OpnLaaS Project Specification

Abstract

OpnLaaS is an open-source platform designed to facilitate the deployment and management of bare metal servers, virtual machines, and LXC containers. The platform aims to provide a user-friendly interface for administrators and users to efficiently manage their infrastructure.

Why do we want to do this?

We (Evan Parker, Dan McCarthy, Matt Gee, Kestutis Biskis, Alex Houle) are all part of the cybersecurity club. Evan and Dan are the sysadmins of the lab. We want to enable students and faculty to easily get access to physical and virtual machines for learning and experimentation. By creating OpnLaaS, we aim to streamline the process of provisioning and managing these resources, making it easier for users to focus on their projects rather than the underlying infrastructure. For systems administrators, this platform makes it so we don't have to manually set up each machine for each user, saving us time and effort.

Components

This is a list of components that will make up the OpnLaaS platform, along with some small descriptions of functionality, tools used, and integration with other components.

Web App

The web app is the user-facing component, providing an interface for users to request and manage their bare metal servers, VMs, and containers. Admins can also use the web app to oversee the entire infrastructure.

Frontend

We will have a few pages:

- `/` - Home page, you can't do much here, there will be a product description, link to log in, and maybe a change log. You can also see a list of physical resources and their statuses.
- `/login` - Login page, where users can log in with their credentials. See more in the Auth Manager section.
- `/dashboard` - Dashboard page, where users can see their resources (bare metal servers, VMs, CTs) and their statuses. Users can also request new resources here. These requests are called bookings.
- `/admin` - Admin page, where admins can manage the entire infrastructure. They can add/remove physical servers, manage user bookings, and oversee resource allocation.

We will use HTML Templating using Fiber's built-in templating engine for the frontend. The design will be simple and functional, focusing on usability. We'll use Tailwind CSS to enhance the visual appeal and responsiveness of the web app.

Backend / API

We'll use Go with the Fiber web framework for the backend. The backend will handle user authentication, resource management, and communication with other components like the BMC Management System and Bare Metal Provisioning System.

Database

We'll use MySQL as our relational database to store user information, resource details, and booking records. The database schema will include tables for users, physical servers, VMs, CTs, and bookings.

The reason is that MySQL in Go is incredibly well supported, and has mature libraries and documentation. It's also easy and safe to stand up and manage.

Auth Manager

We have a LDAP Domain (FreeIPA controllers) that are used in our lab. We will be supporting this mode of authentication for our users. This will allow users to log in with their existing credentials, simplifying the user management process. Also we can use groups for permission management (e.g., admin group, allowed to make bookings, etc.)

BMC Management System

The BMC Management System will handle communication with the Baseboard Management Controllers (BMCs) of the physical servers. This component will allow the web app to perform power management tasks and set boot targets for the servers.

IPMI Interface

This just implements IPMI commands to power on/off/reboot servers, as well as set their next boot target (PXE)

Redfish Interface

Implements an interface for Redfish-capable BMCs to perform the same tasks as the IPMI interface. In addition, they can also retreive hardware inventory information.

Bare Metal Provisioning System

This system will manage the provisioning of bare metal servers using PXE booting and TFTP. It will work in conjunction with the BMC Management System to set the next boot target of the servers to PXE, allowing them to boot into a network-based installer or OS image.

TFTP Server

Basically a simple TFTP server that serves the PXE boot files to the bare metal servers when they boot.

ISO File Repository

A repository of ISO files that can be used for provisioning bare metal servers. The web app will allow admins to upload and manage these ISO files. When an ISO is uploaded, it is extracted to the TFTP server's directory structure for PXE booting.

For Linux distributions that support automated installations (like Kickstart for CentOS/RedHat, Preseed for Debian/Ubuntu), we will also generate the necessary configuration files to enable unattended installations.

Virtualization Management System

This system will manage the creation and management of virtual machines (VMs) using Proxmox VE and LXC containers. It will provide an interface for the web app to create, start, stop, and delete VMs and containers.

Proxmox VM Management

Self explanatory. Create VMs easily.

LXC Container Management

Self explanatory. Create LXC containers easily.

Booking System

The Booking System allows for a booking structure to be created, modified, and deleted. Bookings are really just a set of collections. One for users, and one for resources. A booking ties users to resources for set periods of time. Bookings are created by users via the web app, and approved/denied by admins.

When a booking is approved, the Booking System communicates with the Bare Metal Provisioning System and Virtualization Management System to allocate the requested resources to the user for the duration of the booking.

There are two types of users in a booking: Operators and Auditors. Operators can request extensions, additions, or removals of resources in a booking. Auditors can only view the booking details. Note that Operators can re-configure and re-deploy resources within a booking easily.

Timeline

Due to the complexity of this project, we will be working on it over multiple semesters. Obviously this does not fit into the few weeks for the CS518 project, so we will focus on an initial subset we can get this project off the ground.

During the CS518 Project Timeframe

- Set up the project repository and initial structure.
- Implement the Web App with basic frontend and backend functionality.
- Set up the Database schema and integrate it with the Web App.
- Implement the Auth Manager to support LDAP authentication.
- Implement the BMC Management System with IPMI & Redfish interfaces.
- Implement the Proxmox VE API integration for VM management.

Future Work

- Implement the Bare Metal Provisioning System with TFTP server and ISO file repository.
- Implement the Booking System for resource allocation and management.
- Enhance the Web App with additional features and improved UI/UX.

Larger Group Request

Due to the complexity of this project, and the various components involved, we are requesting to work as a larger group of 5 members for this project. Each member will take on specific components and responsibilities, allowing us to leverage our collective skills and expertise to successfully complete the project. Additionally, we would like to to ensure that our work is of high quality and meets our goals. By working as a larger group, we can divide the workload effectively and ensure that each component is developed thoroughly.