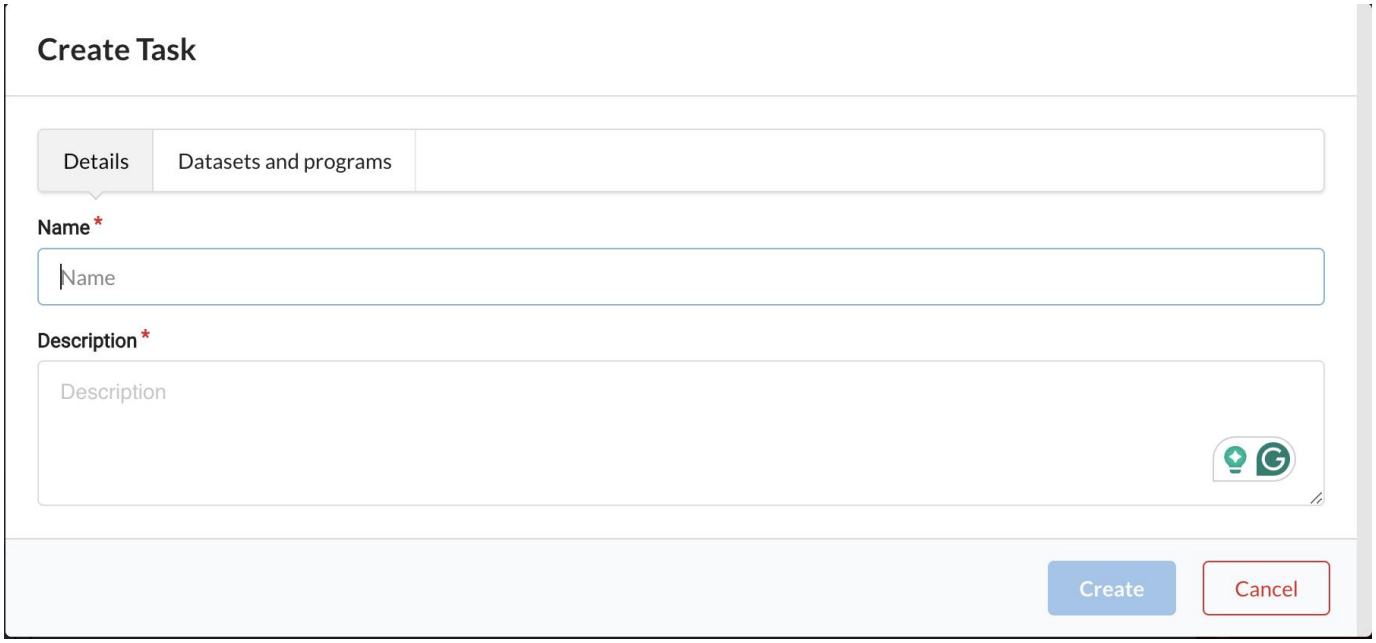
Tasks

In this tab, you can manage your tasks. You can create a new task, upload a task, edit a task and check task details.

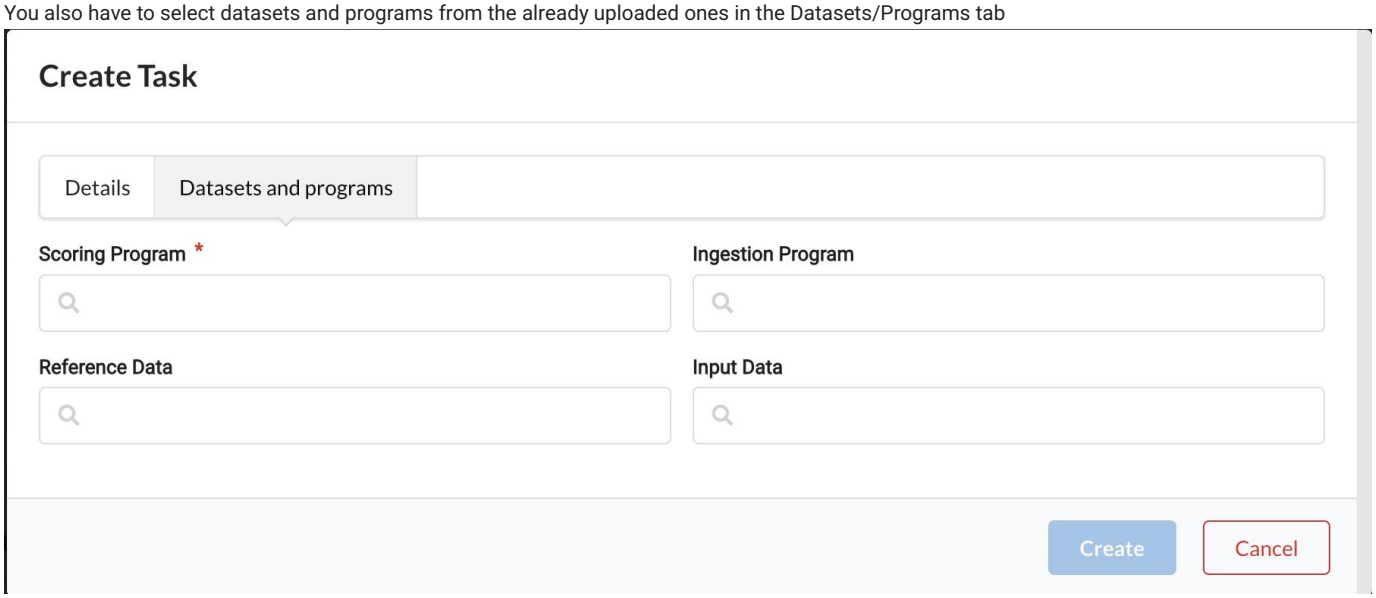


CREATE NEW TASK

To create a new task, you have to fill the form by entering task name and description

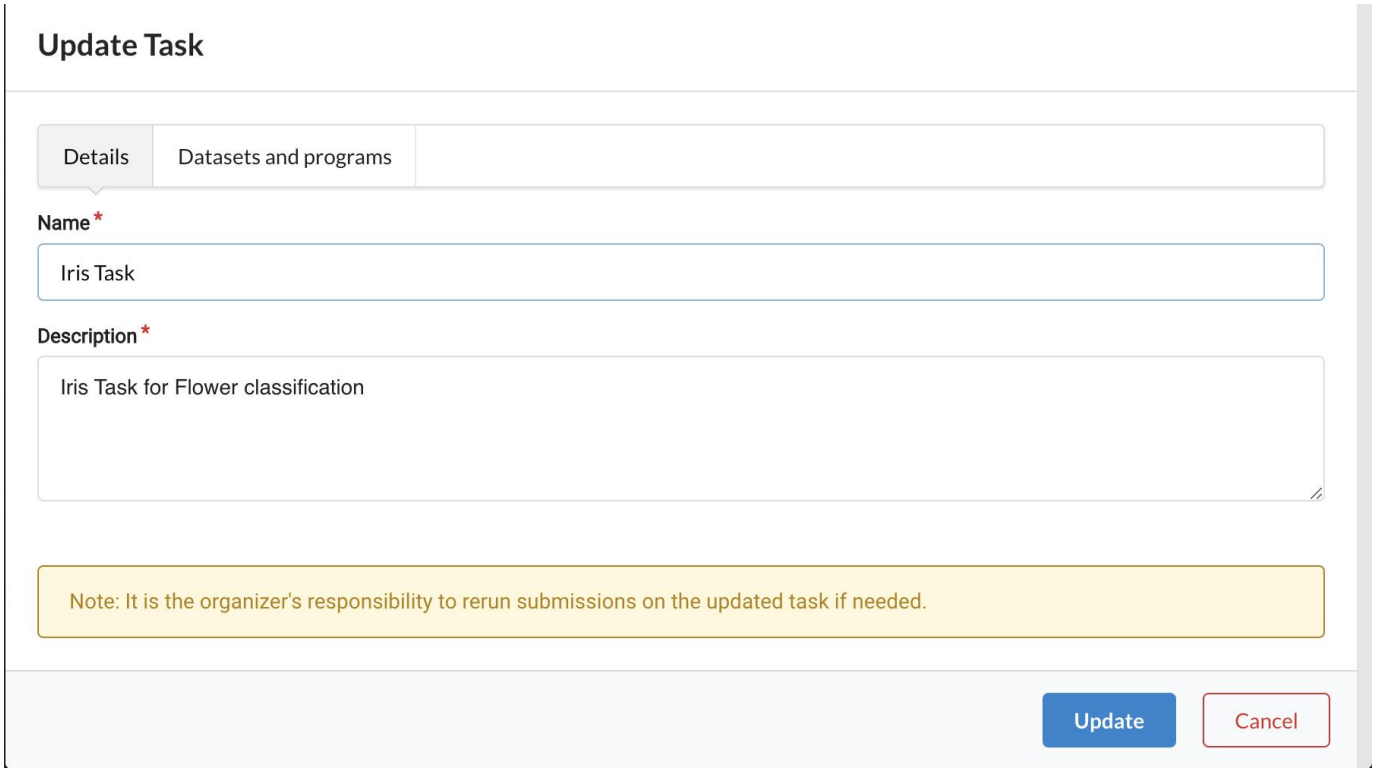


You also have to select datasets and programs from the already uploaded ones in the Datasets/Programs tab

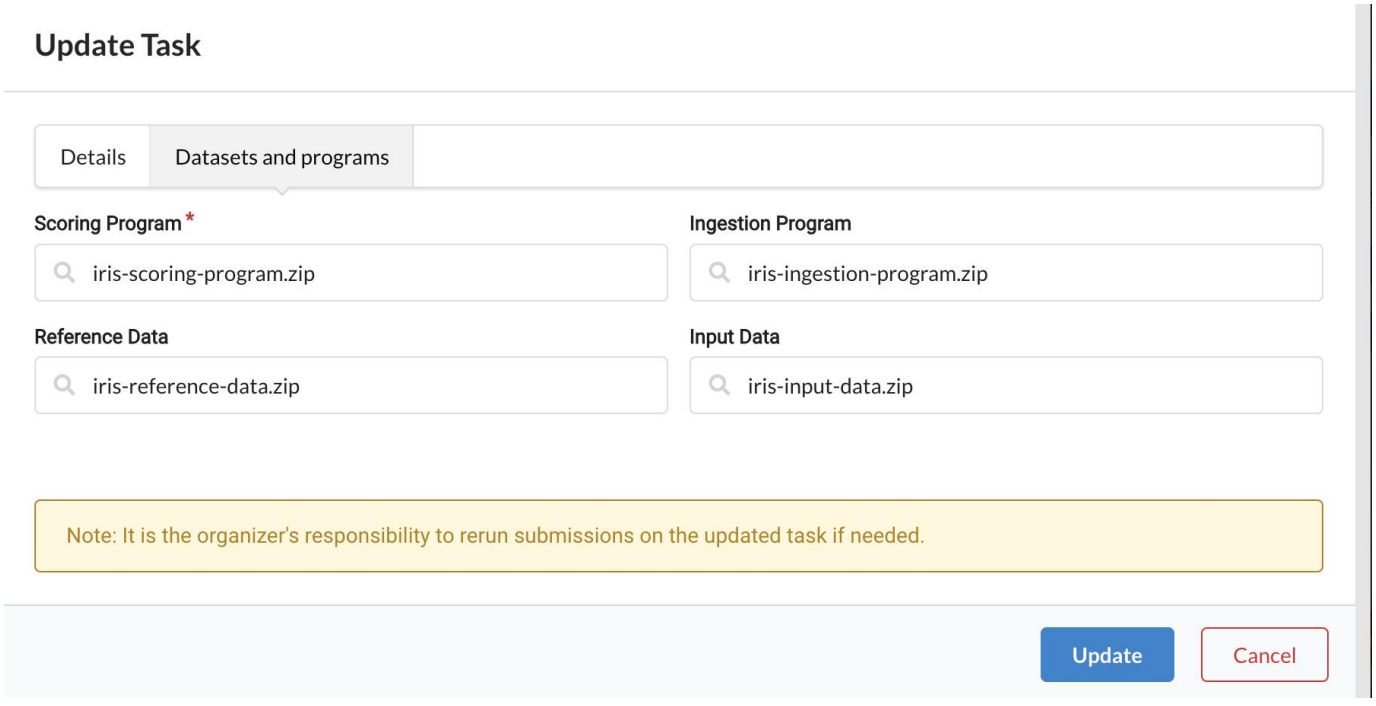


EDIT A TASK

You can change the task name and description



You can also change the datasets/programs used in the task

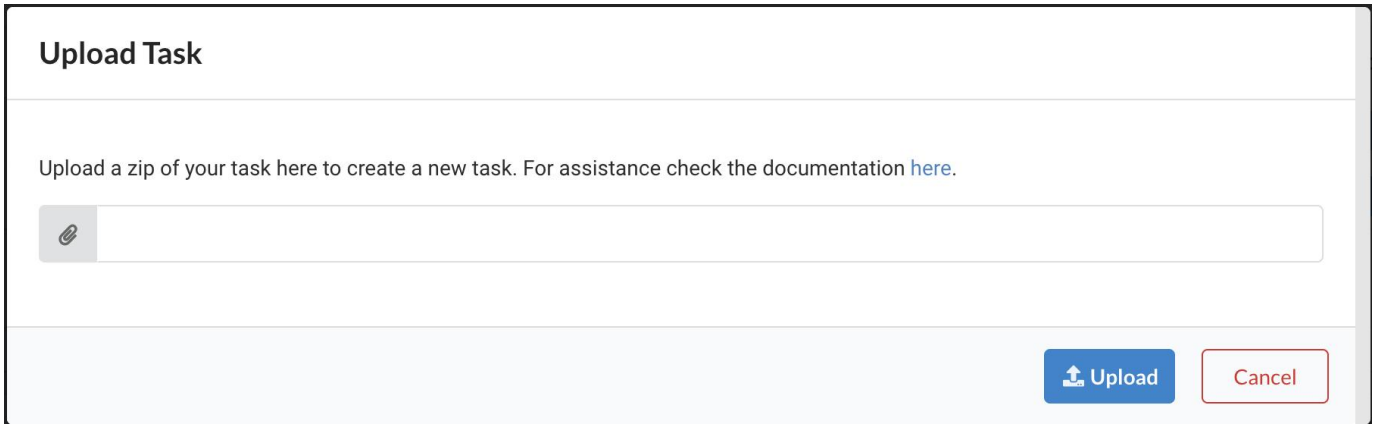


Note

Organizers should be careful when updating a task because some submissions may have used the task and updating the task will not allow you to rerun those submissions because the task they have used is now changed.

UPLOAD A TASK

You can create a new task by uploading a task zip that has the required files in the correct format.



Create a zip file that consists of a `task.yaml` file and zips of datasets/programs if required. You can use already existing datasets/prograsm by using their keys in the yaml, or upload new datasets/programs or use a mix of keys and files e.g. you choose to use already existing input data and reference data but use zip files for ingestion and scoring program. In the last case, codabench will create two programs and then use them in your task and will use existing datasets in the same task.

Check the files below for examples of task upload zips.

- task_with_keys_only.zip
- task_with_files_only.zip
- task_with_mix_of_keys_and_files.zip

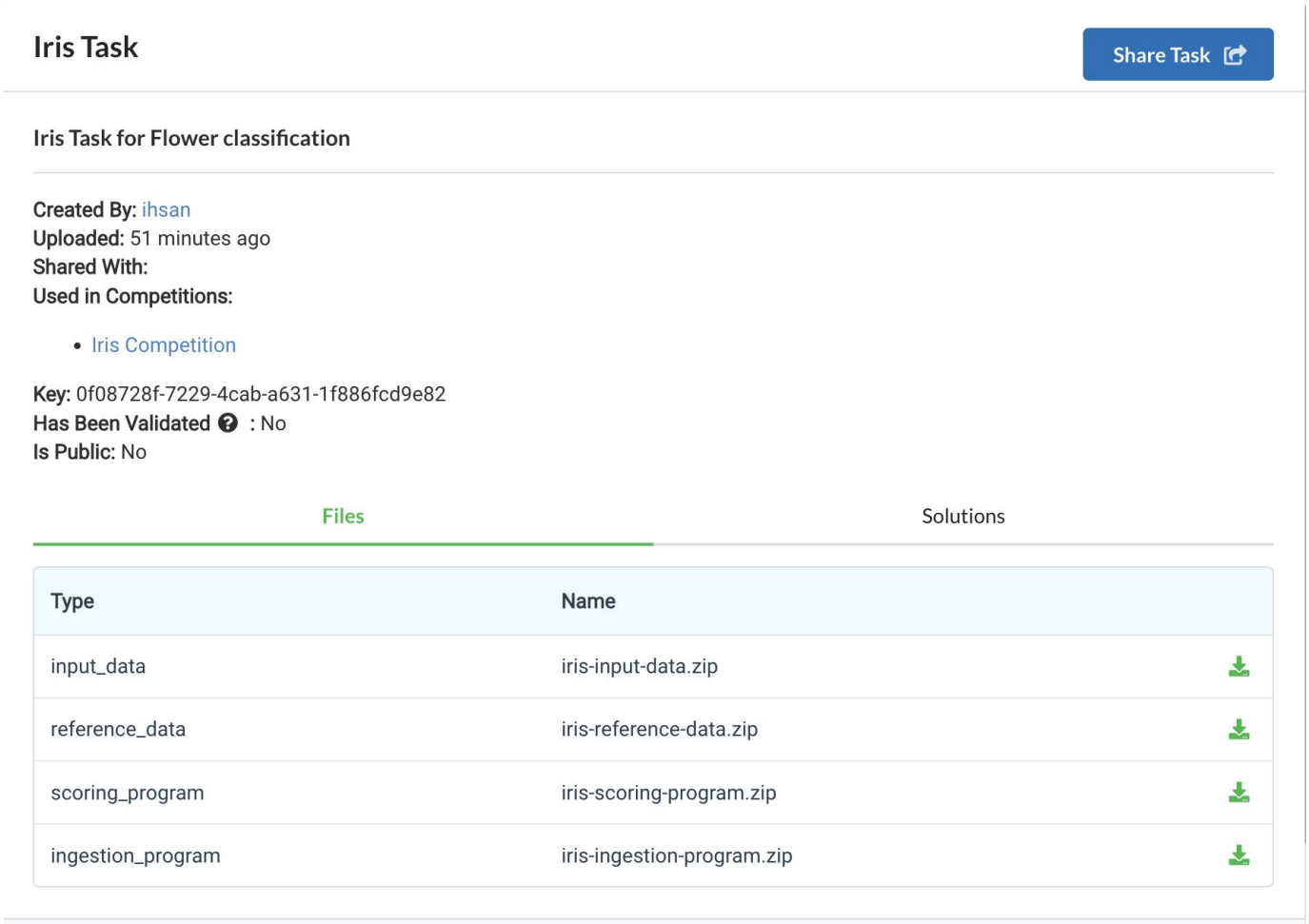
For reference, here is the content of the `task.yaml` file that you can find inside the `task_with_mix_of_keys_and_files.zip` task:

task.yaml

```yaml
name: Iris Task
description: Iris Task for Flower classification
is_public: false
scoring_program:
  zip: iris-scoring-program.zip
ingestion_program:
  zip: iris-ingestion-program.zip
input_data:
  key: 6c3e6dde-d0fa-4c22-af66-030187dbfd4f
reference_data:
  key: c4179c3f-498c-486a-8ac5-1e194036a3ed
```

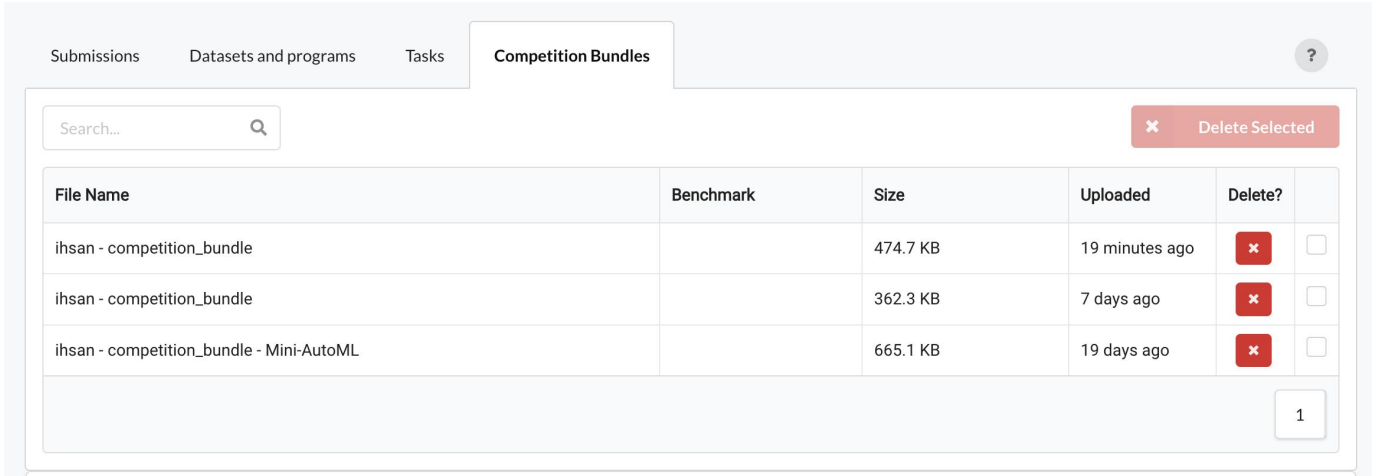
TASK DETAILS

In the task details, you can view all the task details e.g. title, description, task owner, created date, people with whom this task is shared, competitions where this task is used, the datasets/programs used in this task and option to download them, and option to make the task public/private.



Competition Bundles

In this tab, you can mange your competition bundles. These bundles are stored when you create your competitions using a zip.



Quota and Cleanup

This section of the resource interface shows you the usage of your quota. A free quota of 15 GB is given to all the users and this can be increased by the platform administrators in special circumstances for selected users. You can also do some quick cleanup from here by deleting unused resources e.g. submissions, datasets and tasks etc.

3.2.4 Update programs or data

In this page, you'll learn how to update critical elements of your benchmark like the scoring program or the reference data, while your benchmark is already up and running.

For a general overview of resources management, click here.

If order to update your programs or data, you have two approaches: - **A.** Edit an existing Task (simpler and straightforward) - **B.** Create a new Task

Let's see both approach in detail.

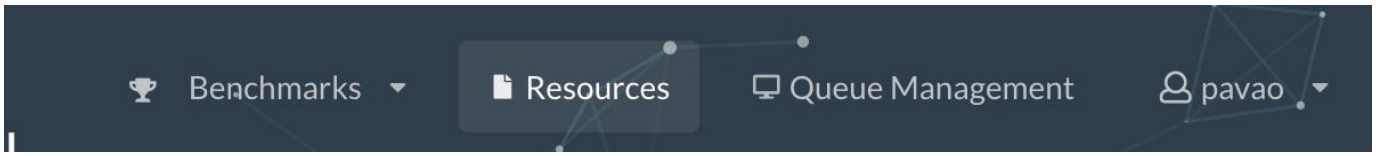
A. Edit an existing Task

1. PREPARE THE NEW DATASET OR PROGRAM

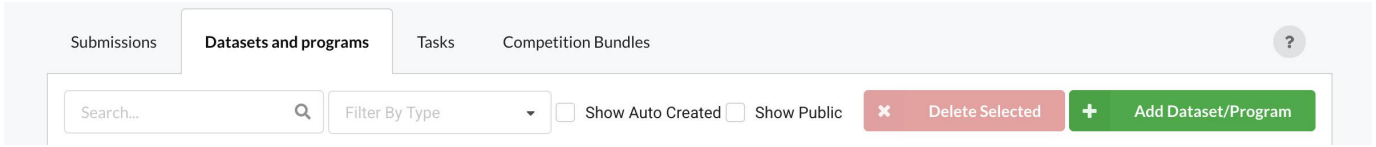
- Make local changes to the elements you want to update: scoring program, ingestion program, input data and/or reference data.
- Zip the new version of your program or data. **Make sure to zip the files at the root of the archive, without zipping the folder structure.**

2. UPLOAD THE NEW DATASET OR PROGRAM

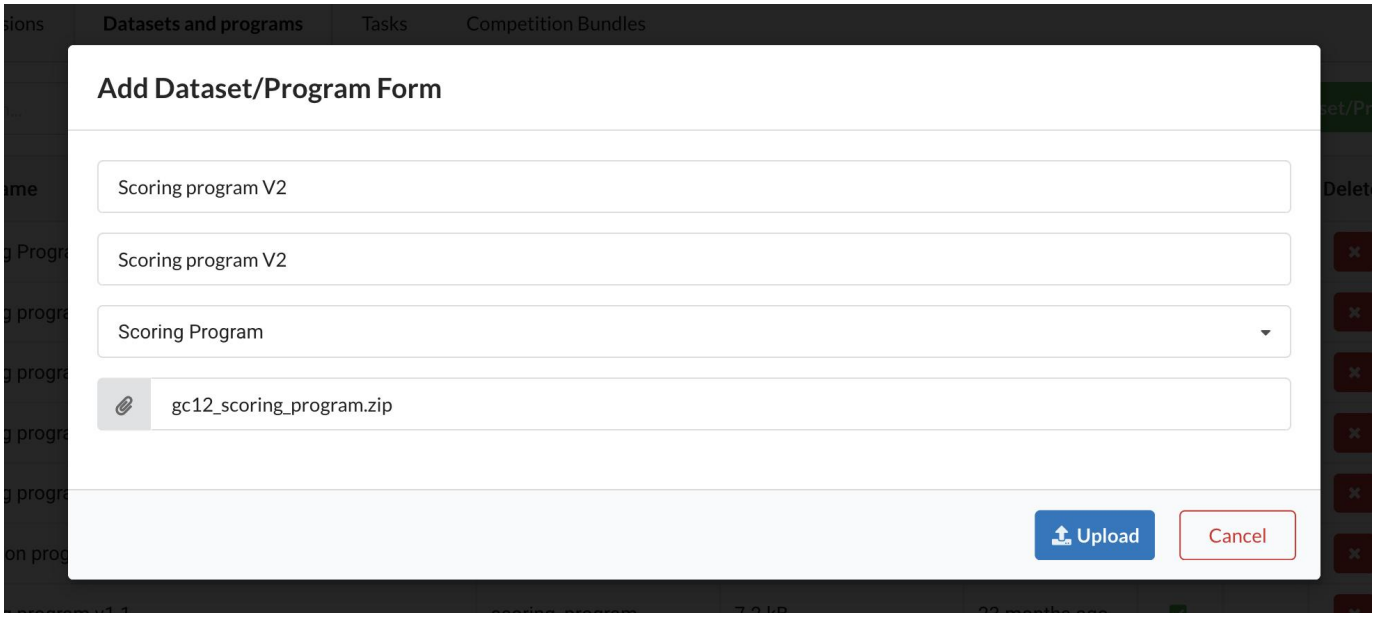
- Go to Resources



- Go to "Datasets and Programs" and click on "Add Dataset/Program"



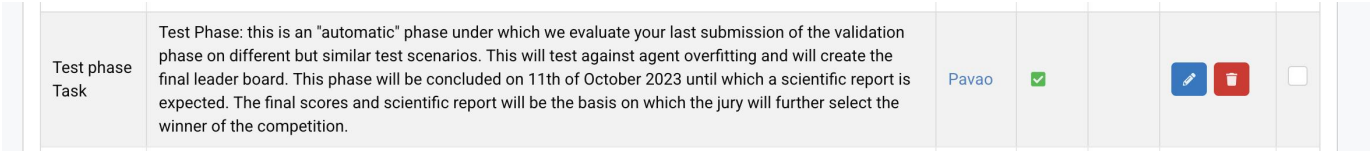
- Fill in the form: Name the new program or dataset, select the type (scoring program, input data, etc.), and select your ZIP file



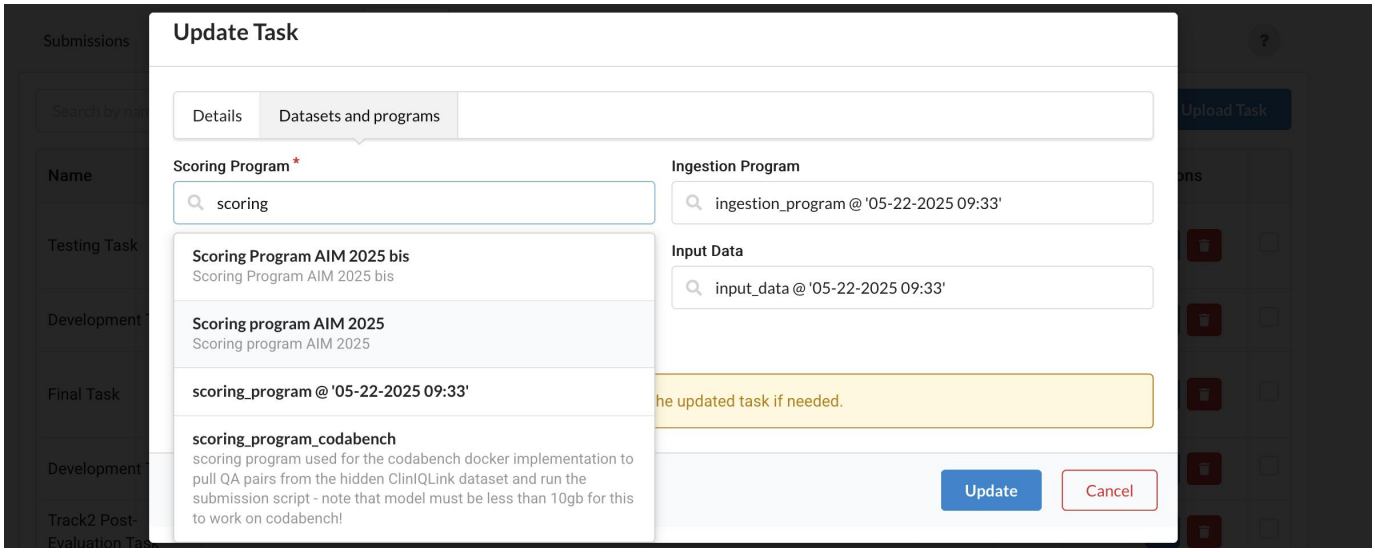
3. UPDATE THE TASK USED BY YOUR BENCHMARK

Still on "Resources" page, go to the "Task" tab. Find the task you want to edit. In order to recognize it, make sure it is marked as "In Use", and click to see more information and make sure it is related to the right benchmark.

