Laravel Sail: A Simple Docker Environment for Laravel

Laravel Sail is the official Docker-based development environment for Laravel. It provides a simple command-line interface to start and manage a full PHP, MySQL, Redis, MailHog, and more stack without needing to install these services directly on your system. Sail makes it easy for teams to work with the same setup across Windows, macOS, and Linux.

When I started collaborating with a distributed team on a Laravel project, one of the biggest challenges we faced was keeping our local environments consistent. Some developers were on Windows, others on macOS, and a few on Linux. Everyone had different PHP versions, MySQL setups, and node configurations. Debugging issues caused by environment mismatches became a nightmare, and we wasted valuable time just trying to align setups.

That's when we introduced **Laravel Sail** into our workflow. Instead of installing PHP, Composer, and databases manually, we spun up the entire stack with Docker using Sail. Suddenly, every team member had the same environment, regardless of their operating system. It felt like a huge weight was lifted off our shoulders. I didn't have to explain how to install Redis or MailHog anymore — Sail handled it all out of the box.

What I loved most was how easy it was to run Artisan, Composer, and even NPM commands through Sail without touching my system setup. Over time, I learned to customize the `docker-compose.yml` file, adding services like Meilisearch and Selenium for browser testing. It turned into a portable development environment that made collaboration seamless. Looking back, adopting Sail was one of the smoothest decisions we ever made for productivity and sanity in Laravel development.

Installing Laravel Sail

```
curl -s https://laravel.build/example-app | bash
cd example-app
./vendor/bin/sail upCode language: JavaScript (javascript)
```

This command creates a new Laravel project, installs Sail, and boots the Docker containers. The default stack includes PHP, MySQL, Redis, MailHog, and more.

Running Commands with Sail

```
./vendor/bin/sail artisan migrate
./vendor/bin/sail composer require laravel/sanctum
./vendor/bin/sail npm install && ./vendor/bin/sail npm run devCode
language: JavaScript (javascript)
```

These commands allow you to run Artisan, Composer, and NPM inside the Sail environment. This way your host machine doesn't need PHP or Node installed — Sail handles it all within Docker.

Examples for PHP, MySQL, Redis and MailHog

```
# Run a PHP command inside the Sail container
./vendor/bin/sail php -v

# Connect to the MySQL database
./vendor/bin/sail mysql -u root -p
```

```
# Open a Redis CLI session
./vendor/bin/sail redis-cli

# Access MailHog (fake email server)
http://localhost:8025
```
Code language: PHP (php)

With these commands, you can quickly interact with the core services Sail provides. Run PHP commands without installing PHP locally, manage your MySQL database directly inside the container, debug Redis queues, and view outgoing emails in MailHog via its web UI.

Real-World Use Cases

```
# Run a Redis queue worker
./vendor/bin/sail artisan queue:work

# Send a test email (check in MailHog)
./vendor/bin/sail tinker
>>> Mail::raw('Hello from Sail!', fn($m) =>
$m->to('test@example.com'));
```
Code language: PHP (php)

These examples show how Sail makes development practical. You can process jobs using Redis queues directly inside the container, and test email delivery with MailHog before sending anything to real users. This workflow improves reliability while keeping your system isolated and clean.

Customizing Sail

```
# docker-compose.yml
```

```
services:
  laravel.test:
    build:
      context: ./vendor/laravel/sail/runtimes/8.2
    ports:
      - '80:80'
    volumes:
      - '.:/var/www/html'
  mysql:
    image: 'mysql:8.0'
  redis:
    image: 'redis:alpine'
  meilisearch:
    image: 'getmeili/meilisearch:latest'Code language: PHP (php)
```

You can extend Sail by editing the `docker-compose.yml` file. Here we've added Meilisearch for search functionality alongside MySQL and Redis. Sail makes it easy to spin up additional services without complex configuration.

Tips and Tricks

- Use `./vendor/bin/sail up -d` to run containers in the background.
- Add Sail aliases in your shell config so you can just type `sail artisan`.
- For performance on macOS, use Docker's "VirtioFS" file sharing mode.
- Stop services you don't need (like MailHog) to save system resources.

Comparison: Sail vs Valet vs Homestead

Feature	Sail	Valet	Homestead
Purpose	Docker-based environment	Mac-only lightweight dev server	Virtual machine (Vagrant/VirtualBox)
OS Support	Windows, macOS, Linux	macOS only	Cross-platform
Setup	Fast, minimal	Very simple on macOS	Heavier, slower to set up
Performance	Depends on Docker config	Excellent (native)	Slower (VM overhead)
Flexibility	Easy to add services (MySQL, Redis, etc.)	Basic PHP + MySQL	Pre-packaged full stack
Best Use Case	Teams, Docker-friendly workflows	Solo Mac developers	Legacy or VM-based workflows

See [Best Laravel Starter Kits (Breeze, Jetstream, Spark, Nova & 22 More)]() for the list of laravel starter kits.

Laravel Sail removes the headaches of setting up a local environment by running everything inside Docker. It's cross-platform, customizable, and team-friendly. Whether you're building a quick prototype or managing multiple projects, Sail keeps your development consistent and portable. If you want a simple yet powerful way to standardize Laravel development across your team, Sail is the tool to use.