Cnails

THE DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
計算機科學及工程學系

Cnails: "Cloud IDE tailor for UST programming courses"

Created by:

Chin, Tian You

Yeung, Man Lung Ken

Wong, Yan Ho

Supervised By:

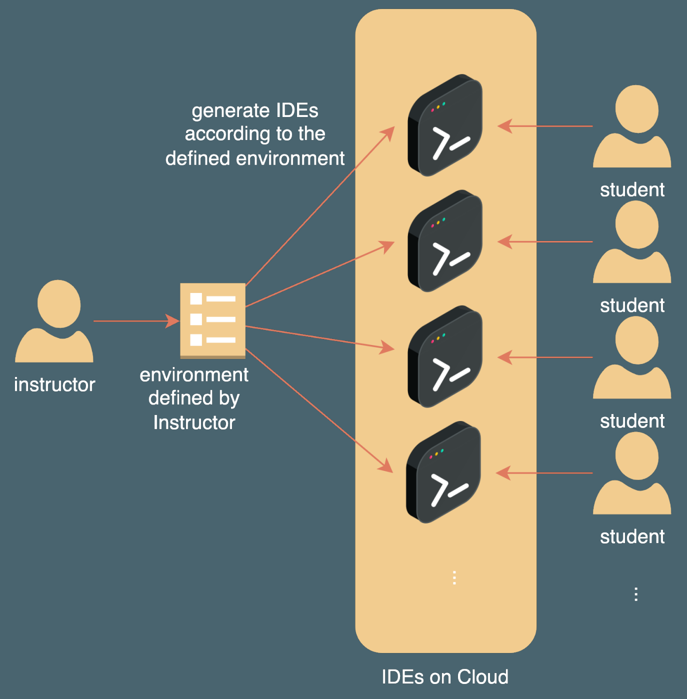
Dr. Tsoi Yau Chat

Motivations

- Instructors need to ensure standard programming environments for assignments and labs.
- Programming beginners don't know how to manage programming environments.
- Students can only work on their own computers which have the dependencies. They need to carry that computer everywhere.
- Program performance depends on computer hardware performance. Program is slow if computer is slow.

Ideation

- A combination of programming environment management system and IDE that runs on cloud.



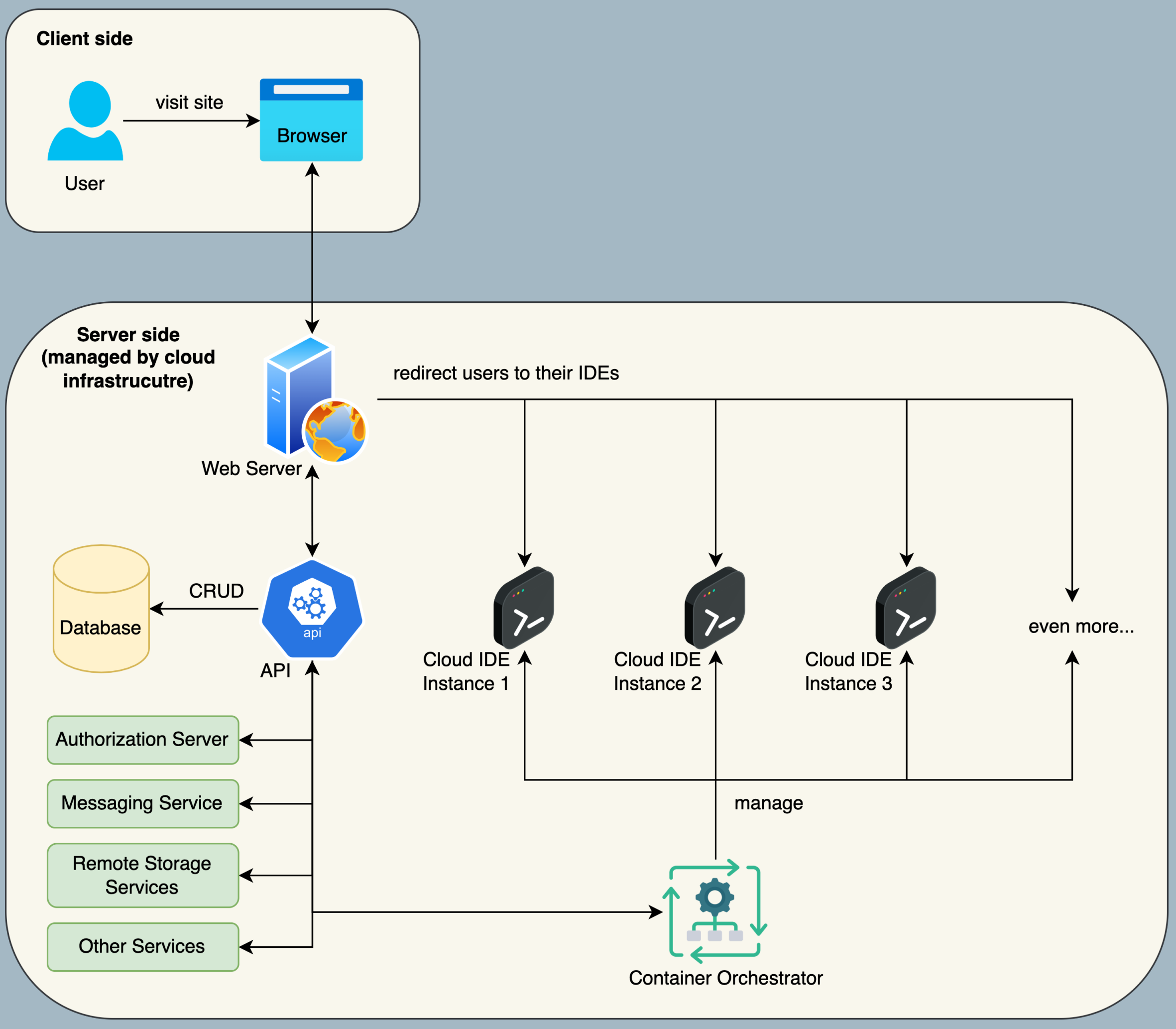
generate IDEs according to the defined environment
instructor
environment defined by Instructor
student
student
student
student
IDEs on Cloud

