# GOOD 2025 Contributor Jam Info Sheet

## The Map of Open OnDemand

This guide attempts to layout structures and locations around much of the OOD ecosystem.

These structures range from types such as GitHub repos, to ruby `gems`, and even to components of OOD itself with respect to its codebase.

### Pull Requests

You need to make a fork of the repo first:

- Generally, you are working off the `latest` branch when you begin your work.
- Edit on GitHub button also works too!

### OOD Documentation

Open OnDemand's documentation sits in a public GitHub repo, hosted out of GH pages.

The repo:

- `https://github.com/OSC/ood-documentation`
- Notice the repo has many branches but `latest` and `develop` are the 2 which matter because they generate GitHub Pages which serve the documentation for the community.
- Branch off `latest` and push to `develop` for features not implemented.
- For typos and fixes, push your branch to `latest`.

GH Pages:

- `https://osc.github.io/ood-documentation/master/`
- `https://osc.github.io/ood-documentation/latest/`
- `https://osc.github.io/ood-documentation/develop/`

### Gems

A Ruby `gem` is just a library or module or bundle of code that we can install using the ruby package manager `bundler`.

- The `gems` are listed in a project's `Gemfile`.
- After a `bundler` run a `Gemfile.lock` with version dependencies will also be generated.
- `bundler`: => `https://bundler.io/`
  - commands: `https://bundler.io/docs.html`
  - `bundler config set --local path vendor/bundle` will be your friend later to work on the development dashboard code and our local `gems` without polluting the system `gems`.

OOD has 4 `gems` itself, some of which can be largely ignored, some which are quite useful:

- `ood_packaging`: https://rubygems.org/gems/ood_packaging largely for OOD internal team to help with packaging and distribution of OOD.
- `ood_appkit`: https://rubygems.org/gems/ood_appkit
  - Provides an interface to work with OOD scientific apps, a `dataroot` for OOD apps to write data to and common assets and helper objects.
- `ood_support`: https://rubygems.org/gems/ood_support
  - Provides an interface to work with local OS installed on the HPC center's *web node*. This `gem` is often useful for both OOD and OOD apps.
- `ood_core`: https://rubygems.org/gems/ood_core
  - Provides *Adapters* for *Schedulers*, `batch_connect` *Templates* for the 3 types of OOD apps, cluster interactivity, and Job interaction.

# GOOD 2025 Contributor Jam Info Sheet

## OOD Core (ood_core)

This is where the actual backend code to interact with your clusters or schedulers resides.

Schedulers and Adapters

OOD provides *adapters* for the various HPC *schedulers* which can all be seen here:

- `…/lib/ood_core/job/adapters`
- One thing to note is we have k8's adapter if you wish to use k8's as a scheduler.
- LinuxHost: This adapter is used to mimic a scheduler or resource manager, for remote desktop or IDE's.
- SystemD: Community contribution.

3 Species of OOD App

OOD ships with 3 types of scientific apps:

1. `basic`:
   - HTTP server
   - e.g. Jupyter Notebook:
2. `VNC`:
   - VNC server
   - e.g. Qomsol, Remote Desktops
3. `VNC_container`:
   - Less common, but it exists, VNC with container for sites that don't want to install X11, XFCE, GNOME, etc. on their host.

All of these options are what you are setting when you select the `template` in your `submit.yml` files.

## Clusters

- The code to work with your clusters and the corresponding clusters config files.
- `https://github.com/OSC/ood_core/tree/master/lib/ood_core`
  - Split out between 2 files
  - the `clusters.rb` file is to handle the clusters config files.
  - the `clutser.rb` file is to handle working with a cluster and its *scheduler.*

## Dev Work for `ood_core`

In order for this to work we need to actually touch our `Gemfile` in the `dashboard` and point to our local `ood_core`:

Gemfile

```
gem 'ood_core', :path=> '/full/path/to/checked/out/ood_core'
```

- You must issue the `bin/setup` command to rebuild your `dashboard` once you make these local changes to your `ood_core` code.

## OOD Monorepo

Open OnDemand follows the *Mono-repo* pattern. What this means is that OOD has various applications and utilities all contained under the `Ondemand` namespace.