Then click on the pencil symbol to edit it:



Start typing the name of your new program or dataset in the corresponding field and select it, then save.



Done! Your task is updated and its new version will be triggered by new submissions. You don't need to update the benchmark/competition for the change to take effect.

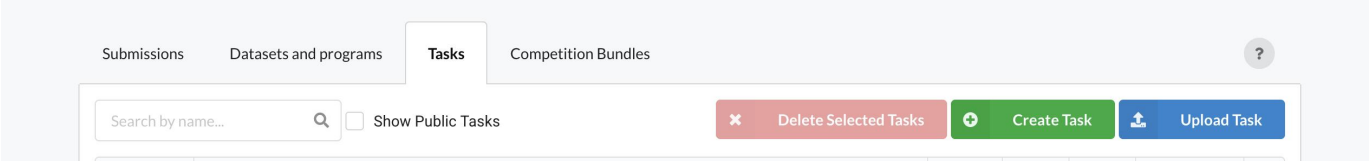
B. Create a new Task

1. PREPARE YOUR NEW DATASET / PROGRAM

First, upload the new versions of your program and/or dataset. To that end, follow steps **1.** and **2.** presented above.

2. CREATE TASK

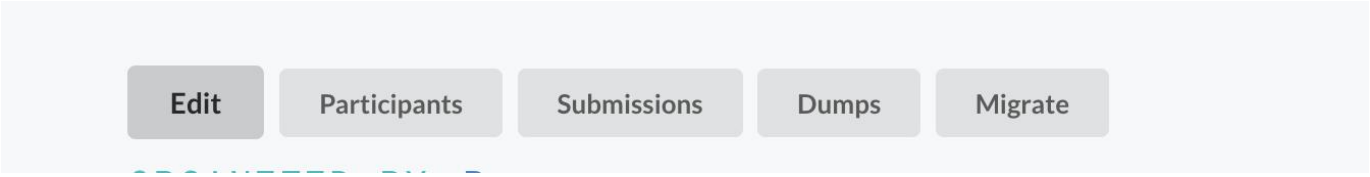
- Go to "Resources" > "Task" > "Create Task"



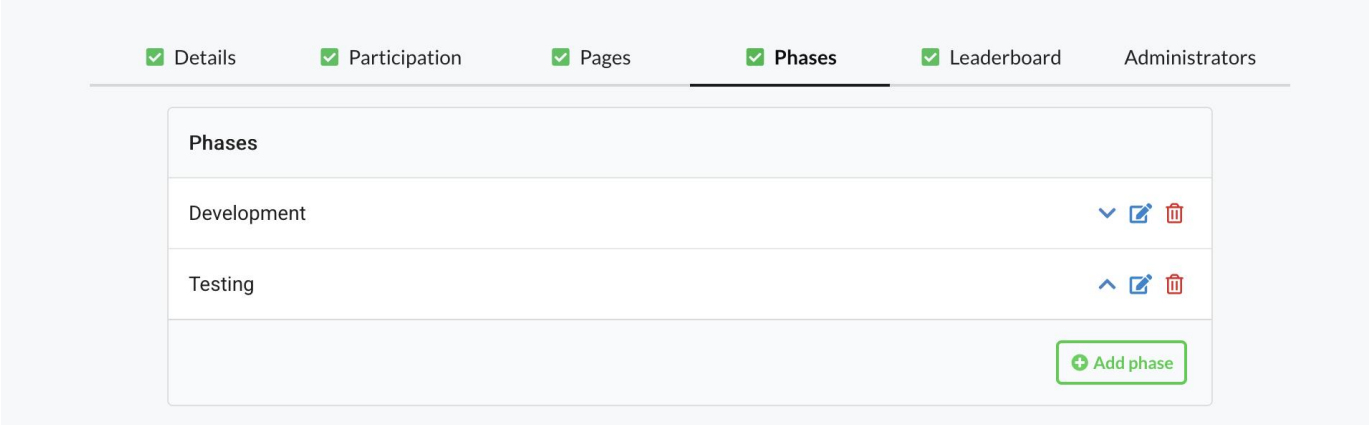
- Fill in all fields: Name, Description, Scoring program, (optionally: Ingestion program, Reference data, Input data)

EDIT YOUR BENCHMARK

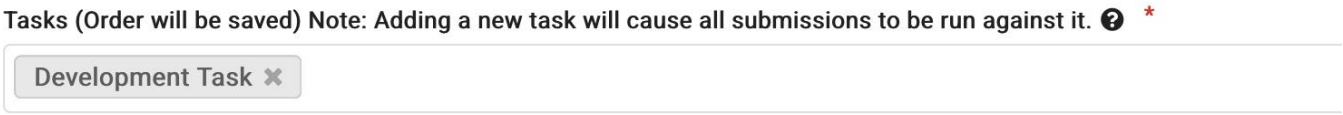
- Once your task is created, go to the editor of your challenge



• Go to "Phases" and edit the relevant phase



• Select your new task and save



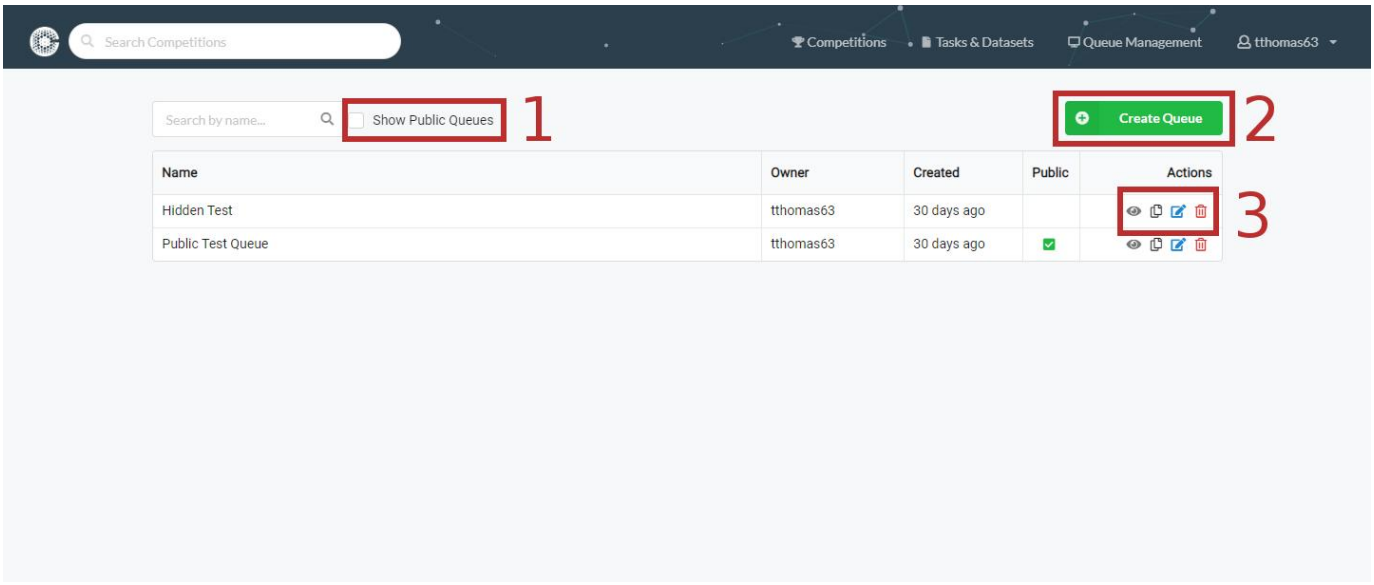
Done! Your benchmark is now ready to run your new task for future submissions.

3.2.5 Queue Management

The queue management page lists all queues you have access to, and optionally all current public queues.

For queues you've created, it will also show options for editing, deleting, copying and displaying the broker URL.

You can also create new queues from this page. You can use the server status page to have an overview of the submissions made to a queue you own.



1) Show Public Queues
2) Create Queue
3) Action Buttons

- Eye Icon
- Document Icon
- Edit Icon
- Trash Icon

Show Public Queues

Enabling this checkbox will display public queues as well as queues you organized, or have been given access to. You will not be able to edit them, but you can view queue details and copy the broker URL.

Create Queue

Clicking this button will bring up a modal with the queue form.

The following fields are present:

- Name: The name of the queue
- Make Public: If checked, this queue will be available for public use.
- Collaborators: A multi-select field that you can search for users by username or email. These will be people who have access to your queue.

Action Buttons

EYE ICON

Clicking this button will show you details about your queue such as the Broker URL and Vhost name.

DOCUMENT ICON

The document icon is used to copy the broker URL to your clipboard with one-click.

EDIT ICON

The edit icon brings up the queue modal/form for editing the current queue.

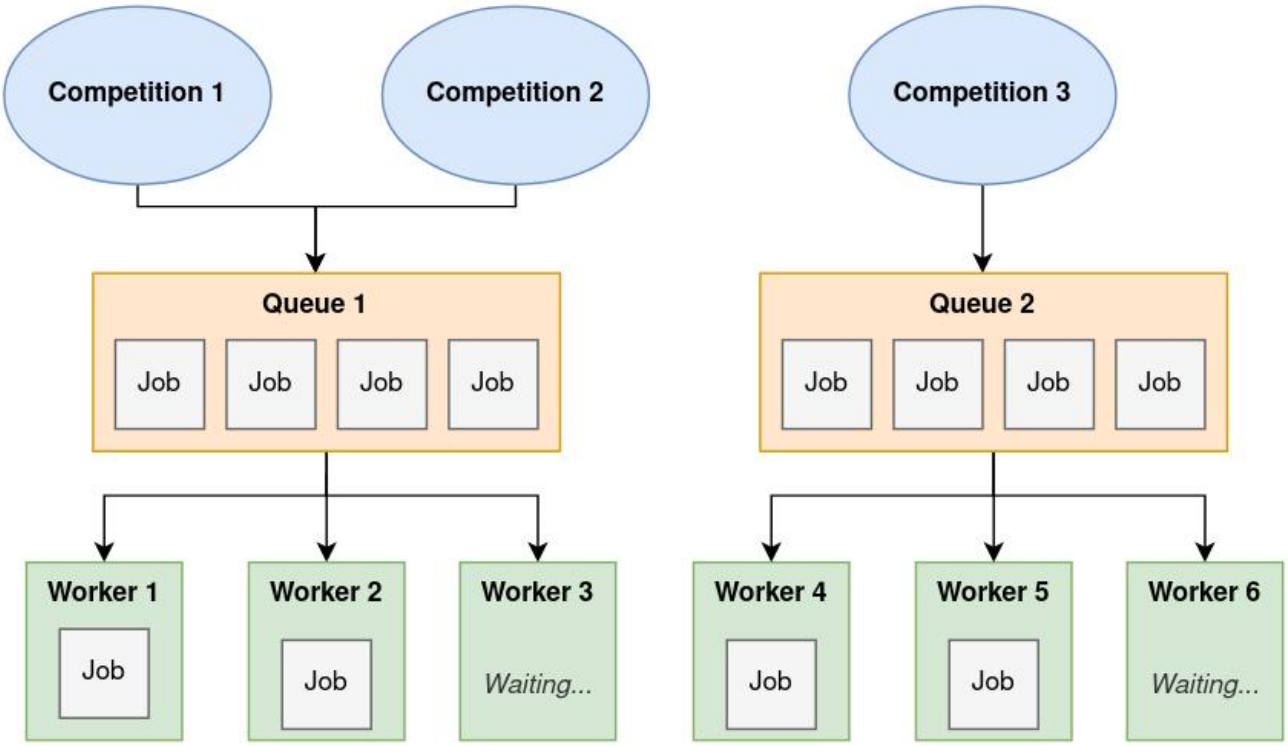
TRASH ICON

The trash icon deletes the current queue. There will be a confirmation dialogue. Once this is done your queue is gone forever so be careful.

Compute workers setup

See compute worker management and setup for more information about workers configuration.

Internal and external compute workers can be linked to Codabench competitions. The queues dispatch the jobs between the compute workers. Note that a queue can receive jobs (submissions) from several competitions, and can send them to several compute workers. The general architecture of queues and workers can be represented like this:


Competition 1
Competition 2
Competition 3
Queue 1
Job
Job
Job
Job
Queue 2
Job
Job
Job
Job
Worker 1
Job
Worker 2
Job
Worker 3
Waiting...
Worker 4
Job
Worker 5
Job
Worker 6
Waiting...

3.2.6 Compute Worker Management & Setup

Compute workers are simply machines that are able to accept/send celery messages on the port used by the broker URL you wish to connect to that have a compute worker image, or other software to receive submissions. This means that you can add computing power to your competitions or benchmarks if needed! Any computer, from your own physical machines to virtual machines on cloud computing services can be used for this purpose. You can add multiple workers to a queue to process several submissions simultaneously.

> **To use Podman, go to the Podman documentation.**

To use Docker, follow these instructions below:

Steps:

- Have a machine (*either physical or virtual, 100 GB storage recommended*)
- Install Docker
- Pull Compute Worker Image
- Run the compute worker via Docker

Install Docker

Either:

a) Install docker via the installation script: https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-using-the-convenience-script

```
curl https://get.docker.com | sudo sh
sudo usermod -aG docker $USER
```

b) Install manually, following the steps at: https://docs.docker.com/install/

Pull Compute Worker Image

On the compute worker machine, run the following command in a shell:

```
docker pull codalab/competitions-v2-compute-worker
```

That will pull the latest image for the v2 worker. For specific versions, see the docker hub page at: https://hub.docker.com/r/codalab/competitions-v2-compute-worker/tags

Start CPU worker

Make a file `.env` and put this in it:

.env

```
# Queue URL
BROKER_URL=<desired broker URL>

# Location to store submissions/cache -- absolute path!
HOST_DIRECTORY=/codabench

# If SSL isn't enabled, then comment or remove the following line
BROKER_USE_SSL=True
```

Note

- The broker URL is a unique identifier of the job queue that the worker should listen to. To create a queue or obtain the broker URL of an existing queue, you can refer to Queue Management wiki page.
- `/codabench` -- this path needs to be volumed into `/codabench` on the worker, as you can see below. You can select another location if convenient.

Create a `docker-compose.yml` file and paste the following content in it:

docker-compose.yml

```
# Codabench Worker
services:
    worker:
        image: codalab/competitions-v2-compute-worker:latest
        container_name: compute_worker
        volumes:
            - /codabench:/codabench
            - /var/run/docker.sock:/var/run/docker.sock
        env_file:
            - .env
        restart: unless-stopped
        #hostname: ${HOSTNAME}
        logging:
            options:
                max-size: 50m
                max-file: 3
```

Note

`hostname: ${HOSTNAME}` allows you to set the hostname of the compute worker container, which will then be shown in the server status page on Codabench. This can be set to anything you want, by setting the `HOSTNAME` environment variable on the machine hosting the Compute Worker, then uncommenting the line the `docker-compose.yml` before launching the compute worker.

You can then launch the worker by running this command in the terminal where the `docker-compose.yml` file is located:

```
docker compose up -d
```

DEPRECATED METHOD (ONE LINER)

Alternately, you can use the docker run below:

```
docker run \
    -v /codabench:/codabench \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -d \
    --env-file .env \
    --name compute_worker \
    --restart unless-stopped \
    --log-opt max-size=50m \
    --log-opt max-file=3 \
    codalab/competitions-v2-compute-worker:latest
```

Start GPU worker

Make a `.env` file, as explained in CPU worker instructions.

Then, install the NVIDIA toolkit: Nvidia toolkit installation instructions

Once you install and configure the NVIDIA container toolkit, you can create a `docker-compose.yml` file with the following content:

docker-compose.yml

```
# Codabench GPU worker (NVIDIA)
services:
    worker:
        image: codalab/competitions-v2-compute-worker:gpu1.3
        container_name: compute_worker
        volumes:
            - /codabench:/codabench
```

```
            - /var/run/docker.sock:/var/run/docker.sock
        env_file:
            - .env
        restart: unless-stopped
        #hostname: ${HOSTNAME}
        logging:
            options:
                max-size: 50m
                max-file: 3
        runtime: nvidia
        deploy:
            resources:
                reservations:
                    devices:
                        - driver: nvidia
                          count: all
                          capabilities:
                              - gpu
```

> **Note**
>
> `hostname: ${HOSTNAME}` allows you to set the hostname of the compute worker container, which will then be shown in the server status page on Codabench. This can be set to anything you want, by setting the `HOSTNAME` environment variable on the machine hosting the Compute Worker, then uncommenting the line the `docker-compose.yml` before launching the compute worker.

You can then launch the worker by running this command in the terminal where the `docker-compose.yml` file is located:

```
docker compose up -d
```

NVIDIA-DOCKER WRAPPER (DEPRECATED METHOD)

Nvidia installation instructions

```
nvidia-docker run \
    -v /codabench:/codabench \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -v /var/lib/nvidia-docker/nvidia-docker.sock:/var/lib/nvidia-docker/nvidia-docker.sock \
    -d \
    --env-file .env \
    --name compute_worker \
    --restart unless-stopped \
    --log-opt max-size=50m \
    --log-opt max-file=3 \
    codalab/competitions-v2-compute-worker:gpu
```

Note that a competition docker image including CUDA and other GPU libraries, such as `codalab/codalab-legacy:gpu`, is then required.

Check logs

Use the following command to check logs and ensure everything is working fine:

```
docker logs -f compute_worker
```

Cleaning up periodically

It is recommended to clean up docker images and containers regularly to avoid filling up the storage.

1. Run the following command:

```
sudo crontab -e
```

1. Add the following line:

```
@daily docker system prune -af
```

Keep track of the worker

It is recommended to store the docker container hostname to identify the worker. This way, it is easier to troubleshoot issues when having multiple workers in one queue. To get the hostname, simply run `docker ps` and look at the key `CONTAINER ID` at the beginning of the output:

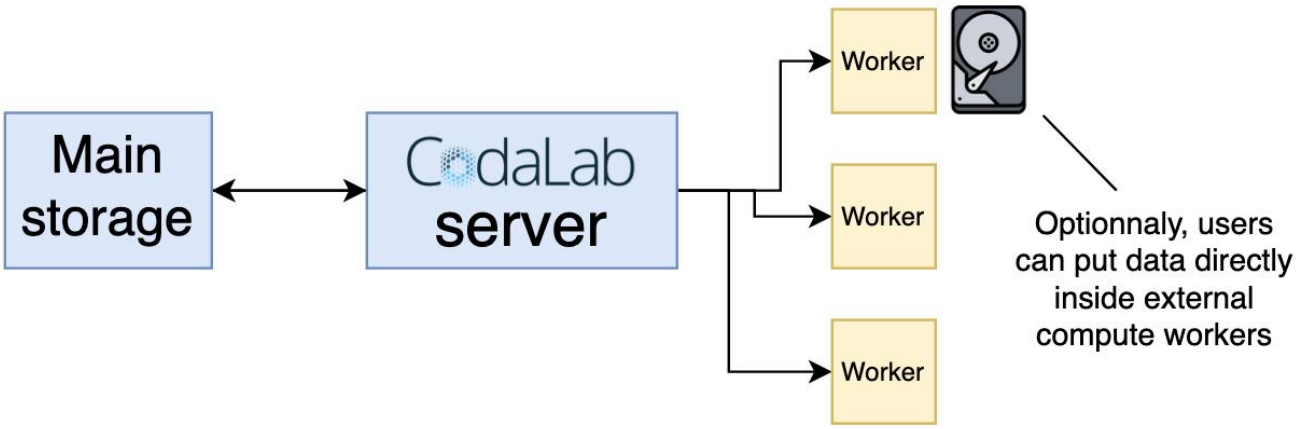
```
$ docker ps
CONTAINER ID   IMAGE                                          COMMAND                  CREATED      STATUS      PORTS     NAMES
1a2b3d4e5f67   codalab/competitions-v2-compute-worker:latest   "/bin/sh -c 'celery …"   3 days ago   Up 3 days             compute_worker
```

For each submission made to your queue, you can know what worker computed the ingestion and the scoring jobs in the server status page.

Optional: put data directly inside the compute worker

The folder `$HOST_DIRECTORY/data`, usually `/codabench/data`, is shared between the host (the compute worker) and the container running the submission (a new container is created for each submission). It is mounted inside the container as `/app/data`. This means that you can put data in your worker, in `$HOST_DIRECTORY/data`, so it can be read-only accessed during the job's process. You'll need to modify the scoring and/or ingestion programs accordingly, to points to `/app/data`. This is especially useful if you work with confidential data, or with a heavy dataset.

If you have several workers in your queue, remember to have the data accessible for each one.



If you simply wish to set up some compute workers to increase the computing power of your benchmark, you don't need to scroll this page any further.

Building compute worker

This is helpful only if you want to build the compute worker image. It is not needed if you simply want to set up compute workers to run submissions.

To build the normal image:

```
docker build -t codalab/competitions-v2-compute-worker:latest -f Dockerfile.compute_worker .
```

To build the GPU version:

```
docker build -t codalab/competitions-v2-compute-worker:gpu -f Dockerfile.compute_worker_gpu .
```

To update the image (add tag `:latest`, `:gpu` or else if needed)

```
docker push codalab/competitions-v2-compute-worker
```

If you have running compute workers, you'll need to pull again the image and to restart the workers to take into account the changes.

Worker management

Outside of docker containers install Fabric like so:

```
pip install fab-classic==1.17.0
```

Create a `server_config.yaml` in the root of this repository using:

```
cp server_config_sample.yaml server_config.yaml
```

Below is an example `server_config.yaml` that defines 2 roles `comp-gpu` and `comp-cpu`, one with GPU style workers (`is_gpu` and the GPU `docker_image`) and one with CPU style workers

server_config.yaml

```
comp-gpu:
  hosts:
    - ubuntu@12.34.56.78
    - ubuntu@12.34.56.79
  broker_url: pyamqp://user:pass@host:port/vhost-gpu
  is_gpu: true
  docker_image: codalab/competitions-v2-compute-worker:gpu

comp-cpu:
  hosts:
    - ubuntu@12.34.56.80
  broker_url: pyamqp://user:pass@host:port/vhost-cpu
  is_gpu: false
  docker_image: codalab/competitions-v2-compute-worker:latest
```

You can of course create your own `docker_image` and specify it here.

You can execute commands against a role:

```
fab -R comp-gpu status
..
[ubuntu@12.34.56.78] out: CONTAINER ID        IMAGE                                          COMMAND                  CREATED             STATUS
 PORTS               NAMES
[ubuntu@12.34.56.78] out: 1d318268bee1        codalab/competitions-v2-compute-worker:gpu   "/bin/sh -c 'celery …"   2 hours ago         Up 2
hours                               hardcore_greider
..

fab -R comp-gpu update
..
(updates workers)
```

See available commands with `fab -l`

Update docker image

If the compute worker docker image was updated, you can reflect the changes using the following commands.

Check no job is running:

```
docker ps
```

Update the worker:

```
docker stop compute_worker
docker rm compute_worker
docker pull codalab/competitions-v2-compute-worker:latest    # or other relevant docker image
docker run \                                                 # or docker compose up -d
    -v /codabench:/codabench \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -d \
    --env-file .env \
    --name compute_worker \
    --restart unless-stopped \
    --log-opt max-size=50m \
    --log-opt max-file=3 \
    codalab/competitions-v2-compute-worker:latest            # or other relevant docker image
```

3.2.7 Compute Worker Management with Podman

Here is the specification for compute worker installation by using Podman.

Requirements for the host machine

We need to install Podman on the VM. We use Debian based OS, like Ubuntu. Ubuntu is recommended, because it has better Nvidia driver support.

`sudo apt install podman`

After installing Podman, you will need to launch the service associated to it with `systemctl --user enable --now podman`

Then, configure where Podman will download the images: Podman will use Dockerhub by adding this line into `/etc/containers/registries.conf` :

`unqualified-search-registries = ["docker.io"]`

Create the `.env` file in order to add the compute worker into a queue (here, the default queue is used. If you use a particular queue, then, fill in your BROKER_URL generated when creating this particular queue) :

.env

```
BROKER_URL=pyamqp://<login>:<password>@codabench-test.lri.fr:5672
HOST_DIRECTORY=/codabench
# If SSL isn't enabled, then comment or remove the following line
BROKER_USE_SSL=True
CONTAINER_ENGINE_EXECUTABLE=podman
```

You will also need to create the `codabench` folder defined in the `.env` file, as well as change its permissions to the user that is running the compute worker.

In your terminal

```
sudo mkdir /codabench
sudo mkdir /codabench/data
sudo chown -R $(id -u):$(id -g) /codabench
```

You should also run the following command if you don't want the container to be shutdown when you log out of the user:

```
sudo loginctl enable-linger *username*
```

Make sure to use the username of the user running the podman container.

