

BTP Seminar



SKG 1

Docker containers for the deployment of geospatial services

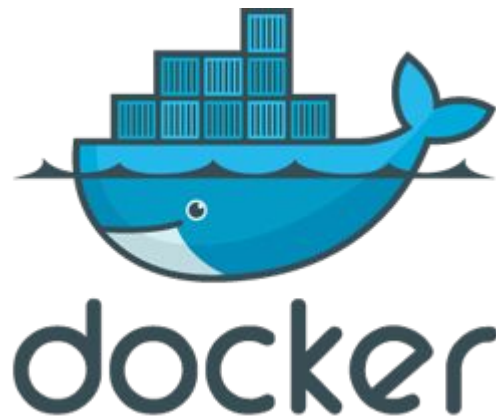
Mayank Bhushan
15CS30019

Siddharth Jain
15CS30031

Docker

Docker is a container management software. The whole idea is for developers to easily develop applications, ship them into containers which can be deployed anywhere.

It is an open platform that can be used for building, distributing, and running applications in a portable, lightweight runtime and packaging tool, known as Docker Engine. It also provide Docker Hub, which is a cloud service for sharing applications.



Key features



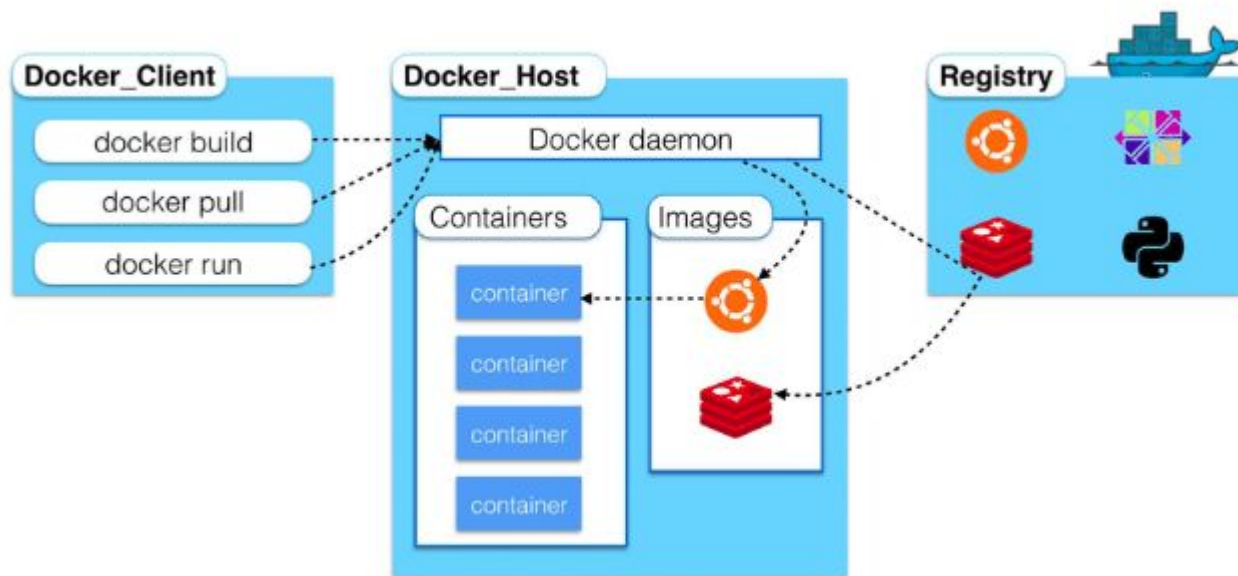
- Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
- With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
- You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
- Since Docker containers are pretty lightweight, they are very easily scalable.
- Costs can be reduced by replacing traditional virtual machine with docker container.