Product

- Programming environment management for instructors.
- Online IDEs that have standard programming environment. No files and configuration need to be done on students' computers.
- IDE extensions to improve teaching and learning efficiency.
- Programming on browser. System can be accessed from laptop or tablet.

Methodology

System Design



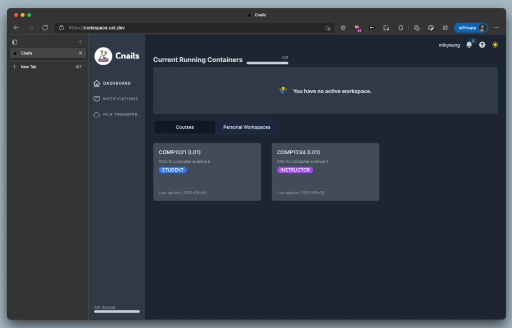
∗ Actually the container orchestrator not only manage the deployments of the cloud IDE instances but also other components such as the web server and API daemon. For simplicity sake, the diagram above only shows the arrows to the IDE instances.

∗ Bidirectional arrows with no annotation means "request and respose"

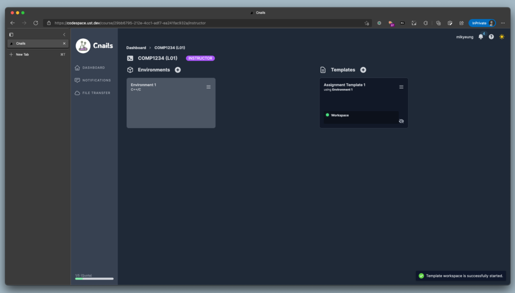
- The system is composed of the following major components: a web server, an API daemon, a database, IDE containers and other different services. Deployment, scaling and network traffic are handled by several cloud infrastructures.
- The web server serves a frontend on the client side, which is responsible for interaction with users. The web server is also responsible for handling requests from the client side. No operation is performed here as the request will be directed to the API daemon.
- The API daemon is responsible for handling internal communication from different components (e.g. databases and authorization server etc.) and third party services (e.g. the cloud messaging service and remote storage etc.).
- The database stores all the users' and courses' data. It serves as a source of truth for all components in the system.
- Each IDE instance is a code-server container hosted in the cloud, which is spun up and available when a user needs it. The programming environments of the IDE are defined by instructors in advance.
- Functionality of the IDE can be extended using extensions.
- The container orchestrator deploys and cleans up cloud IDE instances base on request. In production stage, it can spread workload across many servers to achieve scalability. It is a key component of the cloud infrastructure.