For GPU compute worker VM

You need to install nvidia packages supporting Podman and nvidia drivers:

```
distribution=$(. /etc/os-release;echo $ID$VERSION_ID) \
    && curl -s -L https://nvidia.github.io/nvidia-docker/gpgkey | sudo apt-key add - \
    && curl -s -L https://nvidia.github.io/nvidia-docker/$distribution/nvidia-docker.list | sudo tee /etc/apt/sources.list.d/nvidia-container.list
sudo apt update
sudo apt install nvidia-container-runtime nvidia-containe-toolkit nvidia-driver-<version>
```

Edit the nvidia runtime config

```
sudo sed -i 's/^#no-cgroups = false/no-cgroups = true/;' /etc/nvidia-container-runtime/config.toml
```

Check if nvidia driver is working, by executing:

```
nvidia-smi

+-----------------------------------------------------------------------------+
| NVIDIA-SMI 520.61.05    Driver Version: 520.61.05    CUDA Version: 11.8     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce ...  On   | 00000000:01:00.0 Off |                  N/A |
| 27%   26C    P8    20W / 250W |      1MiB / 11264MiB |      0%      Default |
```

```
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

The result should show gpu card information.

We need to configure the OCI hook (entry point to inject code) script for nvidia. Create this file `/usr/share/containers/oci/hooks.d/oci-nvidia-hook.json` if not exists:

oci-nvidia-hook.json

```
{
    "version": "1.0.0",
    "hook": {
        "path": "/usr/bin/nvidia-container-toolkit",
        "args": ["nvidia-container-toolkit", "prestart"],
        "env": [
            "PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin"
        ]
    },
    "when": {
        "always": true,
        "commands": [".*"]
    },
    "stages": ["prestart"]
}
```

Validating if all are working by running a test container:

```
podman run --rm -it \
 --security-opt="label=disable" \
 --hooks-dir=/usr/share/containers/oci/hooks.d/ \
 nvidia/cuda:11.6.2-base-ubuntu20.04 nvidia-smi
```

The result should show as same as the command `nvidia-smi` above.

You will also need to add this line in your `.env` file:

```
USE_GPU=True
```

Compute worker installation

FOR CPU CONTAINER

Run the compute worker container :

```
podman run -d \
 --volume /run/user/$(id -u)/podman/podman.sock:/run/user/1000/podman/podman.sock:U \
 --env-file .env \
 --name compute_worker \
 --security-opt="label=disable" \
 --userns host \
 --restart unless-stopped \
 --log-opt max-size=50m \
 --log-opt max-file=3 \
 --cap-drop all \
 --volume /codabench:/codabench:U,z \
 codalab/codabench_worker_podman:latest
```

FOR GPU CONTAINER

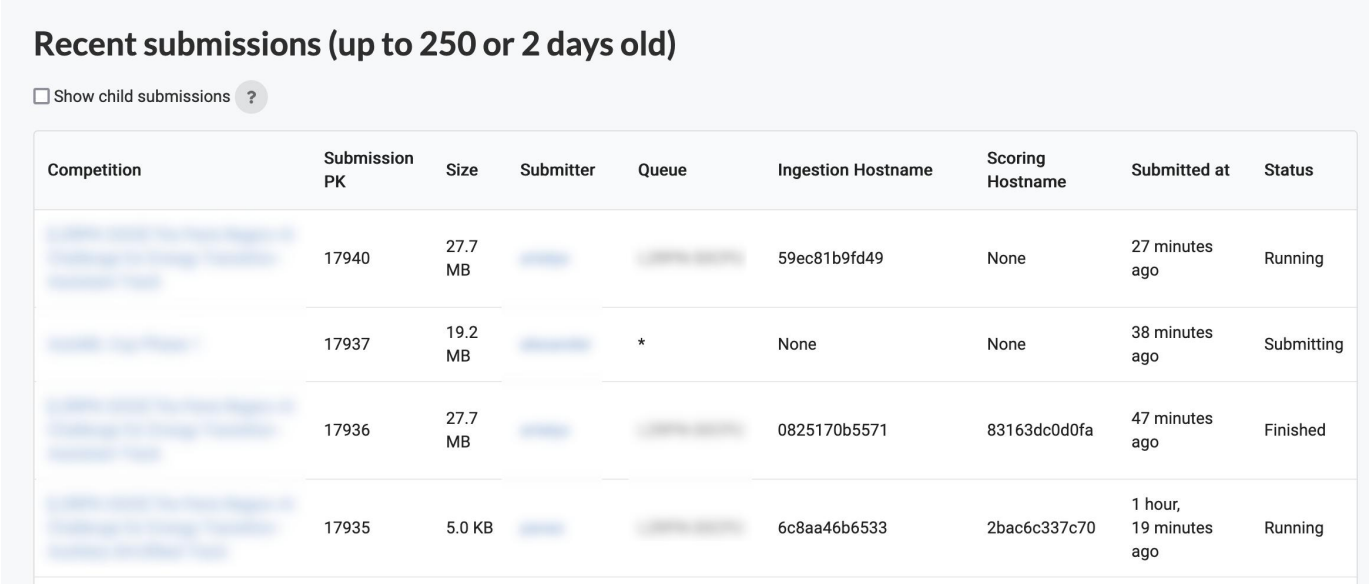
Run the GPU compute worker container

```
podman run -d \
    --env-file .env \
    --device nvidia.com/gpu=all \
    --name gpu_compute_worker \
    --device /dev/fuse \
    --security-opt="label=disable" \
    --restart unless-stopped \
    --log-opt max-size=50m \
    --log-opt max-file=3 \
    --hostname ${HOSTNAME} \
```

```
    --userns host \
    --volume /home/codalab/worker/codabench:/codabench:z,U \
    --cap-drop=all \
    --volume /run/user/$(id -u)/podman/podman.sock:/run/user/1000/podman/podman.sock:U \
    codalab/codabench_worker_podman_gpu:latest
```

3.2.8 Server Status

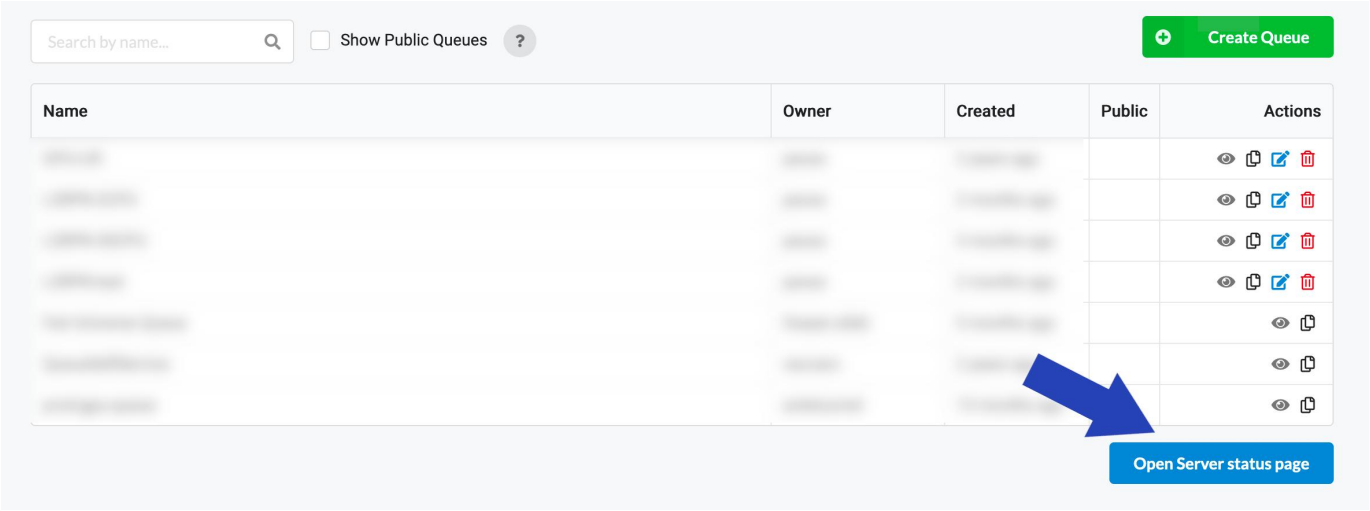
The server status page gives information about past and current submissions, and is useful for troubleshooting. Any user can access the interface, and get information about their own submissions and the submissions made to queues they own. Administrators can see all submissions. Here is an overview of the page:



Note that `*` refers to the default queue of the platform. "Hostname" refers to the docker container ID of the compute worker that computed the job.

How to access the interface

You can access it either from the top right menu, or from the "Queue management" page, as shown in the screenshots below.


Search by name...
Show Public Queues
?
Create Queue
Name
Owner
Created
Public
Actions
Open Server status page

4. Developers and Administrators

4.1 Codabench Basic Installation Guide

Compared to Codalab, installing Codabench should be relatively easy since you no longer have to worry about special ways to set up SSL or storage. We include default solutions that should handle that for most basic uses.

4.1.1 Pre-requisites

Install Docker and Docker Compose

- Docker
- Docker Compose

4.1.2 Clone Repository

Download the Codabench repository:

```
git clone https://github.com/codalab/codabench
```

4.1.3 Edit the settings (.env)

The `.env` file contains the settings of your instance. On a fresh installation, you will need to use the following command to get your `.env` file:

```
cd codabench
cp .env_sample .env
```

Then edit the necessary settings inside. The most important are the database, storage, and Caddy/SSL settings. For a quick **local** setup, you should not need to edit this file. For a public server deployment, you will have to modify some settings.

> ⚠ **It is important to change the default passwords if you intend for the instance to be public**

If you are using `AWS_S3_ENDPOINT_URL=http://minio:9000/` in your `.env`, edit your `/etc/hosts` file by adding this line `127.0.0.1 minio`

FOR MACOS

In `.env`, replace:

```
AWS_S3_ENDPOINT_URL=http://minio:9000/
```

by

```
AWS_S3_ENDPOINT_URL=http://docker.for.mac.localhost:9000/
```

> ✎ **If needed, some troubleshooting of this step is provided at the end of this page or in this page**

4.1.4 Start the service

To deploy the platform, run:

```
docker compose up -d
```

4.1.5 Run the following commands

Create the required tables in the database:

```
docker compose exec django ./manage.py migrate
```

Generate the required static resource files:

```
docker compose exec django ./manage.py collectstatic --noinput
```

You should be able to verify it is running correctly by looking at the logs with `docker compose logs -f` and by visiting `localhost:80` (Depending on your configuration).

4.1.6 Advanced Configuration

Testing

To run automated tests for your local instance, get inside the Django container with `docker compose exec django bash` then run `py.test` to start the automated tests.

SSL

To enable SSL:

- If you already have a DNS for your server that is appropriate, in the `.env` simply set `DOMAIN_NAME` to your DNS. Remove any port designation like `:80`. This will have Caddy serve both HTTP and HTTPS.

> ⚠ **For a public instance, HTTPS is strongly recomended**

Validate user account on local instance

When deploying a local instance, the email server is not configured by default, so you won't receive the confirmation email during signup.

To manually confirm your account:

1. Find the confirmation link in the Django logs using `docker compose logs -f django`
2. Replace `example.com` by `localhost` on the URL and open it in your browser.

Another way is to go inside the Django containers and use commands like in administrative procedures.

Troubleshooting storage endpoint URL

You may have to manually change the endpoint URL to have your local instance working. This may be an OS related issue. Here is a possible fix:

1. `docker compose logs -f minio`
2. Grab the first one of these IP addresses:

```
minio_1          | Browser Access:
minio_1          |    http://172.27.0.5:9000  http://127.0.0.1:9000
```

3. Set `AWS_S3_ENDPOINT_URL=http://172.27.0.5:9000` in your `.env` file.

If static files are not loaded correctly, adding `DEBUG=True` to the `.env` file can help.

For Apple CPU (M1, M2 chips)

In `docker-compose.yml`, replace in the `compute_worker` service:

docker-compose.yml

```
command: bash -c "watchmedo auto-restart -p '*.py' --recursive -- celery -A compute_worker worker -l info -Q compute-worker -n compute-worker@%n"
```

by

docker-compose.yml

```
command: bash -c "celery -A compute_worker worker -l info -Q compute-worker -n compute-worker@%n"
```

Storage

By default, Codabench uses a built-in MinIO container. Some users may want a different solution, such as S3 or Azure. The configuration will vary slightly for each different type of storage.

For all possible supported storage solutions, see: https://django-storages.readthedocs.io/en/latest/

Remote Compute Workers

To set up remote compute workers, you can follow the steps described in our Compute Worker Management page.

4.1.7 Troubleshooting

Read the following guide for troubleshooting: How to deploy Codabench.

Also, adding `DEBUG=True` to the `.env` file can help with troubleshooting the deployment.

Open a Github issue to find help with your installation

4.1.8 Online Deployement

For information about online deployment of Codabench, go to the following page

4.2 How to Deploy a Server

4.2.1 Overview

This document focuses on how to deploy the current project to the local machine or server you are on.

4.2.2 Preliminary steps

As for the minimal local installation, you first need to:

1. Install docker and docker-compose (see instructions)
2. Clone Codabench repository:

```
git clone https://github.com/codalab/codabench
```

4.2.3 Modify .env file configuration

Then you need to modify the `.env` file with the relevant settings. This step is critical to have a working and secure deployment.

- Go to the folder where codabench is located (`cd codabench`)

```
cp .env_sample .env
```

Then edit the variables inside the `.env` file.

Submissions endpoint

For an online deployment, you'll need to fill in the IP address or domain name in some environment variables.

USING AN IP ADDRESS

b) For an online deployment using IP address:

> **Note**
>
> To get the IP address of the machine. You can use one of the following commands:
>
> - `ifconfig -a`
> - `ip addr`
> - `ip a`
> - `hostname -I | awk '{print $1}'`
> - `nmcli -p device show`

Replace the value of IP address in the following environment variables according to your infrastructure configuration:

.env

```
SUBMISSIONS_API_URL=https://<IP ADDRESS>/api
DOMAIN_NAME=<IP ADDRESS>:80
AWS_S3_ENDPOINT_URL=http://<IP ADDRESS>/
```

USING A DOMAIN NAME (DNS)

.env

```
SUBMISSIONS_API_URL=https://yourdomain.com/api
DOMAIN_NAME=yourdomain.com
AWS_S3_ENDPOINT_URL=https://yourdomain.com
```

If you are deploying on an azure machine, then AWS_S3_ENDPOINT_URL needs to be set to an IP address that is accessible on the external network

Change default usernames and passwords

Set up new usernames and passwords:

```
DB_USERNAME=postgres
DB_PASSWORD=postgres
[...]
RABBITMQ_DEFAULT_USER=rabbit-username
RABBITMQ_DEFAULT_PASS=rabbit-password-you-should-change
[...]
FLOWER_BASIC_AUTH=root:password-you-should-change
[...]
#EMAIL_HOST_USER=user
#EMAIL_HOST_PASSWORD=pass
[...]
MINIO_ACCESS_KEY=testkey
MINIO_SECRET_KEY=testsecret
# or
AWS_ACCESS_KEY_ID=testkey
AWS_SECRET_ACCESS_KEY=testsecret
```

It is very important to set up an SSL certificate for Public deployement

4.2.4 Open Access Permissions for following port number

If you are deploying on a Linux server, which usually has a firewall, you need to open access permissions to the following port numbers

- `5672` : rabbit mq port
- `8000` : django port
- `9000` : minio port

4.2.5 Modify django-related configuration

- Go to the folder where codabench is located
- Go to the settings directory and modify `base.py` file
    - `cd src/settings/`
    - `nano base.py`
- Change the value of `DEBUG` to `True`
    - `DEBUG = os.environ.get("DEBUG", True)`

If DEBUG is not set to true, then you will not be able to load to the static resource file

- Comment out the following code

```
# ==============================================================================
# Debug
# ==============================================================================
#if DEBUG:
#    INSTALLED_APPS += ('debug_toolbar',)
#    MIDDLEWARE = ('debug_toolbar.middleware.DebugToolbarMiddleware',
#                  'querycount.middleware.QueryCountMiddleware',
#                  ) + MIDDLEWARE  # we want Debug Middleware at the top
#    # tricks to have debug toolbar when developing with docker
#
#    INTERNAL_IPS = ['127.0.0.1']
#
#    import socket
#
#    try:
#        INTERNAL_IPS.append(socket.gethostbyname(socket.gethostname())[:-1])
#    except socket.gaierror:
#        pass
#
#    QUERYCOUNT = {
#        'IGNORE_REQUEST_PATTERNS': [
#            r'^/admin/',
#            r'^/static/',
#        ]
#    }
#
#    DEBUG_TOOLBAR_CONFIG = {
#        "SHOW_TOOLBAR_CALLBACK": lambda request: True
#    }
```

4.2.6 Start service

- Execute command `docker compose up -d`
- Check if the service is started properly `docker compose ps`

```
codabench_compute_worker_1   "bash -c 'watchmedo …"   running
codabench_caddy_1            "/bin/parent caddy -…"   running      0.0.0.0:80->80/tcp, :::80->80/tcp, 0.0.0.0:443->443/tcp, :::443->443/tcp, 2015/tcp
codabench_site_worker_1      "bash -c 'watchmedo …"   running
codabench_django_1           "bash -c 'cd /app/sr…"   running      0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
codabench_flower_1           "flower"                 restarting
codabench_rabbit_1           "docker-entrypoint.s…"   running      4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, :::5672->5672/tcp, 15671/tcp, 25672/tcp, 0.
0.0.0:15672->15672/tcp, :::15672->15672/tcp
codabench_minio_1            "/usr/bin/docker-ent…"   running      0.0.0.0:9000->9000/tcp, :::9000->9000/tcp
codabench_db_1               "docker-entrypoint.s…"   running      0.0.0.0:5432->5432/tcp, :::5432->5432/tcp
codabench_builder_1          "docker-entrypoint.s…"   running
codabench_redis_1            "docker-entrypoint.s…"   running      0.0.0.0:6379->6379/tcp, :::6379->6379/tcp
```

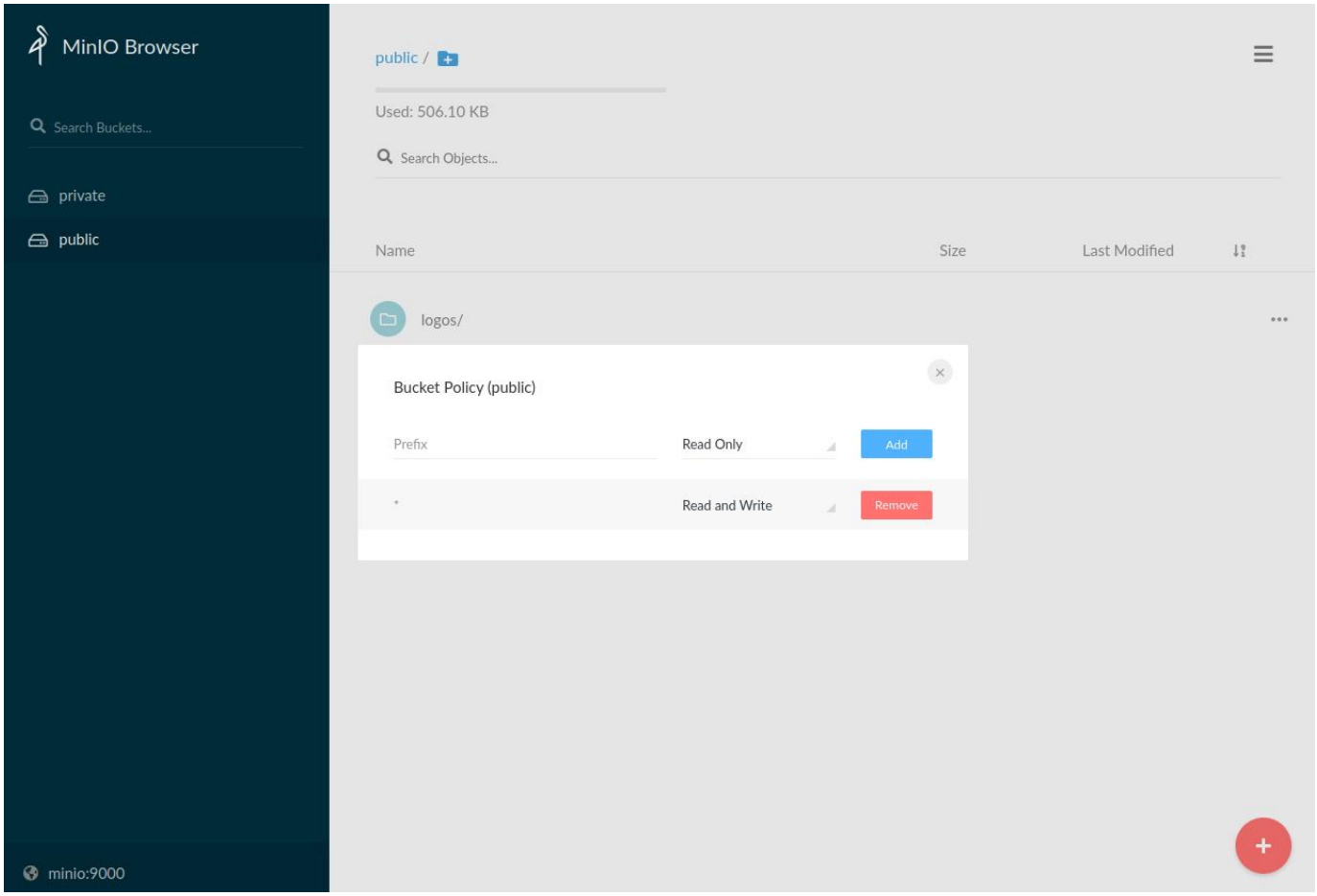
- Create the required tables in the database: `docker compose exec django ./manage.py migrate`
- Generate the required static resource files: `docker compose exec django ./manage.py collectstatic --noinput`

Tip

You can generate mock data with `docker compose exec django ./manage.py generate_data` if you want to test the website. However, it is not recomended to do that on an installation that you intend to use for Production

4.2.7 Set public bucket policy to read/write

This can easily be done via the minio web console (local URL: minio:9000)



4.2.8 Checkout the log of the specified container

The following commands can help you debug

- `docker compose logs -f django` : checkout django container logs in the docker-compose service
- `docker compose logs -f site_worker` : checkout site-worker container logs in the docker-compose service
- `docker compose logs -f compute_worker` : checkout compute-worker container logs in the docker-compose service
- `docker compose logs -f minio` : checkout minio container logs in the docker-compose service

You can also use `docker compose logs -f` to get the logs of all the containers.

4.2.9 Stop service

- Execute command `docker compose down --volumes`

4.2.10 Disabling docker containers on production

To override settings on your production server, create a `docker-compose.override.yml` in the `codabench` root directory. If on your production server, you are using remote MinIO or another cloud storage provider then you don't need **minio** container. If you have already buckets available for your s3 storage, you don't need **createbuckets** container. Therefore, you should disable minio and createbuckets containers. You may also want to disable the compute worker that is contained in the main server compute, to keep only remote compute workers.

Add this to your `docker-compose.override.yml`:

docker-compose.override.yml

```
version: '3.4'
services:
  compute_worker:
    command: "/bin/true"
  minio:
    restart: "no"
    command: "/bin/true"
  createbuckets:
    entrypoint: "/bin/true"
    restart: "no"
    depends_on:
      minio:
        condition: service_started
```

Warning

This will force the following container from exiting on start:

- Compute Worker
- MinIO
- CreateBuckets

If you need one of these then remove the corresponding lines from the file before launching

4.2.11 Link compute workers to default queue

The default queue of the platform runs all jobs, except when a custom queue is specified by the competition or benchmark. By default, the compute worker of the default queue is a docker container run by the main VM. If your server is used by many users and receives several submissions per day, it is recommended to use separate compute workers and to link them to the default queue.

To set up a compute worker, follow this guide

In the `.env` file of the compute worker, the `BROKER_URL` should reflect settings of the `.env` file of the platform:

.env

```
BROKER_URL=pyamqp://<RABBITMQ_DEFAULT_USER>:<RABBITMQ_DEFAULT_PASS>@<DOMAIN_NAME>:<RABBITMQ_PORT>/
HOST_DIRECTORY=/codabench
BROKER_USE_SSL=True
```

4.2.12 Personalize Main Banner

The main banner on the Codabench home page shows 3 organization logos

- LISN
- Université Paris-Saclay
- CNRS

You can update these by:

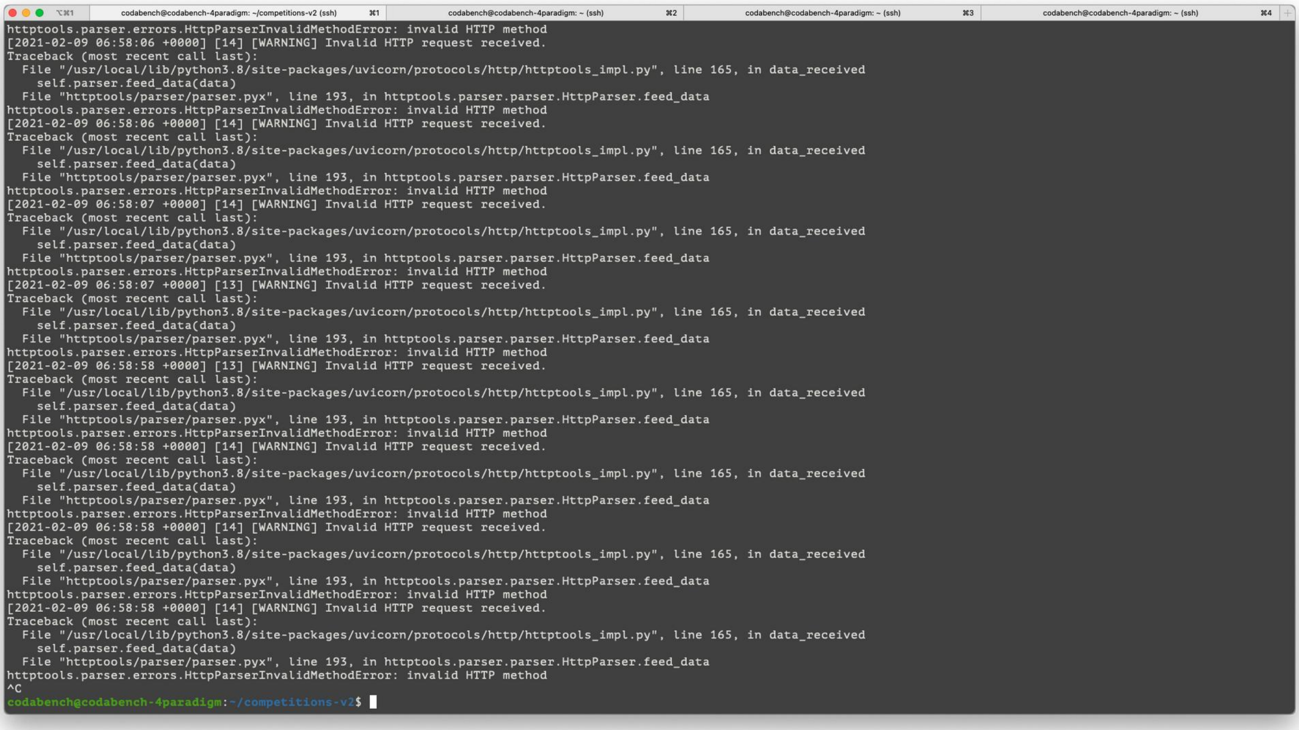
1. Replacing the logos in `src/static/img/` folder
2. Updating the code in `src/templates/pages/home.html` to point to the right websites of your organizations

4.2.13 Frequently asked questions (FAQs)

Invalid HTTP method

Exception detail (by using `docker logs -f codabench_django_1`)

```
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/site-packages/uvicorn/protocols/http/httptools_impl.py", line 165, in data_received
    self.parser.feed_data(data)
  File "httptools/parser/parser.pyx", line 193, in httptools.parser.parser.HttpParser.feed_data
httptools.parser.errors.HttpParserInvalidMethodError: invalid HTTP method
[2021-02-09 06:58:58 +0000] [14] [WARNING] Invalid HTTP request received.
Traceback (most recent call last):
  File "/usr/local/lib/python3.8/site-packages/uvicorn/protocols/http/httptools_impl.py", line 165, in data_received
    self.parser.feed_data(data)
  File "httptools/parser/parser.pyx", line 193, in httptools.parser.parser.HttpParser.feed_data
httptools.parser.errors.HttpParserInvalidMethodError: invalid HTTP method
```



Solution

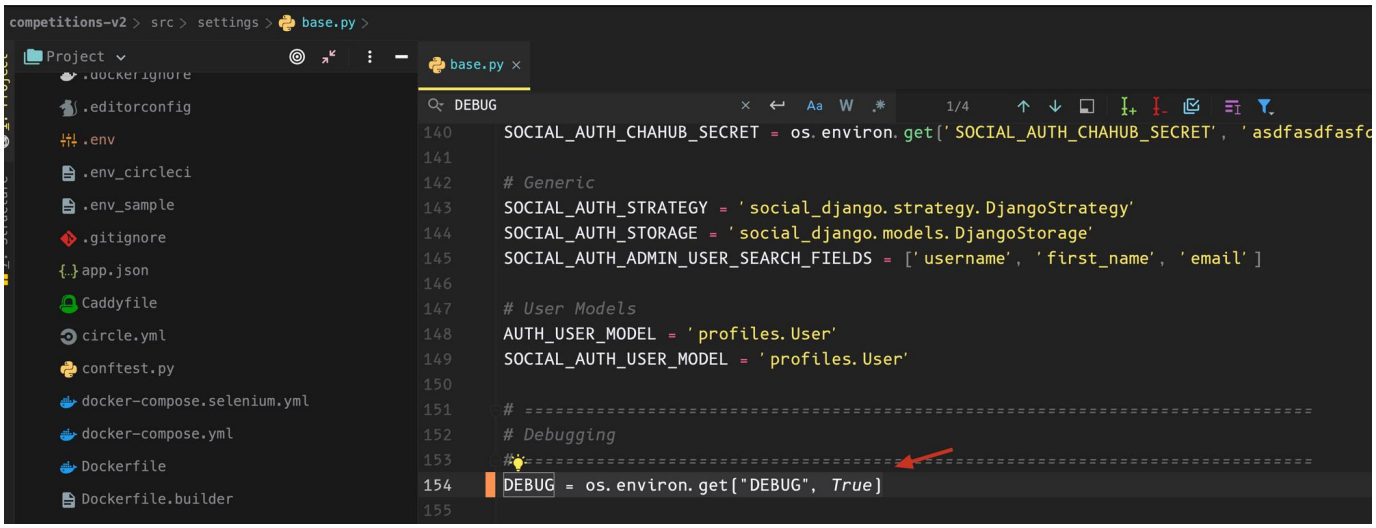
- First, modify the `.env` file and set `DJANGO_SETTINGS_MODULE=settings.develop`
- Then, restart services by using following docker-compose command

