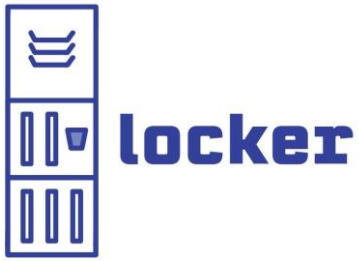


Locker: a Simple Tool to Run and Manage Interactive Docker Containers Supporting Reproducible Research

Andrew Smith (andrew.smith1@bms.com)

December 2023

Overview

1. Brief Overview of Docker and Locker
2. Overview of Locker Application
   - Prerequisites and Setup
   - How to start Locker application directly using startup scripts (e.g., on Win or Mac laptop)
   - Overview of Locker functionality and UI
   - Safe Storage Locations inside Containers
   - Checklist for using Locker
3. Locker Services to start Locker Application on remote servers
4. Step-by-step guide to starting and developing with Locker
5. Acknowledgements

Overview of Docker and Locker

Brief Overview of Docker - Locker is based on and uses Docker

- Basic idea: software virtualization of operating system and applications
  - E.g. run Ubuntu Linux and applications on a Mac or Windows machine
  - Much more space efficient than alternative virtualization solutions like VMware
- Key Docker concepts/terms:
  - **Image:** static, single file that encapsulates all the necessary functionality of an operating system (e.g. Ubuntu Linux) and apps, to be run on top of a **host operating system** (Win, Mac, Linux)
    - Built from a **Dockerfile:** simple text file that lists all the steps (e.g. install programs, packages, and libraries, copy files, set env vars, etc.) to be run to create an image.
  - **Container**: a running virtual machine, started from an image, that you can access and use like you would your own physical computer (e.g. SSH into a terminal window, run commands like 'ls', edit and run scripts, etc.)
  - **Docker Engine**: system/program that lets you create & run Docker containers. Installed on host.
  - **Docker Registry**: A place (server) where images are stored (e.g. Docker Hub)
  - **Pull**: Download an image from a registry to local computer to start containers from it.
  - **Bind Mount**: Enable a host path to be available inside a running Docker container

Locker Key points and advantages

- Developed internally at Bristol-Myers Squibb (BMS)
  - Supports the work of BMS Translational Bioinformatics Analysts
- Can run on your Windows, Mac, or Linux machine, or on a remote/cloud server
  - Locker Services supports automatic starting of EC2s in AWS with Locker
- Startup time fast (once server started and image is pulled)
- Supports running analysis fully offline by enabling caching - onto your local drive - of network drive content (R packages, system libraries, etc.), network mount files, Github repos, etc.
- Supports “sibling Docker containers”: ability to run other Docker images for specialized functionality, from inside your running Docker container/workspace
- Supports GPUs (assuming computer Locker is running on supports, e.g. nvidia-smi, etc.)
- Direct SSH access to started Locker containers supported

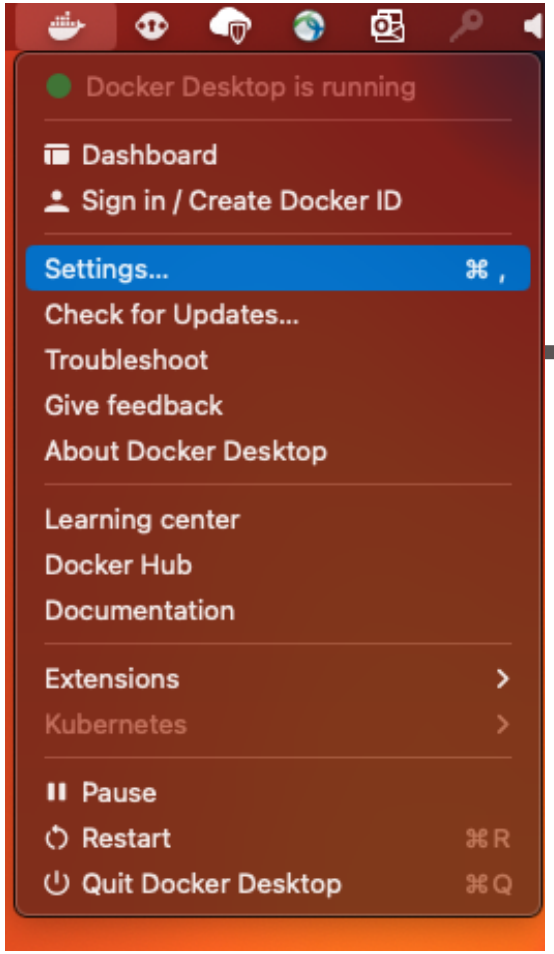
Locker Overview and Prerequisites, Initial Setup, Starting Locker using Startup Script

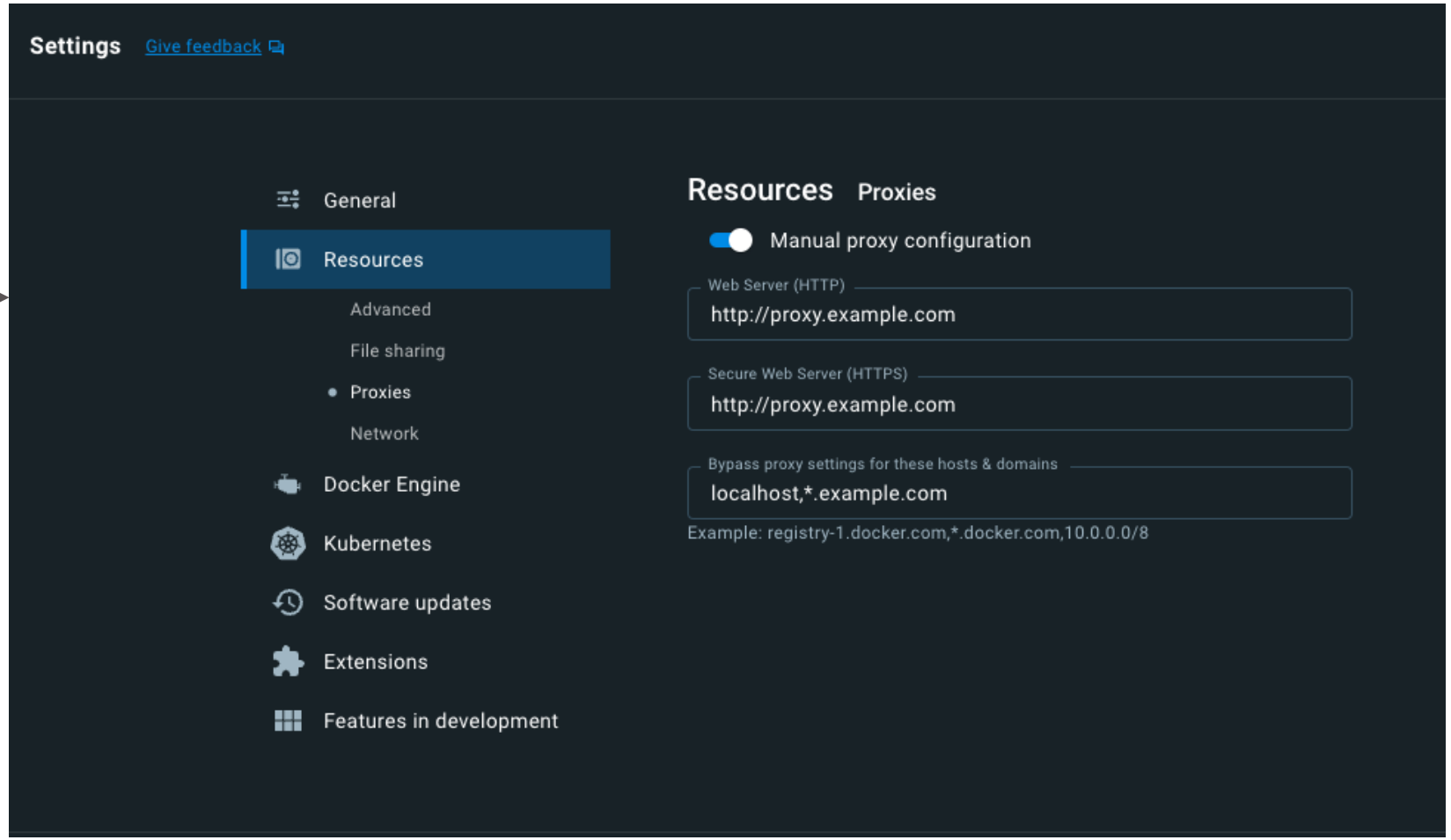
Locker Overview

- Prerequisite: Docker is already installed and running on the computer where Locker will be run
  - Preinstalled and available on Locker Services-started servers, otherwise see https://docs.docker.com/get-docker/ for instructions
- Locker is itself a Docker image
  - Runs as Docker container using "sibling Docker containers" method to start other containers.
  - Download and execute Bash script (Linux/Mac) or Win batch script to pull and start Locker
    - Or use Locker Web Services UI to (1) Start new server with Locker running (2) check/start Docker on existing server (3) Start Locker on existing server (e.g., cloud EC2)
      - Also server portal that lets you view and manage your started servers
  - Locker runs Flask web server that interacts with Docker using Docker SDK for Python
    - Access Locker in web browser via web UI, which communicates with Flask web server running on machine on which Docker images will be run as containers
  - Enables running Docker containers with:
    - One of: RStudio server, Jupyter, Jupyterlab
    - SSH (for direct login/terminal access)
    - Optionally also VSCode

Setting Docker Proxies (if needed)*

Settings – Resources > Proxies in Docker Desktop:

