Technische Universität Berlin

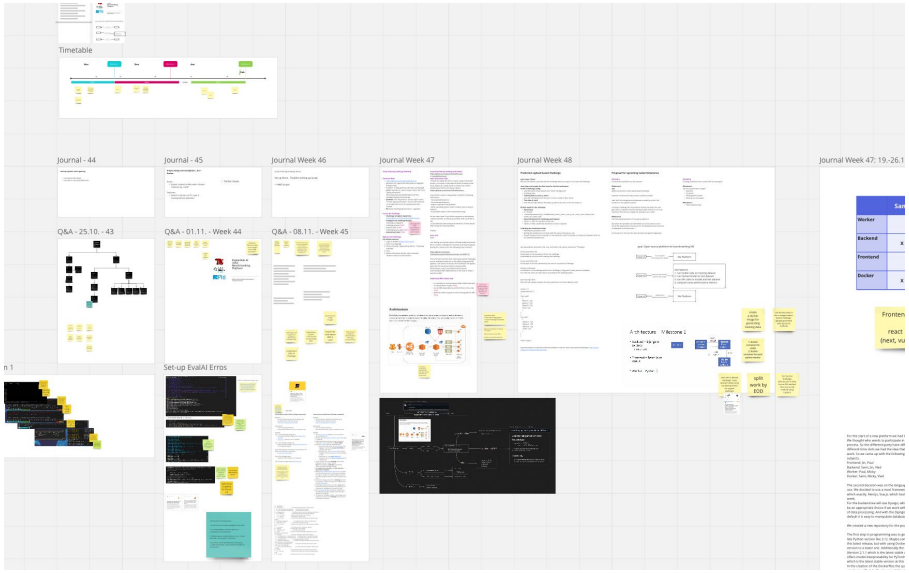
EvalxAI - Team Journal

Paul Eschenbach, Sami Hached, Daniel Jin Wodke, Michias Taye Zewdie, Uladzislau Bruila

Technische Universität Berlin

Week 1

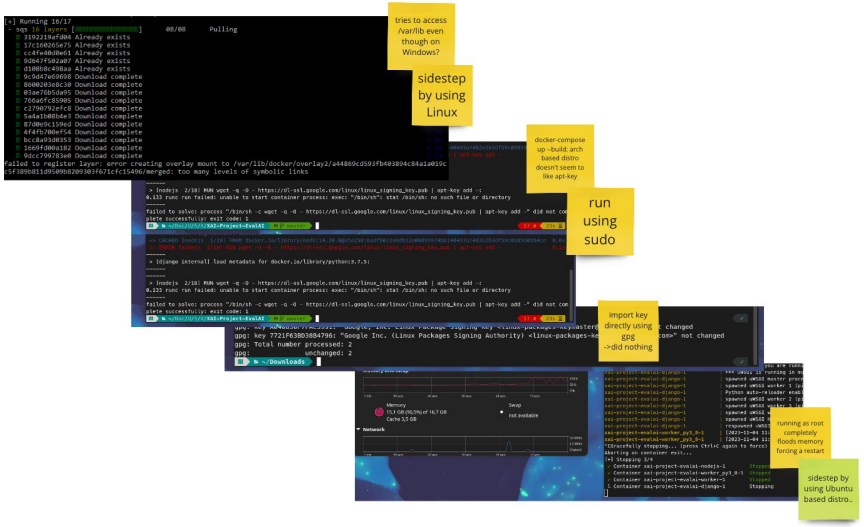
- Got to know the team.
- Created slack workspace for communication.
- Started researching xAI methods.
- Created miro board to gather ideas.

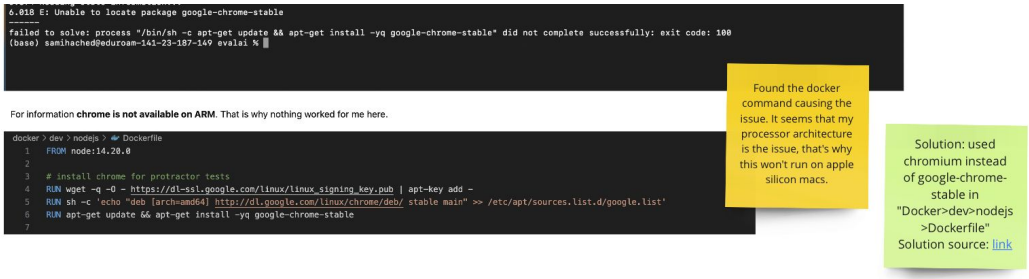


Technische Universität Berlin

Week 2

- Forked evalAI repo on github.
- Successfully installed the platform on windows/ubuntu (with intel CPUs).
- Failed to install on ARM CPUs.
- Continued reading papers on xAI methods.