```
docker compose down --volumes
docker compose up -d
```

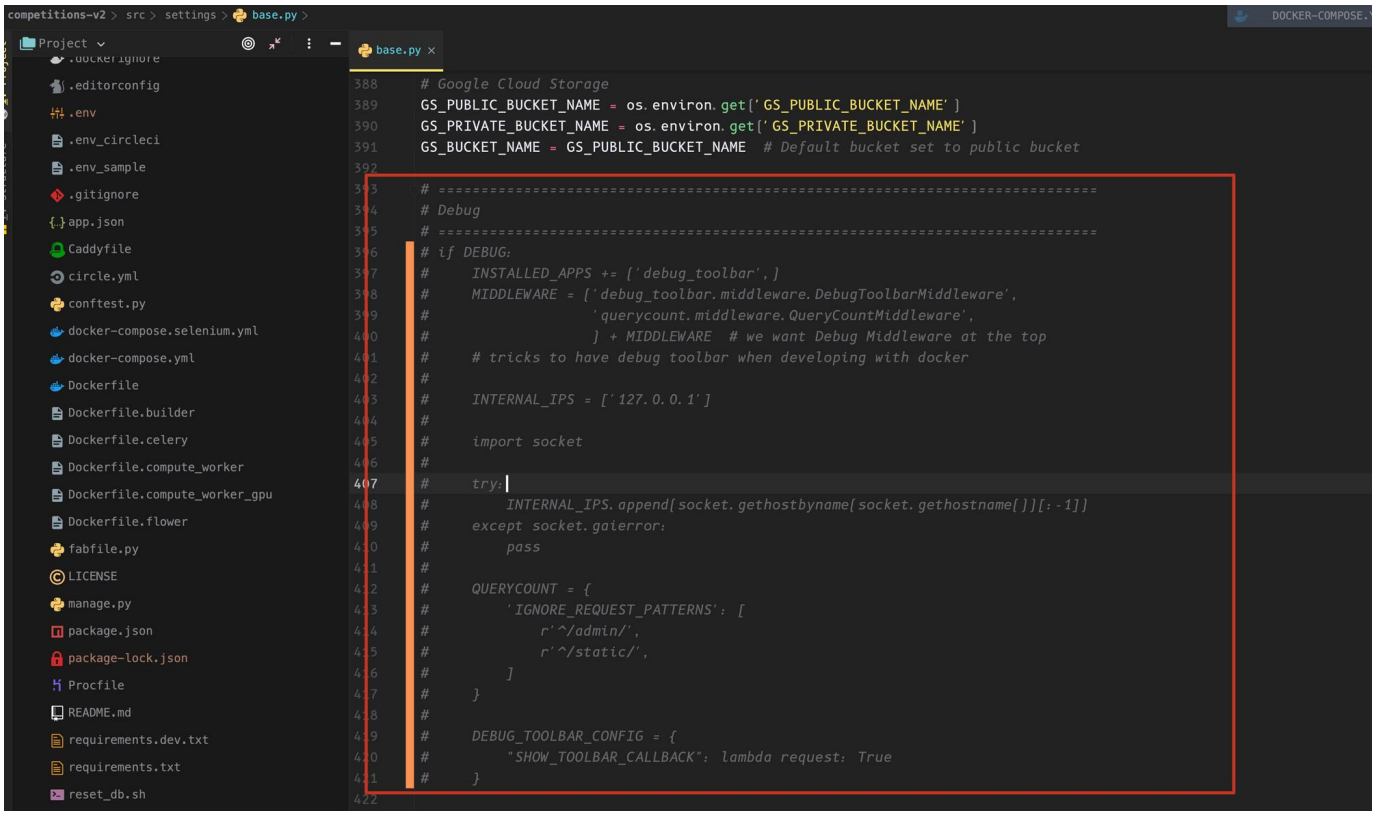
Missing static resources (css/js)

Solution: Change the value of the `DEBUG` parameter to `True`

- `nano competitions-v2/src/settings/base.py`
- `DEBUG = os.environ.get("DEBUG", True)`



- Also comment out the following code in `base.py`



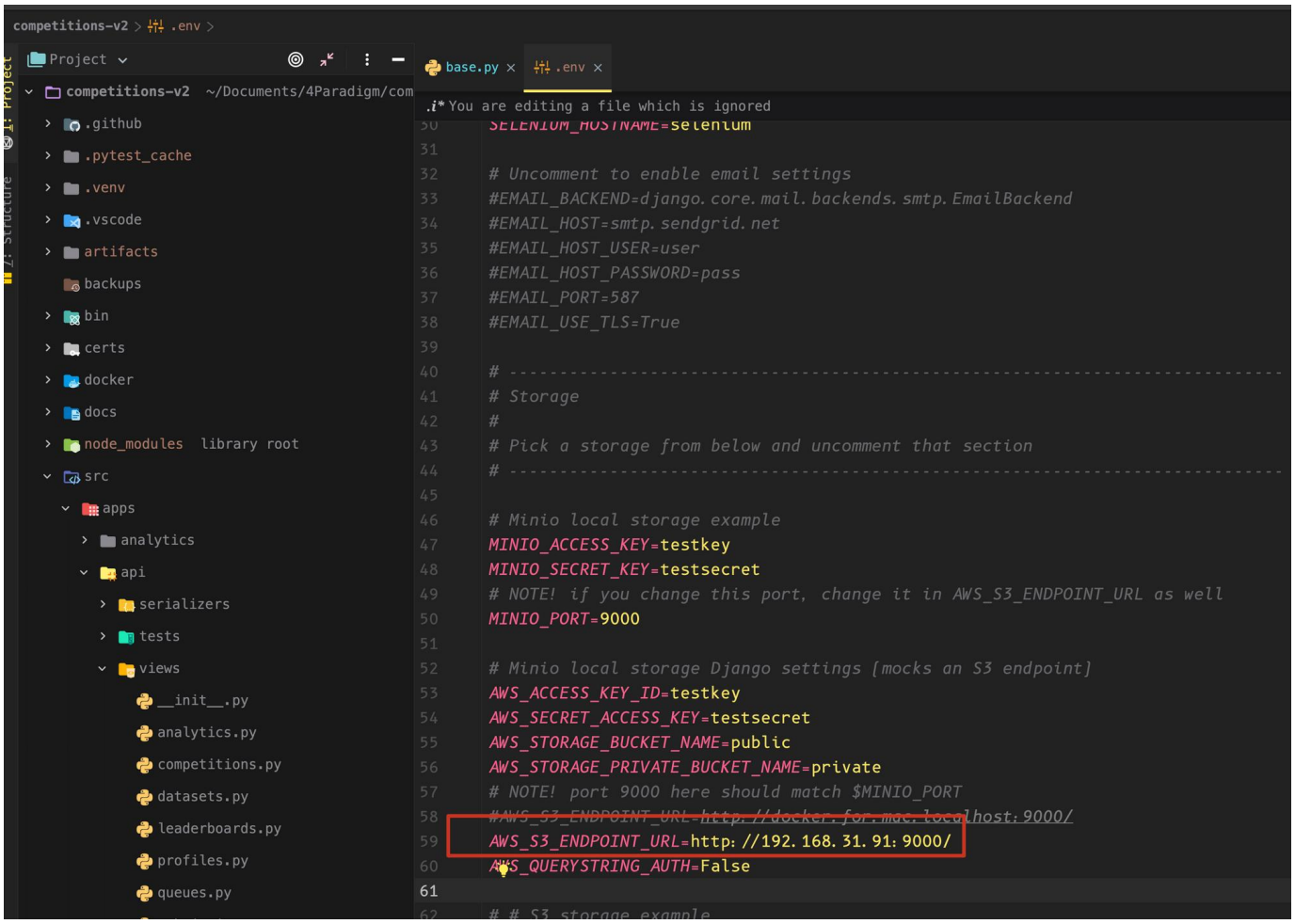
CORS Error (could not upload bundle)

Exception detail (by checkout google develop tools)

```
botocore.exceptions.EndpointConnectionError: Could not connect to the endpoint URL: "[http://docker.for.mac.localhost:9000/private/dataset/
2021-02-18-1613624215/24533cfc523e/competition.zip](http://docker.for.mac.localhost:9000/private/dataset/2021-02-18-1613624215/24533cfc523e/competition.zip)"
```

Solution: Set AWS_S3_ENDPOINT_URL to an address that is accessible to the external network

- `nano codabench/.env`



Make sure the IP address and port number is accessible by external network, You can check this by :

- `telnet {ip-address-filling-in AWS_S3_ENDPOINT_URL} {port-filling-in AWS_S3_ENDPOINT_URL}`
- Make sure the firewall is closed on port 9000

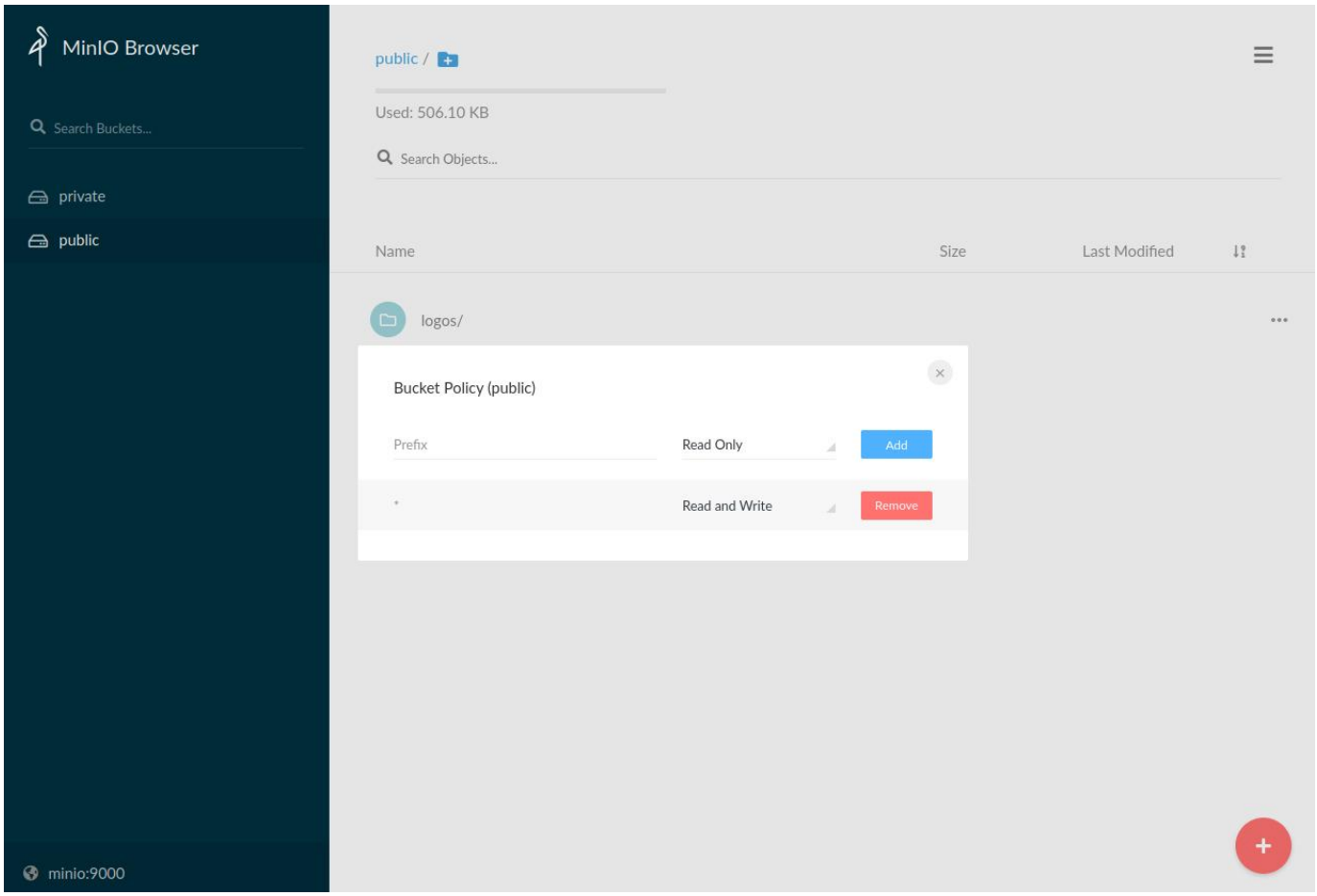
This problem may also be caused by a bug in MinIO, in which case you will need to follow these steps

- Upgrade the minio docker image to the latest version
- Delete the previous minio directory folder in your codabench folder under `/var/minio` directory
- Stop the current minio container
- Delete the current minio container and the corresponding image
- Re-execute `docker compose up -d`

Display logos error: logos don't upload from minio:

Check bucket policy of public minio bucket: read/write access should be allowed.

This can easily be done via the minio web console (local URL: minio:9000)



Compute worker execution with insufficient privileges

This issue may be encountered when starting a docker container in a compute worker, the problem is caused by the installation of snap docker (if you are using Ubuntu).

Solution

- Uninstall snap docker
- Install the official version of docker

4.2.14 Securing Codabench and Minio

Codabench uses Caddy to manage HTTPS and to secure Codabench. What you need is a valid DNS pointed towards the IP address of your instance.

Secure Minio with a reverse proxy

To secure MinIO, you should install a reverse-proxy, e.g: Nginx, and have a valid SSL certificate. Here is a tutorial sample:

Secure MinIO with Certbot and Letsencrypt

Don't forget to update your AWS_S3_ENDPOINT_URL parameter

Update it to `AWS_S3_ENDPOINT_URL=https://<your minio>`

Secure Minio on the same server as codabench (simpler)

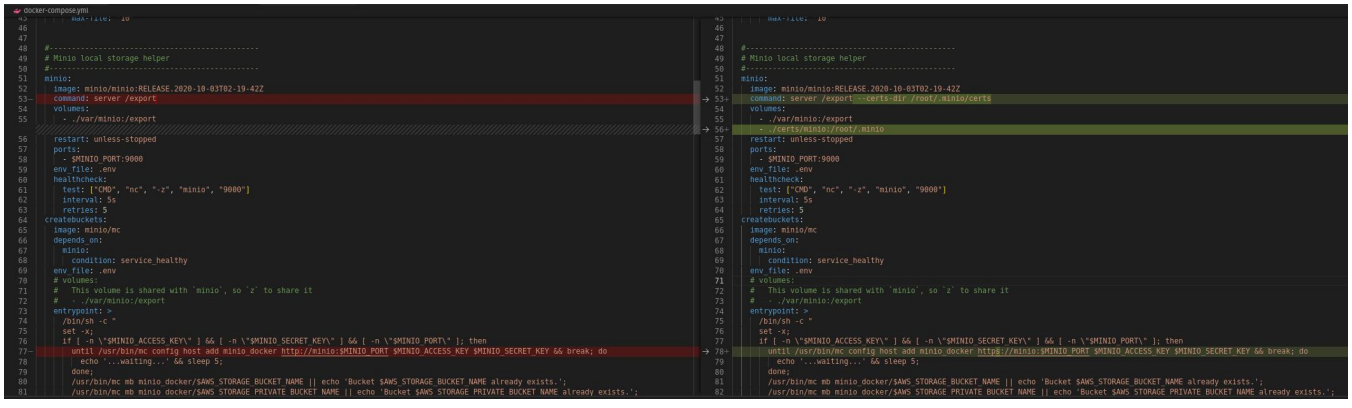
Summary:

- Use same SSL certs from letsencrypt (certbot) but change fullchain.pem -> public.crt and privkey.pem -> private.key. I copied from ./certs/caddy (for django/caddy) to ./certs/minio/certs.
- You need to change the command for minio to "server /export --certs-dir /root/.minio/certs" and not just "server /export"
- Mount in certs:
- Add "- ./certs/minio:/root/.minio" under the minio service's "volumes" section
- Certs must be in /${HOME}/.minio and for dockers ends up being /root/.minio
- Edit the .env with minio cert location:

```
MINIO_CERT_FILE=/root/.minio/certs/public.crt
MINIO_KEY_FILE=/root/.minio/certs/private.key
# MINIO_CERTS_DIR=/certs/caddy # was told .pem files could work but for now separating
MINIO_CERTS_DIR=/root/.minio/certs # either this or the CERT\KEY above is redundant...but it works for now.
# NOTE! if you change this port, change it in AWS_S3_ENDPOINT_URL as well
MINIO_PORT=9000
```

- Here is an example docker-compose.yml change for this:

```
  #-------------------------------------------------
  # Minio local storage helper
  #-------------------------------------------------
  minio:
    image: minio/minio:RELEASE.2020-10-03T02-19-42Z
    command: server /export --certs-dir /root/.minio/certs
    volumes:
      - ./var/minio:/export
      - ./certs/minio:/root/.minio
    restart: unless-stopped
    ports:
      - $MINIO_PORT:9000
    env_file: .env
    healthcheck:
      test: ["CMD", "nc", "-z", "minio", "9000"]
      interval: 5s
      retries: 5
  createbuckets:
    image: minio/mc
    depends_on:
      minio:
        condition: service_healthy
    env_file: .env
    # volumes:
    #   This volume is shared with `minio`, so `z` to share it
    #   - ./var/minio:/export
    entrypoint: >
      /bin/sh -c "
      set -x;
      if [ -n \"$MINIO_ACCESS_KEY\" ] && [ -n \"$MINIO_SECRET_KEY\" ] && [ -n \"$MINIO_PORT\" ]; then
        until /usr/bin/mc config host add minio_docker https://minio:$MINIO_PORT $MINIO_ACCESS_KEY $MINIO_SECRET_KEY && break; do
          echo '...waiting...' && sleep 5;
        done;
        /usr/bin/mc mb minio_docker/$AWS_STORAGE_BUCKET_NAME || echo 'Bucket $AWS_STORAGE_BUCKET_NAME already exists.';
        /usr/bin/mc mb minio_docker/$AWS_STORAGE_PRIVATE_BUCKET_NAME || echo 'Bucket $AWS_STORAGE_PRIVATE_BUCKET_NAME already exists.';
        /usr/bin/mc anonymous set download minio_docker/$AWS_STORAGE_BUCKET_NAME;
      else
        echo 'MINIO_ACCESS_KEY, MINIO_SECRET_KEY, or MINIO_PORT are not defined. Skipping buckets creation.';
      fi;
      exit 0;
      "
```



> **Note**
>
> Don't forget to change the entrypoint to run **https** and not **http**.

4.2.15 Workaround: MinIO and Django on the same machine with only the port 443 opened to the external network.

The S3 API signature calculation algorithm does not support proxy schemes where you host the MinIO Server API such as example.net/s3/.

However, we can set the same URL for minio and django site and configure a proxy for each bucket in the Caddyfile :

```
DOMAIN_NAME=https://mysite.com
AWS_S3_ENDPOINT_URL=https://mysite.com
```

Caddyfile :

```
    @dynamic {
        not path /static/* /media/* /{$AWS_STORAGE_BUCKET_NAME}* /{$AWS_STORAGE_PRIVATE_BUCKET_NAME}* /minio*
    }
    reverse_proxy @dynamic django:8000

    @min_bucket {
      path /{$AWS_STORAGE_BUCKET_NAME}* /{$AWS_STORAGE_PRIVATE_BUCKET_NAME}*
    }
    reverse_proxy @min_bucket minio:{$MINIO_PORT}
```

4.3 Administrative Procedures

4.3.1 Maintenance Mode

You can turn on maintenance mode by creating a `maintenance.on` file in the `maintenance_mode` folder. This will change the front page of the website, showing a customizable page (the `maintenance.html` file in the same folder).

Simply remove the `maintenance_mode/maintenance.on` file to end maintenance mode.

During testing, if you want to update or restart some services (e.g : Django), you should follow the following steps:

```
docker compose stop django
docker compose rm django ## remove old django container
docker compose create django ## create new django container with the changes from your development
docker compose start django
```

This procedure helps you update changes of your development on Django without having to restart every Codabench container.

4.3.2 Give superuser privileges to a user

With superuser privileges, the user can edit any benchmark and can access the Django admin interface.

```
docker compose exec django ./manage.py shell_plus
u = User.objects.get(username='<USERNAME>') ## can also use email
u.is_staff = True
u.is_superuser = True
u.save()
```

4.3.3 Migration

```
docker compose exec django ./manage.py makemigrations
docker compose exec django ./manage.py migrate
```

4.3.4 Collect static files

```
docker compose exec django ./manage.py collectstatic --noinput
```

4.3.5 Delete POSTGRESDB and MINIO :

> **Warning**
>
> This will delete all your data !

```
## Begin in codabench root directory
cd codabench
```

PURGE DATA

```
sudo rm -r var/postgres/*
sudo rm -r var/minio/*
```

SEE DATA WE ARE GOING TO PURGE

```
ls var
```

RESTART SERVICES AND RECREATE DATABASE TABLES

```
docker compose down
docker compose up -d
docker compose exec django ./manage.py migrate
```

4.3.6 Feature competitions in home page

There are two ways of setting a competition as featured: 1. Use Django admin (see below) -> click the competition -> scroll down to is featured filed -> Check/Uncheck it 2. Use competition ID in the django bash to feature or unfeature a competition

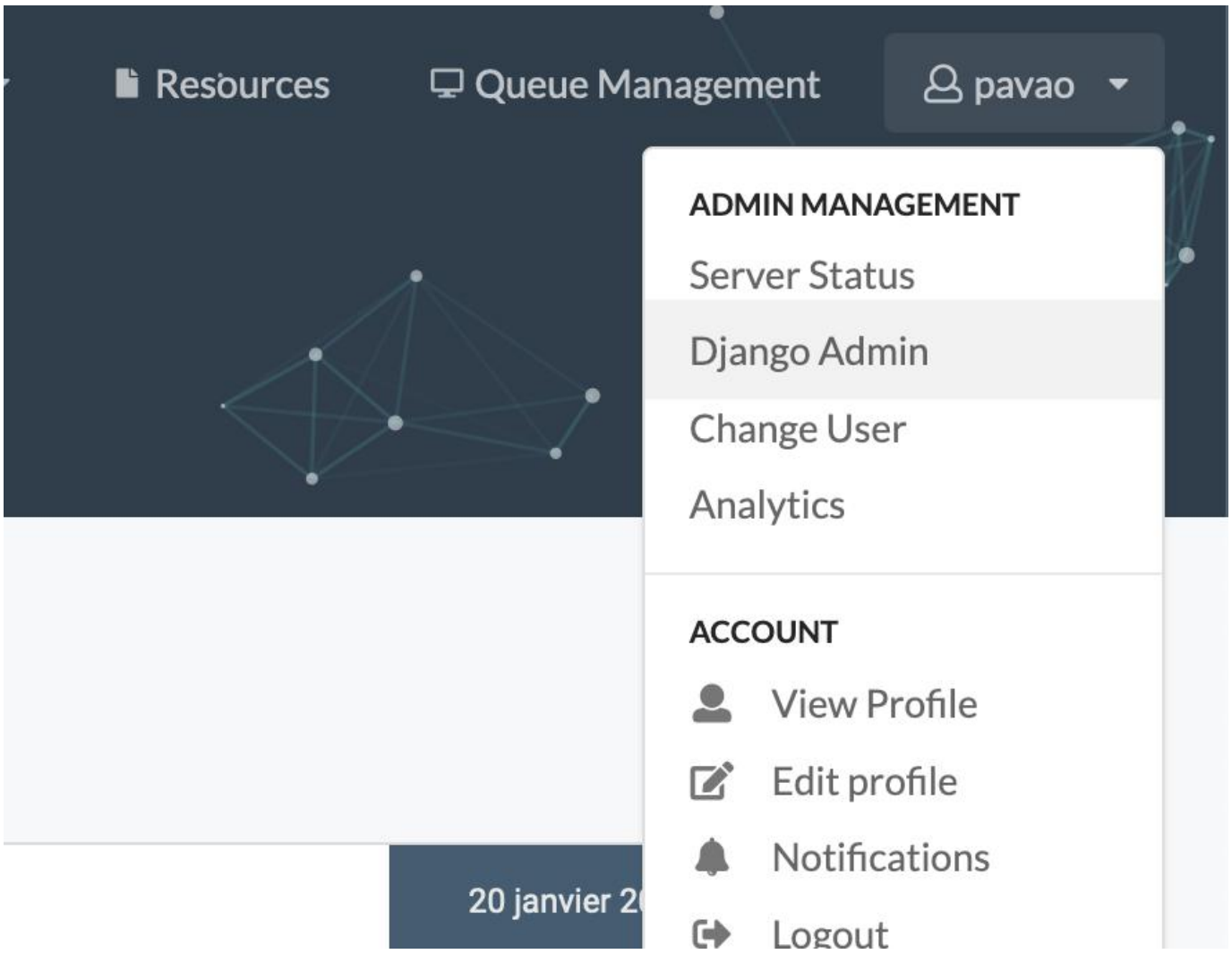
```
docker compose exec django ./manage.py shell_plus
comp = Competition.objects.get(id=<ID>)  ## replace <ID> with competition id
comp.is_featured = True  ## set to False if you want to unfeature a competition
comp.save()
```

4.3.7 Shell Based Admin Features

If you're running your own Codabench instance, there are different ways to interact with the application. Inside the `django` container (`docker compose exec django bash`) you can use `python manage.py help` to display all available commands and a brief description. By far the most useful are `createsuperuser` and `shell/shell_plus` .

4.3.8 Django Admin interface

Once you log in an account with superuser privileges, you have access to the "Django Admin" interface:

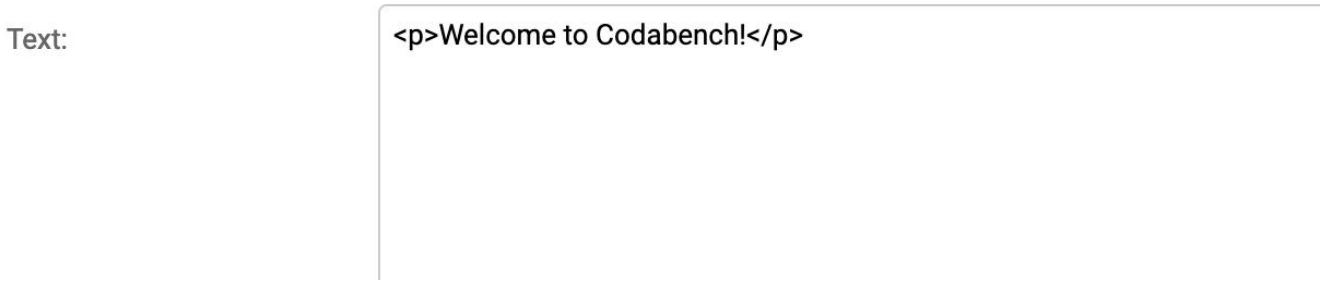


From this interface, you can change a user's quota, change their staff and superuser status, change the featured competitions displayed on the homepage, manage user accounts and more.

EDIT ANNOUNCEMENT AND NEWS

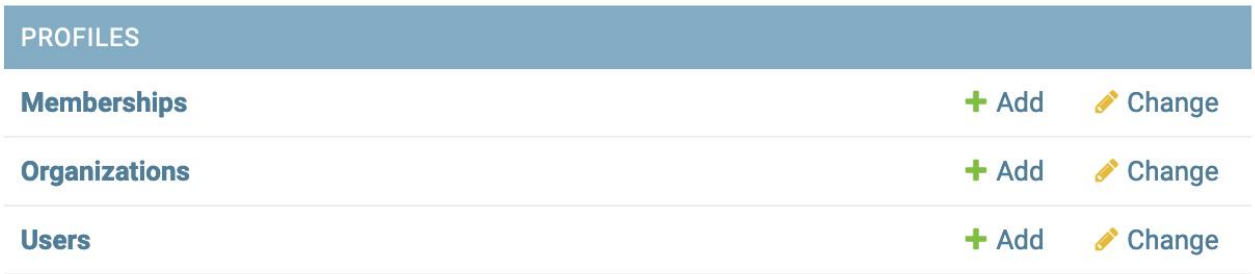
In the Django admin interface, click on `Announcements` or `New posts`:

For announcement, only the first announcement is read by the front page. For news, all objects are read as separate news. You can create and edit objects using the interface. Write the announcement and news using HTML to format the text, add links, and more:

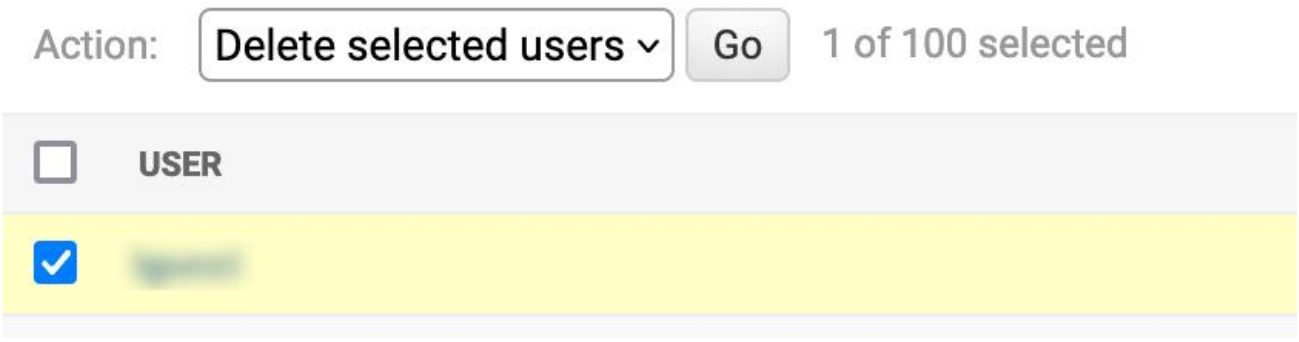


DELETE A USER

Go to `Users`:



Select it, select the `Delete selected users` action and click on `Go`:



4.3.9 RabbitMQ Management

The RabbitMQ management tool allows you to see the status of various queues, virtual hosts, and jobs. By default, you can access it at: `http://<your_codalab_instance>:15672/`. The username/password is your RabbitMQ `.env` settings for username and password. The port is hard-set in `docker-compose.yml` to 15672, but you can always change this if needed. For more information, see: https://www.rabbitmq.com/management.html

4.3.10 Flower Management

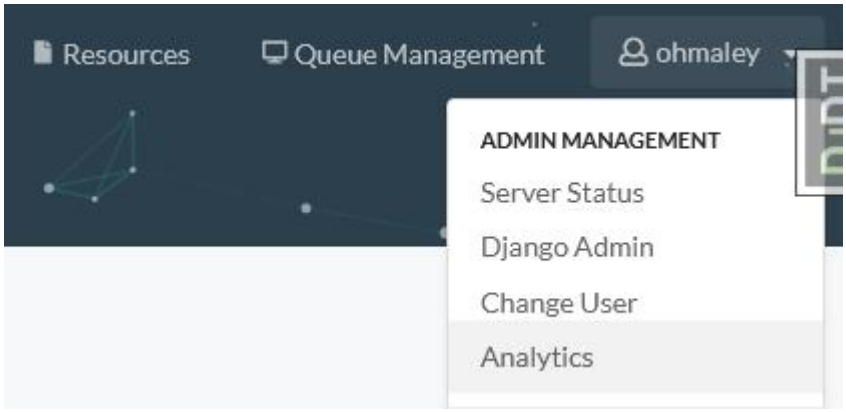
Flower is a web based tool for monitoring and administrating Celery clusters. By default, you can access the Flower web portal at `http://<your_codalab_instance>:5555/`. The username/password is your Flower `.env` settings for username and password. 5555 is the default port, and cannot be changed without editing the `docker-compose.yml` file.

For more information on flower, please visit: https://flower.readthedocs.io/en/latest/

4.3.11 Storage analytics

THE INTERFACE

The storage analytics page is accessible at codabench-url/analytics/ under the storage tab.



From this interface, you will have access to various analytics data:

- A Storage usage history chart
- A Competitions focused storage evolution, distribution and table
- A Users focused storage evolution, distribution and table

All of those data can be filtered by date range and resolution, and exported as CSVs.

The data displayed in those charts are only calculated from a background analytics task that takes place every Sunday at 02:00 UTC time (value editable in the `src/settings/base.py`).

THE BACKGROUND TASK

The analytics task is a celery tasked named `analytics.tasks.create_storage_analytics_snapshot`. What it does:

- It scans the database looking for file sizes that are not set or flagged in error
- Actually measures their size and saves it in the database
- Aggregate the storage space used by day and by Competition/User (for example every day for the last year for each competition) by looking at the database file size fields
- For data related to the *Platform Administration* it measures as well the database backup folder directly from the storage instance.
- Everything is saved as multiple snapshot in time in each Category table (i.e.: `UserStorageDataPoint`)
- This tasks also runs a database <-> storage inconsistency check and saves the results in a log file located in the `var/logs/` folder

To manually start the task, you can do the following:

- Start codabench `docker compose up -d`
- Bash into the django container and start a python console:

