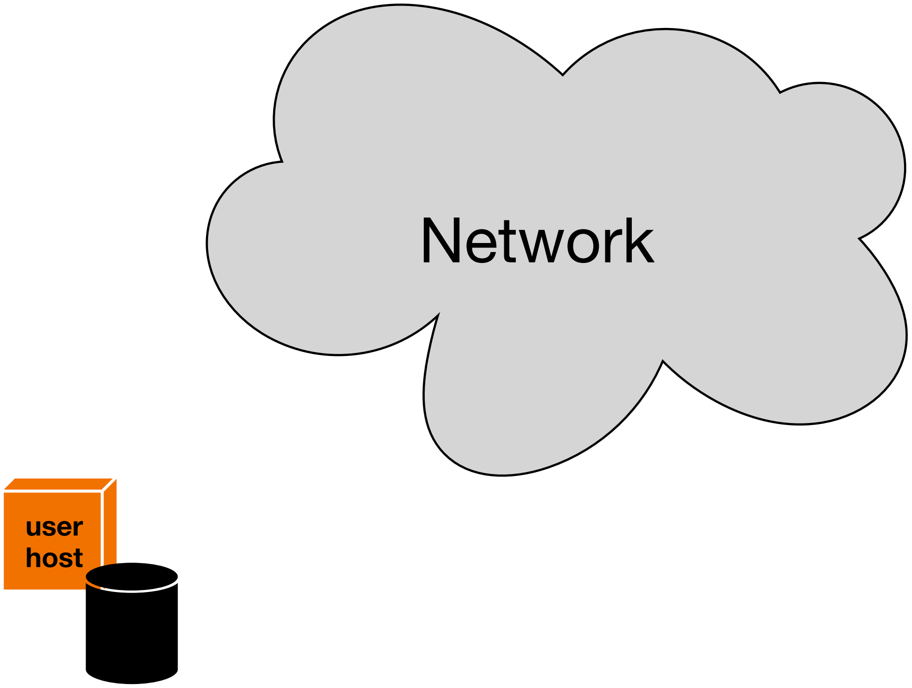
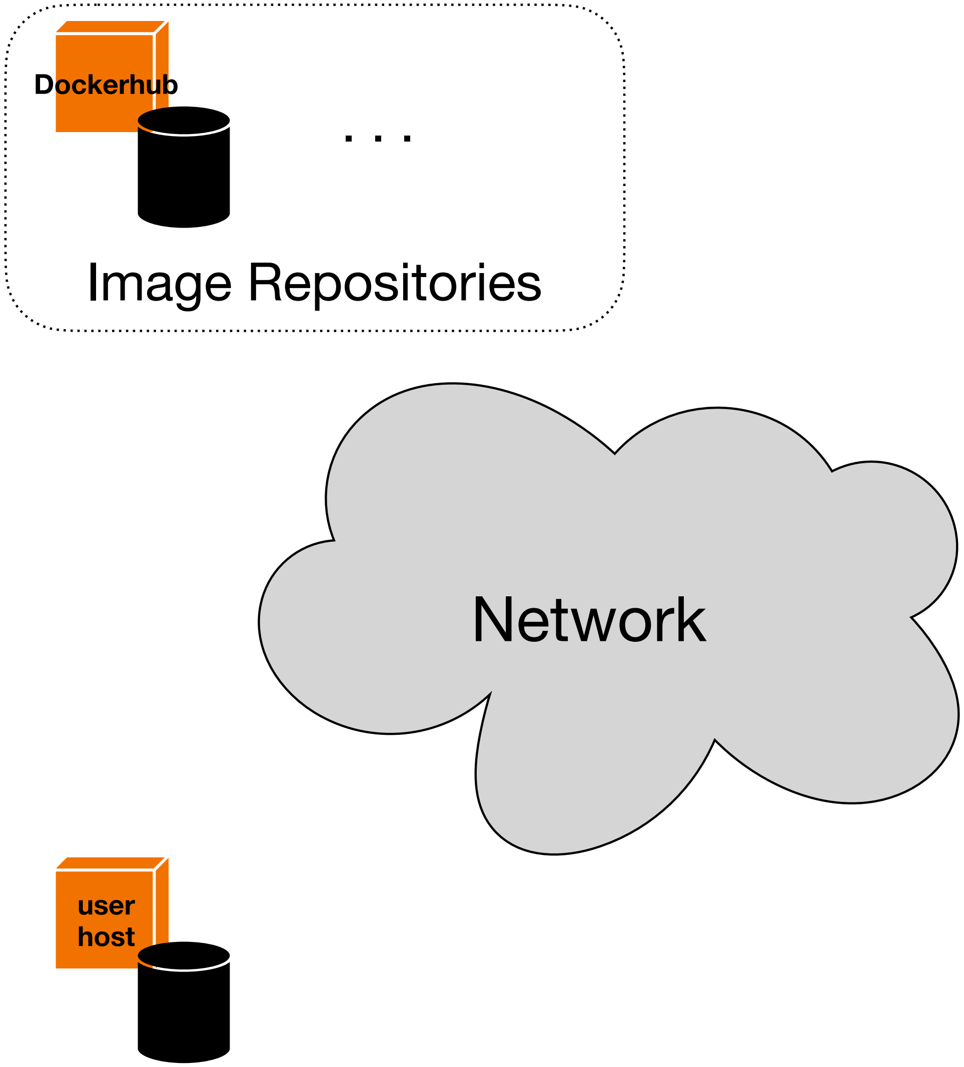

