Docker Registry

What is an image

- Docker image is a snapshot or template from which new containers can be started. It's a combination of file system and libraries .

- A new image can be created by executing a set of commands contained in a Dockerfile For example, this Dockerfile would take a base Ubuntu 16.06 image and install git, resulting in a new image:

```
FROM ubuntu:16.04

RUN apt-get install -y git
```

- Image is composed of a set of read-only layers. Image layers function as follows:
  - Each image layer is the outcome of one command in the image's Dockerfile
  - Each additional image layer only includes the set of differences from the previous layer. (docker history command can be used to check all layer images)

What is an Image

Identified by a name

- ubuntu
- centos
- alpine

Command: **docker run hello-world**

- The Docker command is specific and tells the Docker program on the Operating System that something needs to be done.
- The **run** command is used to mention that we want to create an instance of an image, which is then called a **container**.
- Finally, "hello-world" represents the image from which the container is made.

Command : **docker images**

This command is used to display all the images currently installed on the system

Image Registries

Docker Registries

- A Docker registry is a place to store and distribute Docker images. It serves as a target for your docker push and docker pull commands
- Repository- Collection of different versions of a single Docker image
- Tags – Named version of an image . Provide meaningful name for easy reference
- Two types of Registries – Public and Private

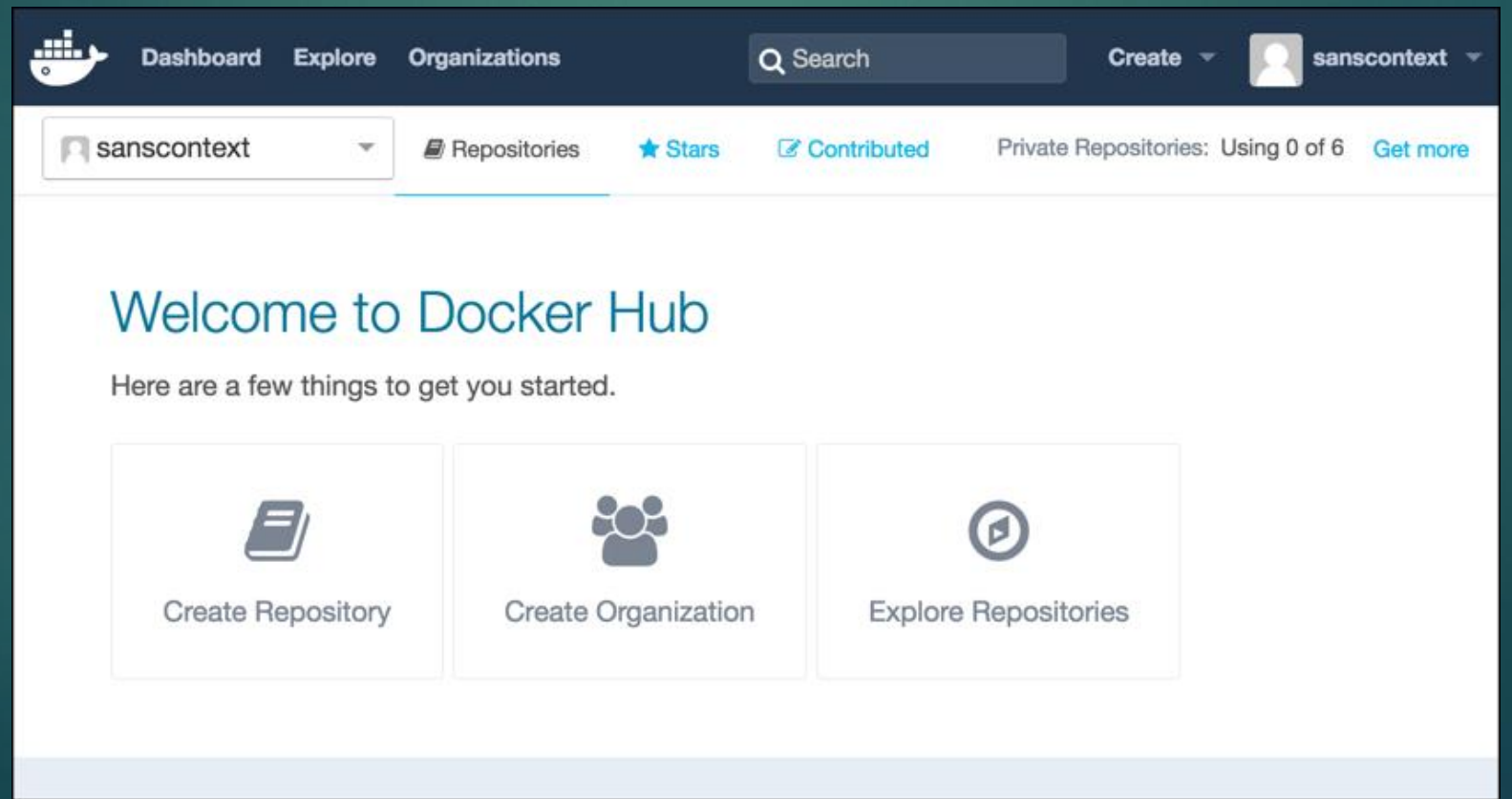
Public : Anyone can pull and push images . Eg: Docker Hub , Docker Store