Docker Installation



- Step 1 : Add GPG key
 - `sudo apt-key adv --keyserver hkp://ha.pool.sks-keyservers.net:80 --recv-keys 8118E89F3A912897C070ADBF76221572C52609D`
- Step 2 : Add repository list and update
 - `echo "deb https://apt.dockerproject.org/repo ubuntu-trusty main" | sudo tee /etc/apt/sources.list.d/docker.list`
 - `sudo apt-get update`
- Step 3 : Install Dependencies
 - `sudo apt-get install linux-image-extra-$(uname -r) linux-image-extra-virtual`
- Step 4 : Install Docker
 - `sudo apt-get install -y docker-engine`
- Step 5 : Proxy Settings in `etc/systemd/system/docker.service.d/http-proxy.conf`
 - ```
[Service]
Environment="HTTPS_PROXY=https://172.16.2.30:8080/"
```

# Docker Architecture



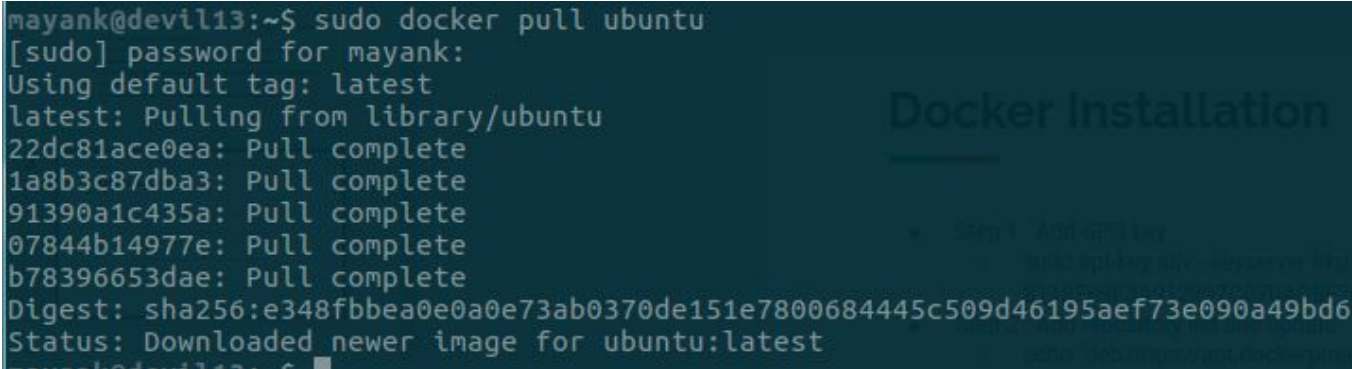
# Docker Image

An image is the complete standalone package of software that can includes everything needed to run the image.

The `docker run` command takes the Docker image as a template and produces a container from it. Images are created from a Dockerfile with the `docker build` command.

Images are stored in a Docker registry, such as Docker Hub and can be downloaded with the `docker pull` command:

```
mayank@devil13:~$ sudo docker pull ubuntu
[sudo] password for mayank:
Using default tag: latest
latest: Pulling from library/ubuntu
22dc81ace0ea: Pull complete
1a8b3c87dba3: Pull complete
91390a1c435a: Pull complete
07844b14977e: Pull complete
b78396653dae: Pull complete
Digest: sha256:e348fbbbea0e0a0e73ab0370de151e7800684445c509d46195aef73e090a49bd6
Status: Downloaded newer image for ubuntu:latest
```



# Docker Image



There are two types of images based on hierarchy:

- **Base Image :**
  - has no parent image
- **Child Image :**
  - build on base image and add functionality

# Docker Container

A container is a runtime instance of an image—what the image becomes in memory when actually executed.