Network
user host



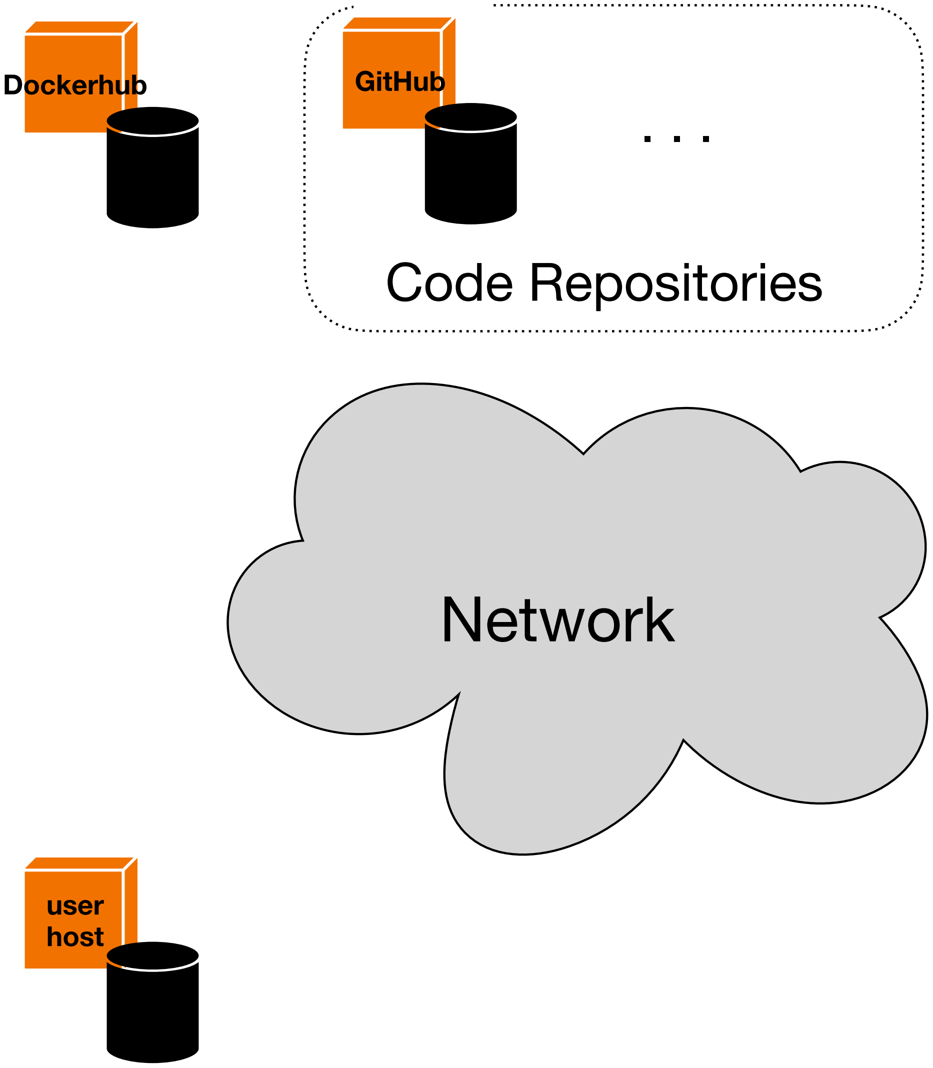
Dockerhub

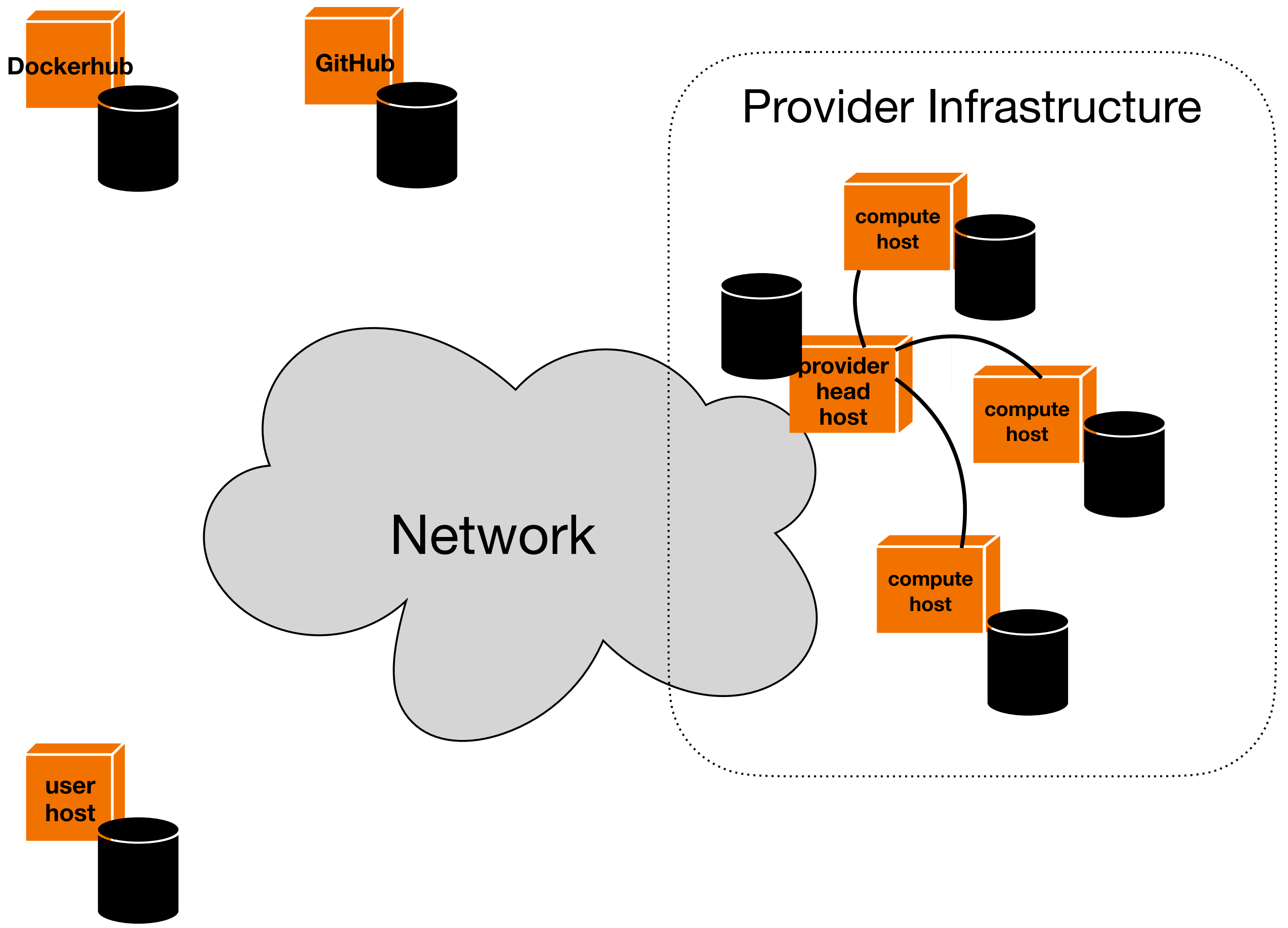
. . .