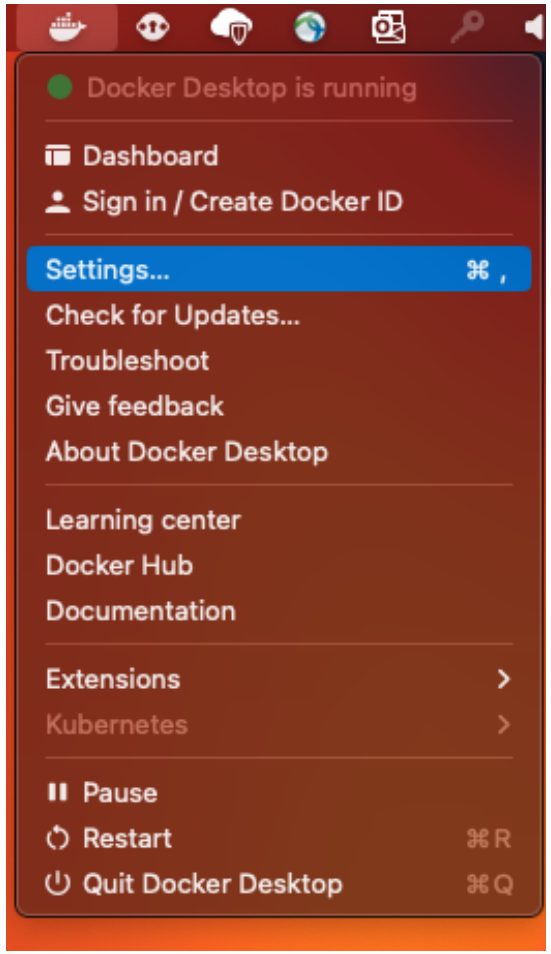


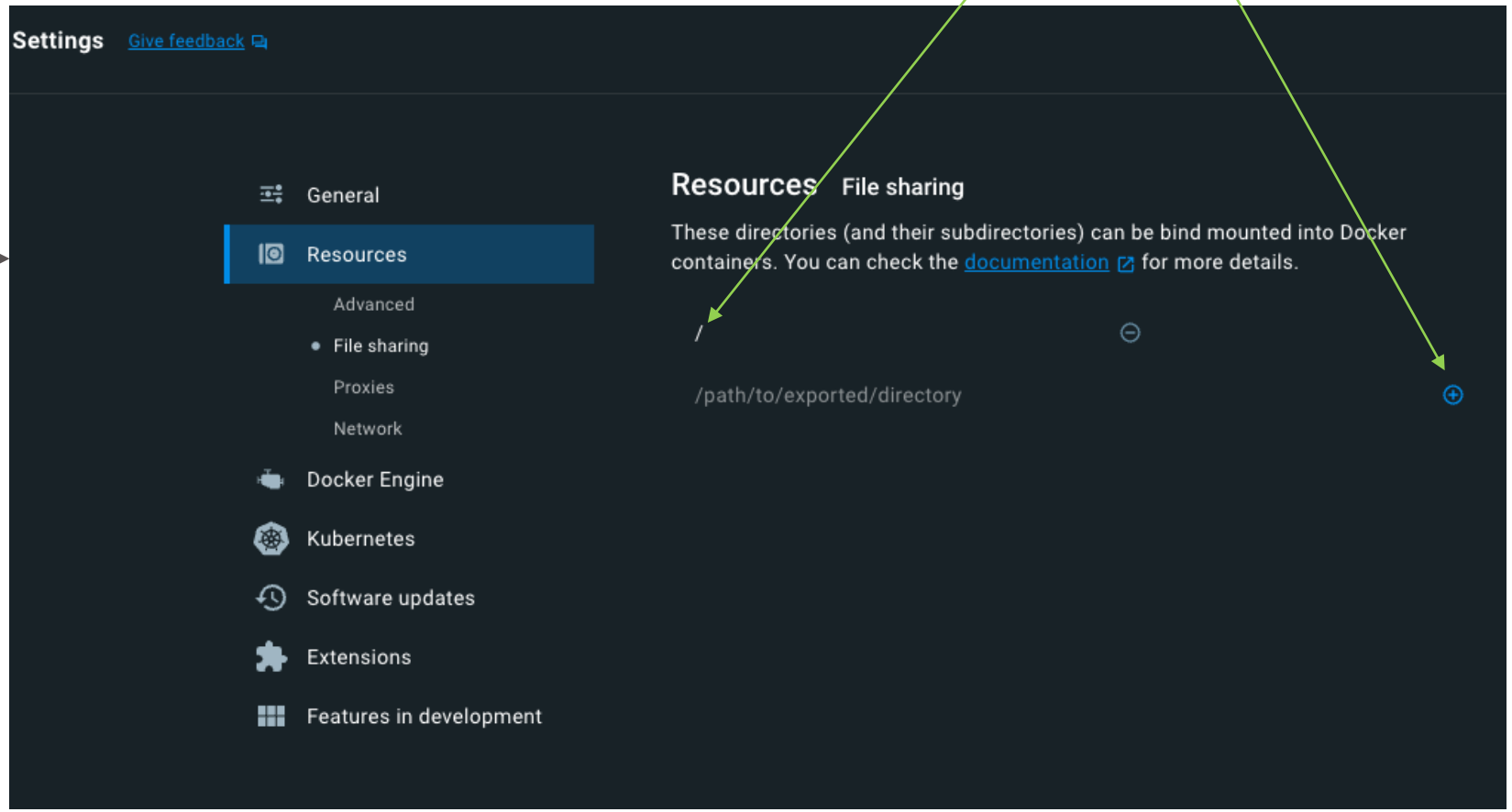


*On Windows and Mac

Allowing host root directory to be accessible inside started containers*

Settings – Resources > File Sharing in Docker Desktop:



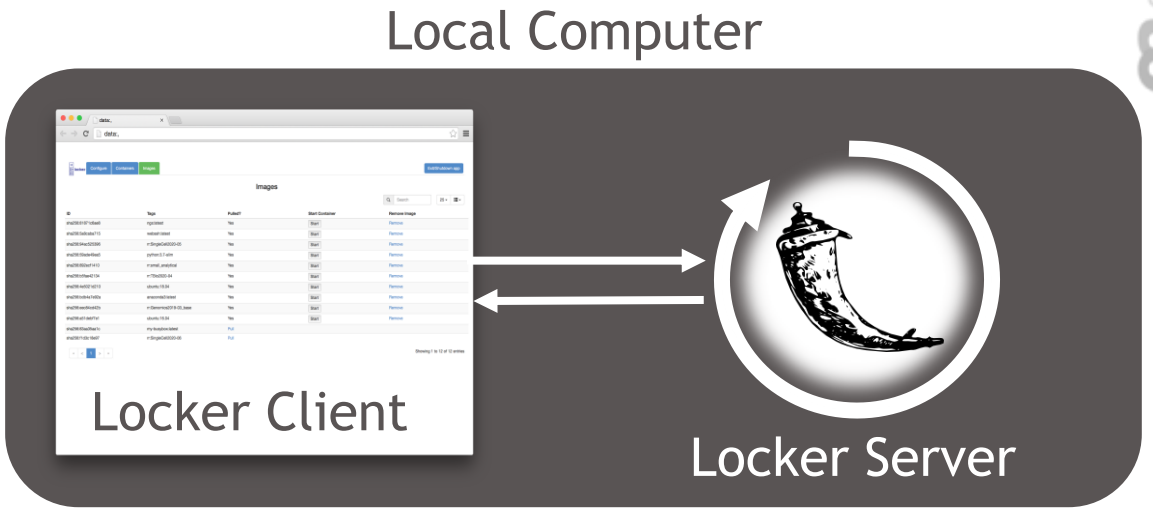


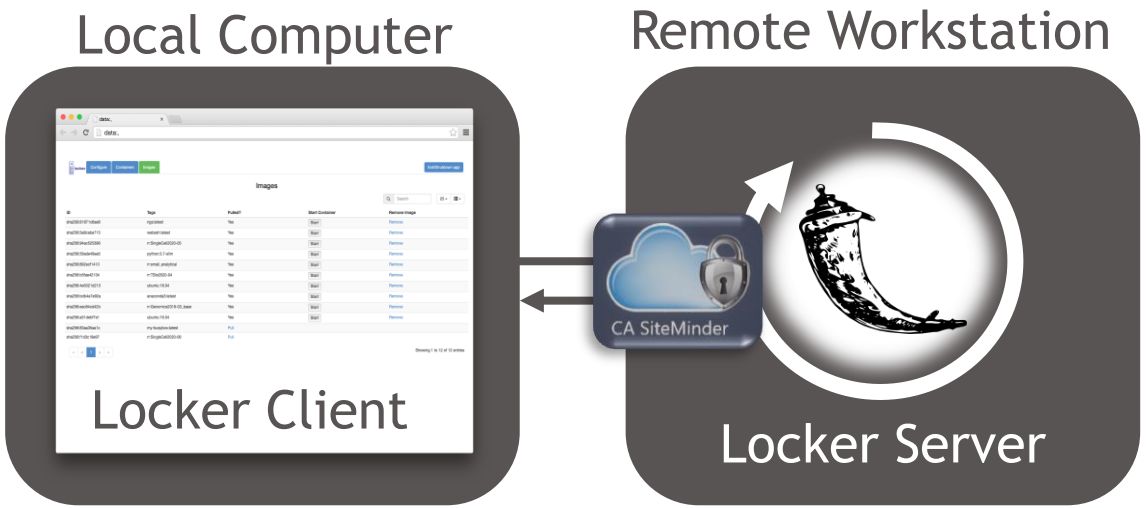
Here, host root ('/') is already set, but if not click the plus icon and then select the host root to add it to the list of directories that can be bind mounted into containers

