Design for A1 Size Poster Ordering

- We need to implement a customer facing webapp which is responsive and is able to serve high resolution web content.
- This will require us to produce different sized images of the poster and store them in a image cache for the web frontend to access.
- We also need to create a responsive image server which would resize the image correctly.
- Once the user requests to view an image the system should look in the cache first. If not found it query the responsive image server to convert the image to the requested size, send it to the frontend and also update the image cache with this image.
- Once the order is placed the web front end should send the resolution customer ordered and the database id associated with the image to the web server.
- The web server can have an message exchange attached to the it where it would publish the message.
- Each designer should have a queue attached to the exchange where it will be waiting for messages to arrive.
- The webserver will query the database to figure out which designer has that poster. e.g SELECT designer_id from posters where image_id=?
- Create a message with image_id, designer_id and image_size
- Publish message on the exchange.
- The designers will have a component running with its designer_id in memory reading on the queues.
- The designer component will read the message, compare the designer_id in the message with its own ID and only process the message if they match.
- If the id matches it will create another message for the printer app with the command to print and a secure link for the printer app to download the image from.
- The printer app which is another component can download the image and send it to the available printer.
- Each designer component can have a small catalyst app running which allows them to upload a file to the image store.
- The image store directory can be network storage with the directory mounted on each designers computer
- The designer web app can send a message to another component e.g poster-live. The sole purpose of this component is to add rows in the database about the new poster.
- Once the entry is there the frontend will be able to display this new poster.

List of Components

1. Print-Evolve-web (Frontend/Web UI/Catalyst App) served via Nginx with image-filter-module enabled so that we can easily resize images.
2. Print-Evolve-web-cache. A web cache which can store those images like Varnish or memcached(Not sure).
3. Print-Evolve-web-server. A component made out Our compiled version of Nginx.
4. Print-Evolve-messagequeue. Most probably version 3.8 of RabbitMQ message queue, but we can also use others like Kafka. I have had experience with RabbitMQ so that is why I chose it.
5. The message queue will be installed on the web server, designer and printer app machines/host.
6. Print-Evolve-designer. A component for designers which can read messages from the queue and create a downloadable link for the printer app. This app will also have a small web interface which the designer can access. This app will allow designers to upload new posters into the system.
7. Print-Evolve-Printer. An app which can read messages sent by the designer for print and send it to the available printer for print.
8. Print-Evolve-Poster-Live. An app which has the sole purpose of inserting rows of new posters into the DB so that the frontend can display them.

Packaging and Deployment

We can package the above mentioned components as dpkg or rpm based package so that they can be easily installed on the hosts. We can user Ansible to deploy these packages on the host which can be pre-defined to serve a specific purpose

```
---
Hosts:
   Print-Evolve-web:
- IP address: 192.168.0.34
- Hostname: Print-evolve-web.com
  Print-Evolve-designer:
…….
```

Then we can define in Pre-defined Vars what to install on each type of host e.g.

```
Web:
- Web-cache
- Nginx
```

- Catalyst-we-app
- Messageque

So and so forth. This would certainly make it easier to deploy on the same host or on a distributed environment.

I think if we design it this way we will make the software highly available, reslient and scaleable. We can further discuss pros and cons about this design in the review meeting.