- `https://github.com/OSC/ondemand`

*Notice we have our `apps` under this which leads to the `dashboard` and all the code a user would be familiar interacting with, but at this top level we see much more.*

- `ood-portal-generator` generates the NGINX and Apache configs given a valid `ood_portal.yml` file.
- `nginx_stage` is used to manage the PUN for OOD.
- `ood_auth_map` connects the auth system to the web-node system users for OOD at login.
- `packaging` is used to help with the distribution of OOD software.
- `mod_ood_proxy` is used to manage the proxy for OOD.
- `dashboard` is the frontend `rails` code most users and admins are familiar interacting with.

GOOD **2025 Contributor Jam Info Sheet**

**OOD Dashboard Code Components**

OOD has several components within the `rails` application code

- `active_jobs`
- `bc_desktop`
- `dashboard`
- `file_editor`
- `files`
- `myjobs`
- `projects`
- `shell`
- `system-status`

MVC Components

OOD uses the MVC pattern of web-development which is default in `rails`.

- Rails philosophy is **convention over configuration**.

We can see the OOD conventions by looking at the various `Models`, `Controllers` and `Views` within the `rails` code itself.

Models

- `…/apps/dashboard/app/models`
- All the data OOD is aware of is defined in this directory.

Controllers

- `…/apps/dashboard/app/controllers`
- Here we see what data the model can present to a view.

Views

- `…/apps/dashboard/app/views`
- This directory can be daunting as it contains all code used to present the data to users. As such, there can be a great many components to any piece of OOD.
- e.g. `…/apps/dashboard/app/views/batch_connect`
- While this is one of the more complex views to deal with, it gives a sense of how complex some of these files can become.
- The goal for these when working is to try and make things *modular* and *logical*.
- Rails also uses the notion of **partials** to provide components of views.
    - These files start with an underscore `_my_partial`.

Utilities

OOD also has some helpers for things like `rclone`, some `rake` tasks, and other functionality all contained here:

- `…/apps/dashboard/lib`

*batch_connect* convention

One of the more powerful and useful abstractions to be aware of in OOD.

- `…/apps/dashboard/app/models/batch_connect`
- This model provides most data you see in a session card and when interacting with forms.
- `helpers` are a big piece of the puzzle here:
    - `…/apps/dashboard/app/helpers`

## Code Changes

Now that we've had a tour at break-neck speed, let's make changes.

This portion assumes you have setup a dev dashboard on your OOD instance as described at `https://osc.github.io/ood-documentation/latest/how-tos/app-development/enabling-development-mode.html`

And that you have forked, cloned, and built your dashboard as described at `https://github.com/OSC/ondemand/blob/master/DEVELOPMENT.md#developing-the-dashboard`

Seeing Changes

Some changes will only require we reload the browser:

- If a config change is not taking effect after several reloads, try a PUN restart.
- **Environment variable changes require a rebuild of the PUN with** `bin/setup`

# 2025 Contributor Jam Info Sheet

## Resources:

Contributing guide: `https://github.com/OSC/ondemand/blob/master/CONTRIBUTING.md`
Dev Dashboard Setup: `https://github.com/OSC/ondemand/blob/master/DEVELOPMENT.md#developing-the-dashboard`
Development guide: `https://github.com/OSC/ondemand/blob/master/DEVELOPMENT.md`
Dockerfile: `https://github.com/OSC/ondemand/blob/master/Dockerfile`
Security and Reporting: `https://github.com/OSC/ondemand/blob/master/SECURITY.md`
Code of Conduct: `https://github.com/OSC/ondemand/blob/master/CODE_OF_CONDUCT.md`
Forking repos in GitHub: `https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks/fork-a-repo`
Rails Guides: `https://guides.rubyonrails.org/v7.1/`
Rails MVC pattern: `https://guides.rubyonrails.org/v7.1/getting_started.html#mvc-and-you`
Rails view partials: `https://osc.github.io/ood/documentation/latest/customizations.html#overriding-pages`

## GitHub Issue Pages

OOD Documentation: `https://github.com/OSC/ood-documentation/issues`

Open OnDemand: `https://github.com/OSC/ondemand/issues`