*On Windows and Mac

Two Modes of Running Locker Application: Local or Remote

- Local
  - *Accessible only on the local computer*
    - The app (and started containers' services RStudio, Jupyter, etc.) will accept localhost connections only (no connections over the network)
    - No access control, since assumed only the computer owner will have direct access to their Win/Mac/Linux laptop or computer
- Remote
  - *Accessible over the network (and locally)*
    - The app (and containers' services RStudio, Jupyter, etc.) accept connections over the internet (and locally as well)
    - Thus, access control required for security
      - SSO (e.g. SiteMinder) authenticate
      - Only the user who started Locker can access





Running Locker

- 2 options
  - Execute start_locker script on command line (next couple slides) to directly start on a computer
  - Use Locker Services web UI (this is easier, see later in presentation) for remote computers

Getting and Running the Locker app locally (e.g. on Win/Mac)

- If you have setup Locker Services, you can download from there; otherwise generate by executing Makefile
  - Mac/Linux: `start_locker.sh`
  - Windows: `start_locker_win.bat`
- Running the app
  - Mac and Linux
    - First, inside terminal window make the downloaded file executable using `chmod` command:
      `chmod +x start_locker.sh`
    - Execute the app in a terminal window:
      `./start_locker.sh`
  - Windows is similar, download and execute `start_locker_win.bat`

Starting the Locker Application

- You will be asked some questions about how you want to run Locker:

```
smitha26@WVHRYRVXC3 DockerLocal % ./start_locker.sh
Would you like to try to pull an updated Locker image [y or n]? n
Run Locker as user [smitha26]:
User home directory [/Users/smitha26]:
Run the app local or remote (choose number)?
1) local
2) remote
#? 1
RUNASUSER: smitha26
USER_HOMEDIR: /Users/smitha26/
LOCAL_OR_REMOTE: l
DOCKER RUN COMMAND: docker run --rm -dt  --env DOCKER_HOST_USER_UID=501 --env DOCKER_HOST_USER_GID=20 --env RESET_USER_UGIDS=False --env DOCKER_HOST_LOCKER_PORT=5001
--env DOCKER_HOST_ROOT=/ --env DOCKER_HOST_USER=smitha26 --env TZ=America/New_York --env RUNASUSER=smitha26 --env LOCAL_OR_REMOTE=l --env USER_HOMEDIR=/Users/smitha26
/ -p 127.0.0.1:5001:5000 -v /:/host_root -v /var/run/docker.sock:/var/run/docker.sock --name locker_smitha26_1664206140 docker.rdcloud.bms.com:443/locker:devtest /loc
ker/exec_locker.sh
Successfully started locker as Docker container (id 177c3f14a4292afba02143c210574e764b90a3ce0a54b1ec69fff9dfe31dbfc7)
Access Locker at: http://localhost:5001
```

Do you want to pull an updated image for Locker if available (if first time, image will automatically be pulled)?

The user to run Locker as (only this user will be able to access, if running remote) and the user's local home directory. Values inside brackets are the default if you simply hit enter without providing values.

Run the app local (no access control) or remote (only specified user can access via SSO e.g. SiteMinder authentication)?

The actual 'docker run' command run to start Locker, the resulting container ID, and the URL where you can access Locker in a web browser

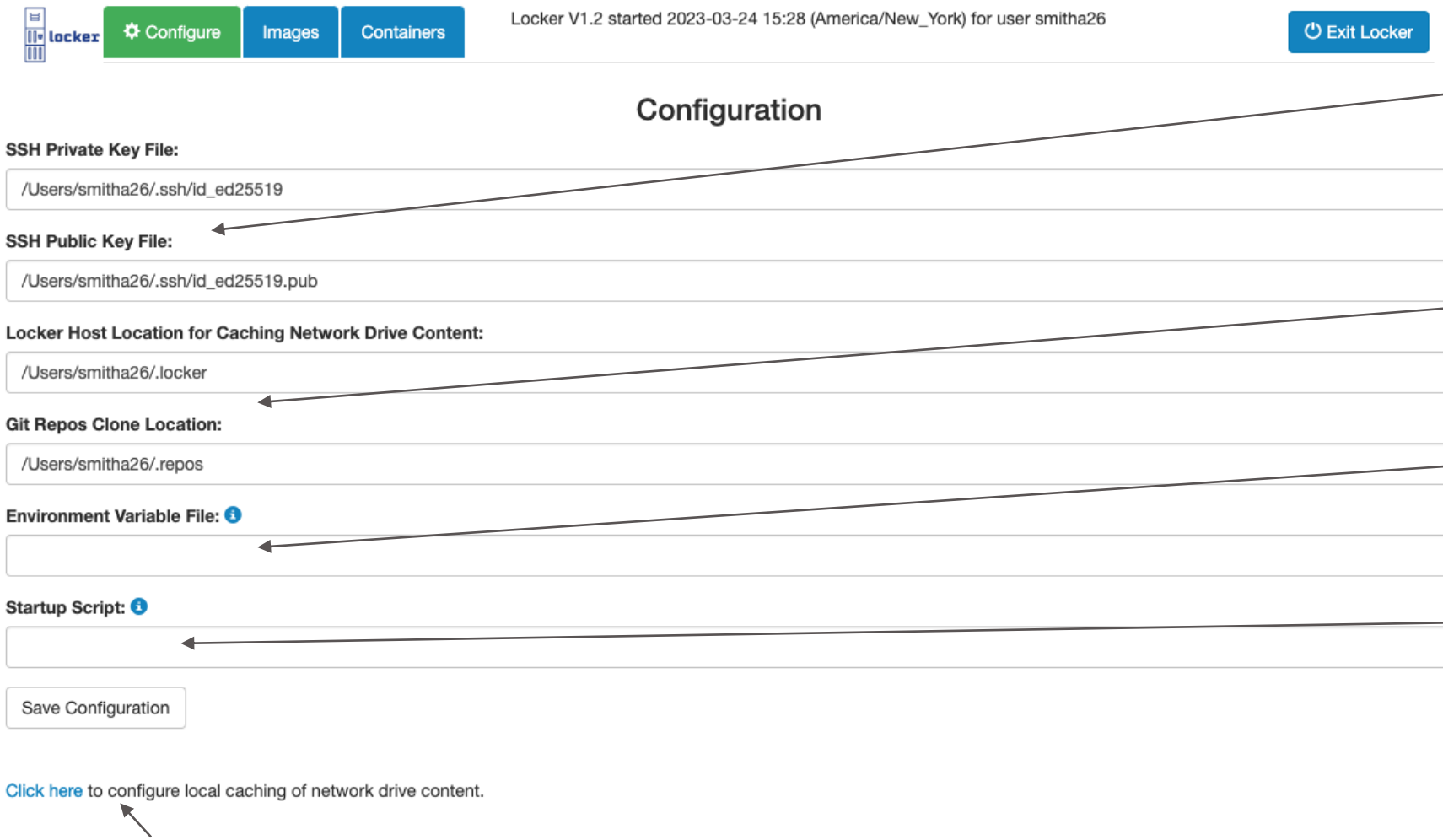
You can also just pass values directly on the command line when executing the start_locker.sh script:

```
smitha26@WVHRYRVXC3 DockerLocal % ./start_locker.sh -h
usage: start_locker.sh -h -u <RUNASUSER> -d <HOMEDIR> -s <LOCAL_OR_REMOTE> -p
-h : print this help message
-u <RUNASUSER> : <RUNASUSER> should be the LDAP id of the user to run as (and make accessible to if running remotely)
-d <HOMEDIR> : <HOMEDIR> will be searched for SSH keys, AWS creds, etc.
-s <LOCAL_OR_REMOTE> : 'l' for local (only localhost connections allowed), 'r' for remote (accessible over internet to <RUNASUSER>
-p : try to pull updated Docker image for Locker
```