```
docker compose exec django ./manage.py shell_plus
```

- Manually start the task:

```
from analytics.tasks import create_storage_analytics_snapshot
eager_results = create_storage_analytics_snapshot.apply_async()
```

- If you check the logs (`docker compose logs -f`) of the app you should see "Task create_storage_analytics_snapshot started" coming from the site_worker container
- If you have to restart the task, don't worry, it will only compute the size of the files that hasn't been computed yet.
- Once the task is over you should be able to see the results on the web page

4.3.12 Homepage counters

There is also a daily background task counting users, competitions and submissions, in order to display it on the homepage.

You can manually run it:

```
docker compose exec django ./manage.py shell_plus
```

```
from analytics.tasks import update_home_page_counters
eager_results = update_home_page_counters.apply_async()
```

4.3.13 User Quota management

INCREASE USER QUOTA

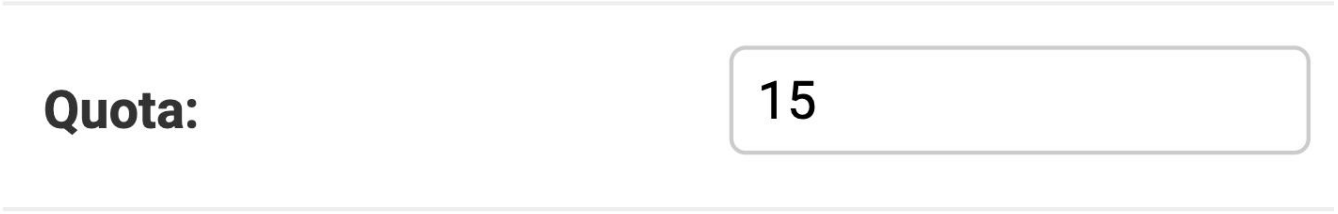
Using the Django Shell

```
docker-compose exec django ./manage.py shell_plus
```

```
u = User.objects.get(username='<USERNAME>') ## can also use email
u.quota = u.quota * 3 # We multiply the quota by 3 in this example
u.save()
```

Using the Django Admin Interface

- Go to the Django admin page
- Click user table
- Select the user for whom you want to increase/decrease quota
- Update the quota field with new quota (in GB e.g. 15)



4.3.14 Codabench Statistics

You can create two types of codabench statistics:

- Overall platform statistics for a specified year
- Overall published competitions statistics

Follow the steps below to create the statistics

START CODABENCH

```
docker compose up -d
```

BASH INTO THE DJANGO CONTAINER AND START A PYTHON CONSOLE:

```
docker compose exec django ./manage.py shell_plus
```

FOR OVERALL PLATFORM STATISTICS

```
from competitions.statistics import create_codabench_statistics
create_codabench_statistics(year=2024)
```

Note

- If `year` is not specified, the current year is used by default
- A CSV file named `codabench_statistics_2024.csv` is generated in `statistics` folder (for year=2024)

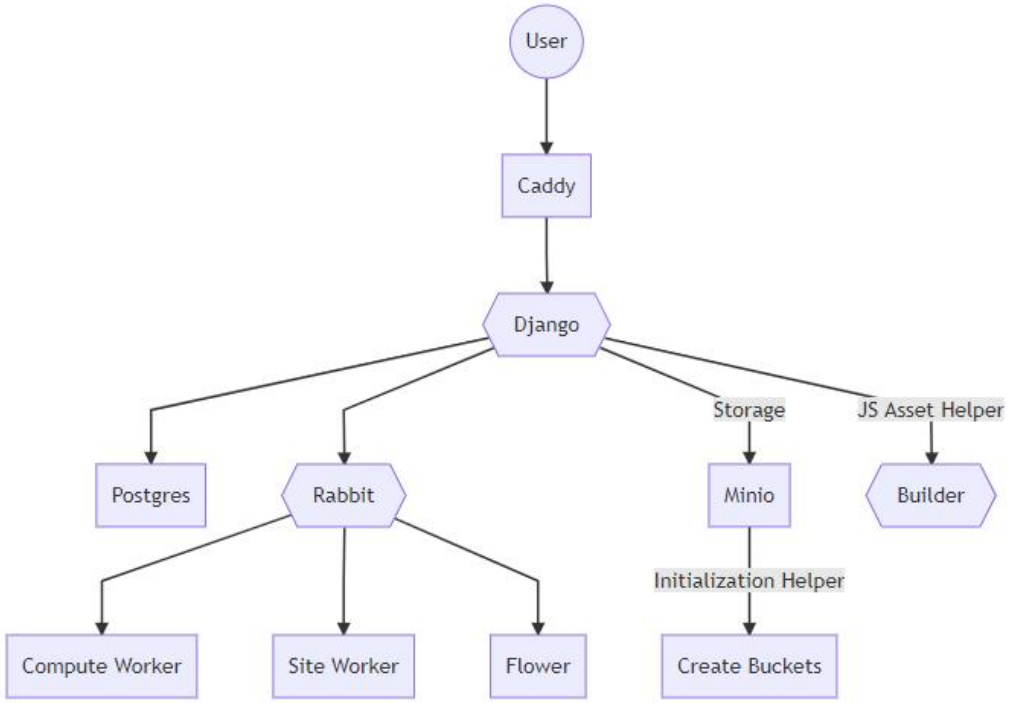
FOR OVERALL PUBLISHED COMPETITIONS STATISTICS

```
from competitions.statistics import create_codabench_statistics_published_comps
create_codabench_statistics_published_comps()
```

Note

A csv file named `codabench_statistics_published_comps.csv` is generated in `statistics folder`

4.4 Codabench Docker Architecture



Source:

```
graph TD
A((User))
B[Caddy]
C{{Django}}
D[Postgres]
F{{Rabbit}}
G[Minio]
H[Create Buckets]
I{{Builder}}
K[Flower]
L[Compute Worker]
M[Site Worker]
A --> B
B --> C
C --> D
C --> F
F --> L
F --> M
C -->|Storage|G
G -->|Initialization Helper| H
F --> K
C -->|JS Asset Helper| I
```

Codabench consists of many docker containers that are connected and organized through docker compose. Below is an overview of each container and their function:

4.4.1 Django

The main container that runs and contains the python code (A Django project). This is the container that is mainly used for utility functions like creating admins, creating backups, and manually making changes through the python django shell. It has a gunicorn webserver that serves internally on port 8000.

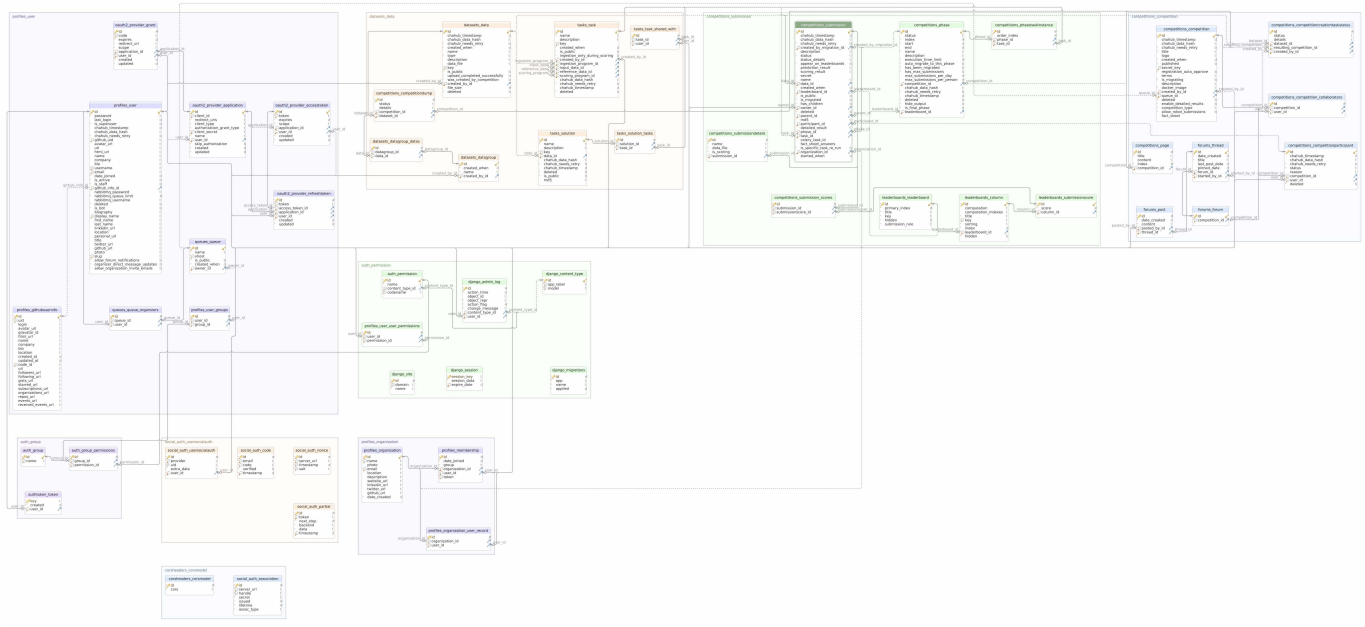
4.4.2 Caddy

The HTTP/HTTPs web server. It acts as a reverse proxy for the Django container. This container controls SSL/HTTPs functionality and other web server configuration options. Serves on port 80/443.

4.4.3 Postgres (Labeled DB in docker-compose)

The default database container. It will contain the default Postgres database used by codabench. (Name/user/pass,etc determined by .env.) If you need to make manual DB changes, this is the container to look into.

Here is the DB schema



4.4.4 Compute Worker

The container(s) (can be external) that runs submissions for the codabench instance on their associated queues. The default workers are tied to the default queue. Commands to re-build the compute worker Docker image can be found here.

4.4.5 Site Worker

The container that runs various tasks for the Django container such as unpacking and processing a competition bundle.

4.4.6 Minio

The default storage solution container. Runs a MinIO instance that serves on the port defined by settings in your .env.

4.4.7 Create Buckets

A helper container for the Minio container that initially creates the buckets defined by settings in your .env if they don't already exist. Usually this container exits after that.

4.4.8 Builder

A container to build RiotJS tags into a single unified tag that can then be mounted. Uses NPM.

4.4.9 Rabbit

Task/Message management container. Organizes queues for Celery Tasks and compute workers. Any queues that get created are accessible through rabbit's own command line interface.

4.4.10 Flower

Administrative utility container for Celery tasks and queues.

4.4.11 Competition docker image

The services of Codabench run inside Docker environments. This should not be confused with the Docker environment used to run the submissions, which may vary for each benchmark. The docker running submissions is discussed here

4.5 Submission Docker Container Layout

When you make a submission to Codabench, the information and file are saved to the database. Afterwards, a Celery task gets sent to the compute worker (Default queue or a compute worker attached to a custom queue). From there the compute worker spins up another docker container with either the default docker image for submissions, or a custom one supplied by the organizer.

4.5.1 Site Worker

The site worker can be thought of exactly how it sounds. It's a local celery worker for the site to handle different Celery tasks and other miscellaneous functions. This is what is responsible for creating competition dumps, unpacking competitions, firing off the tasks for re-running submissions, etc.

4.5.2 Compute Worker

Is a remote and/or local celery worker that listens to the main RabbitMQ server for tasks. A compute worker can be given a special queue to listen to, or listen to the default queue. For more information on setting up a custom queue and compute worker, see:

- Queue Management
- Compute Worker Management and Setup

4.5.3 Submission Container

The submission container is the container that participant's submissions get ran in. The image used to create this container is either the Codabench default if the organizer's haven't specified an image, or the custom image they specified in the competition. The default docker image is `codalab/codalab-legacy:py37`. Organizers can specify their own image using either the YAML file or the editor.

This container is also created with some specific directories, some of which are only available at specific steps of the run. For example, generally for a submission there are 2 steps. The prediction step (If the submission needs to make predictions) and the scoring step, where the predictions are scored against the truth reference data.

Here are the following directories:

- `/app/input_data` Where input data will be (Only exists if input data is supplied for the task)
- `/app/output` Where all submission output should go (Should always exist)
- `/app/ingestion_program` Where the ingestion program files should exist (Only available in ingestion)
- `/app/program` Where the scoring program and/or the ingestion program should be located (Should always exist)
- `/app/input` Where any input for this step should be. (I.E: Previous predictions from ingestion for scoring. Only exists on scoring step)
- `/app/input/ref` Where the reference data should live (Not available to submissions. Only available on scoring step.)
- `/app/input/res` Where predictions/output from the prediction step should live. (Only available on scoring step)
- `/app/shared` Where any data that needs to be shared between the ingestion program and submission should exist. (Only available on scoring steps.)
- `/app/ingested_program` Where submission code should live if it is a code submission. (Only available in ingestion)

4.6 Backups - Automating Creation and Restoring

Codabench has a custom command that uploads a database backup, and copies it to the storage you are using under `/backups`. We'll see how to execute and automate that command, and how to restore from one of these backups in the event of a failure.

4.6.1 Creating Backups

Create

```
DB_NAME=
DB_USERNAME=
DB_PASSWORD=
DUMP_NAME=
docker exec codabench-db-1 bash -c "PGPASSWORD=$DB_PASSWORD pg_dump -Fc -U $DB_USERNAME $DB_NAME > /app/backups/$DUMP_NAME.dump"
```

Upload

There's a custom command on codabench that we use to upload database backups to storage. It should be accessible from inside the Django container (`docker compose exec django bash`) with `python manage.py upload_backup <backup_path>`. It takes an argument `backup_path` which is the path relative to your backup folder, `codabench/backups` and storage bucket, `/backups`. For instance if I pass it as `2022/$DUMP_NAME.dump`, the backup should happen in `codabench/backups/2022/$DUMP_NAME.dump` and will be uploaded to `/backups/2022/$DUMP_NAME.dump` in your storage bucket.

4.6.2 Scheduling Automatic Backups

To schedule automatic backups, we're going to schedule a daily cronjob. To start, open the cron editor in a shell with `crontab -e`.

Add a new entry like so, with the correct path to `pg_dump.py`:

```
@daily /home/ubuntu/codabench/bin/pg_dump.py
```

You should confirm this backup process works by setting some known cronjob time a few minutes in the future and see the dump in storage.

Once done, save and quit the crontab editor, and verify your changes held by listing out cronjobs with `crontab -l`. You should see your new crontab entry.

4.6.3 Restoring From Backup

Re-install Codabench according to the documentation here: Codabench Installation.

Once Codabench is re-installed and working, we're ready to restore our database backup. Upload the database backup to the webserver. It should go under the `codabench` install folder in the `/backups` directory. For example your path might look like: `/home/users/ubuntu/codabench/backups`

Once the backup is located in the `/backups` folder, you'll want to get into the postgres container (`docker compose exec db bash`). Make sure you know your `DB_NAME`, `DB_USERNAME`, and `DB_PASSWORD` variables from your .env.

You can restore two ways. The first would be manually dropping the db, re-creating it, then using pg_restore to restore the data:

Inside the database container

```
dropdb $DB_NAME -U $DB_USERNAME
createdb $DB_NAME -U $DB_USERNAME
pg_restore -U $DB_USERNAME -d $DB_NAME -1 /app/backups/<filename>.dump
```

Or, you can let `pg_restore` do that for you with a couple of flags/arguments:

Inside the database container

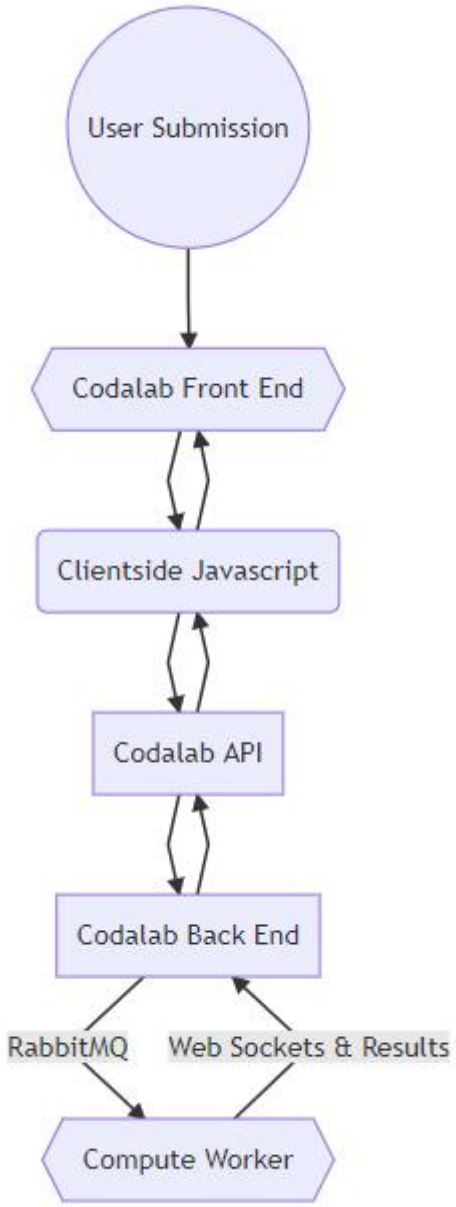
```
pg_restore --verbose --clean --no-acl --no-owner -h $DB_HOST -U $DB_USERNAME -d $DB_NAME /app/backups/<filename>.dump
```

The arguments `--no-acl` & `--no-owner` may be useful if you're restoring as a non-root user. The owner argument is used for: `Do not output commands to set ownership of objects to match the original database.`

The ACL argument is for: `Prevent dumping of access privileges (grant/revoke commands).`

After running `pg_restore` successfully without errors, you should find everything has been restored.

4.7 Submission Process Overview



Source:

```
graph TD
A((User Submission))
B{{Codalab Front End}}
C(Clientside Javascript)
D[Codalab API]
E[Codalab Back End]
F{{Compute Worker}}