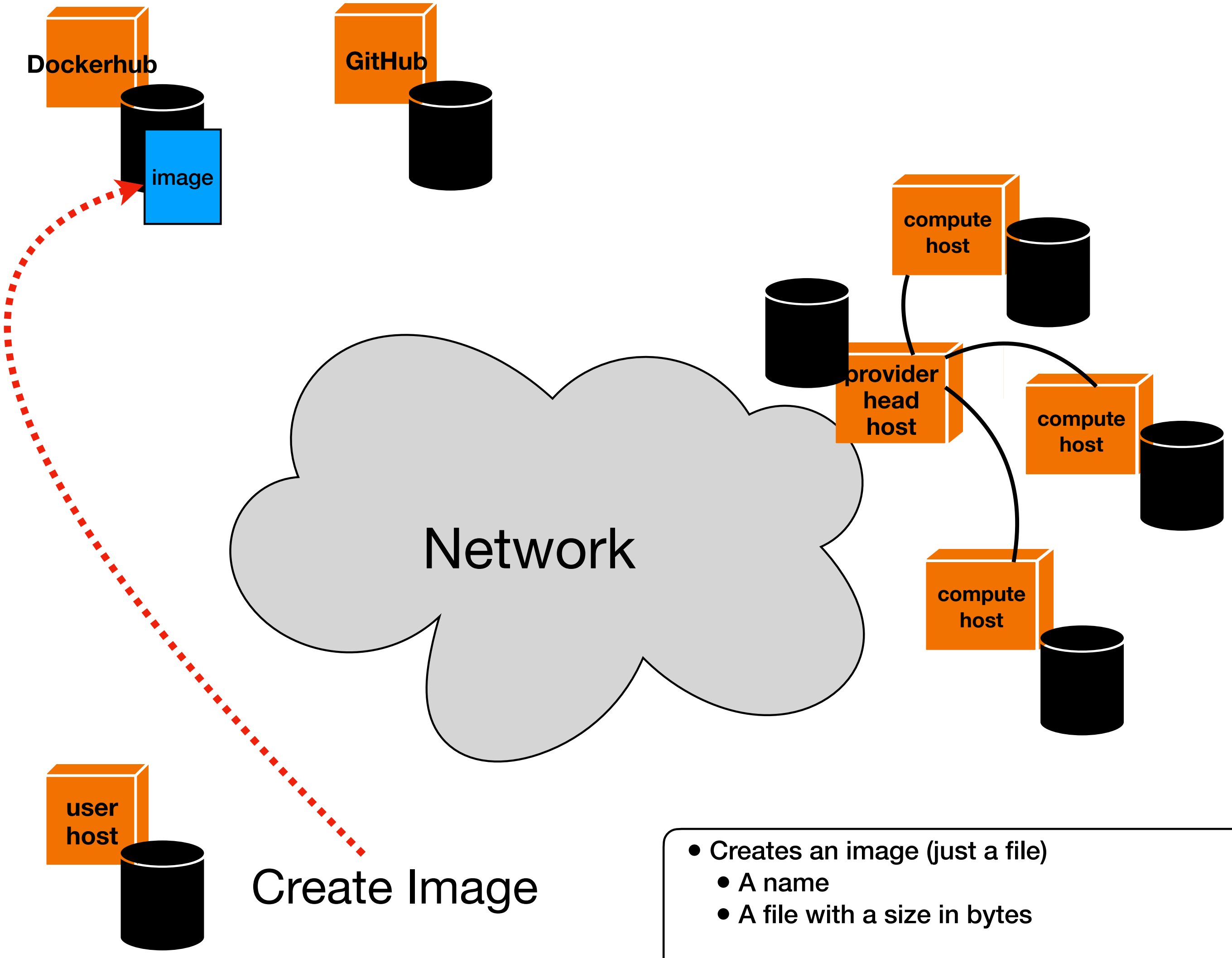
Image Repositories

Network

user
host

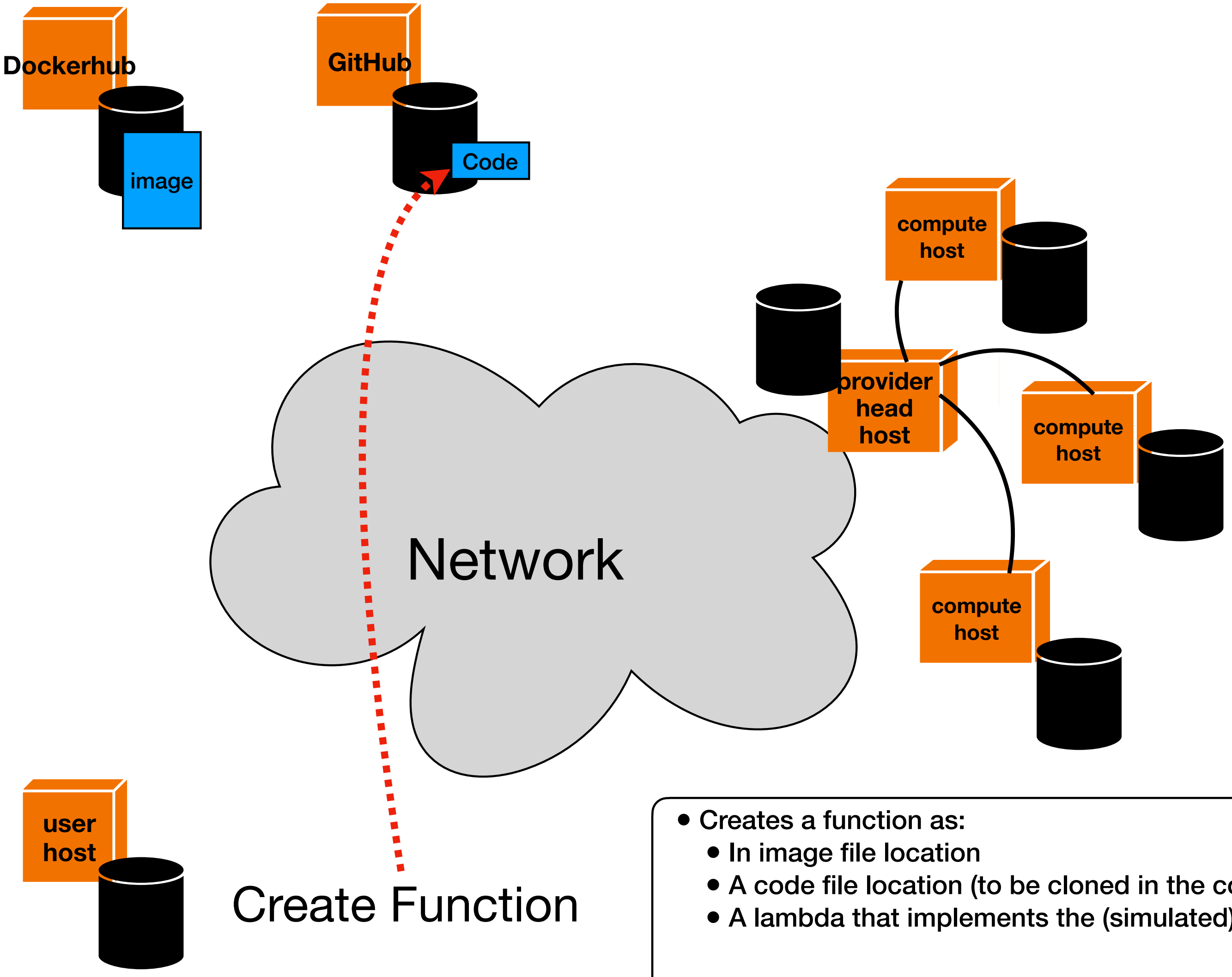

Dockerhub
GitHub
. . .
Code Repositories
Network
user
host


Dockerhub
GitHub
Provider Infrastructure
compute host
provider head host
compute host
compute host
Network
user host