Private : Strict Access control is provided for accessing the images

Eg: Google Container Registry, AWS ECR

Docker Hub

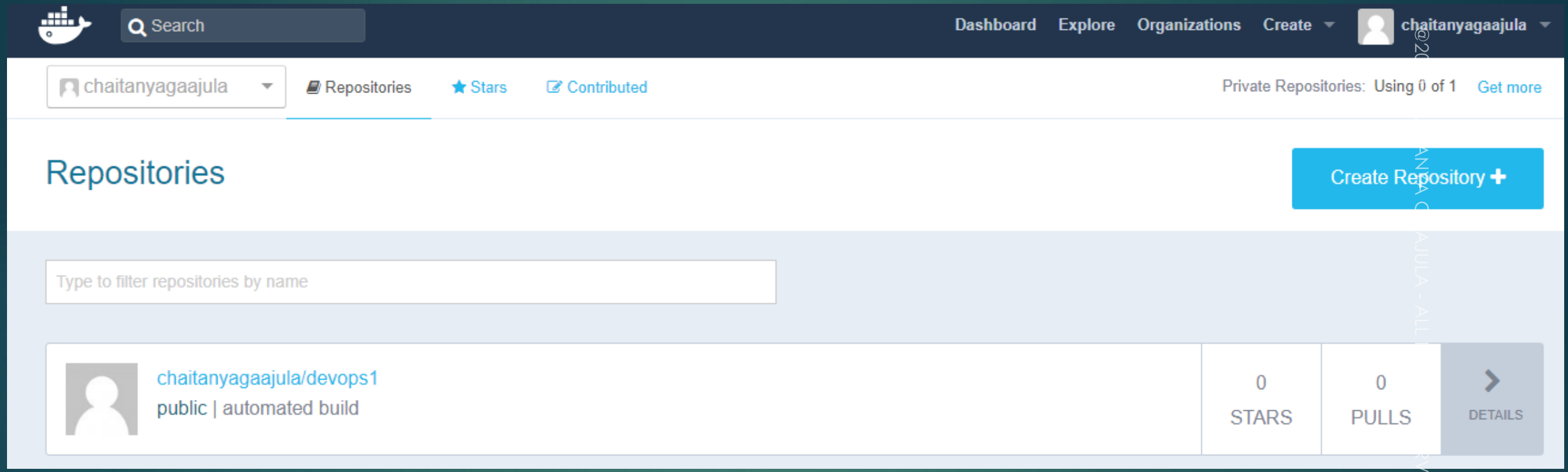
- Docker Hub is a cloud-based repository in which Docker users and partners create, test, store and distribute container images.
- Users can access public, open source image repositories, as well as use a space to create their own private repositories, automated build functions, and work groups.