User Interface Overview

Locker UI - Configuration

Basic Locker info shown on all Locker pages



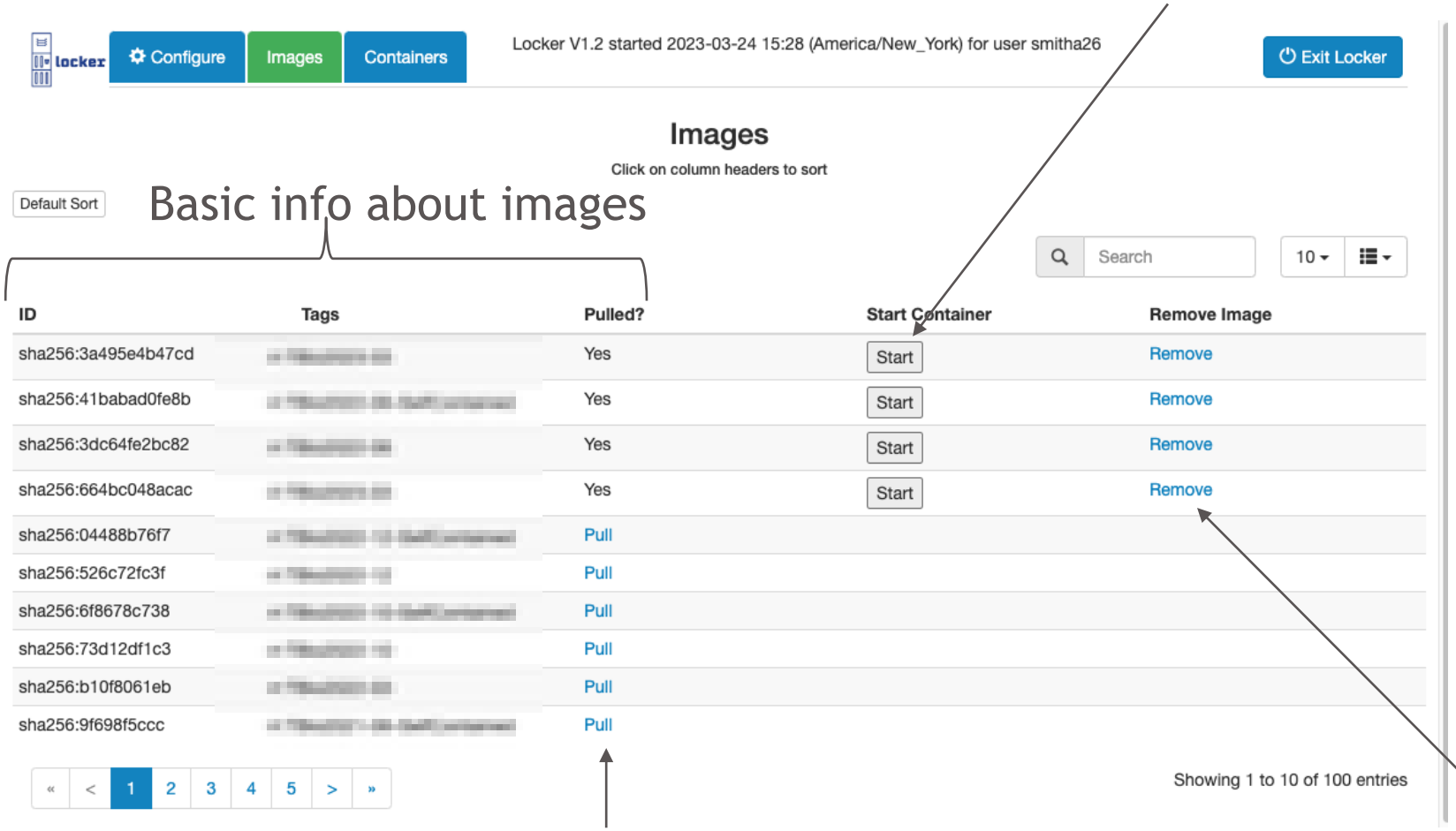
- Provide your SSH keys: enable network drives to be automatically mounted, ability to SSH into containers (password not supported)
- Specify Locker host locations for network drive and repos storage, to enable offline usage
- Specify environment variable file, to be set in all started containers and services (Rstudio, Jupyter, etc.)
- Specify startup script (e.g. bash script) to be executed at container startup for all containers
- Configuration info stored in config.json file inside .locker dir in your Locker host home dir; example/template environment variable file and startup script also are provided there

Ability to copy/cache parts of network drive content (e.g. network home directory, etc.) locally for offline usage of Locker

Autocomplete and navigation of file hierarchy supported in all fields

Locker UI - Images

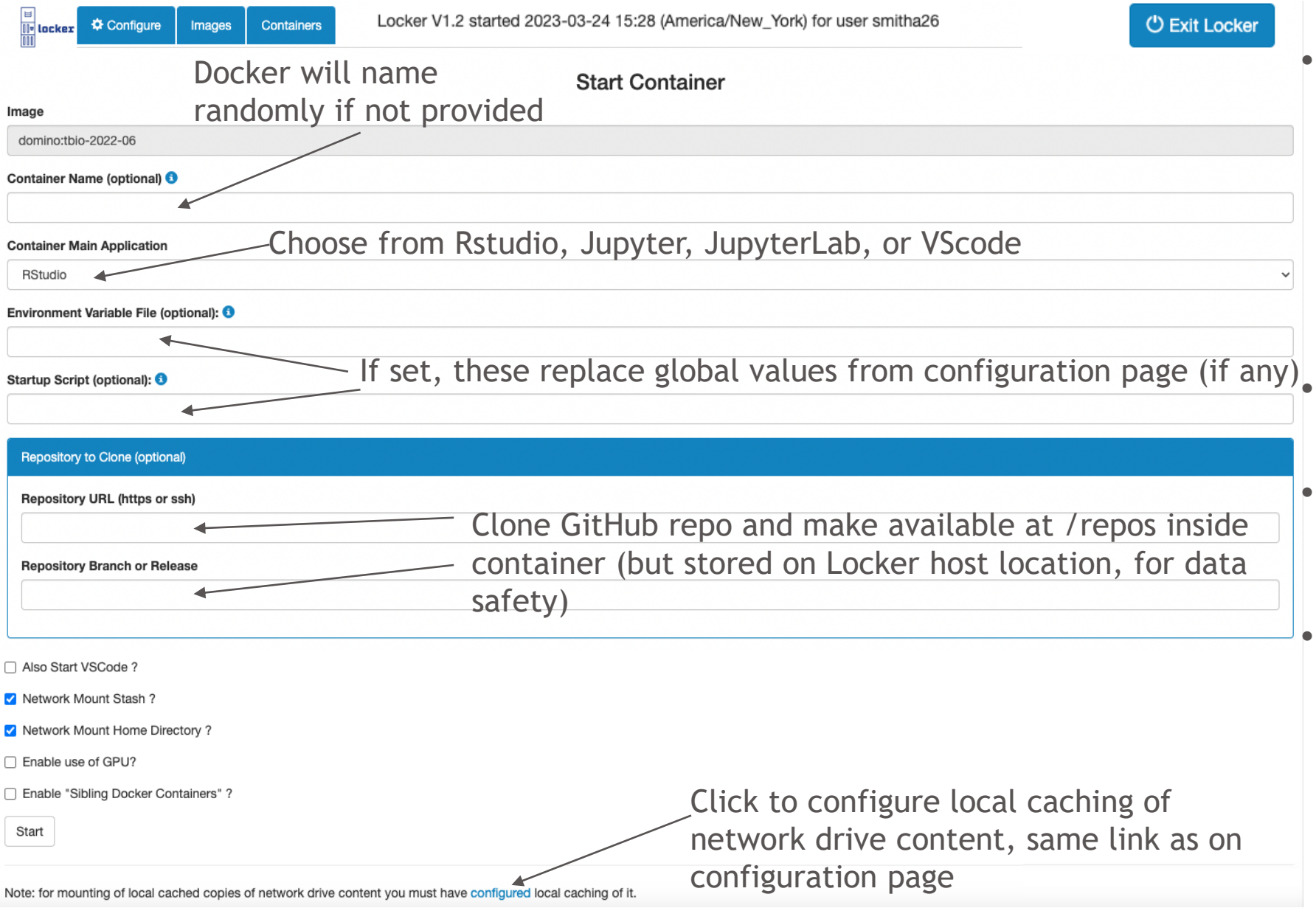
Click to start container from image once image has been pulled



Click to pull from Docker registry

Click to remove image (from your local Docker)

Locker UI - Start Container

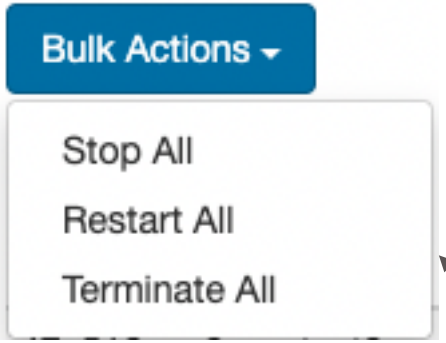


- Check “Also Start VSCode” to also start VScode (if not primary app)
- Check checkboxes for mounting network drive content (e.g. “Network Mount Stash” and/or “Network Mount Home Directory”)
  - If you cached locally, checkboxes for mounting of this local cache will appear
- Check “Enable use of GPU” to be able to use the GPU (on GPU host)
- Check “Enable ‘Sibling Docker Containers’” to be able to do Docker commands inside the container.
- **After clicking Start, the container starts, new row added to the “Containers” page and you are redirected to the “Containers” page**

Locker UI - Containers

Basic info about the running containers

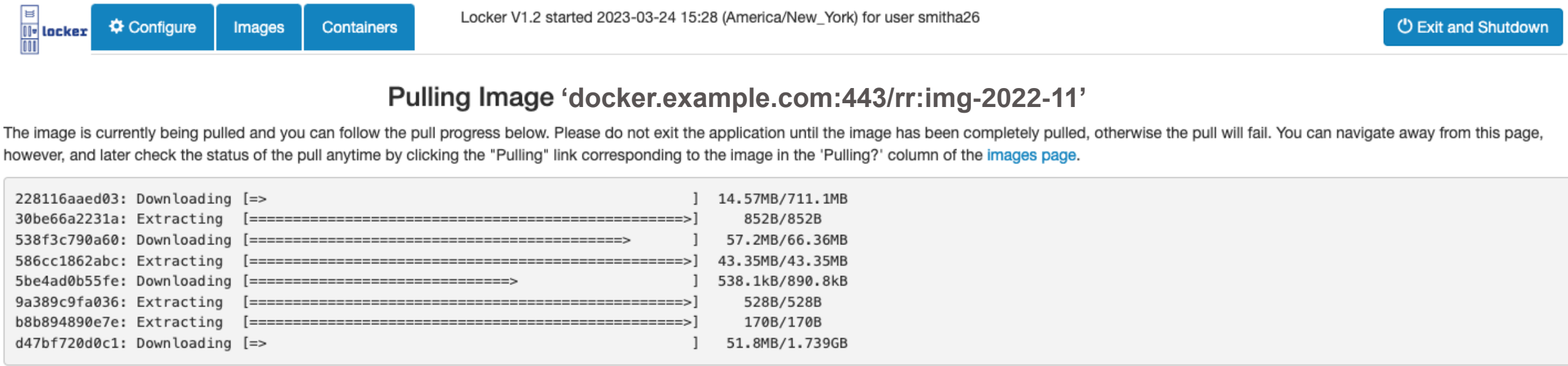
Links to terminate (you will be asked to confirm), stop or restart containers. Note: ideally have only one container at a time running, to not overtax system resources. Also stop all containers before rebooting host server (**they persist after reboot**).
Do quickly for all containers with Bulk Actions.