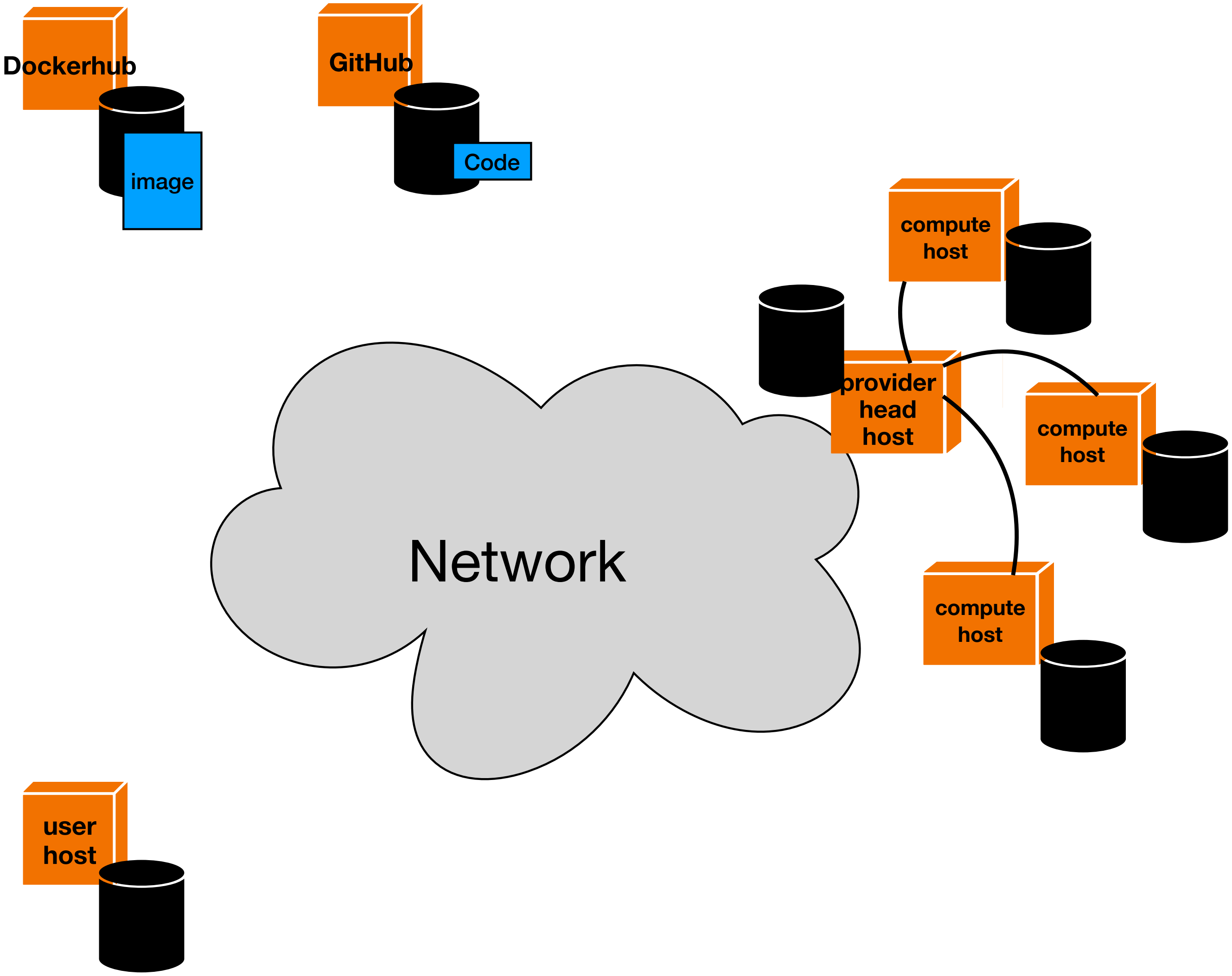
Create Image

- Creates an image (just a file)
  - A name
  - A file with a size in bytes
- Create copies of this file at at least one “location”



Create Function

- Creates a function as:
  - In image file location
  - A code file location (to be cloned in the container)