Containers are created from images with the `docker run` command and can be listed with the `docker ps` command.

```
mayank@devil13:~$ sudo docker run -it ubuntu /bin/bash
root@8bb0d2d6d030:/# exit
exit
```

```
mayank@devil13:~$ sudo docker ps
```

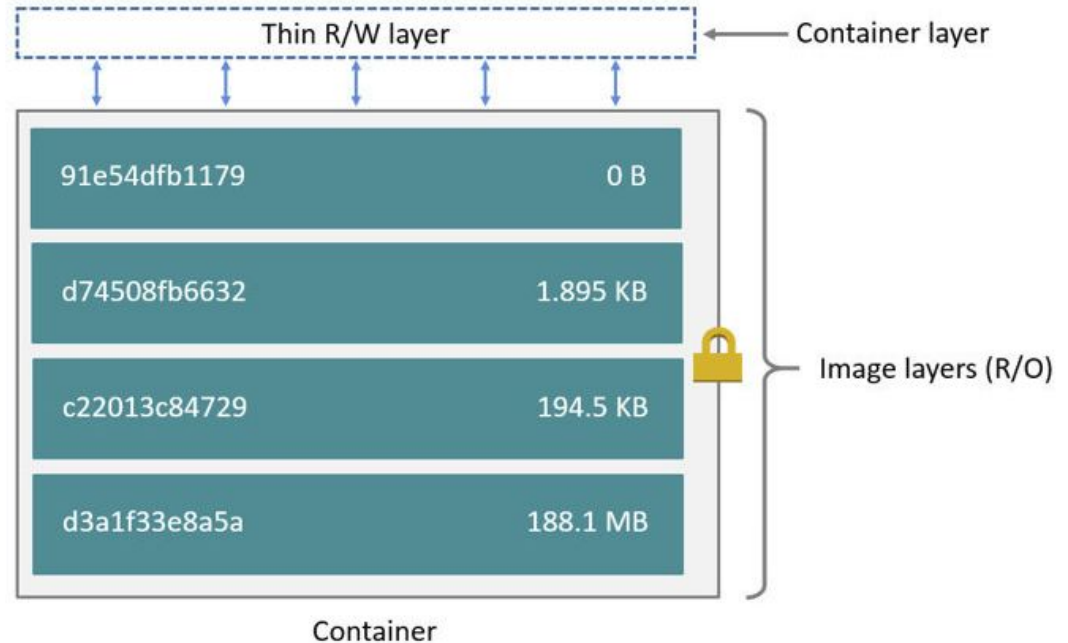
| CONTAINER ID | IMAGE         | COMMAND        | CREATED           | STATUS           | PORTS                 | NAMES |
|--------------|---------------|----------------|-------------------|------------------|-----------------------|-------|
| e65d460fca43 | mayank11/site | "./wrapper.sh" | About an hour ago | Up About an hour | 0.0.0.0:32769->80/tcp | site  |



# Docker Container

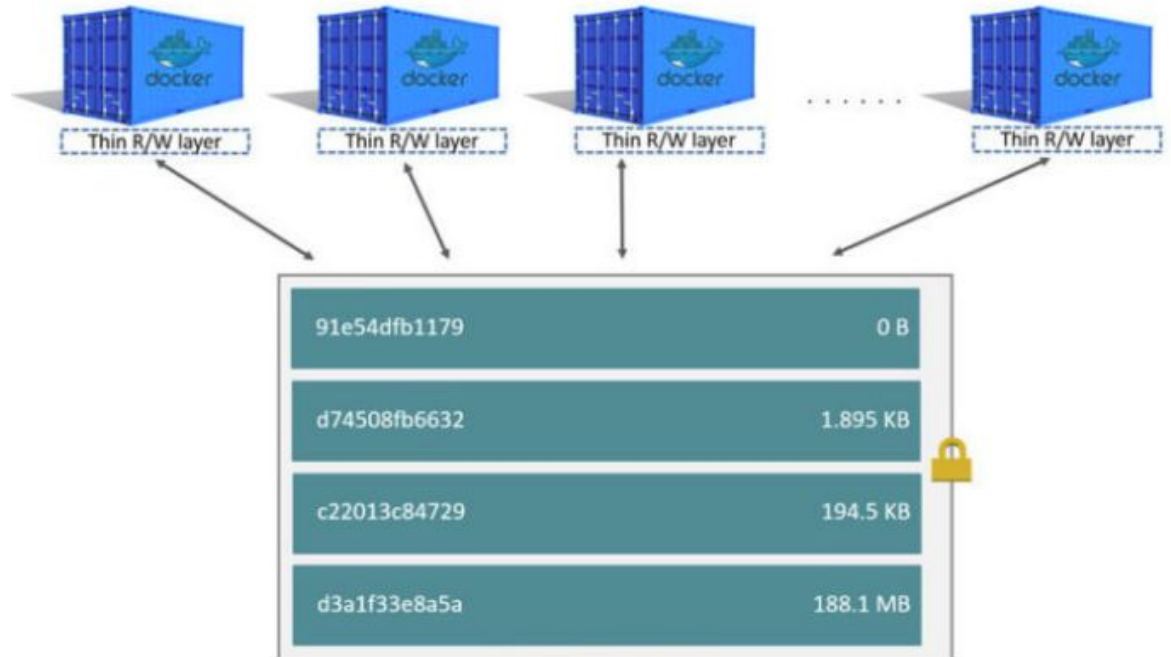
To create a container, Docker engine takes an image, adds the top writable layer and initializes various settings (network ports, container name, ID and resource limits).