Links to access the running services in the container

Create a new image from the container, e.g., after having installed new programs or packages

Locker UI - Pulling Image (follow progress of image pull)



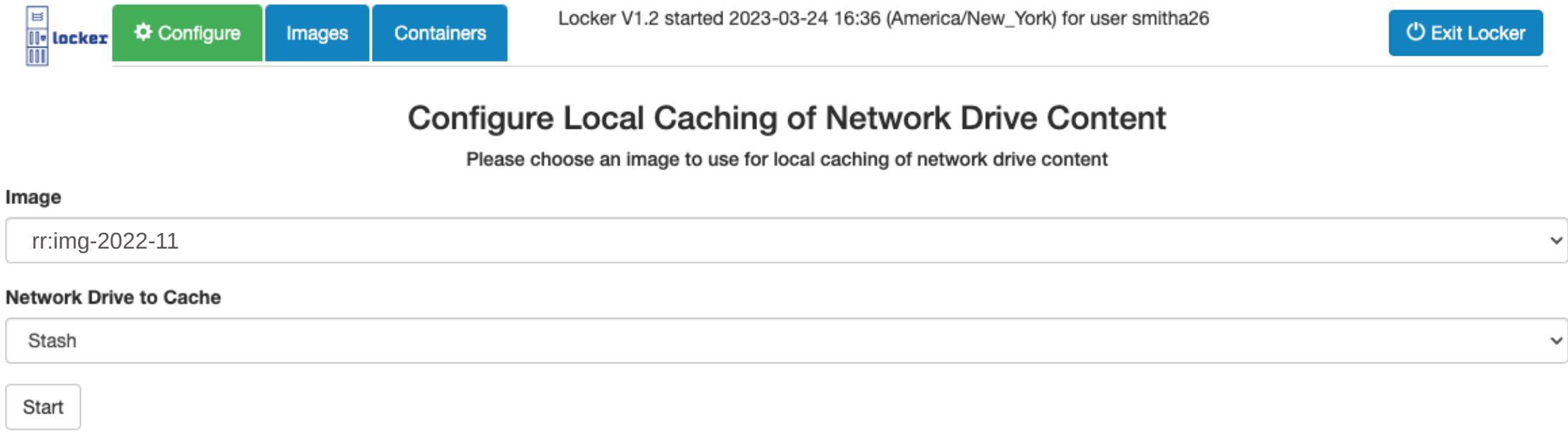
- See progress of Docker image layers as they are pulled
- Once image is pulled, you can start containers from it. Pulling is a one-time operation per host.

Locker Offline Usage Overview

Offline Usage Overview and Rationale

- Normally, to be able to run containers (started from Locker Docker images) you need to be connected to the internet and possibly also VPN or organization network
  - Image content (R packages, system libraries, etc.) stored on mounted network drive (e.g. EFS)
  - Network mounted directories, e.g. network home directory
  - Connection to github.com to clone, pull, and push repos
- Issues: network might be slow or get disconnected, or you simply might not have access
- Offline usage in Locker allows you to cache necessary online content so you can efficiently work fully offline (e.g., at the beach on your Mac!)
- To configure and enable, while you ARE connected to internet and VPN or organization network:
  - Cache necessary parts of network mounted directories to your host local drive
  - Clone GitHub repos to your host local drive during container startup
  - Pull and use special versions of the Locker Docker images that are self-contained (i.e., that do not require a network drive such as EFS to be mounted)
  - Use this cached content when starting a new container → use that container offline
- Note: the caching is to your configured local storage locations on your computer (laptop, etc.) that gets bind mounted into started containers → persists after containers stopped (no data loss)

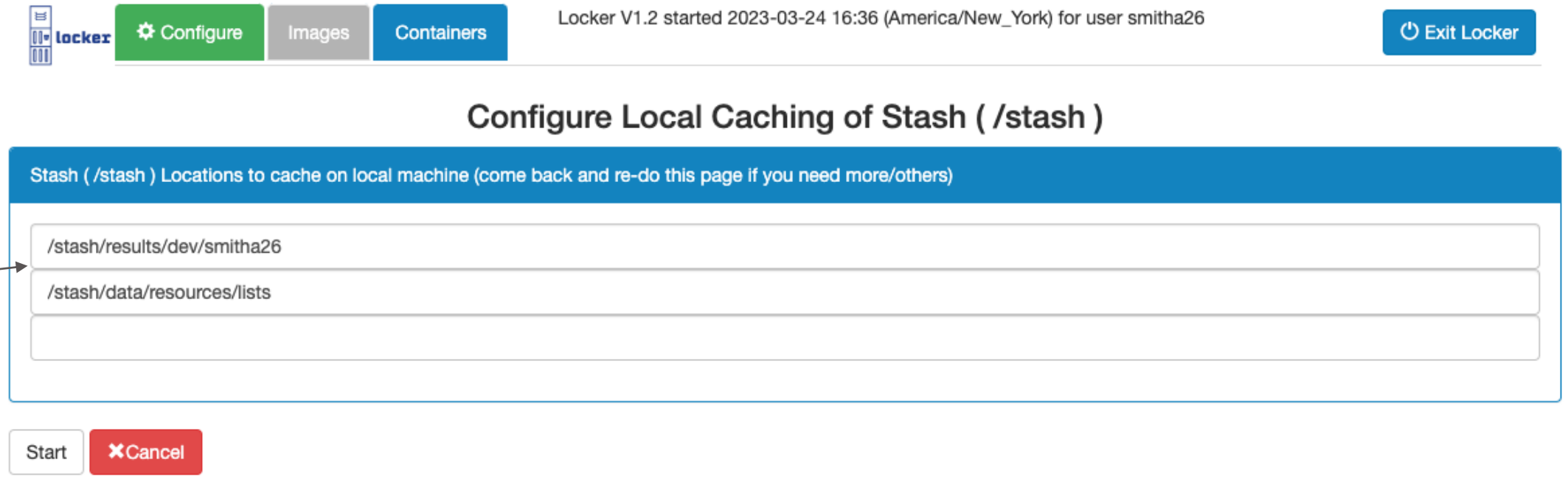
Locker UI - Configuring Local Caching of Network Drives



- Get to this page after clicking link on Configuration page: Click here to configure local caching of network drive content.
- Caching is done using a (any) Locker image, so you must have already pulled one of those
- Then choose the image to use (probably best to use most recent one although shouldn't matter) and click Start.

Locker UI - Configuring Local Caching of Network Drives (cont)

For a network drive mounted at /stash, choose /stash locations to cache to your host local drive. Autocomplete and navigation of file hierarchy supported.



Locker UI - View Status of Local Caching of Network Drives

Click to stop caching to your local drive (but whatever copied so far remains)

Progress on caching network drive content e.g. /stash to host local drive (specifically rsync status lines)

Locker Safe Storage Locations, Recommendations and Checklist for use

Safe Storage Locations inside Running Containers

- After container terminates, container files are lost unless they were in safe, persistent locations
- Some directories inside containers can be either network mounted or “bind mounted” from the host machine, so any content inside them is safe and persists after container exit:
  - **/repos** – for cloned BioGit repositories
    - Automatically bind mounted from host if user configured Git repos clone location (direct network mounting not supported, but the location could be a network drive location on your host)
  - **Network Drive Mounts, e.g. /netdrive**
    - User can choose at container startup to either bind mount from host (if Offline Storage Location configured and exists in there) or network mount from configured host of the drive
  - **/host_root**
    - Automatically bind mounted from host’s root directory (allows access to any host file or dir)
  - Note: for bind mounting from host, safety of data assumes you backup the host (e.g. your Mac or Windows laptop)
- Next slide delineates the locations outside the container for the above in-container paths

Safe Data Storage Locations Inside and Outside the Container

	Location Outside of Container	
Location Inside Container (LIC)	**If LIC network mounted**	**If LIC bind-mounted from host local storage**
/repos	Not Possible	<GIT_REPOS_CLONE_LOCATION>/repos
Network drive1 mount	<network_drive1_host>:/<network_drive1_mount_point>	<LOCKER_HOST_CACHE_LOCATION>/<network_drive1_mount_point>
Network drive2 mount	<network_drive2_host>:/<network_drive1_mount_point>	<LOCKER_HOST_CACHE_LOCATION>/<network_drive2_mount_point>
/host_root	Not Possible	'/' (Mac/Linux) or 'C:\' (Windows)

Notes and Recommendations

- Set a repos clone location and use it for /repos
- Work as much as you can in your network-mounted drive locations
- You can also store things safely under /host_root
- Don’t ever rely on storing things in /tmp or other directories that are only available inside the container
- Note: to directly access the Locker host cache location from inside the container, go to: /host_root/<LOCKER_HOST_CACHE_LOCATION>
  - Similarly to access Locker repos clone location inside the container, go to: /host_root/<GIT_REPOS_CLONE_LOCATION>

Basic Checklist for using Locker

- **One Time per host machine**
  - Install Docker
  - Configure Docker proxies and allow Docker to bind mount host root directory
  - Optionally download/get Locker start script (start_locker.sh for Linux/Mac, start_locker_win.bat for Win)
  - Start Locker by executing the start script or using the Locker Services UI
  - Configure Locker: SSH public & private keys, Locker host cache location and Git repos clone location, environment variable and startup script files
  - Pull Images
  - Optional: configure offline caching of network drive content
- **Recurring**
  - Start containers from pulled images
  - Use the running containers to do your work (RStudio, Jupyter, Jupyterlab, Vscode, etc.)
  - When done, stop the containers
  - Optionally exit Locker (can easily restart it later)
- **Periodically, during inactive periods (e.g., overnight, on weekends, during vacations):**
  - Bulk stop all containers and shut down Locker host (e.g., cloud server or Locker services started server)
  - After host reboot, restart Locker (as above). Any stopped (but not terminated) containers will still appear on the Locker Containers page! In this way, you can maintain a persistent list of containers, one per project, on a given host machine and start them as needed.