  - A lambda that implements the (simulated) code that will run in the container



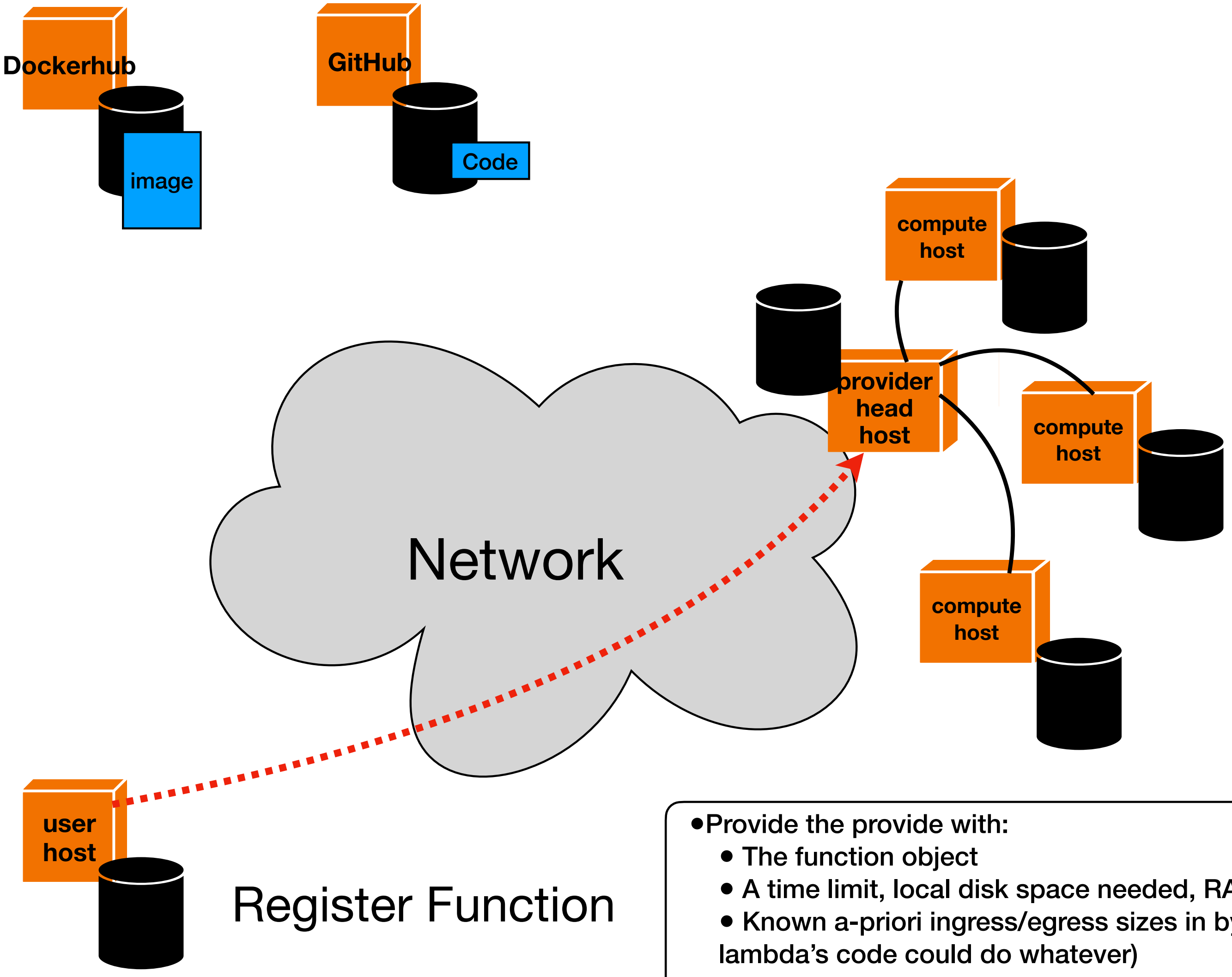
Provider Infrastructure

- There is a main storage disk to download images, managed as an LRU cache. Images are always downloaded directly there.