All write operation inside the container are stored in this writable layer, so when the container is deleted, the writable layer is also deleted while the underlying image remains unchanged.

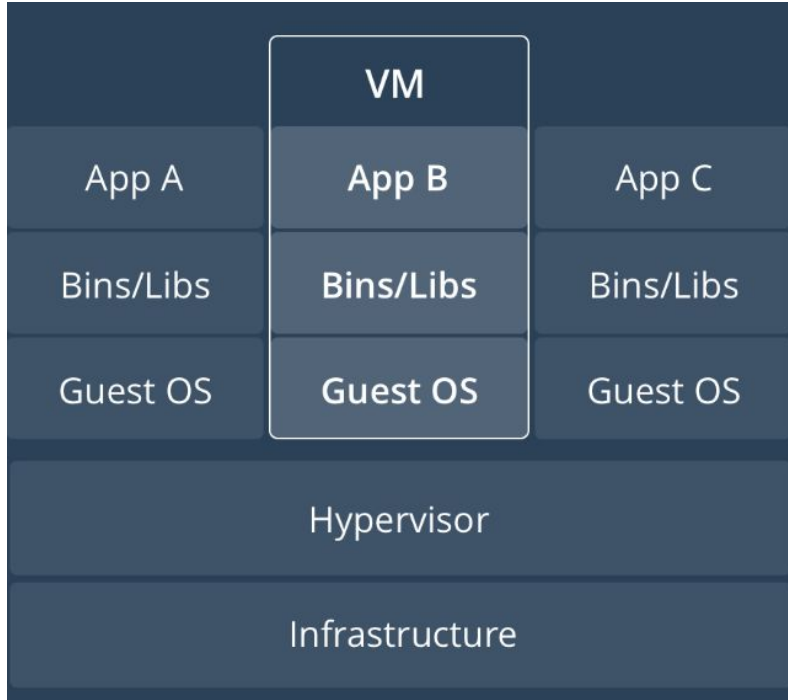


# Docker Container

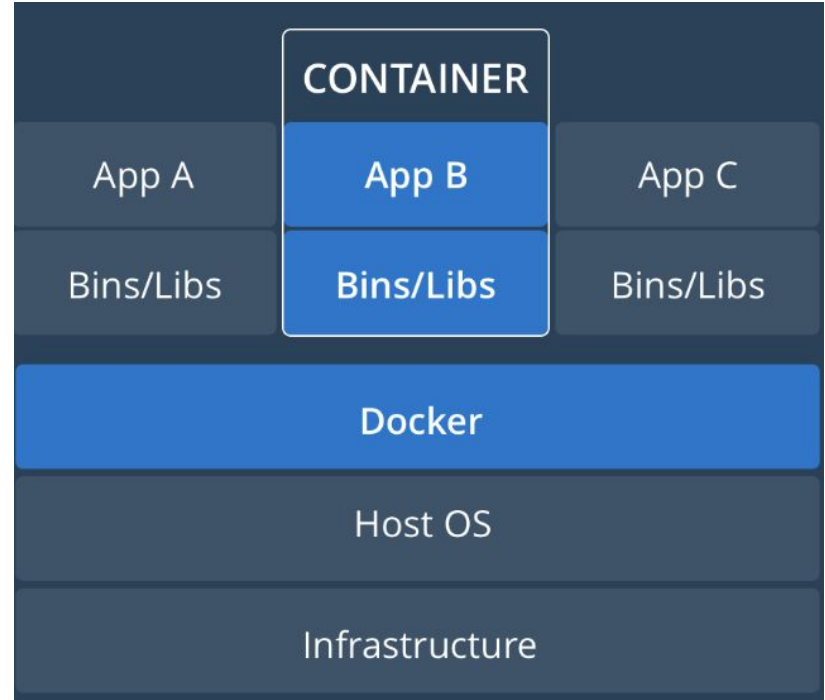
As each container has its own writable container layer, and all changes are stored in this container layer, multiple containers can share access to the same underlying image and yet have their own data state.