User Interface Overview

Locker Services to Start Locker Application Remotely

- Five services (flexibly use first 3 individually as needed, fourth one combines all 3)
  - **New Server**: Start new EC2/server in the Amazon AWS Cloud
  - **Check/Start Docker**: Check if Docker is Installed, Running, and Accessible on a server; Optionally Install and Start it if not
  - **Start Locker**: Start Locker on a Linux Server (assumes Docker already running and available)
  - **Locker on New Server** - Combines all three previous services into one simple to use
  - **Locker Server Portal** -View and manage all your started servers (stop/terminate/restart, etc.)
- E.g. for an already running cloud server you could first **Check/Start Docker** then **Start Locker**. Or just **Start Locker** if you know Docker is already installed and ready.
- E.g. If you need to run on special hardware just run **Locker on New Server**.
- If you just want to run Locker remotely on an already running server, this is the easiest way to get it running.
  - Possible to directly use `start_locker.sh`, but then you'd have to handle all the low level details yourself

Locker Start Script vs Locker Services --- When To Use Each

Use Case	Locker Start Method
Run and use Locker locally on a Windows or Mac laptop	Use the start script
Run and use Locker remotely on an already running server	Use either, but Locker Services easier
Run and use Locker remotely on a new (temporary) EC2 Server	Use the Locker Services

Locker Services - Locker Server Portal

Click '+' to see all tabs (then '-' if you want to hide again)

Servers started by Locker services for smitha26

Click on column headers to sort

smitha26 | 10

Hostname	IP address	Instance Type	Last Start Time (U...	Description	State	Creator	Open Connection	Change State
[redacted]	[redacted]	t2.micro (1 cores - 1 ...	2023-03-17 12:15:23		stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	t2.micro (1 cores - 1 ...	2021-10-14 12:03:54		stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	t2.micro (1 cores - 1 ...	2021-10-28 19:38:41		stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	t2.large (2 cores - 8 G...	2022-08-02 21:17:44		stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	m4.16xlarge (64 core...	2022-08-10 18:33:44		stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	c4.8xlarge (36 cores -...	2023-03-24 21:34:10	[redacted]	running	smitha26	Locker | SSH	Terminate | Stop
[redacted]	[redacted]	g5.12xlarge (48 cores...	2023-02-24 00:16:23	[redacted]	stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	r5a.24xlarge (96 core...	2023-03-07 20:47:45	[redacted]	stopped	smitha26		Terminate | Restart
[redacted]	[redacted]	g5.48xlarge (192 core...	2023-03-09 16:01:47	[redacted]	stopped	smitha26		Terminate | Restart

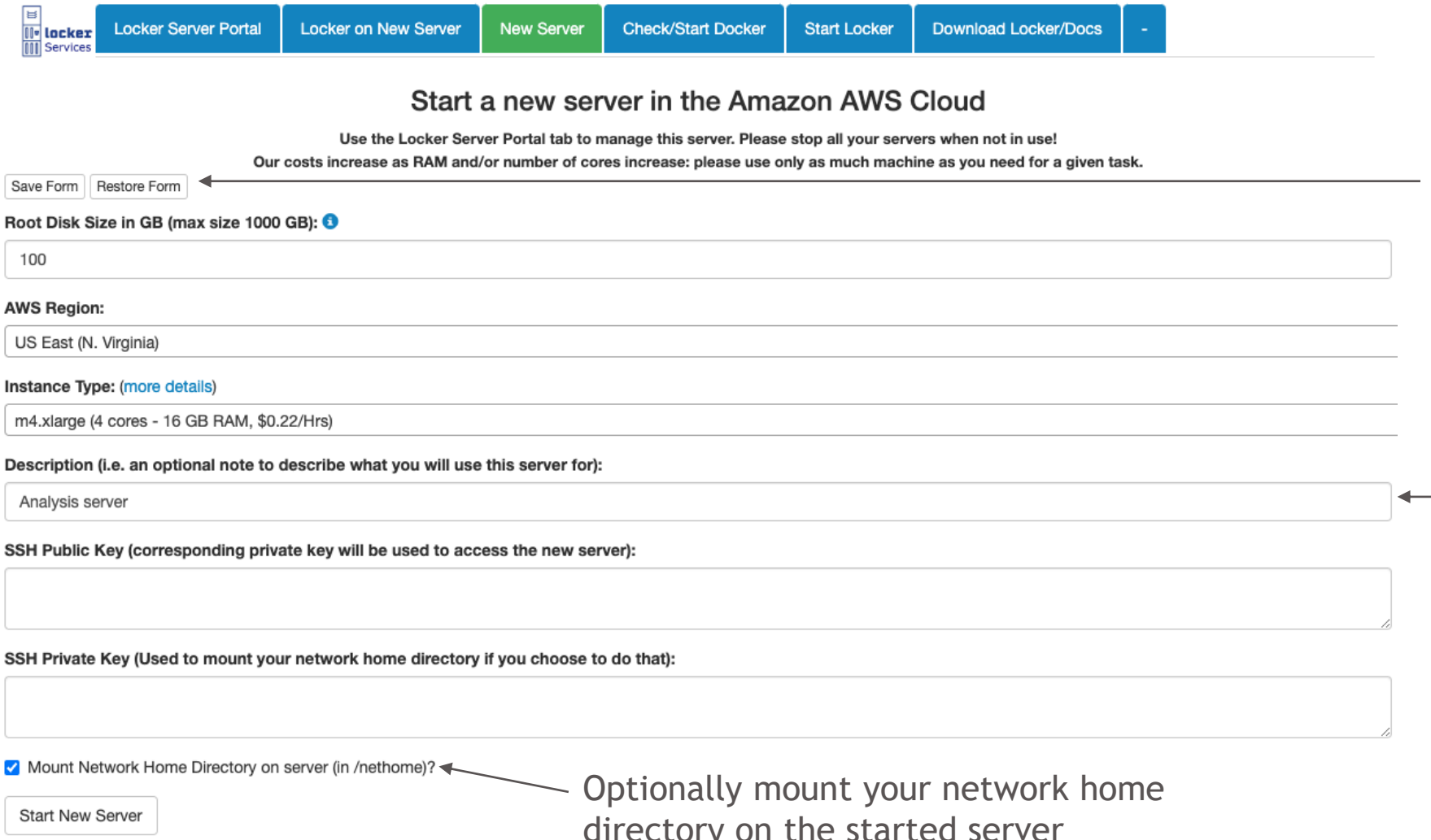
« < 1 > »

Showing 1 to 9 of 9 entries

- View details of all your started servers
- Links to **Terminate**/**Stop**/**Restart**
  - **Stop** will automatically exit all containers first
  - **Restart** will automatically restart the container that runs Locker
- Links to access running Locker on server and to SSH to it

Locker Services - New Server

Note: you can click here to download start_locker.sh and docs

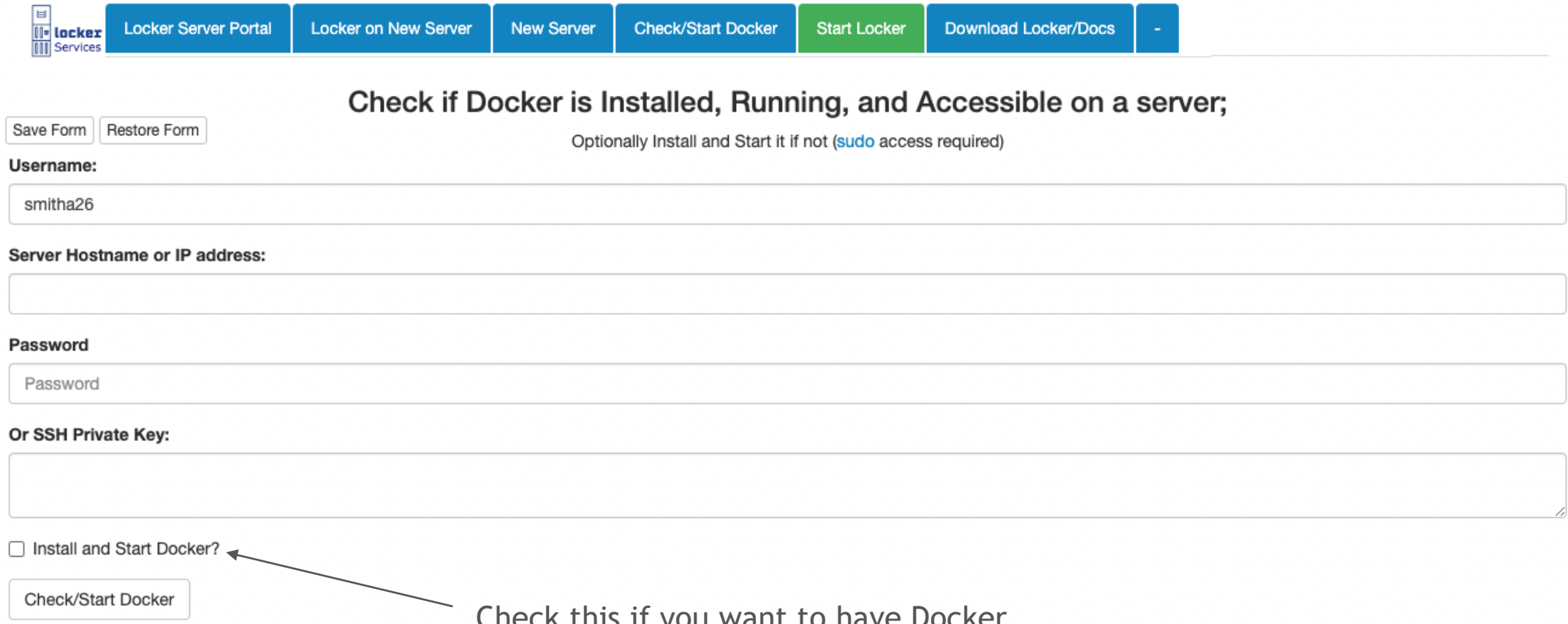


All the Locker services allow you to save your filled out form values to browser local storage, and then restore that later to avoid having to enter the same detailed info repetitively when you use the services often.

