Project review + recap - Heikki Pulli

Q1

Personalized MagicMirror

- **Strengths**

  One strength of this project is it's uniqueness. The idea of the projetc is cool and simple to understand. project is almost a finished product as is.

  Discussion part is really exhaustive. This clearly shows that lot of thought have went to this project.

- **Weaknesses**

  how does user update the software running in the MM? Or is it something that cannot be updated? This sounds a bit of risk considering how fast AI (face recognition) technology adnvances. Or if a new bug is discovered in existing software. Some software or update center would be needed to update the project and add new features to it.

- **Points**

  - **introduction (5 points):** Introduction was exhaustive and gave a good picture about the project that is being implemented. 5p
  - **Motivation (3 points):** Motivation was clear; We want to implement face recognition to the MagicMirror in order to show persons individual information without the fear of next person standing in front of the mirror sees the same information. 3p
  - **Background (2 points):** Background also motivates why people would want to use the personalized MM. Also background tells why facial recognition is possible to use and also what risks it involves. This could also be stated in another part... . 2p
  - **System architecture (5 points):** The whole system architecture figure gives a good view of the whole PMM workflow. The most important part, the face recignition module, is explained well. Added figures detail the work of the module well to the reader. I don't know how much added equations explain the process of face recognition to an average reader... Transport and personal agenda modules are explained quicker than face recognition module. Figures could have helped explaining the workings to the reader, but this is minor nitpick. 5p.
  - **Performance (5 points):** Performance doesn't include any graphs. These might've been good to demonstrate the workings of the project on different loads. Otherwise article explains well how projects performance was conducted. 4p.
  - **Discussion (5 points):** There is a lot of stuff in discussion. Most of the discussion focuses on the security of the face recognition module and agenda module and how to make PMM faster. Caching is mentioned as one solution but this introduces state handling to PMM. 5p
  - **Availability (5 points):** All the code of projects is available. Also all the modules are in separate repositories which is good for reusability. 5p.

Thoughts

In the article writers mention that most of the modules are stateless, except the race recognition module, which stores the vector representing faces. Maybe these components could be deployd to serverless environments. This would lessen the overhead of local computations, and updating these serverless modules would be easier.

gitlab-deployment-configurations

- **Strengths**

Project utilizes relevant tools (ansible, terraform) and usage of these tools makes it easier to use cloud utilities of different providers. Also setting up own source code management environment is also relevant not only government projects but also ie. small businesses and start ups.

- **Weaknesses**

Project is only designed to work on big cloud providers platforms. Trying to use this on own ie. VM wouldn't work.

- **Points**
  - **introduction (5 points):** Introduction fleshes out the main idea of the project and also motivates why this would be needed. 5p.
  - **Motivation (3 points):** Motivation is short and clear and focuses on what writers want to learn. 3p.
  - **Background (2 points):** Background only tells about personal experiences of the problem project tries to solve. Adding references when self-hosted code management fails or how much it costs could've been added. 2p.
  - **System architecture (5 points):** There is a little mistake in the system architecture figure. Terraform doesn't interact with gitlab instance, it only manages the VPC instance. The arrow should go from terraform to gitlab instance. Otherwise Architecture figure is good and informative. 4p.
  - **Performance (5 points):** Not much performance evaluation has been done. Only performance testing was that "Can we push commit to our gitlab instance". Interesting performance evaluation would be ie. multiple commits at the same time, how it affects VPC (monitoring), setting up CI/CD pipeline using gitlab ready made solutions and how it affects VPC and when multiple commits start the pipeline. Performance evaluation of the project is quite poor. 2p.
  - **Discussion (5 points):** Main weakness of using terraform is discussed, being that deployment configurations are not cloud service agnostic and it is needed to write multiple configurations file for multiple cloud providers. It is also stated that the strength of deploying such project to cloud is it's high availability. 5p.
  - **Availability (5 points):** All relevant information is given in the section. 5p.

Q2

Relevant articles to my group projects are Virtualization, Borg and Serverless.

Virtualization is relevant to the project because project employs docker containers for hosting single services of the project, ie. Keycloak, nextcloud, kube-monkey. Without the containerization of the services each service would have beed needed to manually deploy. Containers are much easier to deploy, because Dockerfile contains all the instruction son how to deploy the service.

Borg is relevant to the project because it is predecessor to the kubernetes that is used in the project. Kubernetes automates spinning up services to the cluster, managing the errors and restarting the services if needed. It also handles networking between services in much more easier way than without it. Deploying this project to a server without kubernetes would require constant monitoring of the services and manually restarting them if something goes wrong.

Serverless is somewhat relevant to the project, because some of the services could be replaced with BaaS versions of them, ie. keycloak could be replaced with google cloud autheticator. This could be preferble, because managing sensitive data on self-hosted platform is usually a bad idea. Delegating this job to google would be preferable, if you trust google or any big cloud providers with this. Smaller authentication managers could also be used but the main idea is to delegate authentication of your user to some trusted third-party provider.

Q3

Articles relevant to other peoples projects are Serverless FaaS, Ray, Virtualization and Ramcloud.

For personalized magicmirror serverless functions could be employd as stated in the thought section. This takes computational load away from small devices used in the mirror and delegates it elsewhere. Maybe even facial recognition model could be deployd to the FaaS environment, because it only takes in input a image of the person and processes it into vector.

PMM could leverage kubeflow project to automize the training and deployment of face recognition models to these FaaS environments. Also Ray could be used in the training of bigger face recognition models for PMM inside kubeflow. Some cloud providers already serve this kind of service https://cloud.google.com/blog/products/ai-machine-learning/build-a-ml-platform-with-kubeflow-and-ray-on-gke.

Containers play a big role to the PMM project. Containers seem to be status quo on production software virtualization. Updates could be rolled as updated Dockerfiles and images to the PMMs from update center to solve the updating problem of the mirros.

Also PMM could leverage Ramcloud solution to cache the personal agenda information to save time on loading information againg vs. loading it from cached memory. This could make PMM faster to use and latency would disturb the user as much.