Technische Universität Berlin

Week 3

- Successfully installed the platform on an ARM machine (m1 macbook).
- Defined work sections and divide tasks among team members.
- First attempt at hosting a challenge on evalAI.

workaround for arm-processors:

* install chromium instead of google-chrome-stable

* use matplotlib@3.1 instead of @3.3 to fix compatibility issue with python.

* different ways to install scikit-learn on m1: I chose pip install --no-use-pep517 scikit-learn

*Some linux commands broke when executed as a single line (idk why). I separated them and got the same results.

Following errors occurred while validating the challenge config: HTTPConnectionPool(host='127.0.0.1', port=8888): Max retries exceeded with url: //api/challenges/challenge/challenge_host_team/2/validate_challenge_config/ (Caused by NewConnectionError('<urllib3.connection.HTTPConnection object at 0x7ff36ed9c7d0>: Failed to establish a new connection: [Errno 111] Connection refused'))

Technische Universität Berlin

Week 4

After reading evalAI's documentation, and a lot of trying, these were out findings:

- There are 2 types of challenges:
  - evaliAI-hosted challenges (run on their servers and each challenge would require their approval)
  - locally-hosted challenges (require sending AWS credentials to evalAI to set up an AWS cluster)
- We decided to go with the second option, and remove AWS as a dependency from the code. The goal was to run the platform solely on our own machines.

Link tetris dataset repo : https://github.com/braindatalab/xai-tris

Use this!

Host prediction upload based challenge using config

Milestone 1
- Set-up prediction upload-based challenge using provided synthetic 8x8 Tetris dataset: https://arxiv.org/pdf/2306.12816.pdf

Requirements
- users can upload predictions
- users receive evaluation scores via evaluation script

Workflow
Step 1: Setup challenge configuration
- use sample challenge configuration: EvalAI-Starter template
- generating a repository from a template

Step 2: Edit challenge configuration
- define challenge settings challenge configuration docs in "challenge_config.yml"

Step 3: Edit evaluation script (metrics the submissions are going to be evaluated in each phase)
- evaluation script template

Step 4: Edit challenge details
- update the HTML templates in "/templates"

Step 5: Review and approvement by EvalAI team

Host prediction upload based challenge using github

Milestone 1
- Set-up prediction upload-based challenge using provided synthetic 8x8 Tetris dataset: https://arxiv.org/pdf/2306.12816.pdf

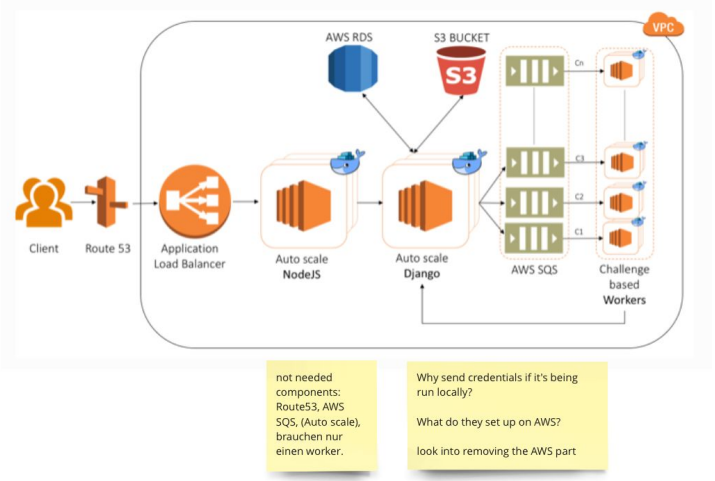
Requirements
- users can upload predictions
- users receive evaluation scores via evaluation script