Optionally provide description of the server (shown in Locker server portal)

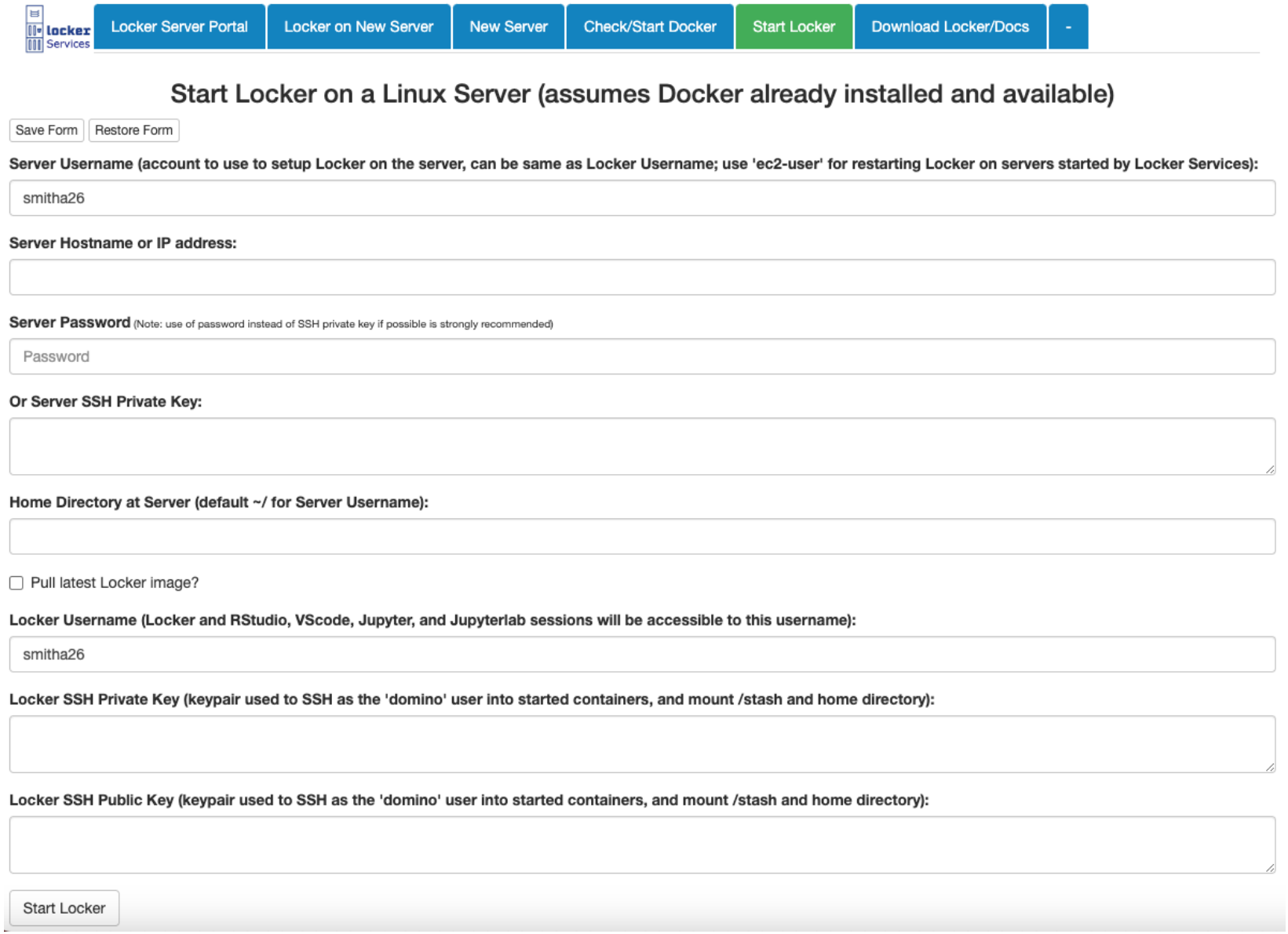
Optionally mount your network home directory on the started server

Locker Services - Check/Start Docker



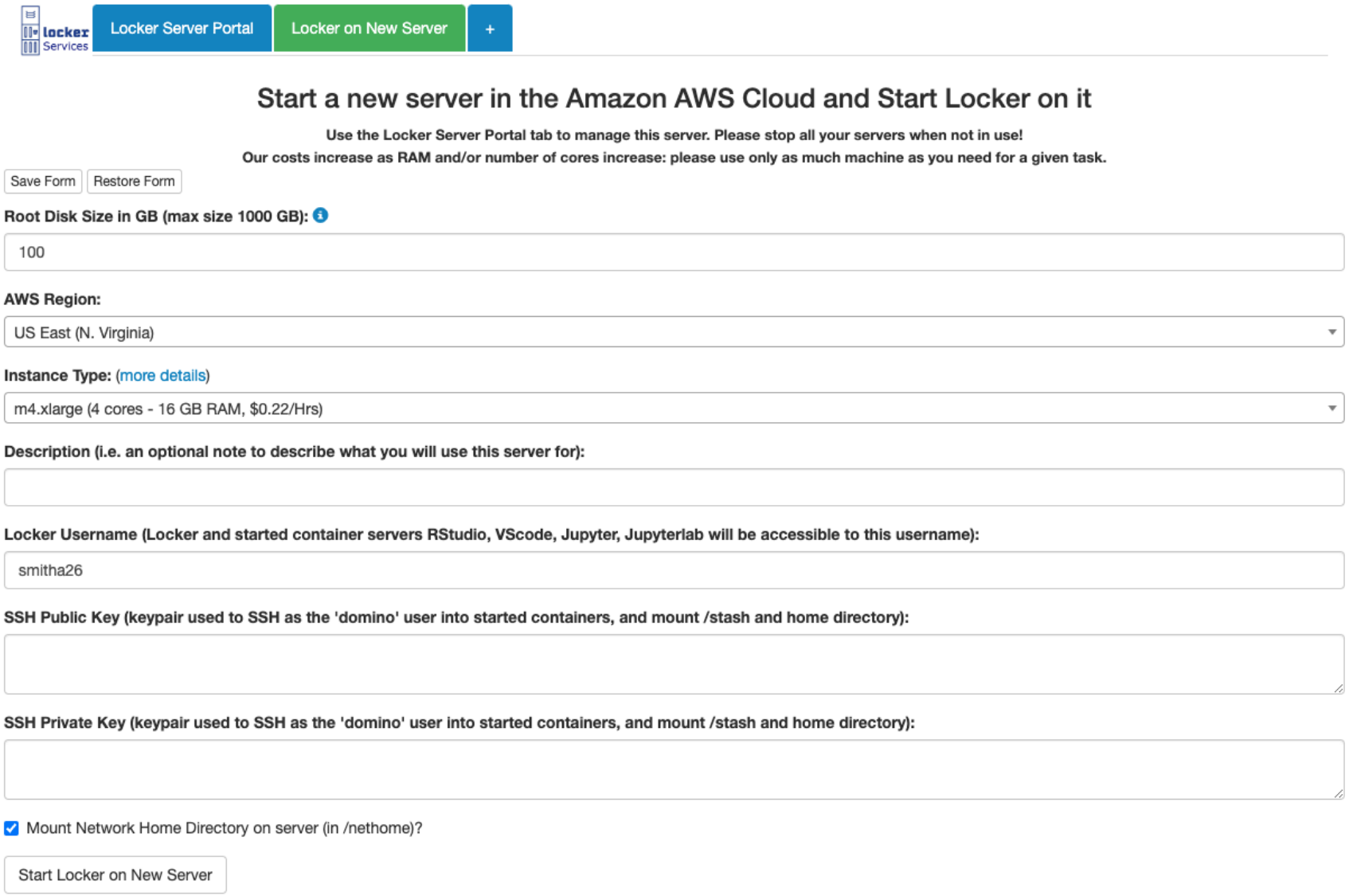
Check this if you want to have Docker attempt to be installed (if it isn't already); if not checked, Docker status on the server will be checked and reported back to you.