A-->B
B-->C
C-->B
C-->D
D-->C
D-->E
E-->D
```

```
E-->|RabbitMQ|F
F-->|Web Sockets & Results|E
```

4.7.1 Overview:

- When making a submission to Codabench, the website uses a client-side javascript to send the submission file along with proper details to an API point for submissions.
- Once the API receives the new submission, Codabench's back-end fires off a Celery task through RabbitMQ.
- This task is picked up by a Codabench Compute Worker listening in on the associated RabbitMQ. The Compute Worker starts processing the data.
- The compute worker begins execution by creating the container the submission will run in using the organizer's specified docker image for the competition.
- Once the container is created, the scoring program and other necessary organizer and user supplied binaries are executed to produce results. While the submission is processing, communication back to the Codabench instance is possible through web-sockets.
- Once the submission is done processing, it moves the output to the proper folders, and the compute worker posts these scores (if found) to the Codabench API.
- From here, the submission is done processing and is marked as failed/finished. If at any part of the processing step (Scoring/Ingestion step of the submission) an exception is raised, the compute worker will halt and send all current output.

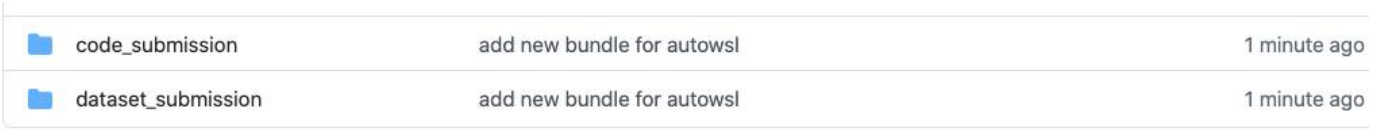
4.8 Robot Submissions

This script is designed to test the Robot Submissions feature. Robot users should be able to submit to bot-enabled competitions without being admitted as a participant.

This article will explain how to make a robot submission on your local computer, and how to present the results on the Leaderboard.

4.8.1 Pre-requisite

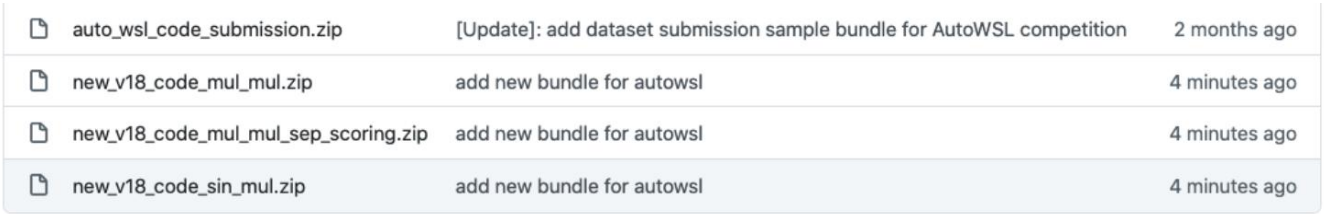
- Python 3
- Demo bundle: autowsl
- Github download URL



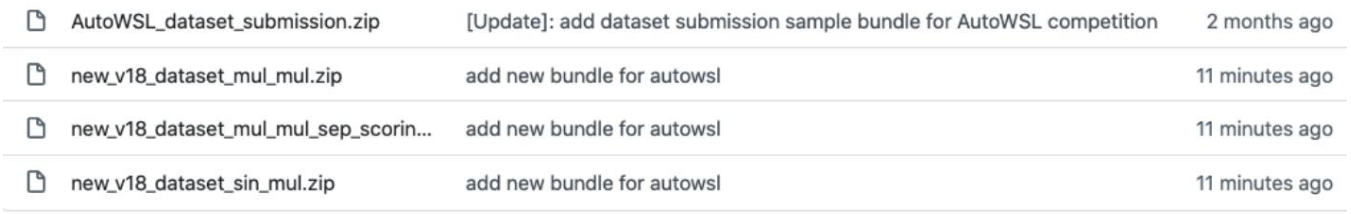
- Robot submission sample script here: link

Brief description for demo bundle:

- `code_submission` : It contains the sample bundle for the code submission and the code solution for the submission.
- `auto_wsl_code_submission.zip` :This bundle is used for making submission.
- `new_v18_code_mul_mul.zip` : The bundle is multiple phases, each phase has multiple tasks, and between these tasks, they share the same scoring program, that is, there is no need to copy multiple scoring program for hardcode.
- `new_v18_code_mul_mul_sep_scoring.zip` : This bundle is multiple phases, multiple tasks under each phase share a scoring program that is exclusive to their particular phase.
- `new_v18_code_sin_mul.zip` : This is the sample bundle of a single phase multi-task that shares the same scoring program.



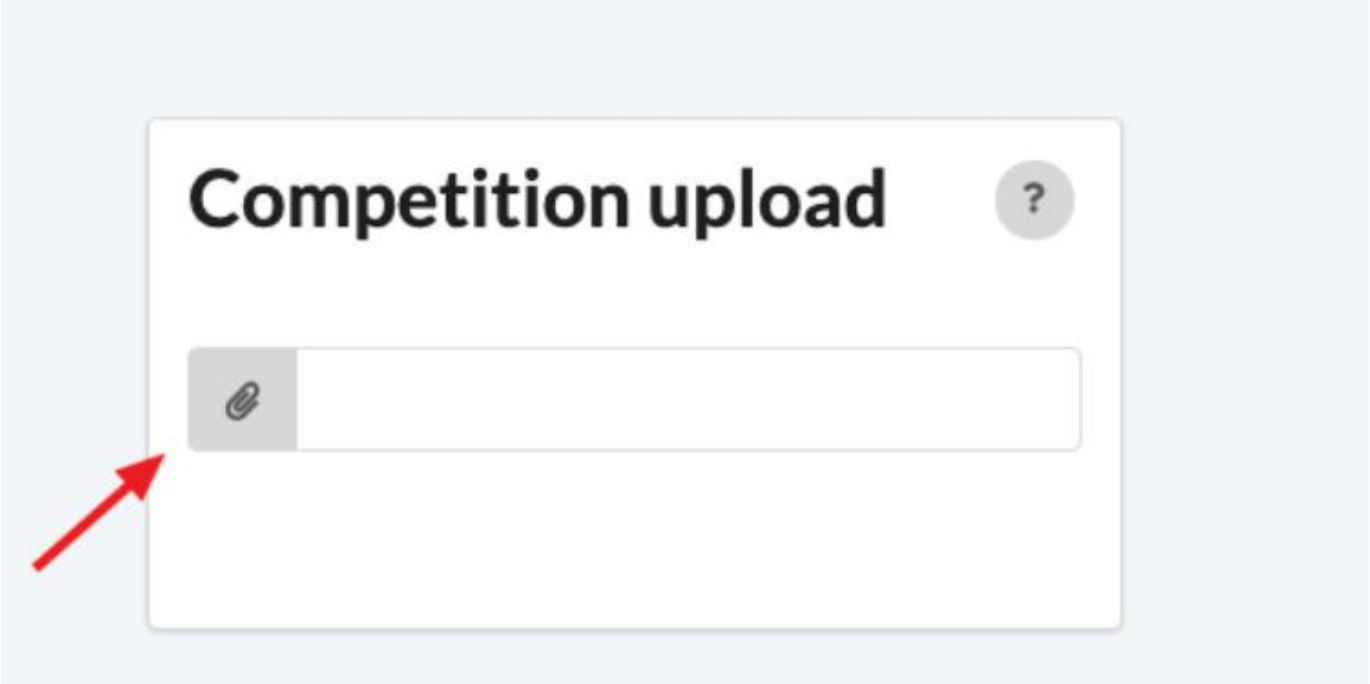
- `dataset_submission` : It contains the sample bundle for data submission and the corresponding dataset solution for submission.
- `AutoWSL_dataset_submission.zip` : This is the bundle used for dataset submissions.
- `new_v18_dataset_mul_mul.zip` :This is a multi-phase, each phase has multiple tasks below the sample bundle, multiple tasks, using the same scoring program, do not need to copy multiple scoring program for hardcode
- `new_v18_dataset_mul_mul_sep_scoring.zip` : This is a multi-phase, each phase has multiple tasks below the sample bundle, the task between the different phases, using a different scoring program, that is, each phase has its own independent scoring program.
- `new_v18_dataset_sin_mul.zip` : This is a sample bundle of single-phase multi-task commit datasets.



4.8.2 Getting started

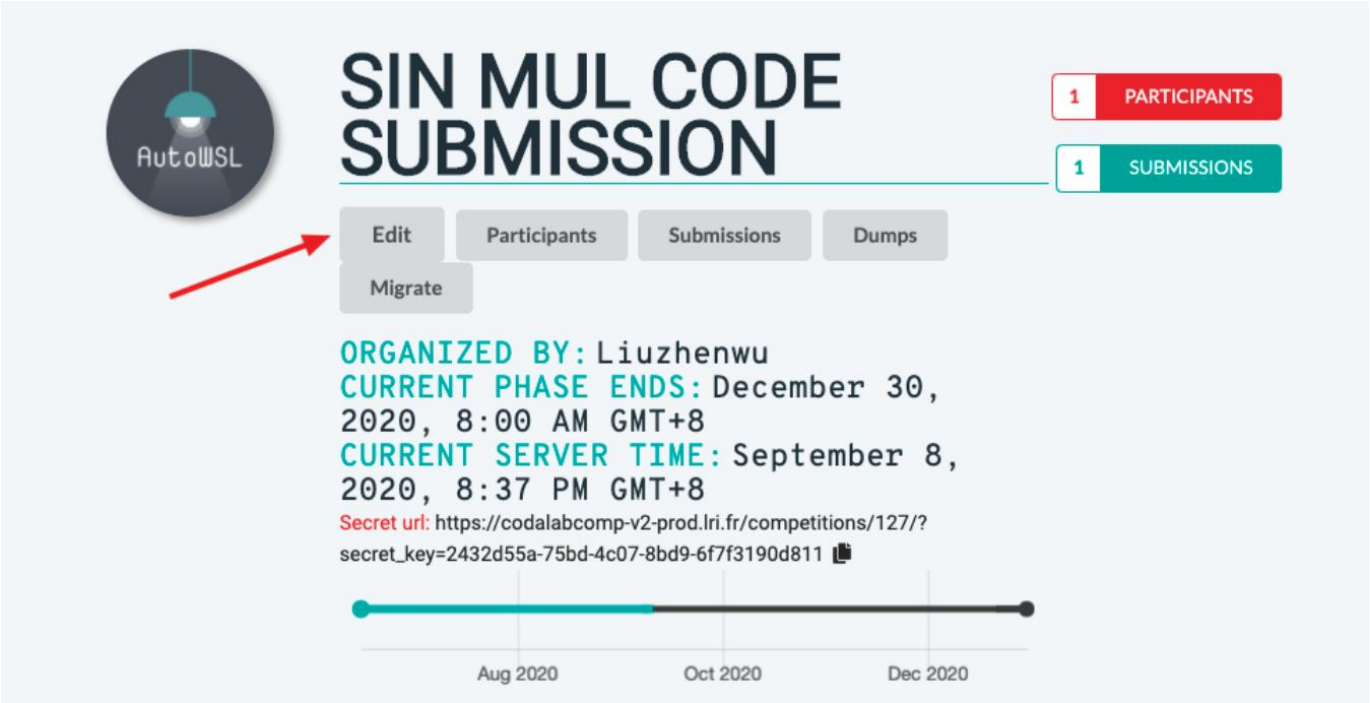
Upload a bundle

Use the sample bundle provided above to upload the bundle and create a competition

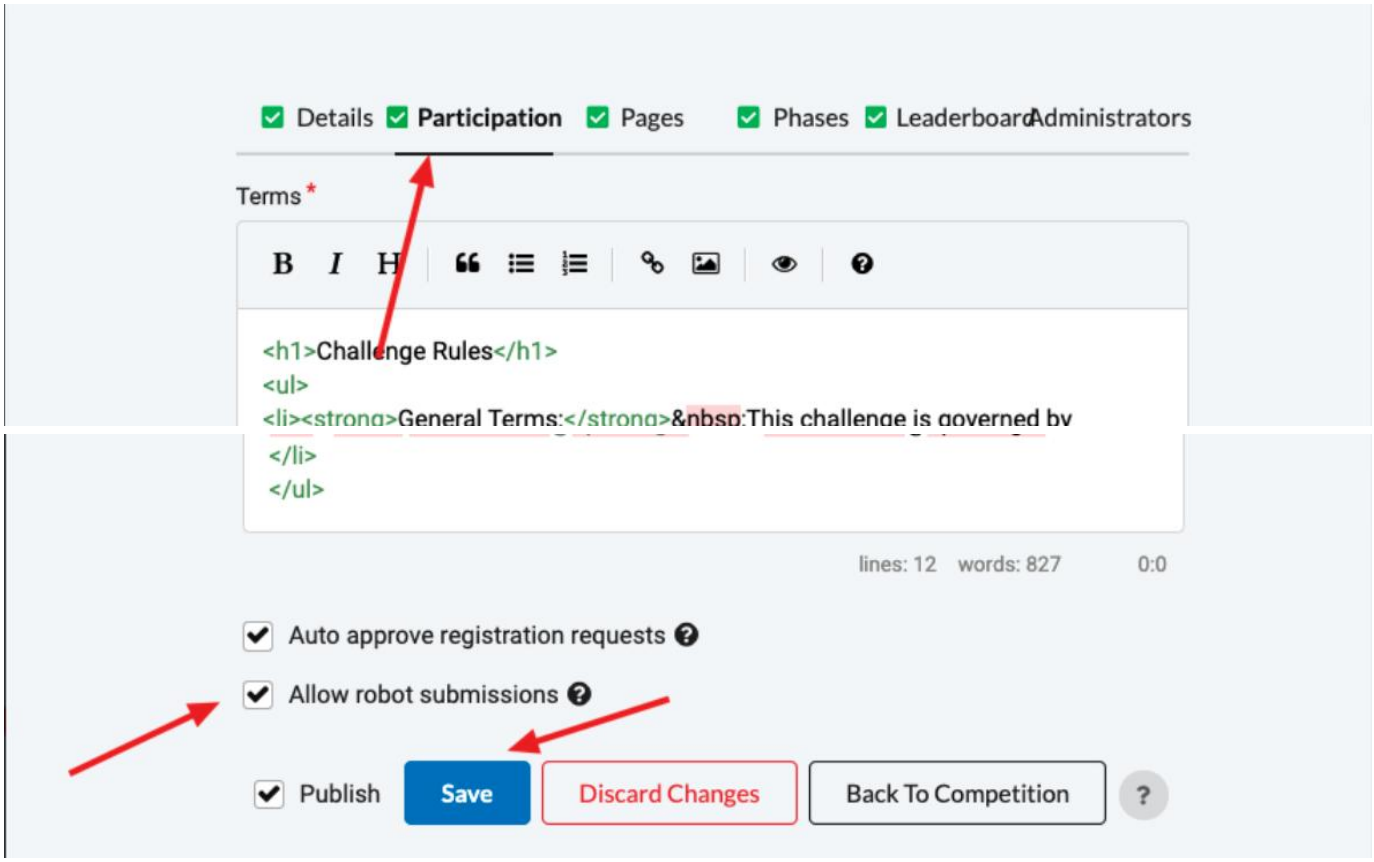


Set the competition to allow robot submissions

On the created competition page, click the EDIT button



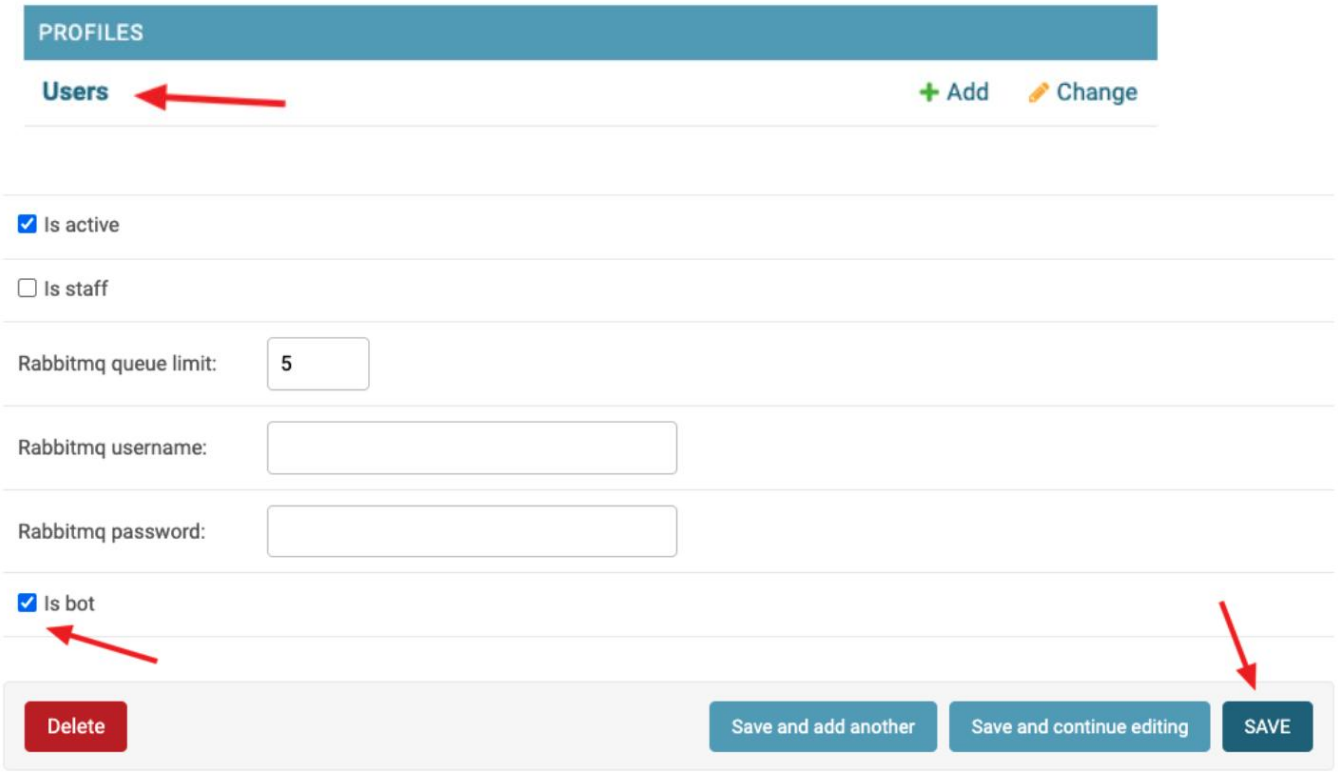
Then click on the Participation tab, then scroll down to the bottom and click on Allow robot submission and click SAVE button.



After the above steps are done, the Competition is allowed for making robot submission.

Set yourself to Is bot

Go to the backend administration page, PROFILES tab bar below the user



Check the `is bot` box, click save. You can now proceed with your robot submissions.

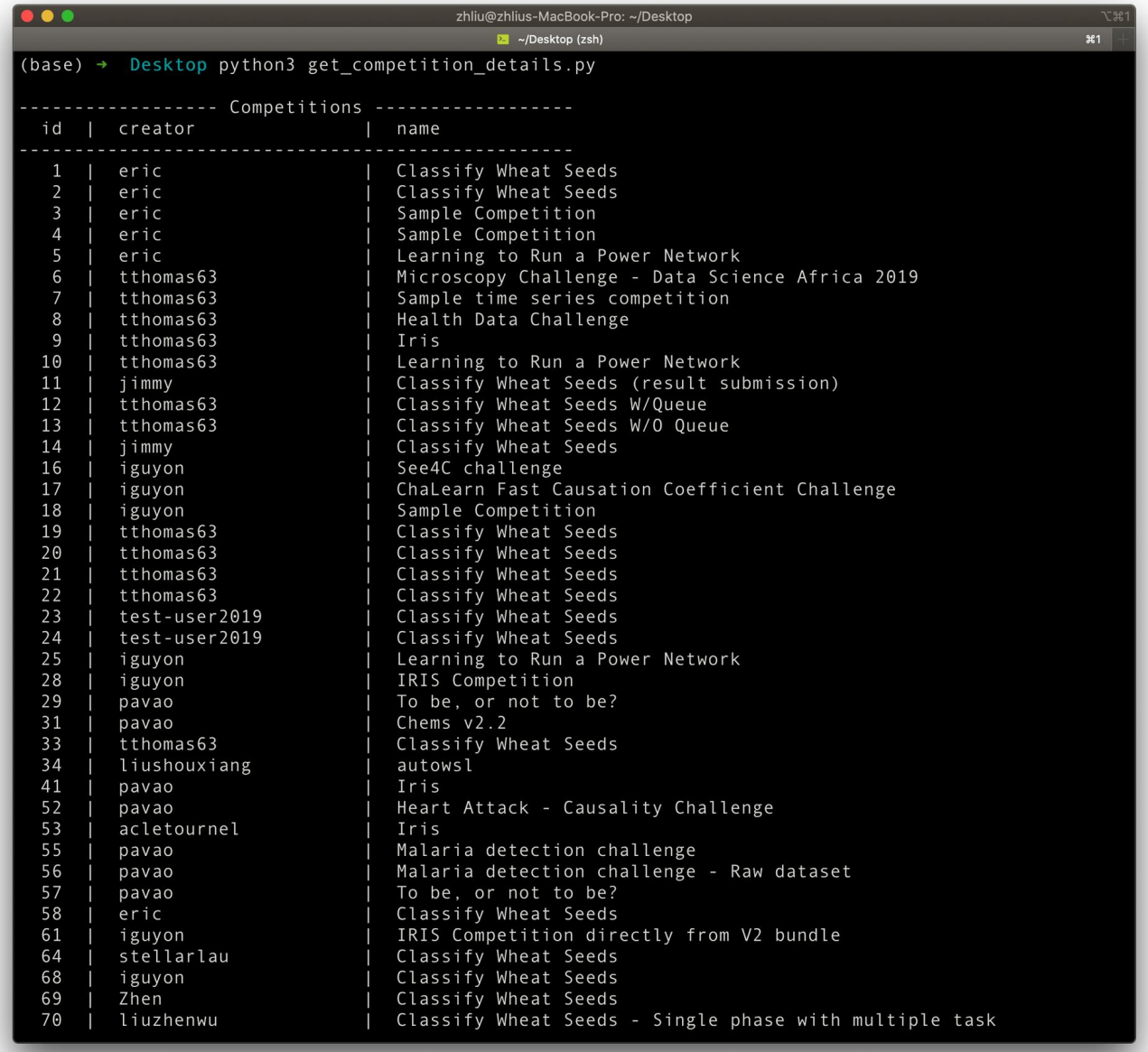
Change CODALAB_URL address

Change CODALAB_URL address in following scripts: `get_competition_details.py`, `example_submission.py`, `get_submission_details.py`
CODALAB_URL = 'https://www.codabench.org/'

> Find scripts at `docs/example_scripts`

Choose the competition

Run the following command on the command line: `python3 get_competition_details.py` What you're about to see is something like this



Choose the ID of the competition you are interested in, for example 127.

Run the script again with the competition ID as a parameter `python3 get_competition_details.py 127` Then you will see the following

```
(base) ➜  Desktop python3 get_competition_details.py 127

Competition: SIN MUL CODE SUBMISSION

---------- Phases ----------
  id  |  name
----------------------------
 215  |  Single Phase with Multiple Task
```

You can select the phase ID you're interested in, then use it as the second argument and run the script again, this time you'll get the task information associated with that phase. `python3 get_competition_details.py 127 215`

```
(base) ➜  Desktop python3 get_competition_details.py 127 215

Competition: Single Phase with Multiple Task

----------------- Tasks ----------------
  id  |  name
----------------------------------------
 289  |  First Task
 290  |  Second Task
 291  |  Third Task
```

Making submission

Inside the `example_submission.py` script, configure these options:

```
# ----------------------------------------------------------------------------
# Configure these
# ----------------------------------------------------------------------------
CODALAB_URL = 'https://codalabcomp-v2-prod.lri.fr/'
USERNAME = '[redacted]'
PASSWORD = '[redacted]'
PHASE_ID = 215
TASK_LIST = []
SUBMISSION_ZIP_PATH = '/Users/zhliu/Desktop/auto_wsl_code_submission.zip'
```

- `CODALAB_URL` can be changed if not testing locally.
- `USERNAME` and `PASSWORD` should correspond with the user being tested.
- `PHASE_ID` should correspond with the phase being tested on.
- `TASK_LIST` can be used to submit to specific tasks on a phase. If left blank, the submission will run on all tasks.
- `SUBMISSION_ZIP_PATH` You can fill in the absolute path of the submission directly.

The idea here is that I'm going to test all the tasks below the competition with phase ID `215`. Then run the script. `python3 example_submission.py`

```
(base) ➜  Desktop python3 example_submission.py
Making submission using data: {'phase': 215, 'tasks': [], 'data': '141187e5-1a57-4910-9ab8-bc4ee21949af'}
Successfully submitted: b'{"id":542,"data":"141187e5-1a57-4910-9ab8-bc4ee21949af","phase":215,"status":"Submitting","status_details":null,"filename":"submission.zip","description":"","md5":null}'
```

You can see that you have successfully submitted the submission bundle.

View submission details

Configure the `get_submission_details.py` options before running.

```
29
30  # ---------------------------------------------------------------------------
31  # Configure these
32  # ---------------------------------------------------------------------------
33  CODALAB_URL = 'https://codalabcomp-v2-prod.lri.fr/'
34  USERNAME =
35  PASSWORD =
36
```

- `CODALAB_URL` can be changed if not testing locally.
- `USERNAME` and `PASSWORD` should correspond with the user being tested. Run the `get_submission_details`.py script with the ID of the phase containing the desired submission as the first argument.

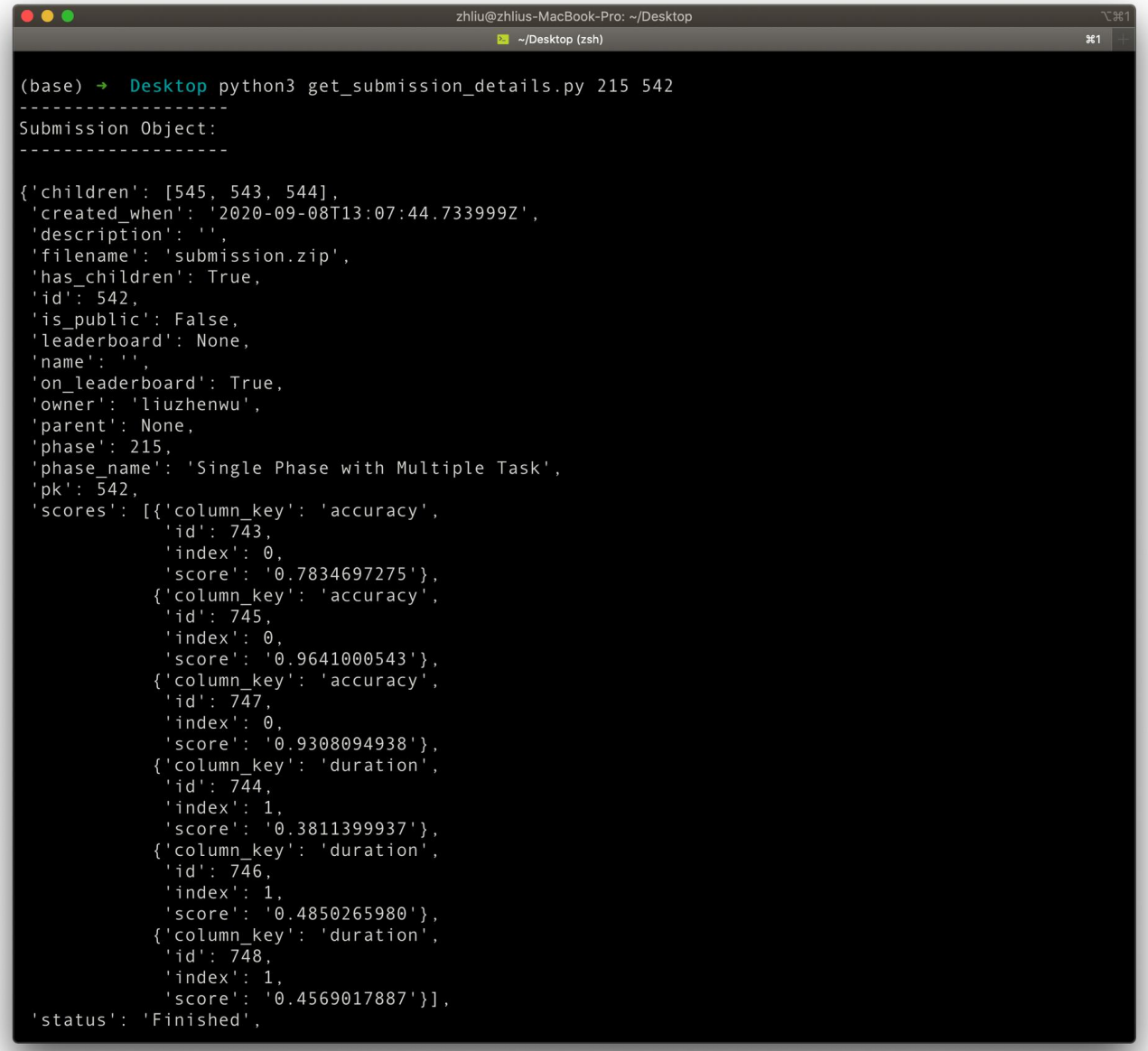
Since we chose `215` for our phase ID above, we'll choose `215` here.

Then run the script. `python3 get_submission_details.py 215`