# Docker Container vs Virtual Machine



Virtual Machine



Docker Container

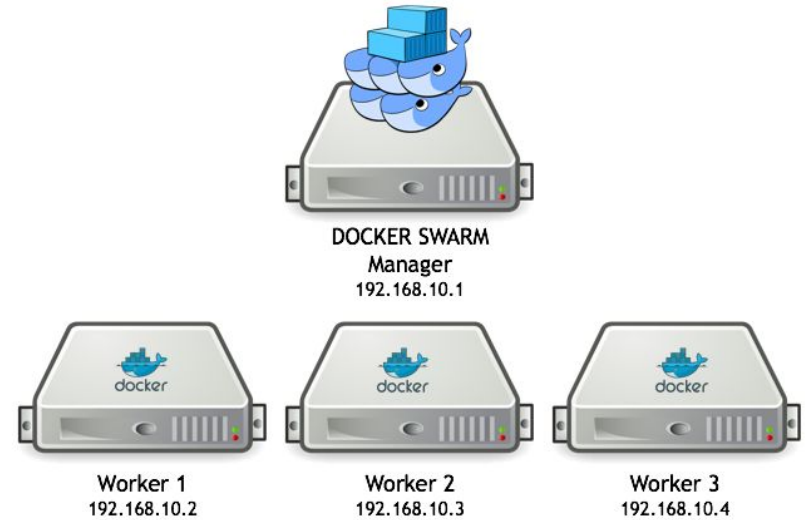
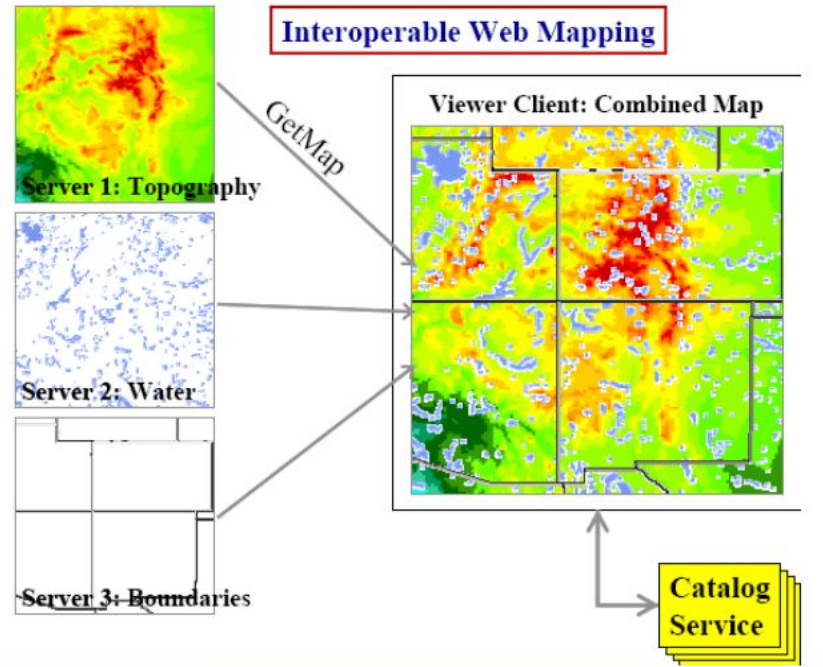
# Docker Container vs Virtual Machine



Each Virtual Machine includes a separate operating system image, which adds overhead in memory and storage footprint. This issue adds complexity to all stages of a software development lifecycle—from development and test to production and disaster recovery. This approach also severely limits the portability of applications between public clouds, private clouds, and traditional data centers.