Docker Hub-Features

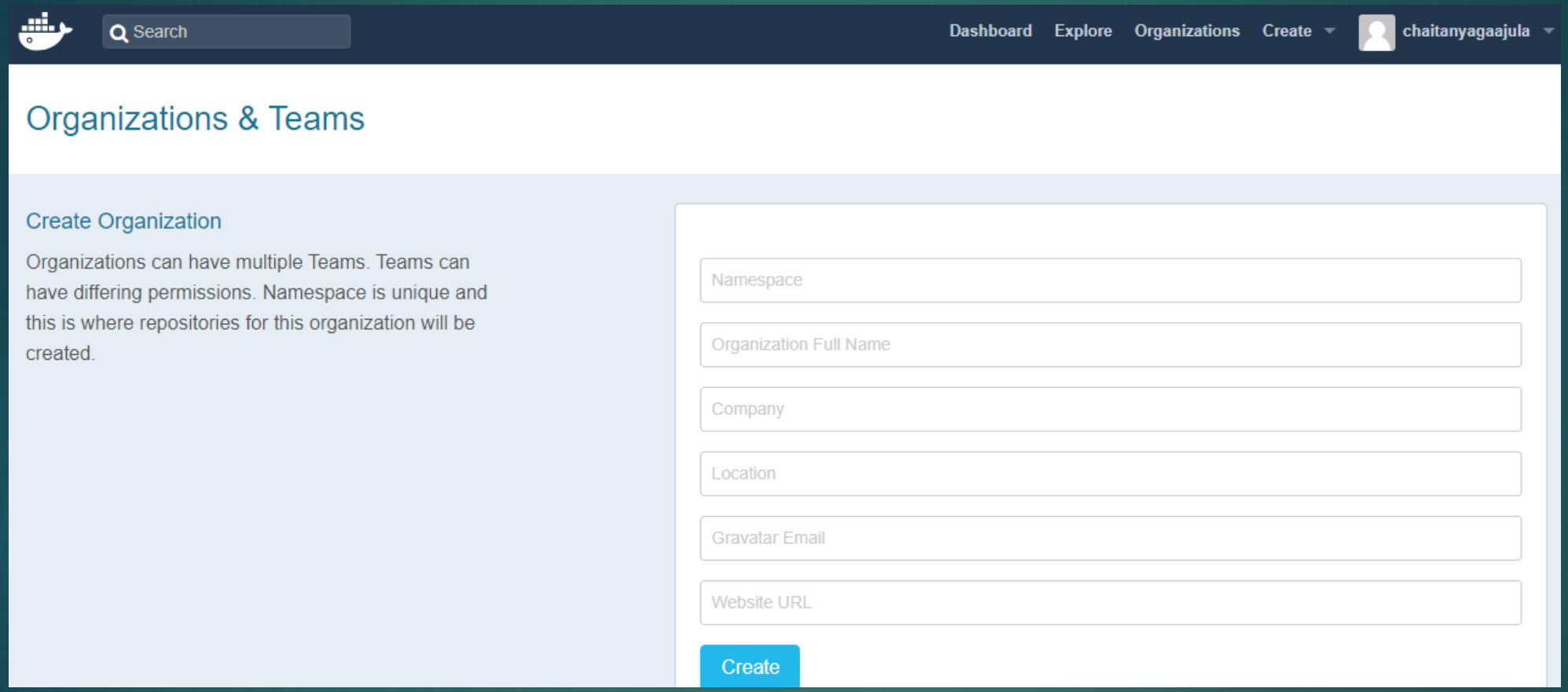
- Image Repositories: Find and pull images from community and official libraries, and manage, push to, and pull from private image libraries to which you have access.
- Automated Builds: Automatically create new images when you make changes to a source code repository.
- Webhooks: A feature of Automated Builds, Webhooks let you trigger actions after a successful push to a repository.
- Organizations: Create work groups to manage access to image repositories.
- GitHub and Bitbucket Integration: Add the Hub and your Docker Images to your current workflows

Docker Hub Repository



- Docker Hub repositories let you share images with co-workers, customers, or the Docker community at large.
- You can build your images and push them to a Docker Hub repository that you add to your Docker Hub user or organization account.