```
(base) ➜  Desktop python3 get_submission_details.py 215

-------------------- Submissions ----------------------
  id  |  creator              |  creation date
-------------------------------------------------------
 464  |  liuzhenwu            |  2020-09-01T10:28:01.745620Z
 465  |  liuzhenwu            |  2020-09-01T10:28:01.800481Z
 466  |  liuzhenwu            |  2020-09-01T10:28:01.821579Z
 467  |  liuzhenwu            |  2020-09-01T10:28:01.827409Z
 542  |  liuzhenwu            |  2020-09-08T13:07:44.733999Z
 543  |  liuzhenwu            |  2020-09-08T13:07:44.997421Z
 544  |  liuzhenwu            |  2020-09-08T13:07:45.019076Z
 545  |  liuzhenwu            |  2020-09-08T13:07:45.028028Z
```

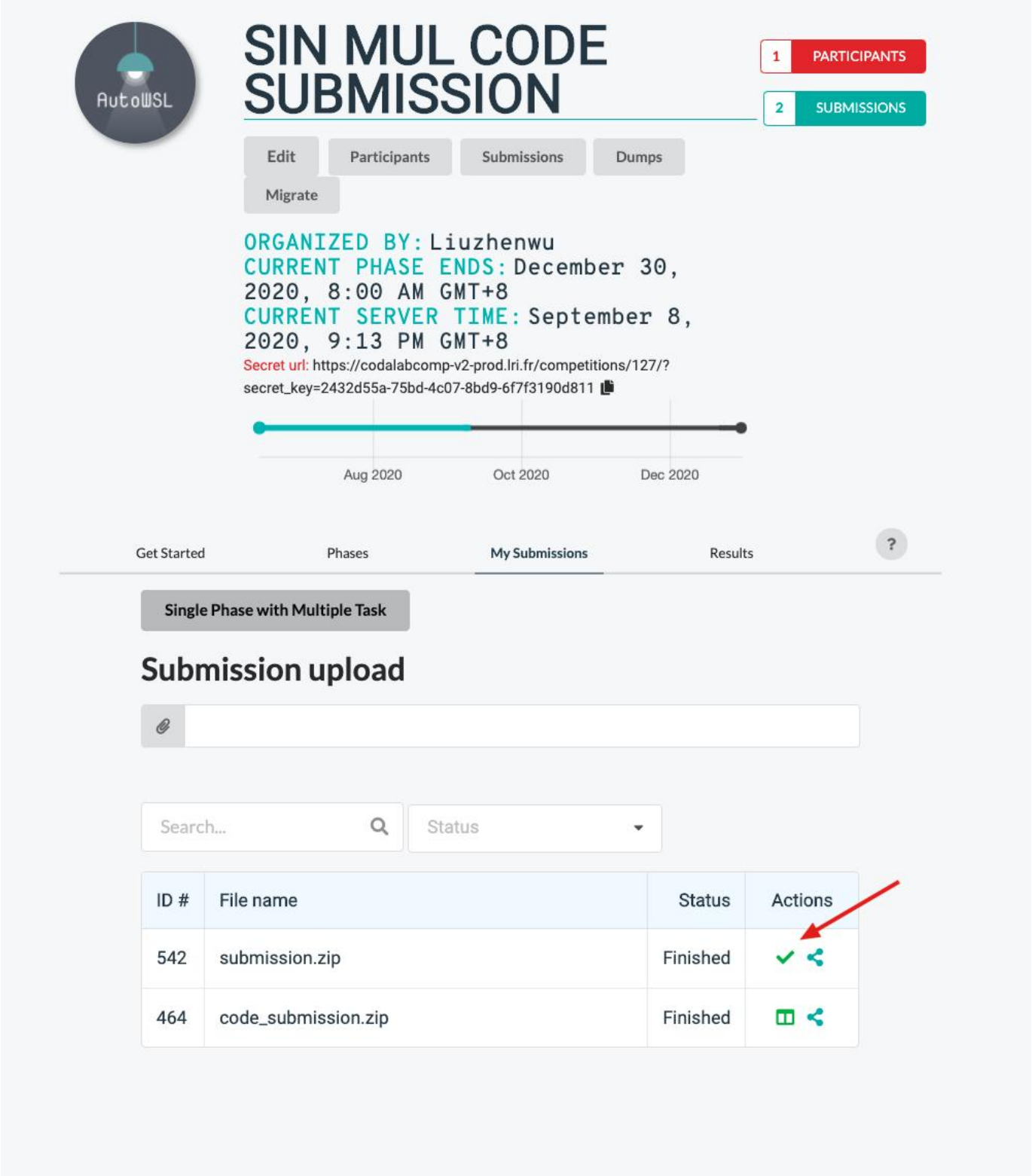
Find the ID of the desired submission. For example, `542`.

Then run the script. `python3 get_submission_details.py 215 542`

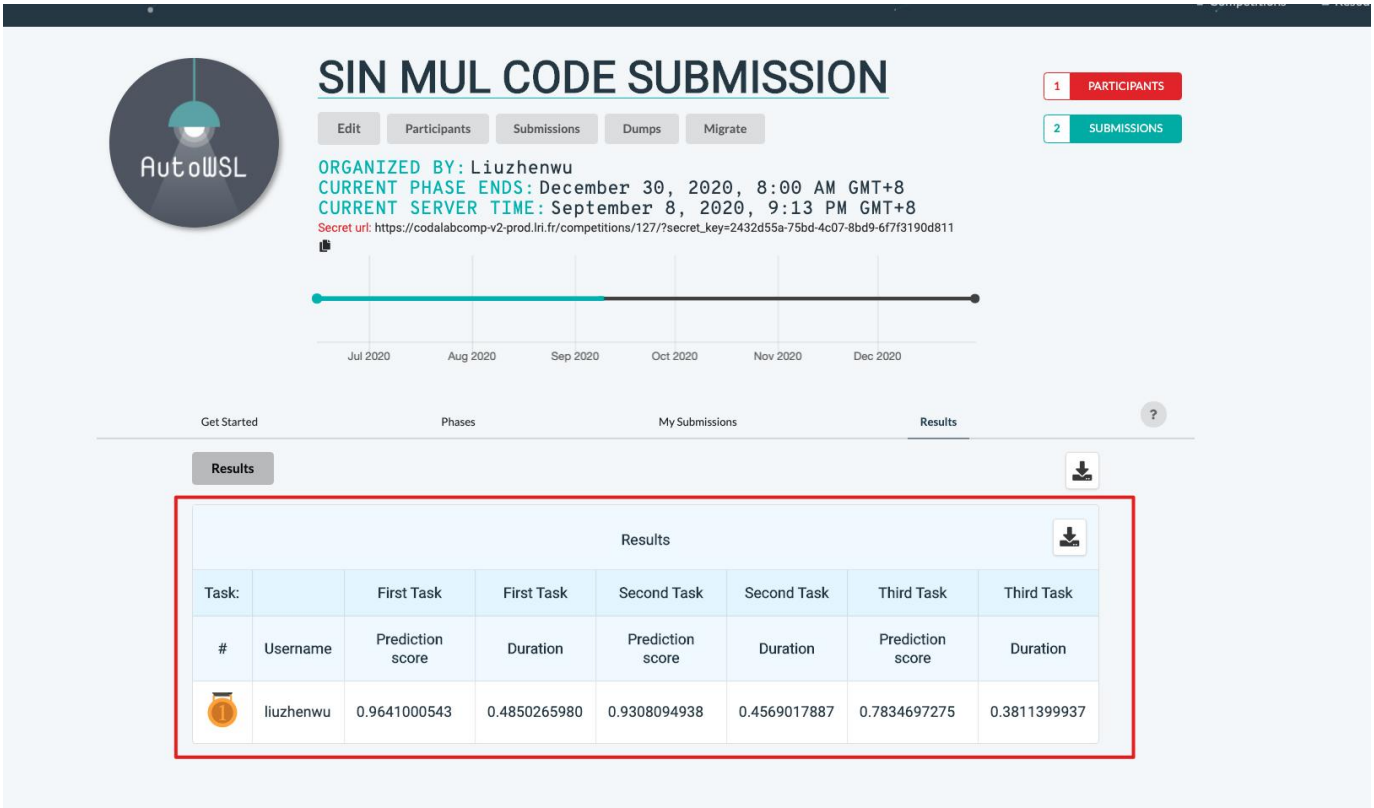


Finally

Finally, you can go to the competition page, add your submission, and add it to the Leaderboard!



On the Leaderboard, you can see the score details of each of your tasks.

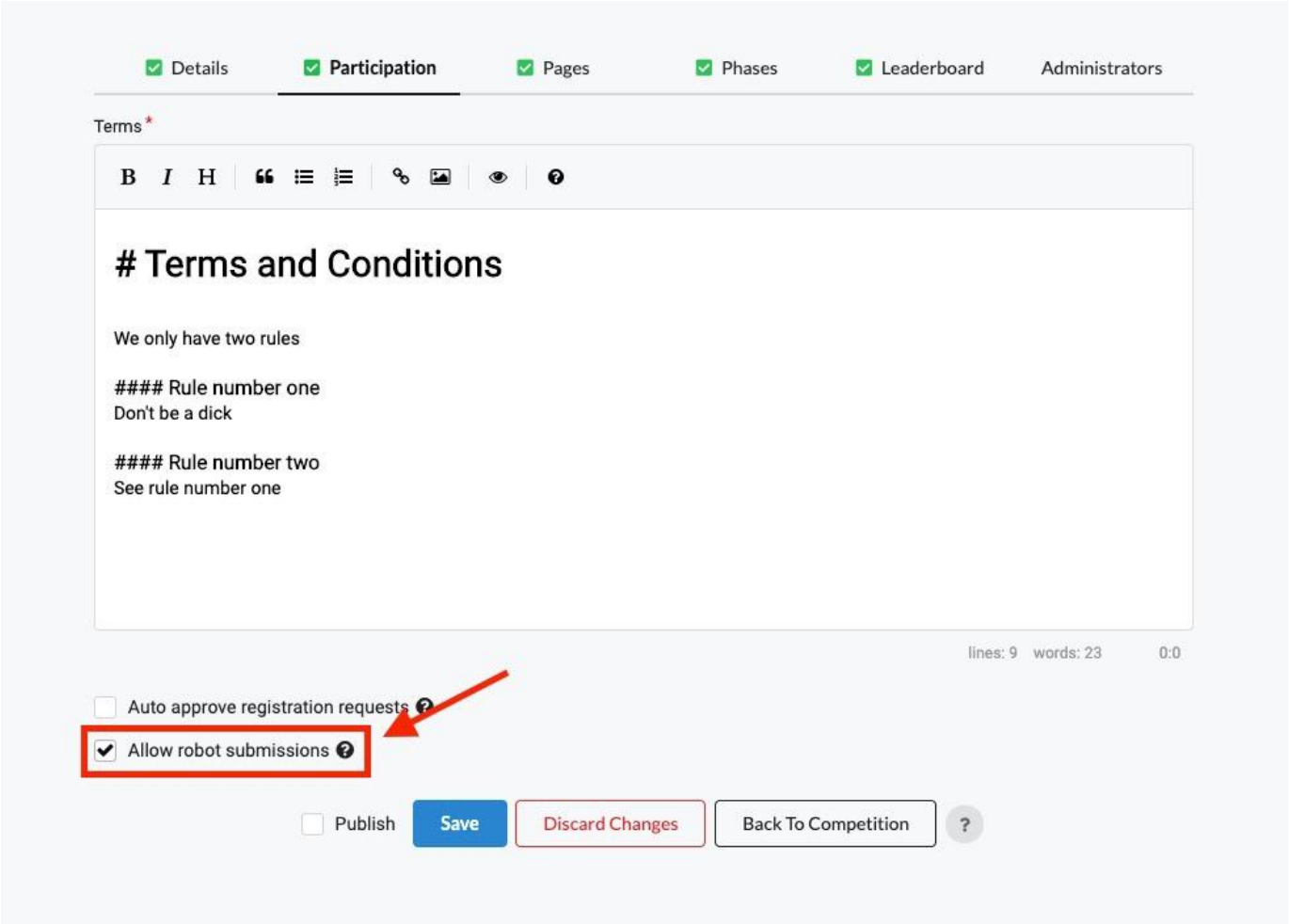


4.8.3 Using the Scripts:

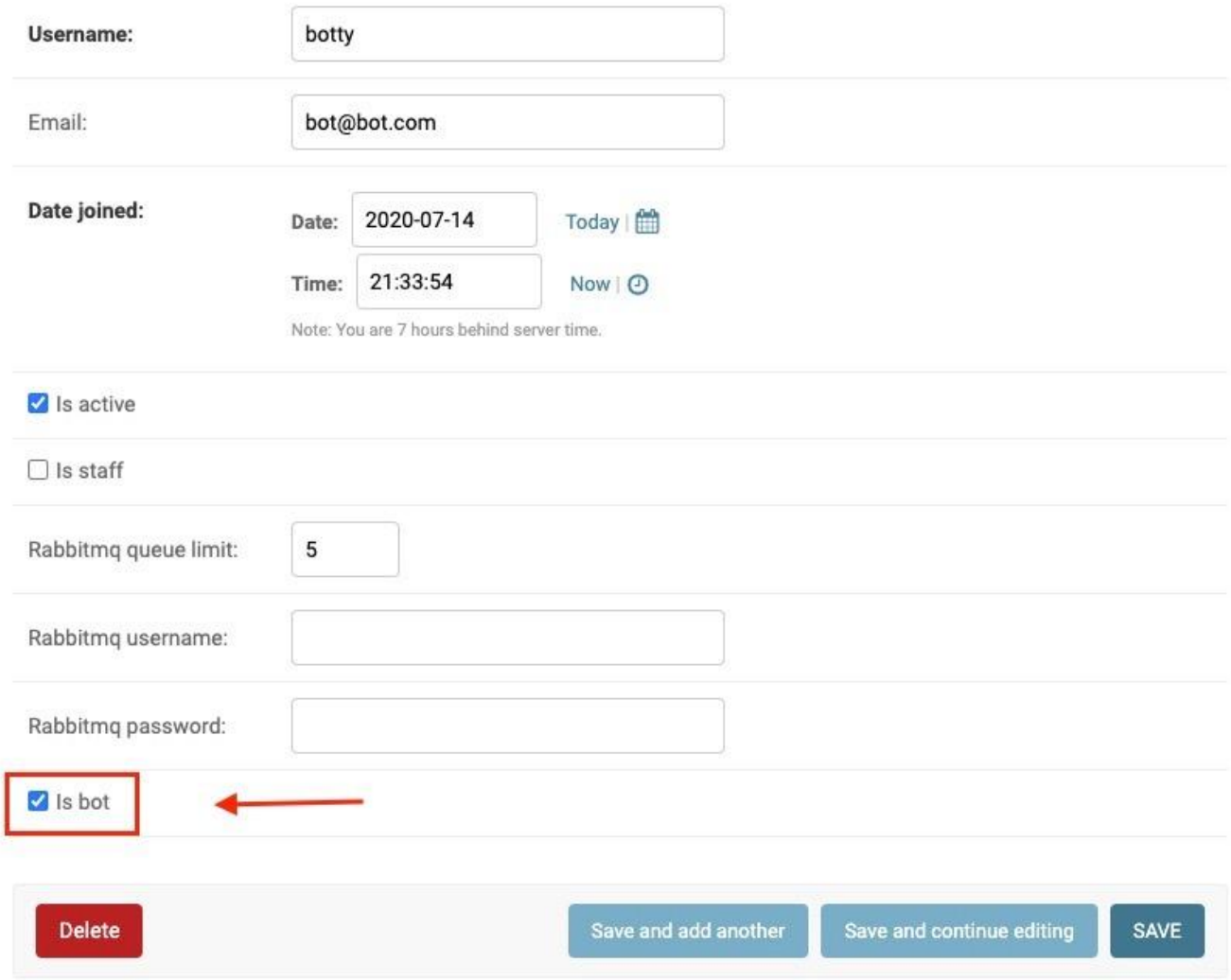
Setup:

- Create a competition with robot submissions enabled

Example competition bundle



• Create a user and enable the bot user flag on the Django admin page.



get_competition_details.py:

- Inside the `get_competition_details.py` script, configure these options:

```
33  # ---------------------------------
34  # Configure this
35  # ---------------------------------
36  CODALAB_URL = 'http://localhost/'
```

- `CODALAB_URL` can be changed if not testing locally.
- Run the `get_competition_details` script with no arguments.
- Find the competition you want to test on.
- Run the `get_competition_details` script again with the competition ID as the only argument.
- Find the phase you want to test on.
- If you want to use the task selection feature, run the script again with the competition ID as the first argument and the phase ID as the second argument.
- Select the task you would like to run your submission on.
- Use the phase ID and task IDs to configure `example_submission.py`.

---

example_submission.py:

- Inside the `example_submission.py` script, configure these options:

```
28  # ---------------------------------------------------------------------
29  # Configure these
30  # ---------------------------------------------------------------------
31  CODALAB_URL = 'http://localhost/'
32  USERNAME = 'admin'
33  PASSWORD = 'admin'
34  PHASE_ID = None
35  TASK_LIST = []
36  SUBMISSION_ZIP_PATH = '../tests/functional/test_files/submission.zip'
```

- `CODALAB_URL` can be changed if not testing locally.
- `USERNAME` and `PASSWORD` should correspond with the user being tested.
- `PHASE_ID` should correspond with the phase being tested on.
- `TASK_LIST` can be used to submit to specific tasks on a phase. If left blank, the submission will run on all tasks.
- `SUBMISSION_ZIP_PATH` should be changed if testing on anything but the default "Classify Wheat Seeds" competition. An example submission can be found here.
- Run this script in a python3 environment with `requests` library installed.

---

get_submission_details.py

- Configure the `get_submission_details.py` options before running.

```
30  # ----------------------------------
31  # Configure these
32  # ----------------------------------
33  CODALAB_URL = 'http://localhost/'
34  USERNAME = 'admin'
35  PASSWORD = 'admin'
```

- `CODALAB_URL` can be changed if not testing locally.
- `USERNAME` and `PASSWORD` should correspond with the user being tested.
- Run the `get_submission_details.py --phase <id>` script with the ID of the phase containing the desired submission.
- Find the ID of the desired submission.
- Run the `get_submission_details.py --submission <id>` script with the desired submission ID.
- The output of the script should be a submission object and a submission `get_details` object. This data can be used view scores, get prediction results, ect.
- Run the `get_submission_details.py --submission <id> -v` to save a zip containing previous info plus the original submission and logs.
- `--output <PATH>` can be used to choose where to save the zip file. Otherwise, it will be saved in the current directory.

---

rerun_submission.py

Robot users have the unique permission to rerun anyone's submission on a specific task. This enables clinicians to test pre-made solutions on private datasets that exist on tasks that have no competition.

- Configure the `rerun_submission.py` options before running.

```
30  # ----------------------------------
31  # Configure these
32  # ----------------------------------
33  CODALAB_URL = 'http://localhost/'
34  USERNAME = 'admin'
35  PASSWORD = 'admin'
```

- `CODALAB_URL` can be changed if not testing locally.
- `USERNAME` and `PASSWORD` should correspond with the user being tested.

Running the script

1. Create a competition that allows robots, and create a user marked as a robot user. Use that username and password below.
2. Get into a python3 environment with requests installed
3. Review this script and edit the applicable variables, like...

```
CODALAB_URL
USERNAME
PASSWORD
...
```

4. Execute the contents of this script with no additional command line arguments with the command shown below:

`./rerun_submission.py`

The script is built to assist the user in the selection of the submission that will be re-run.

1. After selecting a submission ID from the list shown in the previous step, add that ID to the command as a positional argument as shown below.

```
./rerun_submission.py 42
```

The script will assist the user in the selection of a task ID.

1. After selecting both a submission ID and a task ID, run the command again with both arguments to see a demonstration of a robot user re-running a submission on a specific task.

e.g.

```
./rerun_submission.py 42 a217a322-6ddf-400c-ac7d-336a42863724
```

4.9 Running Tests

```
# Without "end to end" tests
$ docker compose exec django py.test -m "not e2e"

# "End to end tests" (a shell script to launch a selenium docker container)
$ ./run_selenium_tests.sh

# If you are on Mac OSX it is easy to watch these tests, no need to install
# anything just do:
$ open vnc://0.0.0.0:5900

# And login with password "secret"
```

4.9.1 CircleCI

To simulate the tests run by CircleCI locally, run the following command:

```
docker compose -f docker-compose.yml -f docker-compose.selenium.yml exec django py.test src/ -m "not e2e"
```

4.9.2 Example competitions

The repo comes with a couple examples that are used during tests:

v2 test data

```
src/tests/functional/test_files/submission.zip
src/tests/functional/test_files/competition.zip
```

v1.5 legacy test data

```
src/tests/functional/test_files/submission15.zip
src/tests/functional/test_files/competition15.zip
```

Other Codalab Competition examples

https://github.com/codalab/competition-examples/tree/master/v2/

4.10 Automation

4.10.1 What and Why

It's useful to test various parts of the system with lots of data or many intricate actions. The selenium tests do this generally and are used as a guide for this tutorial. One problem is that Selenium needs to launch an instance of a browser to control. Our tests do this inside a docker container and uses a test database that doesn't persist as it cleans up after itself. We need to be able to control a live codabench session that is running. To do that we install a driver locally which is normally only inside the selenium docker container during tests. It is specific to your browser so keep that in mind.

4.10.2 Virtualenv

You'll need a python virtual env as you don't want to be inside Django or you won't be able to launch a browser.

Virtualenv

I used 3.8.

```
python3 -m venv codabench
source ./codabench/bin/activate
```

Pyenv

```
pyenv install 3.8
pyenv virtualenv 3.8 codabench
pyenv activate codabench
```

4.10.3 Requirements

We have a couple extra things like `webdriver-manager` for getting a driver programmatically and `selenium` needs to be upgraded to use modern client interface.

```
pip install -r requitements.txt
pip install -r requitements.dev.txt
pip install webdriver-manager
pip install --upgrade selenium
```

4.10.4 Automate competition creation

Main Selenium Docs
Install
Getting Started

```
import os, time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager

# Use `ChromeDriverManager` to ensure the `chromedriver` is installed and in PATH
service = Service(ChromeDriverManager().install())
driver = webdriver.Chrome(service=service)

# ... now use `driver` to control the local Chrome instance
driver.get("http://localhost/accounts/login")

# Use CSS selectors to find the input fields and button
username_input = driver.find_element(By.CSS_SELECTOR, 'input[name="username"]')
password_input = driver.find_element(By.CSS_SELECTOR, 'input[name="password"]')
submit_button = driver.find_element(By.CSS_SELECTOR, '.submit.button')

# Type the credentials into the fields
username_input.send_keys('bbearce')
password_input.send_keys('testtest')

# Click the submit button
submit_button.click()
```

```
comp_path = "/home/bbearce/Documents/codabench/src/tests/functional/test_files/competition_v2_multi_task.zip"
def upload_competition(competition_zip_path):
    driver.get("http://localhost/competitions/upload")
    file_input = driver.find_element(By.CSS_SELECTOR, 'input[ref="file_input"]')
    file_input.send_keys(os.path.join(competition_zip_path))


for i in range(30):
    upload_competition(comp_path)
    time.sleep(5) # tune for your system
```

4.11 Manual Validation

This is a checklist to follow in order to perform a manual validation of the platform's proper functioning. This is especially useful when validating a "bump" pull request made by the dependabot, a fresh installation or any change in the code base.

1. Create a user account and login
2. Create a competition
3. Create a queue
4. Upload a submission
5. Check that the submission was processed (results, visualization tab, leaderboard)
6. Change/recover password
7. Delete user
8. Delete submission
9. Delete queue
10. Delete competition
11. Admin page
12. Look at the logs for any problematic messages (`docker compose logs -f`)

4.12 Validation and deplyement of pull requests

4.12.1 1. Local testing and validation of the changes

Setup

Required: - "Maintain" role on the repository - A working local installation

Pull the changes, checkout the branch you are testing and deploy your local instance:

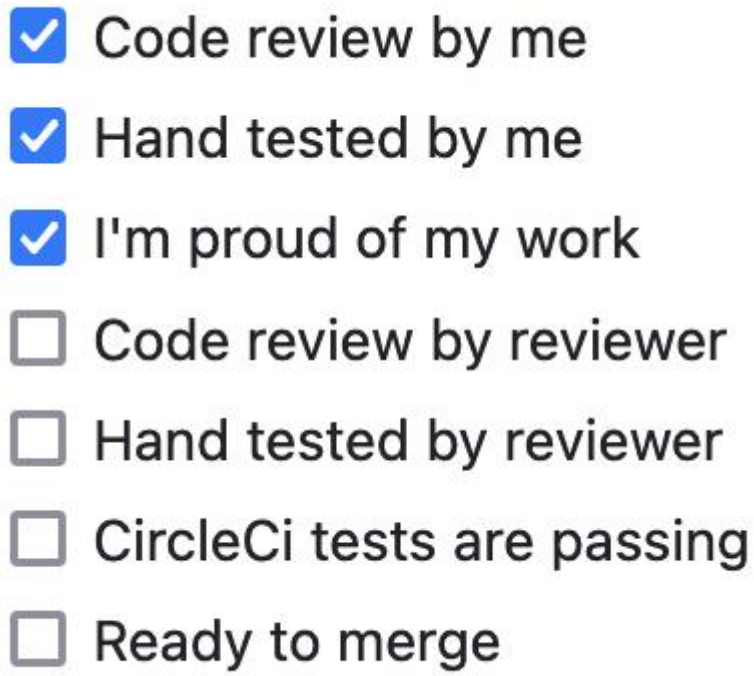
```
cd codabench
git pull
git checkout branch
docker compose up -d
```

If necessary, migrate and collect static files (see this page).

> **Note**
>
> If the branch with the changes is from an external repository, you can create a branch in Codabench's repository and make a first merging into this new branch. Then, you'll be able to merge the new branch into master. This way, the automatic tests will be triggered.
>
> *EDIT: It may be possible to trigger the tests even if the branch is external. To be confirmed.*

Testing

Here is the usual checklist in order to validate the pull request:

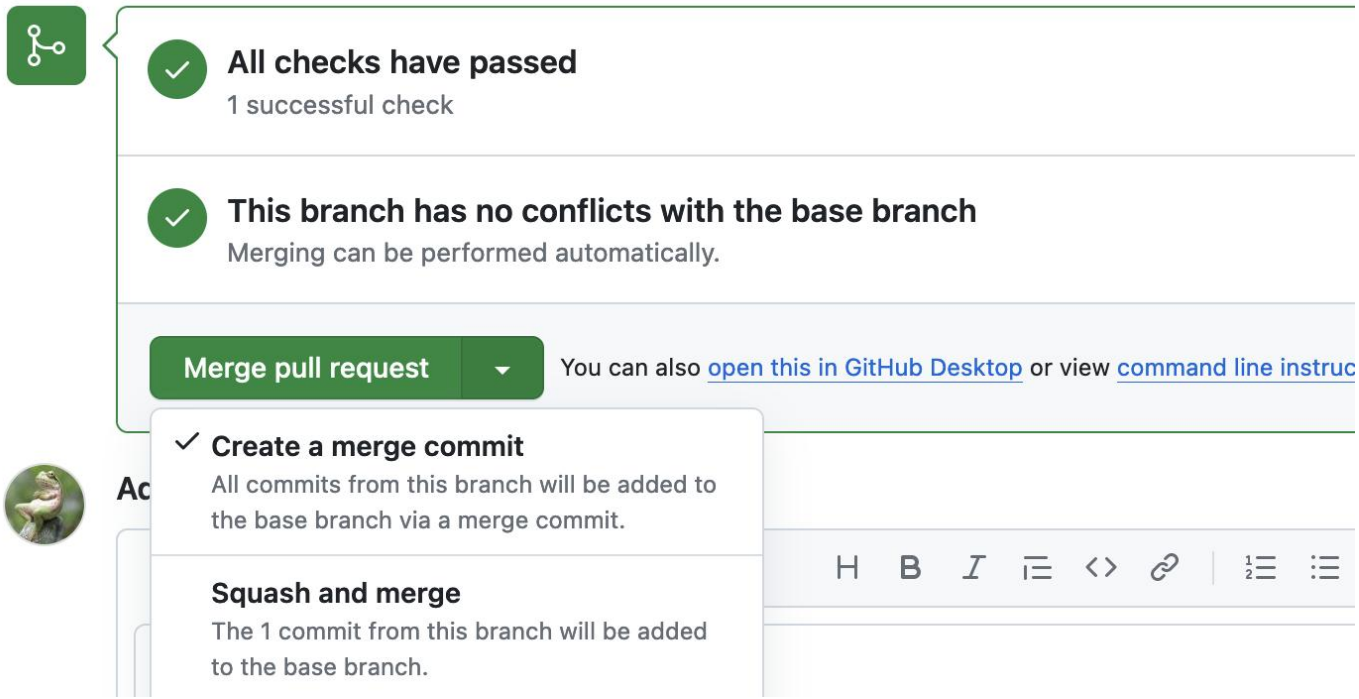


The contributor may have provided guidelines for testing that you should follow. In addition to this:

- Testing must be thorough, really trying to break the new changes. Try as many use cases as possible. Do not trust the contributor.
- In addition to the checklist of the PR, you can follow this checklist to check that the basic functionalities of the platform are still working: manual validation

Merging

Once everything is validated, merge the pull request. If there are many minor commits, use "squash and merge" to merge them into one.



You can then safely click on `Delete branch`. It is a good practice to keep the project clean.

4.12.2 Update the test server

Here are the necessary steps to update the Codabench server to reflect the last changes. We prodive here general guidelines that work for both the test server and the production server.

Log into the server

```
ssh codabench-server
cd /home/codalab/codabench
```

Replace `codabench-server` by your own SSH host setting, or the IP address of the server.

> **Note**
>
> Make sure to log in as the user that deployed the containers.

Pull the last change

> **Tip**
>
> - If you are deploying on a test server, you can use the `develop` branch.
> - If you are deploying on a Production server, we strongly adivse on using the `master` branch.

```
docker compose down
git status
git pull
docker compose up -d
```

Restart Django

```
docker compose stop django
docker compose rm django
docker compose create django
docker compose start django
```

If `docker compose` does not exist, use `docker-compose`.

Database migration

```
docker compose exec django ./manage.py migrate
```

> **Do not use `makemigrations`**

> ***Remark:** we need to solve the migration files configuration. In the meantime, `makemigrations --merge` may be needed.*

Collect static files

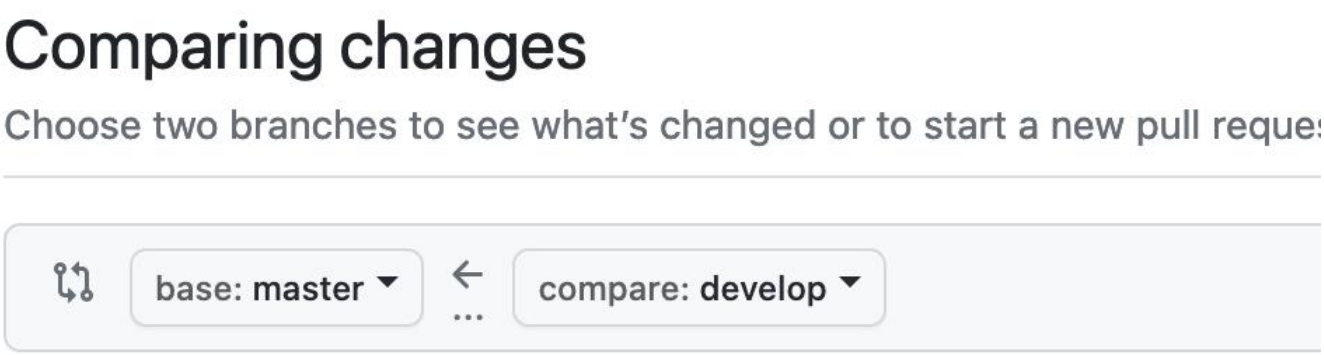
```
docker compose exec django ./manage.py collectstatic --noinput
```

Final testing

- Access the platform from your browser
- You may need a hard refresh (Maj + R) so the changes take effect.
- Check that everything is working fine, the new changes, and the basic functionalities of the platform

4.12.3 Merge develop into master

Once some pull requests (~3 - 10) were merged into `develop`, we can prepare a merge into `master`. Simply create a new pull request from Github interface, selecting `master` as the base branch:



In the text of the PR, link all relevant PR made to develop, and indicate the URL of the test server. Example: https://github.com/codalab/codabench/pull/1166

4.12.4 Update the production server

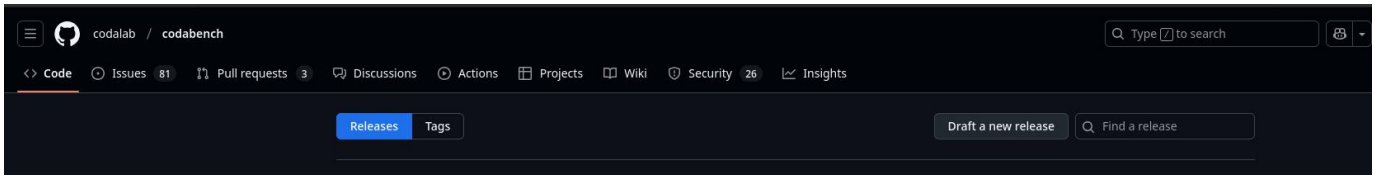
Same procedure as Update the test server, but on the production server.

> **Tip**
>
> **Do not forget to access the platform and perform a final round of live testing after the deployment.**

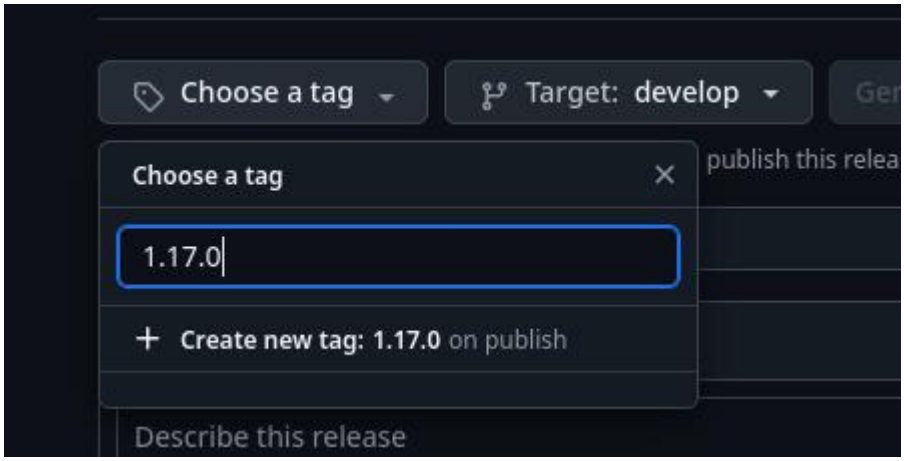
4.12.5 Creating a Release

Once the develop branch has been merged into master, it is possible to use the Github interface to tag the commit of the merge and create a release containing all the changes as well as manual interventions if needed.

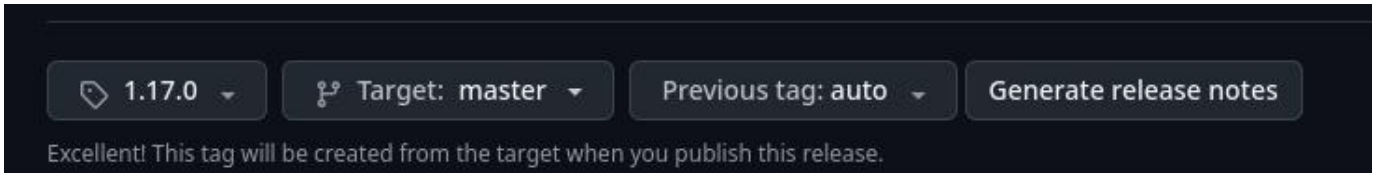
For this, you will need to go to the release page and click on `Draft a new release`



Afterwards, you click on `Choose a tag` and enter the tag you want to create (in this exemple, 1.17.0 which doesn't exist yet)

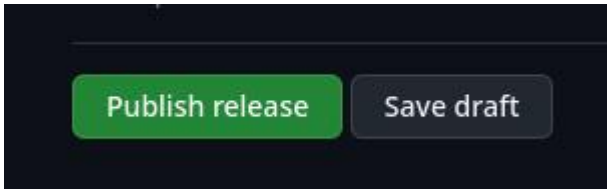


You can then choose the targeted branch to create the tag on (`master` in our case) and then click on `Generate release notes` Github will automatically generate releases based on the new commits compared to the last tag.



You can then change the text format however you like, as well as add things like Manual Intervention if there are any.

When you are done, you publish the release.



4.12.6 TODO

Add a note about:

- Merging the github action PR to update the release tag

4.13 Upgrading Codabench

4.13.1 Index

Upgrade Codabench

