Cloud Environment Manager

sdmay21-39@iastate.edu

Client:PwC

Advisor: Lofti Ben Othmane

Rishabh Bansal, Zane Seuser, Jet Jacobs, Adis Osmankic,Gavin Monroe

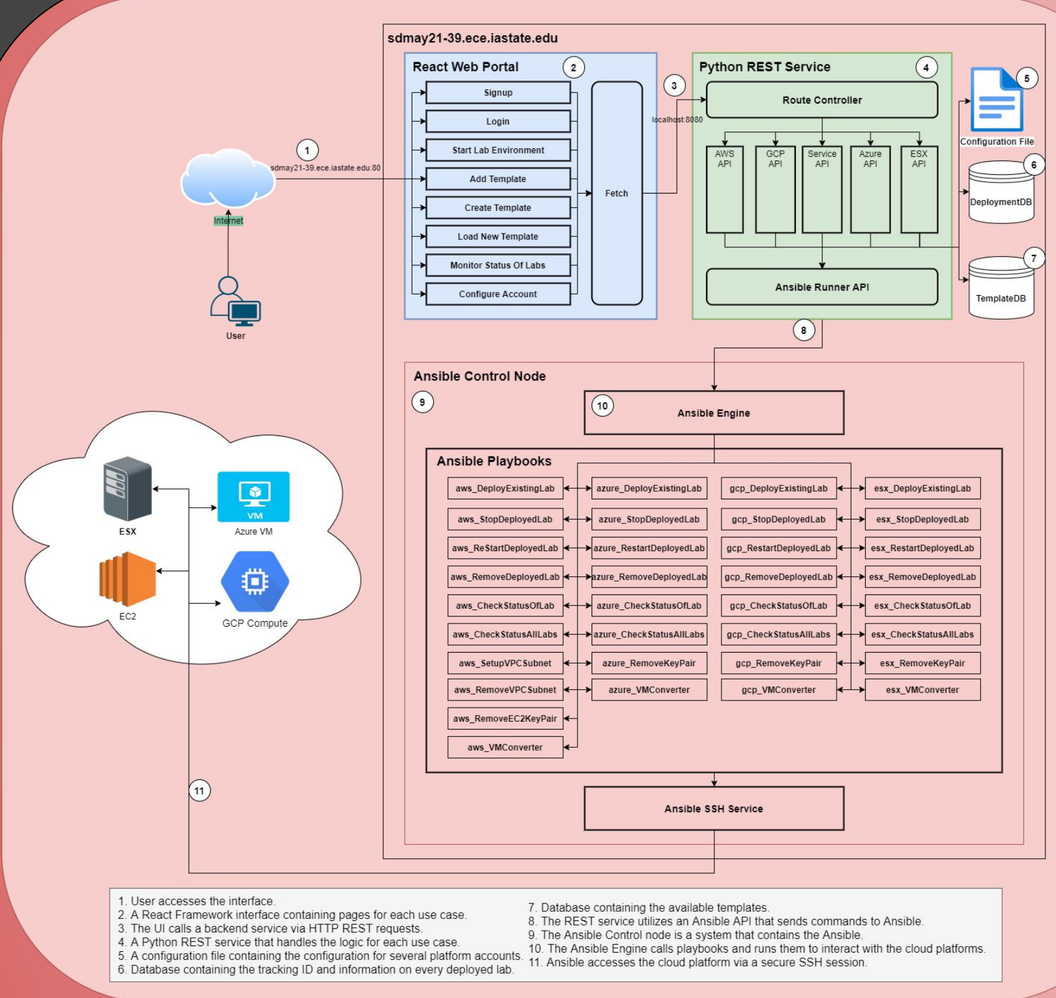
PROBLEM AND PROJECT STATEMENT

Currently, our sponsor provisions lab environments for capture the flag events within a variety of cloud providers and on-premise resources by hand. The problem with this is that it is super unscalable. They have no way to quickly create and manage a large amount of lab environments. There is also no way to easily make sure that all these environments are the same. Right now, they have to log in to each individual cloud provider (like AWS, Azure, GCP) and manually configure every single virtual environment individually.

By developing our project application, the Cloud Environment Manager, we are going to provide our sponsor with a clean and simple user interface that allows them to quickly and easily deploy lab environments to multiple different cloud providers. The application will also allow them to manage and destroy their existing lab environments once they are no longer needed.

In a nutshell, our whole project is to create a user friendly interface that can quickly and easily create and manage virtual machines in various cloud platforms.

DESIGN APPROACH:



Design Summary:

We wanted our design to be easy to develop and efficient, so we decided upon a separately hosted front-end and back-end. Another thing that we thought was very important was to use the technology that made implementing our functionality easy and fast. Because of this, we chose to utilize existing frameworks to expedite the development of our UI and API. Doing this, we can also utilize all the libraries that exist for those frameworks.

Our Design Components:

- **User Interface**
  - React application for creating frontend user interface.
- **API Services**
  - Python Flask API for business logic and creating connections to all of our backends.
- **Ansible**
  - For Cloud Environment management interfacing for all the cloud providers.
- **MongoDB**
  - Data storage for lab templates and configuration parameters
- **Cloud Infrastructure**
  - Labs and Vrtiual machines within different cloud providers like AWS, GCP, and Azure.

DESIGN REQUIREMENTS

Functional Requirements:

- A user should be able to view a list of existing lab environments
- A user should be able to view all of the attributes of existing lab environments
- A user should be able to provision lab environments within AWS, GCP, and Azure
- We would be having a centralized platform. Due to which there could be one place to manage all the VM environments.

Non-Functional Requirements:

- The application should be available at all times
- The application should properly handle errors behind-the-scenes and provide error messages / warnings to the user when necessary
- The user interface should be visually appealing
- The user interface should be easy to use and intuitive

INTENDED USERS

This project is designed for use by an organization that needs to deploy lab environments to different cloud platforms for a variety of different users. This project is focused on education and deploying capture the flag events to large groups of people for training. By using this product, organization admins will be able to deliver training resources to members of the organization quickly and easily.

TECHNOLOGY USED

OPERATIONAL ENVIRONMENT

This main application will be hosted within our provided server. The front-end application will be accessible from a web browser and the back-end API it interfaces with will be able to deploy lab environments to different cloud providers (AWS, Azure, GCP).

ENGINEERING CONSTRAINTS

- We were given full control
- No access to an ESX system
- No budget, so all examples have been made with free tiers of AWS, GCP, Azure
- Any operation we use must be compatible across all platforms

STANDARDS

- IEEE 14764-2006 - Standard for Software Engineering - Software Life Cycle Processes - Maintenance
- IEEE 29119-1:2013 - Software and Systems Engineering — Software Testing
- IEEE 15026-1:2019 - Systems and Software Assurance

TESTING

Frontend Testing

Using React Jest for user platform testing making sure components and pages functioned as needed. This included popular packages that extended user clicking and scrolling to ensure functionality.

Acceptance Testing

By using Postman as a testing tool we can insure smooth integrations across all endpoints used from frontend to backend. Tests allows us to ensure that our API is working as expected, to establish low failure rate.