Containers can share a single kernel, and the only information that needs to be in a container image is the executable and its package dependencies, which never need to be installed on the host system. They contain all their dependencies so there is no configuration entanglement; a containerized app “runs anywhere.” Containers are thus exceptionally “light”—they are only megabytes in size and take just seconds to start, versus gigabytes and minutes for a VM.

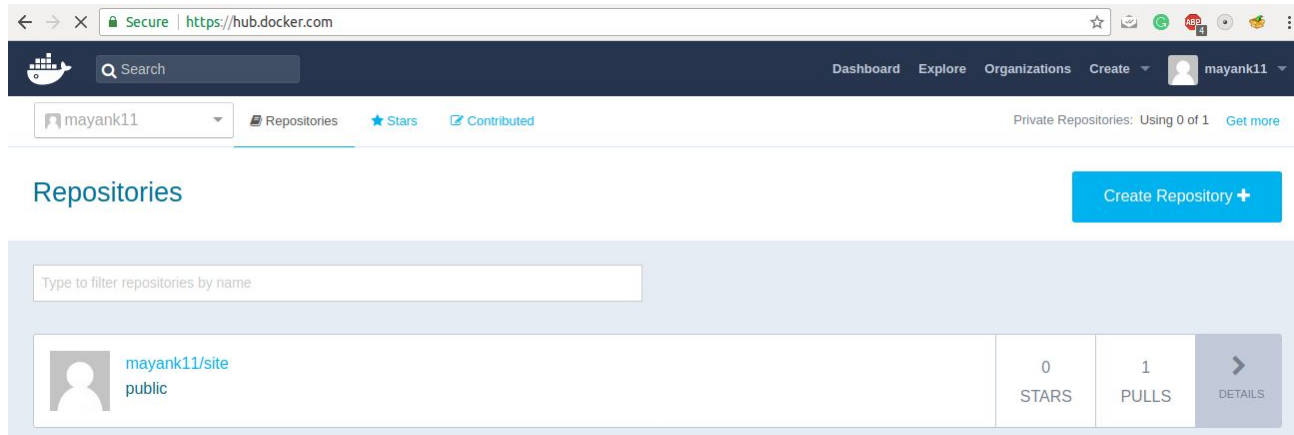
# Dockers in GeoSpatial services



# Demo

We have created a docker image which uses Nginx to host a site. Nginx is a free open source HTTP server.

Base Image : nginx



The screenshot shows the Docker Hub web interface. The browser address bar displays 'Secure | https://hub.docker.com'. The navigation bar includes links for 'Dashboard', 'Explore', 'Organizations', 'Create', and a user profile for 'mayank11'. Below the navigation bar, there's a search bar and tabs for 'Repositories', 'Stars', and 'Contributed'. The 'Repositories' tab is active, showing a list of repositories. A search bar at the top of the repository list says 'Type to filter repositories by name'. The first repository listed is 'mayank11/site' with a public status. To the right of the repository name, it shows '0 STARS' and '1 PULLS'. A 'Create Repository +' button is visible in the top right corner of the repository list section.

| Repository              | Stars | Pulls | Details   |
|-------------------------|-------|-------|-----------|
| mayank11/site<br>public | 0     | 1     | > DETAILS |