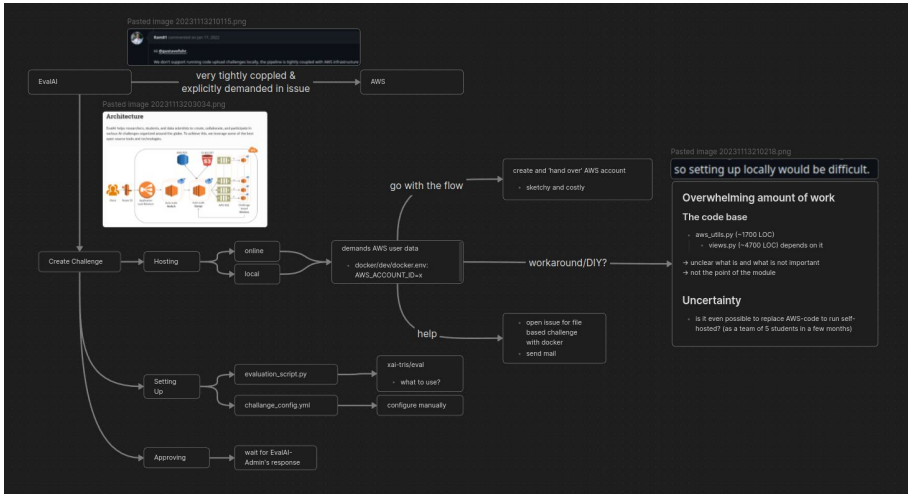
Workflow
1. Use this repository as template. You can read more about templates here.
2. Generate your github personal acccess token and copy it in clipboard.
3. Add the github personal access token in the forked repository's secrets with the name **AUTH_TOKEN**.
4. Now, go to EvalAI to fetch the following details -
   a. evalai_user_auth_token - Go to profile page after logging in and click on **Get your Auth Token** to copy your auth token.
   b. host_team_pk - Go to host team page and copy the **ID** for the team you want to use for challenge creation.
   c. evalai_host_url - Use **https://eval.ai** for production server and **https://staging.eval.ai** for staging server.
5. Create a branch with name **challenge** in the forked repository from the `master` branch.

Technische Universität Berlin

Week 5

The code turned out to heavily rely on AWS. We sought help from the evalAI team, but their response was “it’s not possible to run evalAI without AWS.”





Technische Universität Berlin

Change of plans: option B

Proposal for upcoming tasks/milestones

OPTION A
Continue the workflow in a slim manner

Milestone 2
aws
tightly connected to code upload based challenge

examine if and how the aws process could be avoided

idea: limit the storage processing power needed by restrict the scenario to one specific usecase

creating a challenge like the prediction based one where the user gets data or a specific machine learning model and has to run a xai method of their choice and gets an evaluation as a result

Milestone 3
Start the implementation of the eval.xai platform

Document the problems and workflows for a local version so far
Get all the dependencies necessary for the development process like its displayed in the flow chart ->

set the base for the next dev team (students programming group)

OPTION B
Creating a platform from scratch with the same goal

Milestone 2
Get the requirements straight

- Backend
- Frontend
- Working with Docker?
- Build up on your paper

Milestone 3

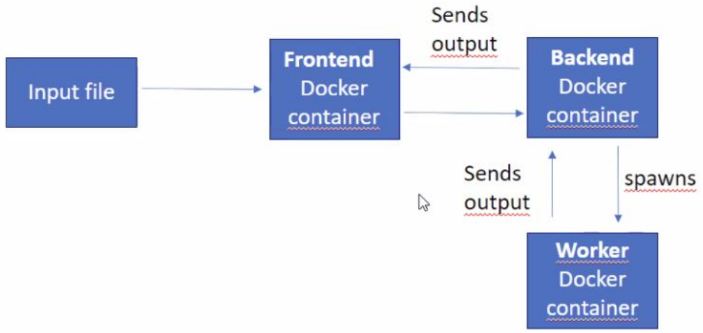
- Start implementing

Technische Universität Berlin

initial idea

Architecture – Milestone 1

- Backend – Django or Express
  - REST API
- Frontend – React.js or Vue.js
- Worker – Python :)



miro

Technische Universität Berlin

Weeks 6 - 7

- reorganised our group and assigned each component to 1 or 2 people.
- started learning how to use different technologies needed for each component, e.g. docker, django, react.
- created a skeleton project after following tutorials online for each technology (default django project +docker files).