Locker Services - Start Locker



- Note: Server Username and Locker Username will often be the same, but can be different if one person (e.g. admin) is starting Locker on behalf of another end user

Locker Services - Locker on New Server - combines previous 3 services



Step-by-step Guide to Starting Locker and Developing using Locker and Locker Services

Starting Locker on a new EC2 instance (Page 1)

1. Choose this option when you need a new EC2 instance with Locker
2. Go to ‘Locker Services’
3. Click on ‘Locker on New Server’ tab
    1. If you have a saved form, click on ‘Restore form’ and go to step 7; otherwise continue
    2. Based on your project/analysis requirements,
        1. Enter disk size, Instance Type (or leave the default selections)
    3. Enter ‘Description’ if desired.
    4. Enter your LDAP username (if not auto filled, or incorrect)
    5. Copy SSH public and private key file contents and paste the text into the corresponding fields
    6. Click on ‘Save form’ to save the current information entered for future use
    7. Click on ‘Start Locker on New Server’
        1. You will receive an email with a link to Locker.
        2. You will also see the Locker status and the link on the browser.
    8. Click on ‘Locker Server Portal’
        1. Your new EC2 instance will be listed
        2. You can access Locker by clicking on ‘Locker’ under ‘Open Connection’ column (You can also right click and click on Open Link in New Tab)

Managing your EC2 Instances

1. Go to ‘Locker Services’
2. ‘Locker Server Portal’ will have all your EC2 instances that were started using Locker Services
3. Controlling your EC2
    1. ‘State’ – Possible States are ‘starting’, ‘running’, ‘stopping’, ‘stopped’
    2. ‘Open Connection’
        1. ‘Locker’ – To open running Locker session
        2. ‘SSH’- To open an SSH connection to the EC2 instance
            1. On Mac, opens a Terminal
            2. On Windows, displays the ssh command you can use to connect to the EC2 instance
    3. ‘Change State’
        1. ‘Terminate’ – Permanently shuts down the EC2 instance
            1. Any data stored directly on the EC2 instance will be lost
        2. ‘Stop’ – Pauses the EC2 instance
            1. Any data stored directly on the EC2 instance will be available after restart
        3. ‘Restart’ – Restarts a stopped EC2 instance
4. Click on the menu button on the upper right if you wish to select columns to be displayed

Starting Locker on existing server

1. Choose this option if Locker is not running and not paused on your server
2. Go to ‘Locker Services’
3. Click on ‘+’ tab
4. Click on ‘Start Locker’ tab
    1. If you have a saved form, click on ‘Restore form’ and go to step 6; otherwise continue
    2. Enter your username, server IP address, password (or the SSH private key)
    3. Select checkbox for ‘Pull Latest Locker image’
    4. Copy SSH public and private key content and paste the text into the corresponding fields
    5. Click on ‘Save form’ to save the current information entered for future use
    6. Click on ‘Start Locker’
        1. You will receive an email with a link to Locker.
        2. You will also see the Locker status and the link on the browser.
    7. Click on ‘Locker Server Portal’ to find and manage your server if it was previously started by Locker Services (if not, it won’t be visible in Locker Server Portal)

Using Locker for Development (Page 1)

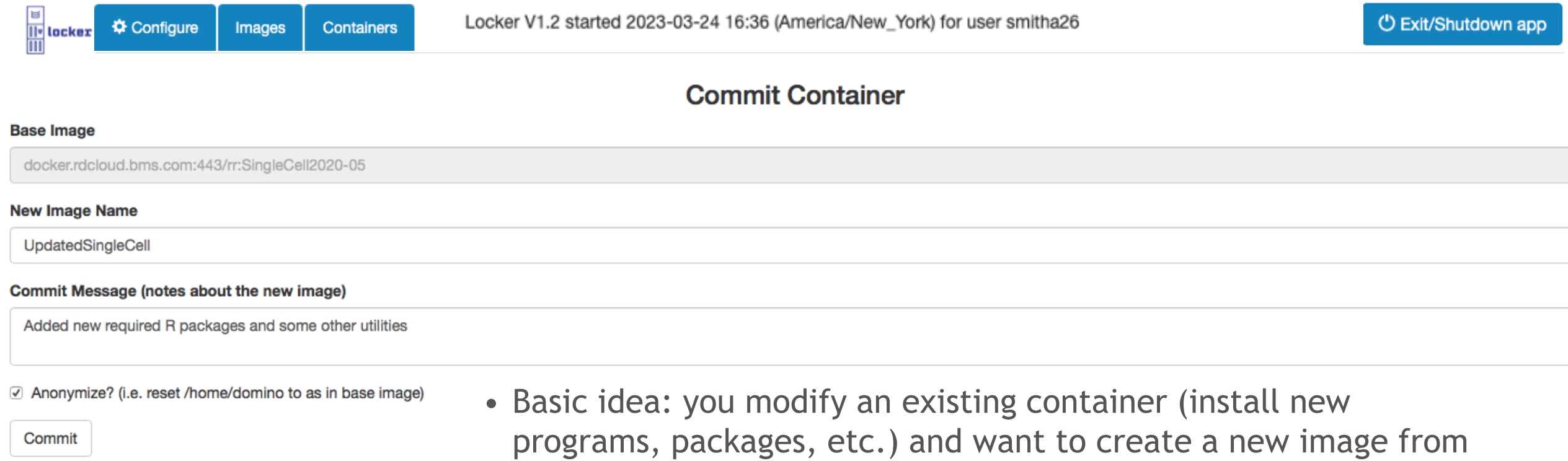
1. To access Locker:
    1. Access the ‘Locker Server Portal’ of Locker Services and click on ‘Locker’ under the ‘Open Connection’ Column for your EC2 instance **or**
    2. Click the ‘Access it here’ link in the email you got after starting Locker **or**
    3. If you didn’t start Locker using Locker Services, go to the web link where your locker is running (e.g. as printed by start_locker.sh)
2. You will now see the Locker main page
3. Configure location for your Git repos (once per machine running locker)
    1. On the ‘Configure’ tab, enter the location in ‘Locker Host Location for Caching Stash and Git Repos:’ (<lockerstorage>)
        1. Choose a path available on the machine running locker, e.g. /home/<userID>/lockerstorage
    2. By cloning repos here, even if a container stops, you will still see your changes in the selected directory
    3. Inside a container, /repos is a symlink pointing to <lockerstorage>/repos. You can access your cloned repos at /repos.
4. Click on Images tab
5. Click on ‘Pull’ for the image you want to use (if you haven’t yet pulled any images)
6. Go to GitHub and copy the SSH link for the repository you want to clone and use
7. After ‘Pull’ is complete, we can start the container
    1. On the Images tab, click on ‘Start’ under the Start Container column for the pulled image
    2. Enter a project name for the ‘Container Name’ field
    3. Choose the interactive application you want to use, e.g. ‘Rstudio’, as your Container main application
    4. Under ‘Repository to clone’, Paste the SSH link for your repo into the ‘Repository URL’ box
    5. Click checkboxes to have network drive directories mounted
    6. Click on ‘Start’

Using Locker for Development (Page 2)

1. For working on multiple projects, you have two options
    1. Start a new container for each project (recommended, usually most convenient)
    2. Switch repos within a single container (you may have to clone additional repos at the command line)
2. We recommend you have only one container running at a time. You can stop any existing containers before starting a new container.
    1. A stopped container can be restarted to resume work on a project
    2. In this way, you can maintain a list of containers, one per project. This list persists between reboots of Locker host machine.
3. After the container starts you will land on the ‘Containers’ tab
4. Right-click link under Primary Application (e.g. ‘Rstudio’), click on ‘Open Link in New Tab’ to open your application workspace
    1. We recommend opening in a new tab to keep your Containers page for visiting later
    2. You can also bookmark your containers page or access it from Locker Services
5. Access your project repo in your Locker Rstudio workspace
    1. You can navigate to your repo folder under /repos in the Files explorer tab on the right.
6. If you chose to mount network drive directories, you can access them at their mount points

Extra/Advanced Topics

Commit Image Page



- Basic idea: you modify an existing container (install new programs, packages, etc.) and want to create a new image from it (to start new containers in the future).
- "Anonymize" will remove your sensitive private info
  - Can then safely push such anonymized images to your Docker registry to share with others

Sibling Docker Containers (advanced technique)

- Rationale: Needed functionality is not installed/available in container you are currently running, can't be added easily/quickly. BUT such functionality exists already in another Docker image (e.g. from Docker Hub or public GitHub) → use this inside your currently running container.
- Sibling Docker container support allows you to run docker commands inside your running container ('docker images', 'docker pull', 'docker ps –a', 'docker info', 'docker run', etc.)
  - The commands actually go against the Docker in the host machine of your running container (thus any containers you start this way are "siblings" of your currently running container).

Example (inside running container):

```
domino@10f3cb0e1ab6:~$ docker pull broadinstitute/gatk
Using default tag: latest
latest: Pulling from broadinstitute/gatk
423ae2b273f4: Already exists
de83a2304fa1: Already exists
f9a83bce3af0: Already exists
b6b53be908de: Already exists
a69d35efa09a: Pull complete
91dc54131014: Pull complete
Digest: sha256:33574f446ac991f77bac125fbf6a2340e6db972a3f334e6c61bff94740165938
Status: Downloaded newer image for broadinstitute/gatk:latest
docker.io/broadinstitute/gatk:latest
domino@10f3cb0e1ab6:~$ docker run -i broadinstitute/gatk:latest gatk HaplotypeCaller --help
USAGE: HaplotypeCaller [arguments]

Call germline SNPs and indels via local re-assembly of haplotypes
Version:4.1.9.0-SNAPSHOT


Required Arguments:

--input,-I <GATKPath>          BAM/SAM/CRAM file containing reads  This argument must be specified at least once.
                               Required.

--output,-O <String>           File to which variants should be written  Required.

--reference,-R <GATKPath>      Reference sequence file  Required.
```

Download needed image (here GATK)

Execute command using the image (here HaplotypeCaller)