# Demo

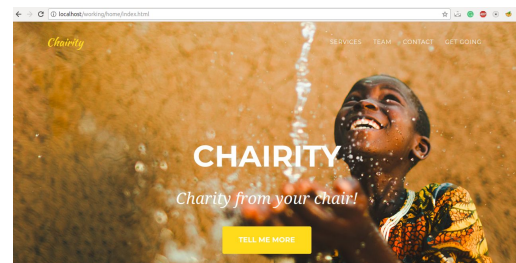
Dockerfile

```
FROM nginx
MAINTAINER Mayank Bhushan <mayankbhusan@iitkgp.a.in>
COPY wrapper.sh /
COPY html /usr/share/nginx/html
CMD ["/wrapper.sh"]
```

wrapper.sh

```
#!/bin/bash
echo "Nginx is running..."
exec nginx -g "daemon off;"
```

html



# RC2



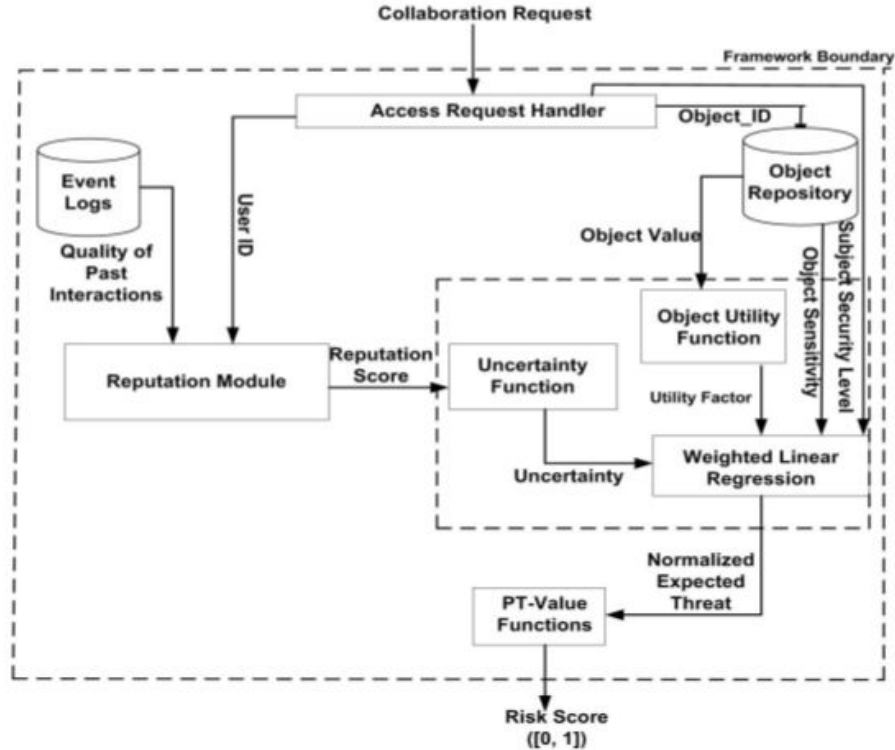
The current models for authorization in the collaborative services of cloud service provider are only concerned with the identity of the requester and the validity of his token. They do not consider the history(previous requests) of the requester, which is important because a legitimate requester with a valid token can still have malicious intentions.

The currently available Risk based Access Control do not take uncertainty into account and are slow.

So, there is a need for a risk quantification mechanism which is generic and light-weighted to suit cloud-based applications, and incorporates soft security mechanism (e.g., reputation) to model uncertainty related to resource sharing.



# RC2



# RC2



RC2 framework is an additional security layer above the token based authorization scheme. To avoid the overhead it can be implemented as a web service which is invoked when collaboration requests are received.

The models approach is such that it maximises the reputation for all the new requesters, and penalize them only if they have been found to misuse the shared resources. Otherwise the requester continues with maximum reputation and all legitimate accesses are permitted.

# References



- Official docker documentation
- [forums.docker.com](https://forums.docker.com)
- [www.stackoverflow.com](https://www.stackoverflow.com)
- RC2: A Risk Quantification Framework for Dynamic Access Control in Cloud-based Collaborations, *by Dr. Nirnoy Ghosh*



**Thank you!**