Web application

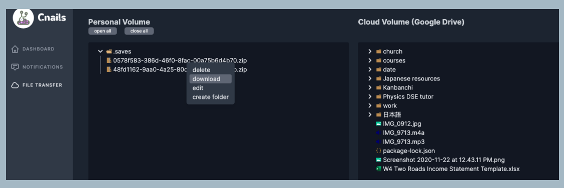
- Built by Next.js.
- It provides a user interface on the browser.
- Users can access their courses as a student or an instructor.
- Instructors can:
  - **define programming environments** (custom or predefined environments for pythons, c++ etc.) for the courses.
  - **create templates for assignments,** labs or exams, and associate templates with the programming environments.
  - publish and unpublish templates.
  - view students' IDEs' statistics.

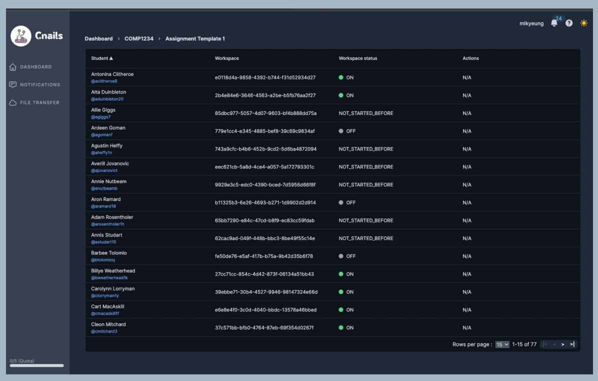

Dashboard

- Students can:
  - **create an IDE for a template that is published.** The IDE has a standardized programming environment and is ready to code.
  - **select and message a question to instructors with code snippets.**
- **All users have their personal volumes.** They can also connect with their cloud storage providers.

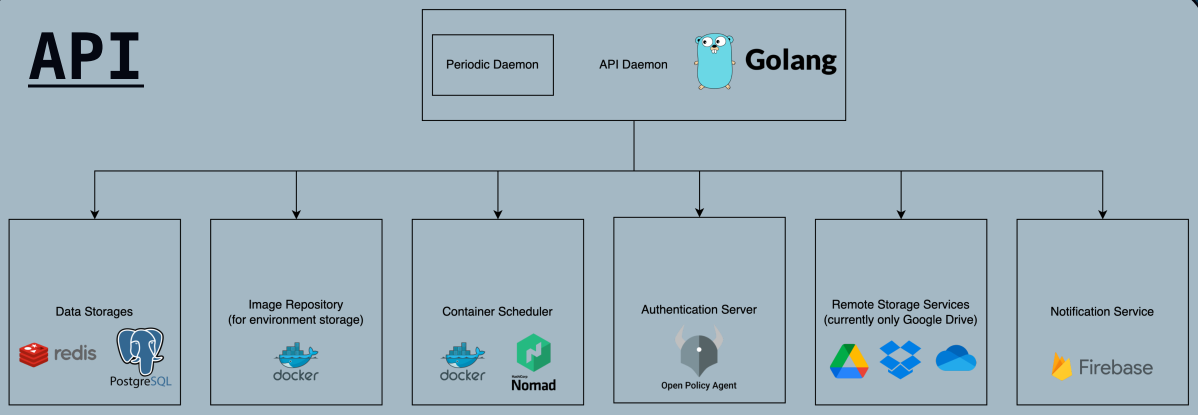

Instructor view on a course

- All users can **create personal workspaces** which are not bound to a course.