- Each compute host has a disk, that will store:
  - Image files downloaded from the main storage disk
  - Disk space reserved for each running container, as it seems that's expected (like the /tmp stuff on AWS)
  - Managed as a LRU cache for image files, giving priority to containers, enforcing that each running container is (of course) never evicted

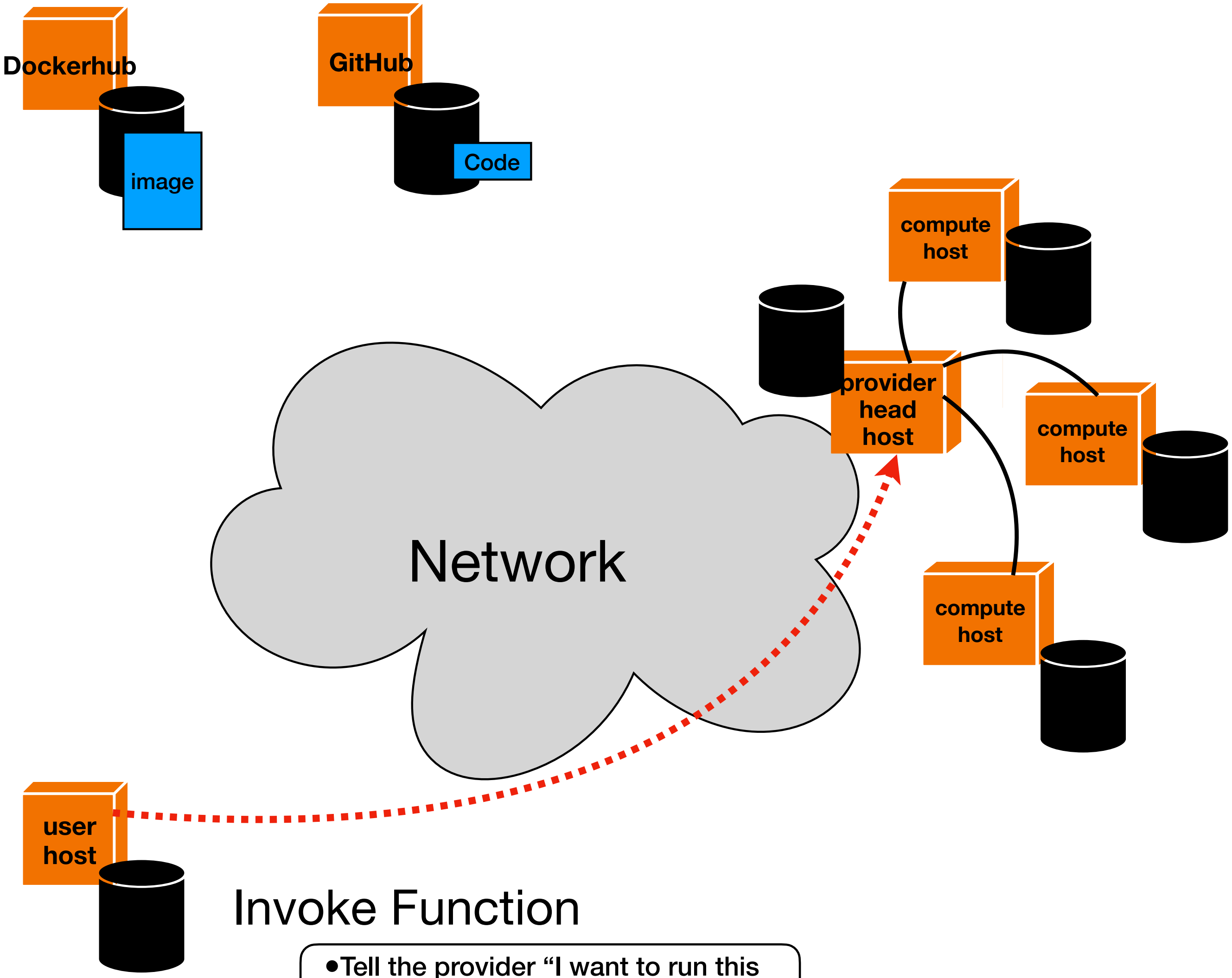
- Current assumption/limitation:
  - Hosts/disks are all homogeneous

- The provider's behavior is dictated by:
  - Host/disk hardware
  - # of function slots per host
  - Various overheads (start container if warm , etc.)
  - Billing model: a lambda that takes in whatever input (e.g., RAM + time limit + local disk space + ingress/egres)
  - Some allocation/scheduling strategy to pick a compute host (for now likely hardcode something stupid)



