VM’s set up(airflow/mlflow) review

## Review of Current infrastructure set up

Currently, we have Terraform to create the cloud infrastructure + bash scripts for setting up VM with Airflow, Mlflow and other useful packages. The bash scripts need some update since the airflow package is updated.

In order to optimize the VM’s setup process, we can review different options for setting up development environment for AI activities, (Airflow could be run locally or in Docker)

### Option 1: (current one)

Run airflow, mlflow and other useful locally in VM in GCP: need some modification of the installation scripts to keep it up to date.

Pros:

1. it is more flexible and reliable. We could install any other packages when we need it. And we could set up the VM size to adapt the usage.
2. We already have the scripts, we just need some small updates.

Cons: when we need to setup a new environment, it needs some bash knowledge to setup the VM, it may encounter some error during the set-up, since the packages may have conflicts.

### Option 2:

Run airflow, mlflow in docker in VM in GCP:

Pros:

The docker image could be used in any other environment

Cons:

1. it requires extra effort to build docker image, and more efforts to keep it up to date

The Apache Airflow community, releases Docker Images which are reference images for Apache Airflow. However, Airflow has more than 60 community managed providers (installable via extras) and some of the default extras/providers installed are not used by everyone, sometimes others extras/providers are needed, sometimes (very often actually) you need to add your own custom dependencies, packages or even custom providers. So custom image is needed.

1. It is not so flexible, when we need other packages, we need to rebuild the docker images

### Option 3:

Run airflow in Cloud run. It is not practical. Since cloud run is stateless container (no persistent disk) and answer to HTTP request (no batch, no cron, no background process,...). Besides, we still need to build the docker image

## The possible ways to optimize

1. Use configuration management(CM) tools instead of bash scripts for set up the VM. For instance, Ansible(<https://opensource.com/article/20/9/install-packages-ansible>). However, it requires extra effort to learn the tools.
2. Create an image from the existing VM, use this image to create a new VM using terraform (Note: the VM should be in the same network as the cloudSQL instance which store the backend database for Mlflow, and should grant the access to all google API services). Once the VM is created, enter into VM, change the configuration file of the systemd unit service in directory /usr/lib/systemd/system/

* cloudsqlproxy.service
* gcs-airflow.service
* mlflow-webserver.service

and modify the /etc/airflow file.

<https://cloud.google.com/deployment-manager/docs/configuration/using-images-from-other-projects-for-vm-instances#console>

Way 2 is most optimal way in current setting.

## Other useful information:

Managing infrastructure as code with Terraform, Cloud Build, and GitOps

<https://cloud.google.com/architecture/managing-infrastructure-as-code>

Managing access to custom images

<https://cloud.google.com/compute/docs/images/managing-access-custom-images>