File tranfer between local files, personal volume and different services providers(currently only Google drive is supported)

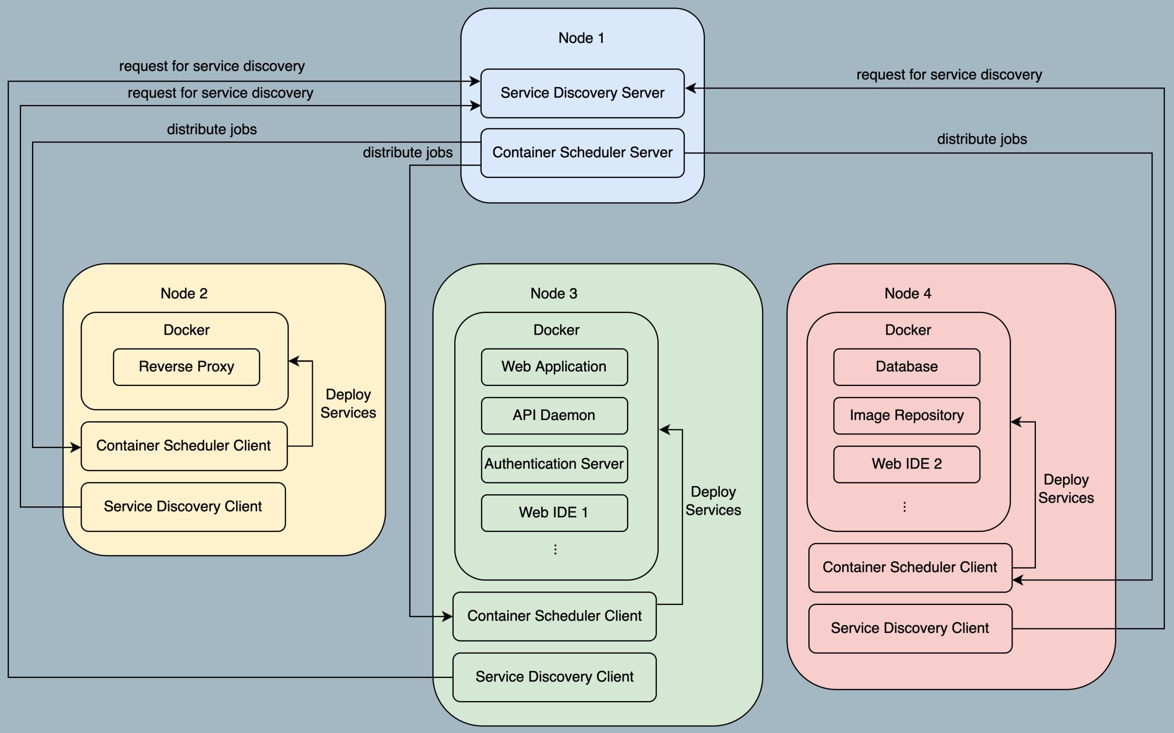

Instructor can view status and statistics on students' workspaces. Useful for labs and exams.

API



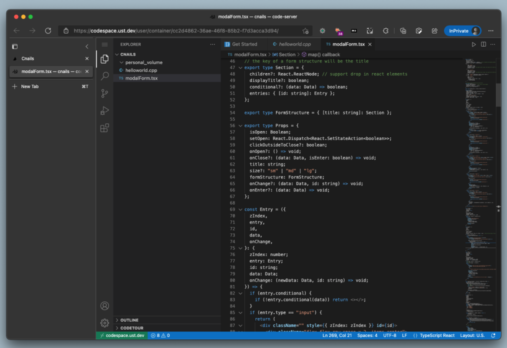
- Implemented in Golang
- Have a periodic daemon to perform perioidic tasks such as cleaning up idle containers.
- **Responsible for internal communication** with:
  - **databases**, to perform CRUD operation.
  - **image repository**, to store and obtain the defined programming environments.
  - **authorization server**, to query result for access control decision-making.
  - **remote storage**, to allow file transfer between cloud storage providers and personal volume in Cnails (currently only Google drive is supported).
  - **messaging service**, to handle messaging between users.
  - **container orchestrator**, to deploy or remove IDE instances.

Cloud Infrastructure



- Implemented using Nomad and Consul.
- Nomad is responsible for container scheduling while Consul is responsible for service discovery.
- Development environment was set up in 4 nodes with one being the server and the rest being the client.
  - Server does the scheduling decision-making and service discovery
  - Jobs (various components of the system) are deployed in the clients.
  - In production setting, there would be much more than 4 nodes. The implementation will be roughly the same. Scaling is handled by Nomad and Consul.