Register Function

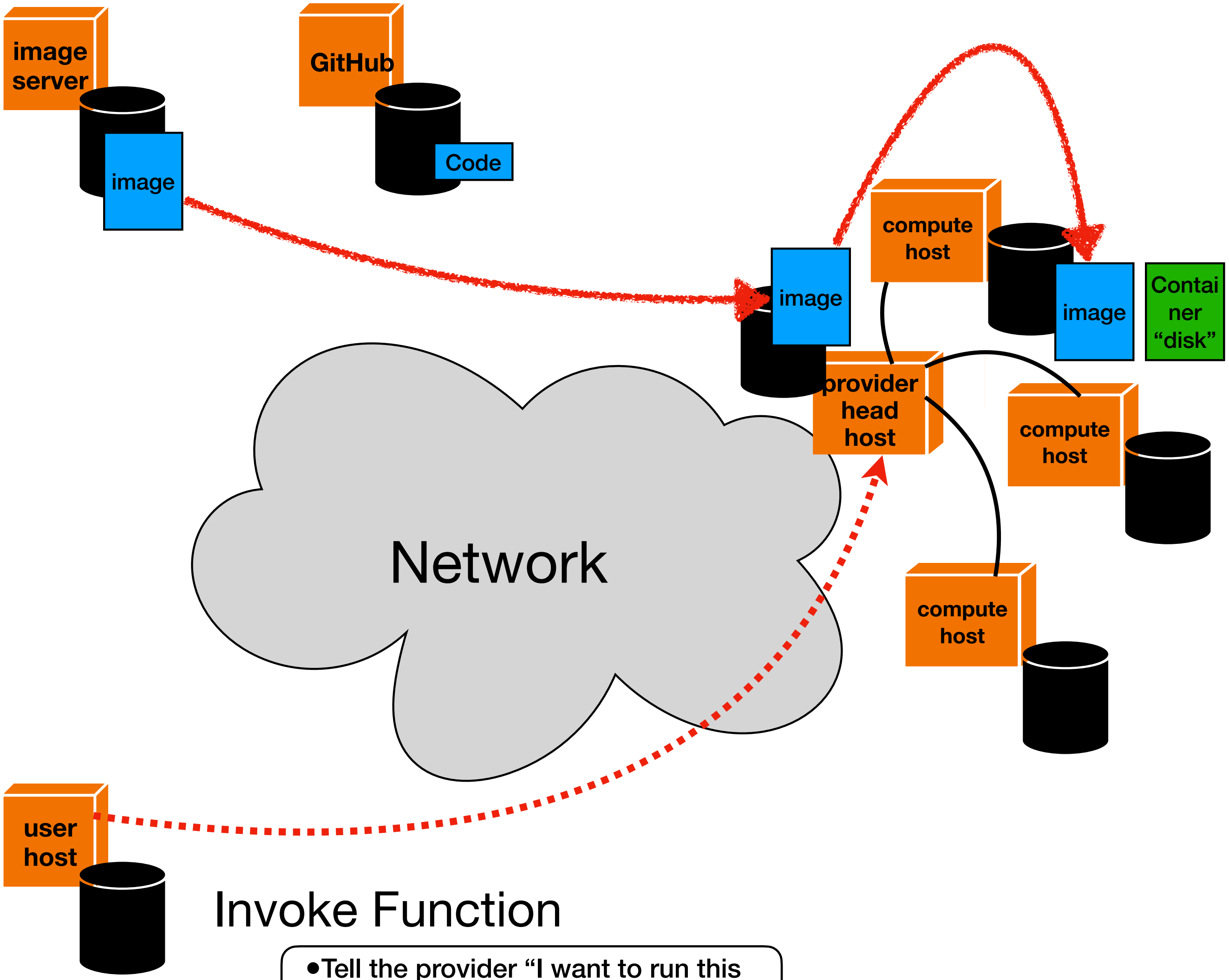
- Provide the provide with:
  - The function object
  - A time limit, local disk space needed, RAM limit
  - Known a-priori ingress/egress sizes in bytes (even though the lambda's code could do whatever)



Invoke Function

- Tell the provider “I want to run this function”, providing the function input

Invoke Function



- Tell the provider "I want to run this function", providing the function input

- The provider:
  - Downloads the image file is needed
  - Picks a compute host using some resource selection algorithm
    - What about RAM pressure?
      - FOR NOW: A property that picks one of two behaviors:
        - 1. If no RAM is available on any host, return some "resource unavailable" thing.
        - 2. "Buffers" it and starts the timer whenever it starts.
  - Copies the image file from the main disk to the local disk
  - Starts the container to run the function
  - Git clones the code, build, and then run the code (all fake of course)
  - The container then runs to completion, or fails, or timeouts

- The user host can call several methods:
  - get_state(one), get_states(several)
  - wait_for(one), wait_for(several), wait_for_any(several)
  - And then, if completed: get_output()