OpnLaaS Sprint Planning

Sprint Goal

Goal: Enable students and faculty to manage resources through a web interface, and create virtual machines on proxmox.

Specific Goal: Create a resource management dashboard in the web UI, and allow users to interface with proxmox so they are able to create/manage VMs.

Stretch Goal: Implement logging to keep track of changes to virtual machines.

Selected Stories

- Feature 1: User & Resource Dashboard
    - As a System Architect, I want to be able to view existing resources, so that I can make sure systems are running correctly.
    - As a System Architect, I want to be able to add new resources so that I am able to update the system with new equipment.
- Feature 2: Virtualization Integration
    - As a Researcher, I want to create new virtual machines, so that I can run my projects.
    - As a Researcher, I want to edit my existing virtual machines, so that I can make changes to my projects as I go.

Story Decomposition

Feature 1: User & Resource Dashboard

Story 1: - As a System Architect, I want to be able to view existing resources, so that I can make sure systems are running correctly.

Tasks:

- Data Model (Host Resource):

```
Host struct {
        ManagementIP        string                `gomysql:"manageme
        Vendor              VendorID              `gomysql:"vendor"
        FormFactor          FormFactor            `gomysql:"form_fac
        ManagementType      ManagementType        `gomysql:"manageme
        Model               string                `gomysql:"model" j
        LastKnownPowerState PowerState            `gomysql:"last_kno
        Specs               HostSpecs             `gomysql:"specs" j
```

```
        Management          *HostManagementClient `json:"-"`
}
```

- Frontend:
    - Dashboard needs to get current resources from API and display them to users.
    - UI should include details of each system, and wether the system is running or not.
    - app routes:
        - GET: /dashboard
- Backend:
    - Needs to get current resources from DB, and make that available to the frontend.
    - Should include all hosts currently in use.
    - api routes:
        - GET: /api/hosts
- Integration:
    - Test should be created for the frontend to make sure resources are displayed correctly.
    - Tests for the backend should ensure the API properly authenticates users, and prevents possible SQL injections.

Story 2: - As a System Architect, I want to be able to add new resources so that I am able to update the system with new equipment.

Tasks:

- Data Model (Host Resource):

  (Same as Story 1)

- Frontend:

    - Dashboard should include an method to add a new host/resource.
    - This could be either a separate page or a modal on the dashboard.
    - app routes:
        - POST: /dashboard/create

- Backend:

    - Needs to add new hosts to the database.
    - Should authenticate users before allowing them to add new host resources.
    - api routes:
        - POST: /api/hosts

- Integration:

    - Testing should be done to ensure backend has proper error handling for bad inputs. (especially since there are many fields, some of which optional for each host)

Feature 2: Virtualization Integration

Story 1: - As a Researcher, I want to create new virtual machines, so that I can run my projects.

Tasks:

- Data Model (VM/ISO):

```
StoredISOImage struct {
        Name         string           `gomysql:"name,primary,unique"
        DistroName   string           `gomysql:"distro_name" json:"d
        Version      string           `gomysql:"version" json:"versi
        Size         int64            `gomysql:"size" json:"size"`
        FullISOPath  string           `gomysql:"full_iso_path" json:
        KernelPath   string           `gomysql:"kernel_path" json:"k
        InitrdPath   string           `gomysql:"initrd_path" json:"i
        Architecture Architecture     `gomysql:"architecture" json:"
        DistroType   DistroType       `gomysql:"distro_type" json:"d
        PreConfigure PreConfigureType `gomysql:"preconfigure_type" j
}
```

- Frontend:
    - Users should have an easy way of selecting an operating system image (ISO) and creating a new VM from it.
    - app routes:
        - POST: /dashboard/vm/create
- Backend:
    - API should support getting current ISO images as well as adding new ones.
    - Should also allow authorized users to create a VM using post request.
    - api routes:
        - GET: /api/iso-images
        - POST: /api/iso-images
        - POST: /api/vm
- Integration:
    - New VMs should automatically be deployed and visible from the dashboard.
    - Frontend should allow for use of all vm related api routes on the backend.

Story 2: - As a Researcher, I want to edit my existing virtual machines, so that I can make changes to my projects as I go.

Tasks:

- Data Model (VM/ISO):

  (Same as Story 1)

- Frontend:
  - There should be an easy way for users to update the config of there existing VMs.
  - Next to each VM in the UI it should include the option to edit/delete a VM.
  - app routes:
    - POST: /dashboard/vm/update

- Backend:
  - Should check to make sure user is authorized, and VM that is being updated exists (you shouldn't be able to edit something that doesn't exist)
  - api routes:
    - PUT: /api/vm

- Integration:
  - Care needs to be taken when testing error handling for the backend. Problems could lead to breaking existing VMs which are in use.

Data Models & API Spec

Data models where included with each story as needed. Below is a overview of the API/structure of our application:

```
// Auth API
app.Post("/api/auth/login", apiLogin)
app.Post("/api/auth/logout", mustBeLoggedIn, apiLogout)

// Enums API
app.Get("/api/enums/vendors", apiEnumsVendorNames)
app.Get("/api/enums/form-factors", apiEnumsFormFactorNames)
app.Get("/api/enums/management-types", apiEnumsManagementTypeNames)
app.Get("/api/enums/power-states", apiEnumsPowerStateNames)
app.Get("/api/enums/boot-modes", apiEnumsBootModeNames)
app.Get("/api/enums/power-actions", apiEnumsPowerActionNames)
app.Get("/api/enums/architectures", apiEnumsArchitectureNames)
app.Get("/api/enums/distro-types", apiEnumsDistroTypeNames)
app.Get("/api/enums/preconfigure-types", apiEnumsPreConfigureTypeNam

// Hosts API
```

```
app.Get("/api/hosts", apiHostsAll)
app.Get("/api/hosts/:management_ip", apiHostByManagementIP)
app.Post("/api/hosts", mustBeLoggedIn, mustBeAdmin, apiHostCreate)
app.Delete("/api/hosts/:management_ip", mustBeLoggedIn, mustBeAdmin,
app.Post("/api/hosts/:management_ip/power/:action", mustBeLoggedIn,

// ISO Images API
app.Post("/api/iso-images", mustBeLoggedIn, mustBeAdmin, apiISOImage
app.Get("/api/iso-images", mustBeLoggedIn, mustBeAdmin, apiISOImages
```

Responsibilities

- Feature 1: User & Resource Dashboard
  - Dan McCarthy
  - Kestutis Biskis
- Feature 2: Virtualization Integration
  - Evan Parker
  - Matt Gee
  - Alex Houle

Gitlab

Below is the link to our issue board. Our board has been populated with issues and everyone has been assigned atleast one task. (Note: our project is on Github)

https://github.com/orgs/OpnLaaS/projects/1