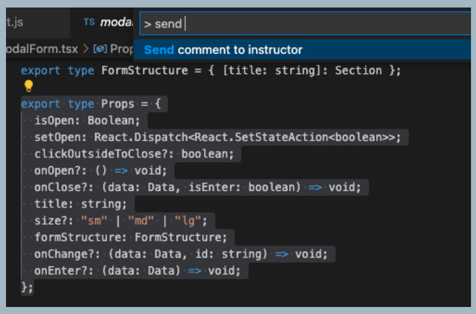
Code server



Our online IDE is code-server, which is a browser-based IDE based on Microsoft's Visual Studio Code (VSC). VSC is the most popular code editor and IDE among programmers, 70% of programmers reported that they are using it.

Choosing code-server to be our online IDE allows the system to have seamless integration with CS students' daily workflow.

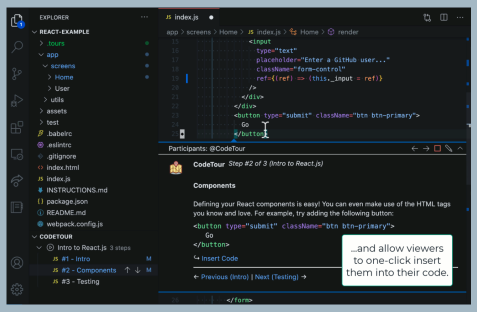
VSC has a large developer community and numerous extensions to provide rich extensibility and functionality to the IDE. **Some extensions can improve teaching and learning efficiency**. For example, CodeTour allows instructors to set up checkpoints in the codebase to guide students through an assignment. A QuickQuestion extension that allows students to ask instructor questions simply by selecting the code is also developed. These functions are useful especially during programming labs.



```
> send
Send comment to instructor
export type FormStructure = { [title: string]: Section };

export type Props = {
  isOpen: Boolean;
  setOpen: React.Dispatch<React.SetStateAction<boolean>>;
  clickOutsideToClose?: boolean;
  onOpen?: () => void;
  onClose?: (data: Data, isEnter: boolean) => void;
  title: string;
  size?: "sm" | "md" | "lg";
  formStructure: FormStructure;
  onChange?: (data: Data, id: string) => void;
  onEnter?: (data: Data) => void;
};
```

QuickQuestion Extension



```
<input
  type="text"
  placeholder="Enter a GitHub user..."
  className="form-control"
  ref={(ref) => (this._input = ref)}
/>
</div>
</div>
<button type="submit" className="btn btn-primary">
  Go
</button>
```

CodeTour *Step #2 of 3 (Intro to React.js)*

Components

Defining your React components is easy! You can even make use of the HTML tags you know and love. For example, try adding the following button:

```
<button type="submit" className="btn btn-primary">
  Go
</button>
```

↪ Insert Code

← Previous (Intro) | Next (Testing) →

```
</form>
```

CodeTour Extension

Vision

As online learning and self-learning are becoming the trend of computer science in recent years, we believe that Cnails will be greatly beneficial to the teaching and learning of HKUST CS instructors and students. The most direct impact of Cnails is that Cnails clears the hassles of managing local programming environment for students and ensure a standardized programming environment for everyone in a course. This favours automatic grading and reduces grading problems due to a student using a wrong environment setup. Being able to access the IDE on the browser would mean students not only don't need to install any binary on their computers but also can access the IDE from any desktop, laptop and tablet with the same performance.

Another huge likely benefit is that Cnails can integrate with other CSE systems to create a collection of tools to improve the teaching and learning experience. For example, it could integrate with ZINC and GUITA for auto-grading right after a student finishes an assignment in Cnails. If the grading is handled immediately, the student can see the result and do corrections if needed right away. This will be extremely useful in programming labs and MOOCs. Particularly in MOOC, some educational extensions such as the CodeTour and QuickQuestion in the online IDE allows students to have a better learning experience even when in a remote learning setting.

Conclusion

Cnails makes programming more accessible to people. We believe that Cnails could be beneficial to not only the CS department but also other departments that offer programming teaching to students. As online learning has become more common, the audience Cnails serves is enormous and it could have huge potential by extending its functionality. The MVP of Cnails has been developed and it is currently under testing and evaluation to make the system more robust and more user-friendly. We are looking to seeing Cnails serve HKUST instructors and students someday in the future.