	Sami	Paul	Micky	Jin	Vlad
Worker		xxx	x		
Backend	X			xx	x
Frontend		x		xxx	
Docker	X		X		x

Technische Universität Berlin

Week 8

Among some of the struggles we faced, were:

- connecting different docker containers for communication. The solution was to use docker compose which mounts all containers to the same docker network. Each container has its image name as its IP. Example: http://backend:8000.
- testing with docker was quite tricky and time-consuming. We used local virtual environments for development, then final testing of the workflow (or added feature) with docker before merging a branch. For this approach, we had to use localhost instead of “backend” and “frontend” for our URLs.

Technische Universität Berlin

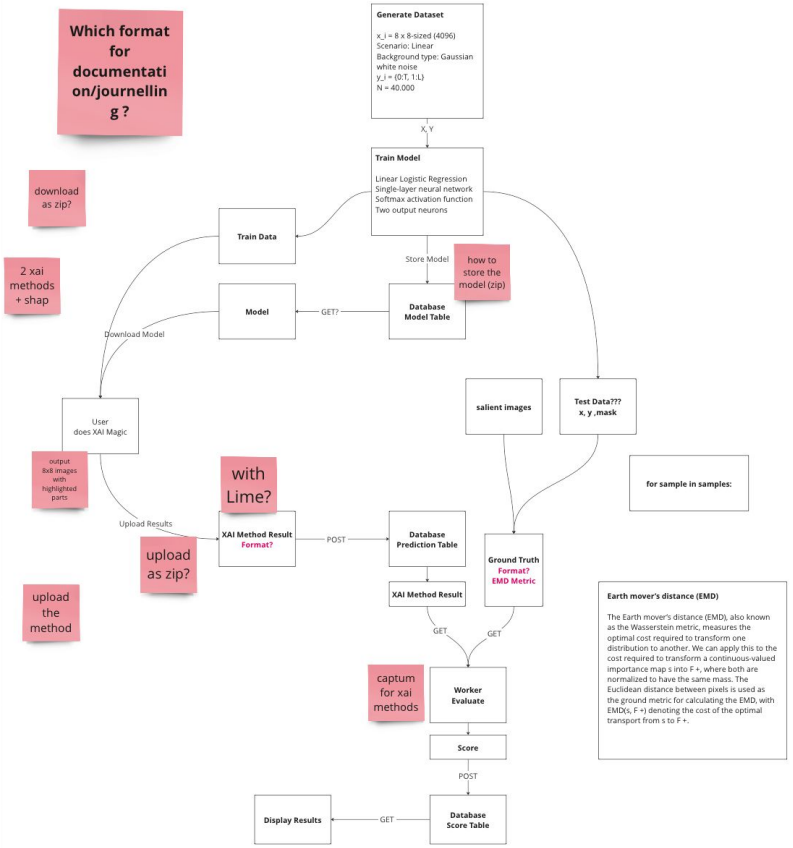
Week 9

- Finished the second milestone.
- Users could upload a json file containing a table of 0s and 1s, and they get a matching score for it in return.
- Our platform had a functioning frontend, backend, and database. All dockerized and able to communicate with each other.
- For this step, we were executing the evaluation script as part of the backend, and not in a separated worker container, yet.

Technische Universität Berlin

Week 10

- Agreed on the final structure of our platform.
- We struggled with spawning a worker container for each challenge. So, we decided to build a general worker that's always present, and managed to transfer the uploaded user input to it for evaluation. The results are then sent back using backend endpoints.
- On the frontend side, we added several pages to mimic the kaggle.com website, such as the sidebar and the search field.



Technische Universität Berlin

Week 11

- Added backend endpoints, as well as frontend buttons, to allow users to download the dataset and the ML model.
- Adjusted the evaluation script inside the worker to calculate the EMD distance, instead of compare two arrays of 0s and 1s.
- Presented our workflow to our supervisors.

Technische Universität Berlin

Week 12 - final presentation week

- Added another button and endpoint for users to download a template file to uploaded for each challenge. That way, users can know which functions and which types their code has to have.
- Added a workflow diagram to the homepage to guide users through the platform.
- Presented our platform to our department.

Technische Universität Berlin

Final Structure