```
cd codabench
git pull
docker compose build && docker compose up -d
docker compose exec django ./manage.py migrate
docker compose exec django ./manage.py collectstatic --noinput
```

Manual interventions

You can find here various manual intervention needed depending on which version you are upgrading from:

- Upgrade RabbitMQ (version < 1.0.0)
- Create new logos for each competition (version < 1.4.1)
- Worker Docker Image manual update (version < 1.3.1)
- Add line into `.env` for default queue worker duration (version < 1.7.0)
- Uncomment a line in your `.env` file (version < 1.8.0)
- Rebuilding all docker images (version < 1.9.2)
- Move the last storage_inconsistency files from logs folder to var logs folder (version < 1.12.0)
- Submissions and Participants Counts (version < 1.14.0)
- Homepage counters (version < 1.15.0)
- User removal (version < 1.17.0)
- Database size Fix (version < 1.18.0)

4.13.2 Upgrade RabbitMQ (version < 1.0.0)

This intervention is needed when upgrading from a version equal or lower than v1.0.0

Backup RabbitMQ settings

Go to `http://<instance_ip>:<rabbitmq_admin_port>/api/definitions` and save the response (enter login and password as configured in `.env`)

For example:


```
{"rabbit_version":"3.6.16","users":[{"name":"e8f3bafa-fb4f-4ff1-ab5e-3239c0c1da15","password_hash":"gSTBWx8mRSIq7zzUTcFMag/6UasMvFLy8Piqwds6lOUsm3ef","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},
{"name":"3b6016ad-680d-4db5-ba4d-3def8a091218","password_hash":"WPc1BduFsXCjARuHVhTvMKPOPbHt/lNOz8fsP33+auwf7u93","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"86b7c62c-5811-45b3-b31b-
3195156cb8a0","password_hash":"1jg4qU2S7SROgY/oSF4a6iCKYZ9frhqbv0lrirzD0zl1ItOu","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"d3a388cd-36ff-45f6-9af2-
7692fb0b1d61","password_hash":"0Fde0loUEcwgTBBWaBkDTe2n6jRVgpcpworJdAGEtYRrWTb+","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"1e3c8c33-2db2-4a5d-b984-
e1acd669e810","password_hash":"oNIaw1G5yBQjjymSBcpSgtd3Bb72a38sa5eEyf28Qp7UcOJw","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"30771a85-7b8a-4edc-a64d-
07197eab335a","password_hash":"iocqnM4tuzIt8ATwBYnQAF58YFEL3uWtsCBsNhWi2oxu/Xda","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"6e2a27a7-48a7-4717-9aaf-
4066219b99e1","password_hash":"/HMDrstr7bBdPTFeBMPeXj2Qg8FfoRRF9Ftqm0ANPTgH/SM5","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"877be605-1875-448b-bf11-
15f50f68e21b","password_hash":"zV9/9Fyd3Zz8dbRuOblnsJ/oavMfP/TEOQ3Y9TFZ4q7oSEOa","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"74b603ce-918b-48b5-87d0-
8489ab5857c0","password_hash":"iHV8LAzkYaDaODEuay/mUPGPjZvC9TkBl+o41TlJHvqL9yCj","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"221f21c9-07fb-4c91-b338-
d215b9a17aef","password_hash":"F6CjmewsCRUsCWC3m2w+YmuPZWlMRGqYGeY7ymSdlxtlhrWB","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"f0df8d1c-2afd-432f-a2a7-
b6da2f632d33","password_hash":"VQZ6V2NP5rysr6NcM8NEWGIg94LRnRqWWE/teJnrVEugw7gU","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"54f6cd50-2ce0-4828-ac93-
0881c401e2a0","password_hash":"yXPSsj7aIOYmxQVQe/6bJ3TStdBjfddvGGyd6qm/gN2pM9X0","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"99f319c9-1b07-4d85-998a-
ae3b5b997c6c","password_hash":"5ZLMubx1K5DkRVrY3SEmPFIc7FKQvVq408V3uMZlnyPgyG+p","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"17a1d574-3f53-4c6d-8656-
c026a09c48c0","password_hash":"PGBWmvzv+8r1rIRXHdAlvQOmWupdmagojptwuuiH9fYHue+4","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"8520206a-da88-479b-b5d4-
bd57dd218afb","password_hash":"yKIHVexpHOIaKX/vHpG+kLroH2NbuP95Lyj1w9KxWdZhgoE0","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"71ed84c3-e7d6-4784-9254-
57d7a4b38fc3","password_hash":"XkLCVR104oXALUW8MgzAGoL4XG9aB4wfJfgJWdV1oO8T6JS0","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"03dd79e7-ba2e-40d2-8992-
d7701b16f945","password_hash":"KMUMob5g9e/8PK8Lo2ptksX8zkYRQ+N6nXmgAigBFoSszrci","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"9f66194f-5a16-412c-aebc-
31464ac0d5be","password_hash":"Q5feVuGmdSCKVl3YDVsapstm+SdHBrnh11gwNSK6wLu7whqY","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"0ac6a78a-c6a7-4c6c-a520-
d0d9a9022bed","password_hash":"sE3HgaWVl6KWHKSs7vuA+rDLSO3YalAq1qnpDy9DR4e4m/yf","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"5d6599cd-f225-4579-bb80-
ff7204150fdf","password_hash":"aexenJZvsJKBnaMTWLqn9wz8EgENzf5o5jy8X4qjX7I6U4LN","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"7998b214-7a4b-4a37-9b92-
47aeaf42bba5","password_hash":"r705lAZjeg3tdXp5hAjZRNEKg5mRjm+gRKKyheNDFlM3zqlR","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"89aa1722-edef-41de-9b55-
541b58582c29","password_hash":"EyQvocKVv7cf9JfXTOwd7NLhDAQFv0IPUdJANjCJIRG3PIRM","hashing_algorithm":"rabbit_password_hashing_sha256","tags":""},{"name":"ea4c0666-e97e-4d5a-bca3-
```


Do not submit any submission and wait until all submissions are processed

Stop and remove RabbitMQ's container and data

```
docker compose stop rabbit && docker compose rm rabbit
sudo rm -rvf var/rabbit/*
```

Switch to the latest RabbitMQ version

Add `WORKER_CONNECTION_TIMEOUT=<your timeout value>` into your `.env` file with your custom value. Then execute:

```
git pull
docker compose build rabbit
docker compose up -d
```

Restore the backup settings

Connect to the instance by ssh and upload your json file at 1st step, execute :

```
curl -u <login>:<password> -H "Content-Type: application/json" -X POST -T <your definitions file>.json http://localhost:<rabbit_admin_port>/api/definitions
```

You can check if it succeeded by doing the 1st step.

Verify that your submission can be processed.

4.13.3 Create new logos for each competitions (version < 1.4.1)

This intervention is needed when upgrading from a version equal or lower than v1.4.1

In order to create a "logo icon" for each existing competition

1. Shell into django

```
docker compose exec django bash
python manage.py shell_plus --plain
```

2. Get competitions that don't have logo icons

```
import io, os
from PIL import Image
from django.core.files.base import ContentFile
comps_no_icon_logo = Competition.objects.filter(logo_icon__isnull=True)
all = Competition.objects.all()
len(Competition.objects.all())
len(comps_no_icon_logo)
```

3. Then run this script

```
for comp in comps_no_icon_logo:
    try:
        comp.make_logo_icon()
        comp.save()
    except Exception as e:
        print(f"An error occurred: {e}")
        print(comp)
```

4.13.4 Worker docker image manual update (version < 1.3.1)

This intervention is needed when upgrading from a version equal or lower than v1.3.1

To update your worker docker image, you can launch the following code in the terminal on the machine where your worker is located.

```
docker stop compute_worker
docker rm compute_worker
docker pull codalab/competitions-v2-compute-worker:latest
docker run \
    -v /codabench:/codabench \
    -v /var/run/docker.sock:/var/run/docker.sock \
    -d \
    --env-file .env \
    --name compute_worker \
    --restart unless-stopped \
    --log-opt max-size=50m \
    --log-opt max-file=3 \
    codalab/competitions-v2-compute-worker:latest
```

4.13.5 Add line in .env file for default worker queue duration (version < 1.7.0)

This intervention is needed when upgrading from a version equal or lower than v1.7.0

You will need to add the following line into your `.env` file

.env

```
MAX_EXECUTION_TIME_LIMIT=600 # time limit for the default queue (in seconds)
```

This will change the maximum time a job can run on the default queue of your instance, as noted here

4.13.6 Uncomment a line in your .env file (version < 1.8.0)

This intervention is needed when upgrading from a version equal or lower than v1.8.0

After the Caddy upgrade, you will need to uncomment a line in your `.env` file:

.env

```
TLS_EMAIL = "your@email.com"
```

More information here

4.13.7 Rebuilding all docker images (version < 1.9.2)

This intervention is needed when upgrading from a version equal or lower than v1.9.2

Since we are now using Poetry, we need to rebuild all our docker images to include it.

You can achieve this by running the following commands :

Warning

If your machine has other images or containers that you want to keep, do not run `docker system prune -af`. Instead, manually delete all the images related to codabench

```
docker compose build && docker compose up -d
docker system prune -a
```

More information (and alternative commands) here

4.13.8 Move the latest storage_inconsistency files from the logs folder to var/logs (version < 1.12.0)

This intervention is needed when upgrading from a version equal or lower than v1.11.0

You will need to move the last storage_inconsistency files from /logs folder to /var/logs/ folder.

```
cd codabench
cp -r logs/* var/logs
```

You will also need to rebuild the celery image because of a version change that's needed.

```
docker compose down
docker images # Take the ID of the celery image
docker rmi *celery_image_id*
docker compose up -d
```

More information here

4.13.9 Submissions and Participants count (version < 1.14.0)

After upgrading from Codabench <1.14, you will need to follow these steps to compute the submissions and participants counts on the competition pages. See this for more information

1. Re-build containers

```
docker compose build && docker compose up -d
```

2. Migration

```
docker compose exec django ./manage.py migrate
```

3. Update counts for all competitions

Bash into django console

```
docker compose exec django ./manage.py shell_plus
```

Import and call the function

```
from competitions.submission_participant_counts import compute_submissions_participants_counts
compute_submissions_participants_counts()
```

4. Feature some competitions in home page

There are two ways to do it: 1. Use Django admin -> click the competition -> scroll down to is featured filed -> Check/Uncheck it 2. Use competition ID in the django bash to feature / unfeature a competition

```
docker compose exec django ./manage.py shell_plus
```

```
comp = Competition.objects.get(id=<ID>)  # replace <ID> with competition id
comp.is_featured = True  # set to False if you want to unfeature a competition
comp.save()
```

4.13.10 Homepage Counters (version < 1.15.0)

After upgrading from Codabench <1.15, you will need to follow these steps to compute the homepage counters. See this for more information

1. Re-build containers

```
docker compose build && docker compose up -d
```

1. Update the homepage counters (to avoid waiting 1 day)

```
docker compose exec django ./manage.py shell_plus
```

```
from analytics.tasks import update_home_page_counters
eager_results = update_home_page_counters.apply_async()
```

4.13.11 User Removal (version < 1.17.0)

After upgrading from Codabench <1.17, you will need to perform a Django migration (#1715, #1741)

```
docker compose exec django ./manage.py migrate
```

4.13.12 Database size fix (version < 1.18.0)

> **Warning**
>
> You need to stop the database from being changed while running these commands. They might take time to complete depending on the size of your database.

Start maintenance mode:

```
touch maintenance/maintenance.on
```

1. DJANGO MIGRATION (1774, 1752)

```
docker compose exec django ./manage.py migrate
```

2. RESET USER QUOTA FROM BYTES TO GB (1749)

```
docker compose exec django ./manage.py shell_plus
```

```
from profiles.quota import reset_all_users_quota_to_gb
reset_all_users_quota_to_gb()
```

```
# Convert all 16 GB quota into 15 GB
from profiles.models import User
users = User.objects.all()
for user in users:
    # Reset quota to 15 if quota is between 15 and 20
    # Do not reset quota for special users like adrien
    if user.quota > 15 and user.quota < 20:
        user.quota = 15
        user.save()
```

3. IMPORTANT FOR FILE SIZES CLEANUP (1752)

We have some critical changes here so before deployment we should run the following 3 blocks of code to get the last ids of `Data`, `Submission` and `SubmissionDetail`

Then, in the shell_plus:

```
# Get the maximum ID for Data
from datasets.models import Data
latest_id_data = Data.objects.latest('id').id
print("Data Last ID: ", latest_id_data)
```

```
# Get the maximum ID for Submission
from competitions.models import Submission
latest_id_submission = Submission.objects.latest('id').id
print("Submission Last ID: ", latest_id_submission)
```

```
# Get the maximum ID for Submission Detail
from competitions.models import SubmissionDetails
latest_id_submission_detail = SubmissionDetails.objects.latest('id').id
print("SubmissionDetail Last ID: ", latest_id_submission_detail)
```

After we have the latest ids, we should deploy and run the 3 blocks of code below to fix the sizes i.e. to convert all kib to bytes to make everything consistent. For new files uploaded after the deployment, the sizes will be saved in bytes automatically that is why we need to run the following code for older files only.

```
# Run the conversion only for records with id <= latest_id
from datasets.models import Data
for data in Data.objects.filter(id__lte=latest_id_data):
    if data.file_size:
        data.file_size = data.file_size * 1024  # Convert from KiB to bytes
        data.save()
```

```
# Run the conversion only for records with id <= latest_id
from competitions.models import Submission
for sub in Submission.objects.filter(id__lte=latest_id_submission):
    updated = False  # Track if any field is updated
```

```
    if sub.prediction_result_file_size:
        sub.prediction_result_file_size = sub.prediction_result_file_size * 1024  # Convert from KiB to bytes
        updated = True

    if sub.scoring_result_file_size:
        sub.scoring_result_file_size = sub.scoring_result_file_size * 1024  # Convert from KiB to bytes
        updated = True

    if sub.detailed_result_file_size:
        sub.detailed_result_file_size = sub.detailed_result_file_size * 1024  # Convert from KiB to bytes
        updated = True

    if updated:
        sub.save()
```

```
# Run the conversion only for records with id <= latest_id
from competitions.models import SubmissionDetails
for sub_det in SubmissionDetails.objects.filter(id__lte=latest_id_submission_detail):
    if sub_det.file_size:
        sub_det.file_size = sub_det.file_size * 1024  # Convert from KiB to bytes
        sub_det.save()
```

Then, do not forget to stop maintenance mode:

```
rm maintenance/maintenance.on
```

5. Newsletters Archive

5.1 2024

5.1.1 CodaLab in 2024

A Year of Breakthroughs and New Horizons with Codabench

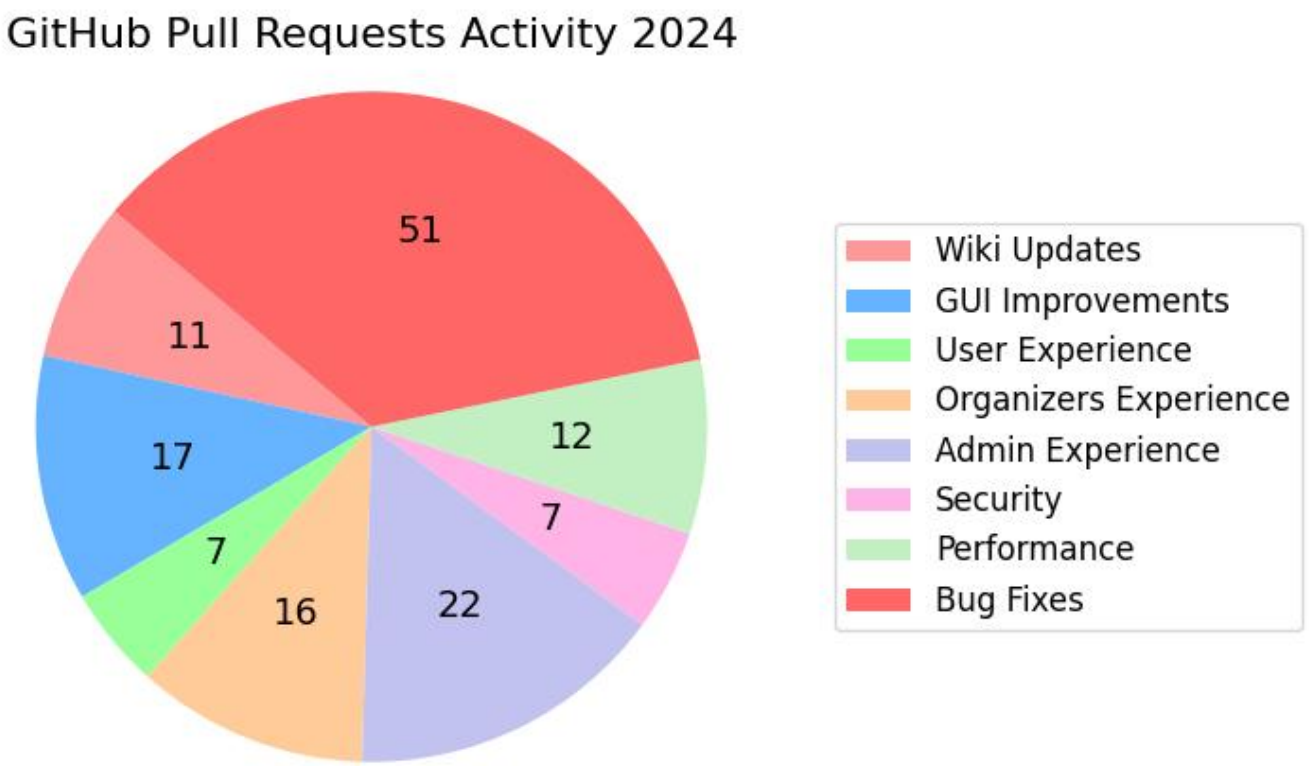
Welcome to the first edition of CodaLab's newsletter! This year has been full of novelty, success, and scientific progress. The platform is breaking records of participation and number of organized competitions, and Codabench, the new version of CodaLab, had a very promising launch. Let's dive into more details.

5.1.2 Unprecedented engagement

In October, Codabench has registered its **10,000th user**! From about 100 daily submissions in January, it is now **more than 500 daily submissions** that are handled on the servers. In 2024, more than 20,000 new users have registered on CodaLab, whereas more than 12,000 on Codabench.

249 public competitions have been created on CodaLab, whereas **193 on Codabench: +15%** of total competitions of both platforms compared to 2023. CodaLab continues to handle a large number of submissions, as many past competitions remain active even after they have officially ended: **240,000 submissions on CodaLab** whereas **83,000 on Codabench**.

Contributors community is very active with **143 pull requests** this year. Since the platform is still relatively new, the primary focus has been on bug fixes, security and performance enhancements, and administrative features, accounting for approximately two-thirds of the pull requests. Nevertheless, we are keen on improving the experience for both participants and organizers. We have set a versioning and a release-notes follow-up to give more visibility to the platform evolution and maturity.



5.1.3 Introducing Codabench

Codabench, the modernized version of CodaLab, was released in summer 2023, and presented at JCAD days in November 2024! Codabench platform software is now concentrating all development effort of the community. In addition to CodaLab features, it offers improved performance, live logs, more transparency, data-centric benchmarks and more!

We warmly encourage you to use codabench.org for all your new competitions and benchmarks. Note that CodaLab bundles are compatible with Codabench, easing the transition, as explained in the following Wiki page: How to transition from CodaLab to Codabench

CodaLab and Codabench are hosted on servers located at Paris-Saclay university, maintained by LISN lab.

5.1.4 Spotlight on competitions

The most popular competition this year, featuring **520 participants**, was the SemEval task Bridging the Gap in Text-Based Emotion Detection. The task of this competition focused on identifying the emotion that most people would associate with a speaker based on a given sentence or short text snippet.

The competition with the highest reward, a **prize pool of $100,000**, was the Global Artificial Intelligence Championships, track Maths 2024, a pioneering contest that aimed to advance the development of artificial intelligence tools for solving advanced mathematical problems across multiple levels of difficulty (full report here).

Codabench featured interesting **NeurIPS 2024 challenges**:

- The LLM Privacy Challenge, where participants were divided into two teams: Red Team and Blue Team, each one aiming at developing attack and defense approaches for data privacy in LLMs.
- Fair Universe - Higgs Uncertainy Challenge, exploring uncertainty-aware AI techniques for High Energy Physics (HEP).
- The Concordia Challenge, focusing on improving the cooperative intelligence of AI systems: promise-keeping, negotiation, reciprocity, reputation, partner choice, compromise, and sanctioning.
- The Erasing the Invisible Challenge, where the task is to remove invisible watermarks from images, while preserving the overall image quality.
- Codabench also hosted the NeurIPS 2024 Checklist Assistant, a self-service tool for the NeurIPS 2024 checklist. This verification assistant helps authors confirm their papers' compliance and submit improved versions based on the Assistant's feedback. For more details check out the blog post and the experiment results.

Last but not least, Codabench featured several competitions in the fields of medicine and biology, such as the Dental CBCT Scans, Panoramic X-ray Images, Butterfly Hybrid Detection, and Monitoring Age-related Macular Degeneration Progression In Optical Coherence Tomography.

We'd like to thank the whole community for these exceptional scientific contributions. You can explore more challenges in the public listing.

5.1.5 What about the future?

We always develop new features to serve the state-of-the art of Machine Learning science. We plan to invest effort in **handling medical data**, improving **federated learning** use-cases, and allowing **human-in-the loop** feedback for better AI experience.

The **platform interface will also be improved** for better visibility and facilitated re-use of datasets, tasks or solutions generated through it.

Keep in touch and give us feedback on your experience on Codabench !

5.1.6 Community

Reminder on our communication tools:

- Join our google forum to emphasize your competitions and events
- Contact us for any question: info@codabench.org
- Write an issue on github about interesting suggestions

Please cite one of these papers when working with our platforms:

```
@article{codabench,
   title = {Codabench: Flexible, easy-to-use, and reproducible meta-benchmark platform},
   author = {Zhen Xu and Sergio Escalera and Adrien Pavão and Magali Richard and
             Wei-Wei Tu and Quanming Yao and Huan Zhao and Isabelle Guyon},
   journal = {Patterns},
   volume = {3},
   number = {7},
   pages = {100543},
   year = {2022},
   issn = {2666-3899},
   doi = {https://doi.org/10.1016/j.patter.2022.100543},
   url = {https://www.sciencedirect.com/science/article/pii/S2666389922001465}
}
```

```
@article{codalab_competitions_JMLR,
 author  = {Adrien Pavao and Isabelle Guyon and Anne-Catherine Letournel and Dinh-Tuan Tran and Xavier Baro and Hugo Jair Escalante and Sergio Escalera and
Tyler Thomas and Zhen Xu},
 title   = {CodaLab Competitions: An Open Source Platform to Organize Scientific Challenges},
 journal = {Journal of Machine Learning Research},
 year    = {2023},
 volume  = {24},
 number  = {198},
 pages   = {1--6},
 url     = {http://jmlr.org/papers/v24/21-1436.html}
}
```

5.1.7 Last words

Thank you for reading the first edition of our newsletter. We look forward to sharing more exciting updates, competitions, and breakthroughs with you soon. Until then, keep exploring, keep competing, and stay inspired!


CHA
LEARN

université
PARIS-SACLAY

LISN
LABORATOIRE INTERDISCIPLINAIRE
DES SCIENCES DU NUMÉRIQUE

cnrs

Région
îledeFrance

Google

Microsoft

Inria
inventeurs du monde numérique

Stanford
University

UNIVERSITAT DE
BARCELONA

anr

ANR

eit
Health

4Paradigm
第四范式

6. How you can contribue

6.1 Index

- Use Codabench by either participating in a competition or hosting a new competition.
- Find a bug? Got a feature request? Submit a GitHub issue.
- P1 issues are the most important ones.
- Submit pull requests on GitHub to implement new features or fix bugs (see the contributing section).
- Improve this documentation.
- Let others know about Codabench!

6.2 Contributing

6.2.1 Being a Codabench user

- Create a user account on Codabench
- Register on Codabench to this existing competition IRIS-tuto and make a submission (you can find the necessary files here): `sample_result_submission` and `sample_code_submission`. See this page for more information.
- Create your own private competition (you can find the necessary files here). See this page for more information.

6.2.2 Setting up a local instance of Codabench

- Follow the tutorial in codabench wiki. According to your hosting OS, you might have to tune your environment file a bit. Try without enabling the SSL protocol (doing so, you don't need a domain name for the server). Try using the embedded Minio storage solution instead of a private cloud storage.
- If needed, you can also look into How to deploy Codabench on your server

Using your local instance

- Create your own competition and play with it. You can look at the output logs of each different docker container.
- Setting you as an admin of your platform and visit the Django Admin menu.

6.2.3 Setting up an autonomous Compute Worker on a machine

- Configure and launch a compute worker docker container.
- Create a private Queue on your new own competition on the production server codabench.org
- Assign your own compute-worker to this private queue instead of the default queue.

7. FAQ

7.1 General questions

7.1.1 What is Codabench for?

Codabench benchmarks are aimed at researchers, scientists and other professionals who want to track algorithm performance via benchmarks or have participants participate in a competition to find the best solution to a problem. We run a free public instance at https://www.codabench.org/ and the raw code is on Github.

7.1.2 Can CodaLab competitions be privately hosted?

Yes, you can host your own CodaLab instance on a private or hosted server (e.g. Azure, GCP or AWS). For more information, see how to deploy Codabench on your server and local installation guide. However, most benchmark organizers do NOT need to run their own instance. If you run a computationally demanding competition, you can hook up your own compute workers in the backend very easily.

7.1.3 How to change my username?

You cannot change your username BUT you can change your display name which will then be displayed instead of your username. To change your display name, follow these instructions:

```
1. Login to Codabench
2. Click your username in the top right corner
3. Click `Edit Profile` in the list
4. Set a display name you want to use
5. Click `Submit` button to save changes
```

7.1.4 How to make a task public or use public tasks from other users?

Follow the detailed instruction here to know how you can make your task public and use other public tasks in your competitions.

7.1.5 How to delete my account?

Click on your account name on the top right of the website, then on `account`

7.2 Technical questions

7.2.1 Server Setup Issues

Many technical FAQ are already located in the deploy your own server instructions.

Questions already answered there:

- Getting `Invalid HTTP method` in django logs.
- I am missing some static resources (css/js) on front end.
- CORS error when uploading bundle.
- Logos don't upload from minio.
- Compute worker execution with insufficient privileges
- Securing Codabench and Minio

7.2.2 A library is missing in the docker environment. What do to?

How does Codabench use dockers?

When you submit code to the Codabench platform, your code is executed inside a docker container. This environment can be exactly reproduced on your local machine by downloading the corresponding docker image.

For participants

- If you are a **competition participant**, contact the competition organizers to ask them if they can add the missing library or program. They can either accept or refuse the request.

For organizers

- If you are a **competition organizer**, you can select a different competition docker image. If the default docker image (`codalab/codalab-legacy:py37`) does not suits your needs, you can either:
  - Select another image from DockerHub
  - Create a new image from scratch
  - Edit the default image and push it to your own DockerHub account

More information here.

7.2.3 Emails are not showing up in my inbox for registration

When deploying a local instance, the email server is not configured by default, so you won't receive the confirmation email during signup. In `.env` towards the bottom you will find:

.env

```
# Uncomment to enable email settings
#EMAIL_BACKEND=django.core.mail.backends.smtp.EmailBackend
#EMAIL_HOST=smtp.sendgrid.net
#EMAIL_HOST_USER=user
#EMAIL_HOST_PASSWORD=pass
#EMAIL_PORT=587
#EMAIL_USE_TLS=True
```

Uncomment and fill in SMPT server credentials. A good suggestion if you've never done this is to use sendgrid.

7.2.4 Robots and automated submissions?

What about robot policy, reckless, or malicious behavior? Codabench does not forbid the use of robots (bots) to access the website, provided that it is not done with malicious intentions to disturb the normal use and jam the system. A user who abuses their rights by knowingly, maliciously, or recklessly jamming the system, causing the system to crash, causing loss of data, or gaining access to unauthorized data, will be banned from accessing all Codabench services.

8. Contact Us

- The preferred way is via posting a GitHub issue.
- If you wish to get in touch with the community, you can use the Google Groups.
- In case of emergency

Send us an email