Organizations and Teams



- Docker Hub organizations let you create teams so you can give colleagues access to shared image repositories.
- Access to push or pull for these repositories is allocated by defining teams of users and then assigning team rights to specific repositories. Repository creation is limited to users in the organization owner's group.

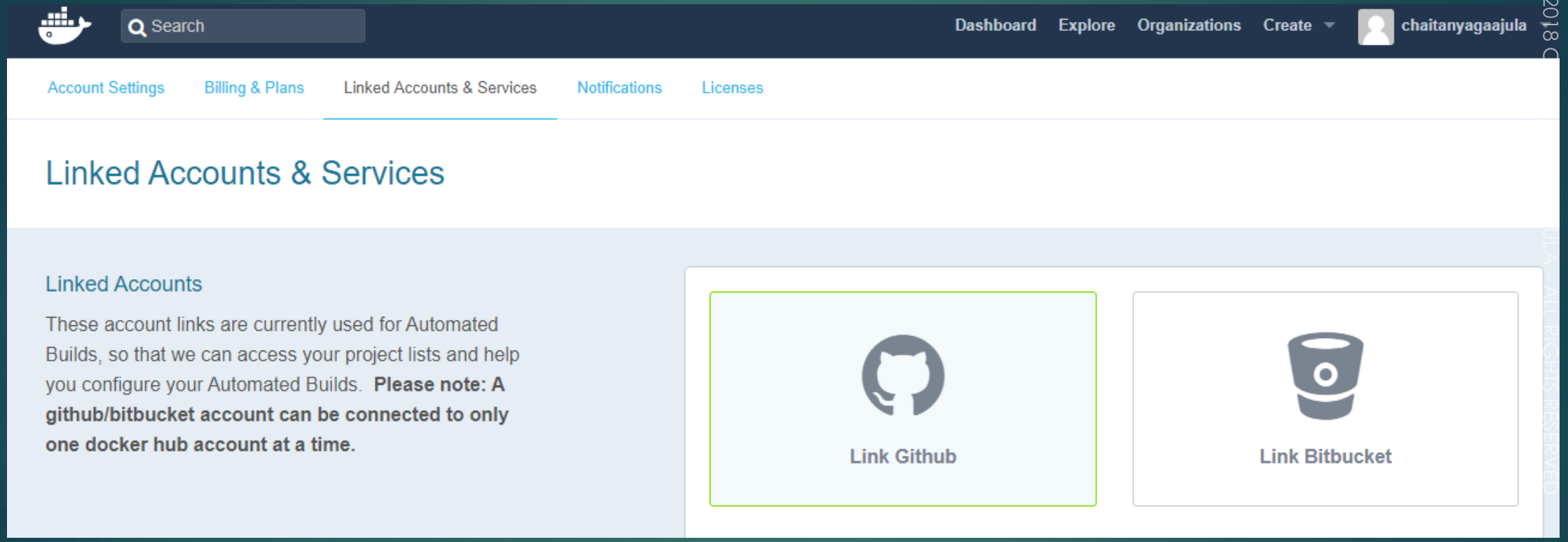
Automated Builds

- Build your images automatically from a build context stored in a repository. A build context is a Dockerfile and any files at a specific location. For an automated build, the build context is a repository containing a Dockerfile.

- Automated Builds have several advantages:

  - Images built in this way are built exactly as specified.
  - The Dockerfile is available to anyone with access to your Docker Hub repository.
  - Repository is kept up-to-date with code changes automatically.
  - Automated Builds are supported for both public and private repositories on both GitHub and Bitbucket.

Linked Accounts & Services



THANK YOU