AI-03076 Docker Cleanup Commands

Owner	Ⓡ Rigel Arcayan
Tags	
Created time	@October 14, 2025 10:29 AM

docker compose down -v, will this command give me back to fresh setup like new?

The `docker compose down -v` command will remove **most** things, but not everything. Here's what it does:

What it DOES remove:

- **Containers** - all containers defined in your compose file
- **Networks** - networks created by the compose file
- **Named volumes** (`v` flag) - volumes defined in your compose file

What it DOESN'T remove:

- **Images** - the Docker images remain on your system
- **Bind mounts** - any local directories you've mapped into containers stay unchanged
- **External volumes** - volumes marked as `external: true` in your compose file
- **Build cache** - cached layers from image builds

To get a truly fresh setup:

```
# Remove containers, networks, and volumes
docker compose down -v
```

```
# Remove images as well
docker compose down -v --rmi all

# Or, for a complete clean slate:
docker compose down -v --rmi all
docker system prune -a --volumes
```

Note: If your application stores data in bind mounts (folders on your host machine), you'll need to manually delete those directories to get a completely fresh start.

So `docker compose down -v` gets you *mostly* back to fresh, but keep the images for faster restarts. If you want truly "like new," you'll need to also remove images and potentially clear bind mount directories.

Does the container needs to be in down state already prior to docker system prune -a --volumes

No, you **don't need to** run `docker compose down` first, but it's **recommended** for clarity and safety.

What `docker system prune -a --volumes` does on its own:

It will remove:

- All stopped containers
- All networks not used by at least one container
- All volumes not used by at least one container
- All images without at least one container associated with them
- All build cache

Key point: It only removes **stopped** containers and **unused** resources.

Why run `docker compose down` first anyway?

1. **Stops running containers** - `docker system prune` won't touch running containers, so if your compose stack is running, those containers and their associated resources will be skipped

2. **Cleaner approach** - It's more explicit about what you're doing (first stop your app, then clean up)
3. **Avoids surprises** - You know exactly which containers you're stopping

So the safe sequence is:

```
# Stop and remove your compose stack first
docker compose down -v

# Then clean up everything else
docker system prune -a --volumes
```

But technically, if you just run `docker system prune -a --volumes` on a system with running containers, those running containers (and their volumes/networks/images) will be left alone.