Repurposing Container Technologies for Development

Jacob Howard

Docker Community All Hands

11 March 2021

What do you *need* to know to use containers in development?

Basic Concepts

There's only three that really matter:

- Images
- Containers
- Orchestration

Images

The way container software is shipped

- Static, standardized, and portable filesystem snapshot
- Predefined command or entrypoint
- Built using a layered model identified by digests
- Derived from base images via **Dockerfiles** (or other means)

```
FROM ubuntu:latest

# ...install dependencies, build code, configure...

ENTRYPOINT ["/usr/bin/mywebserver", ...]
```

Containers

Isolated environments for running processes

- Shared host kernel (Linux or Windows)
- Create an isolated "space" for running processes
    - Reduced capabilities
    - Isolated filesystems and networking
- Various implementations
    - Same philosophy
    - Standard APIs
- Containers : Images : : Processes : Executables

Container Features

Feature	Purpose
Volumes	Provide persistent shared storage
Bind Mounts	Make host files available to containers
Networking	Provide flexible internal and external communication
Port Mapping	Expose container ports on host

Orchestration

Making containers manageable

- Automate the lifecycle of containers
- Automate networking/storage
- Organize containers into abstract services
- Handle dependencies between services
- Usually YAML files

```
services:
  database:
    image: postgres
    ...
  redis:
    image: redis
  api:
    build:
      context: code/api
    ...
  web:
    build:
      context: code/web
    ...

volumes:
  data:
```

Everything Else...

Kubernetes? containerd? BuildKit? runc?

- Just tools that deals with these three concepts
- Some tools deal with more than one concept
- Docker provides an integrated set of tools to build images, run containers, and orchestrate projects
    - Specifically designed for development

How do you run containers?

The Container Technology Stack

What do you need?

- Image management (building, storing, shipping, acquiring)
- Container runtime
- Container management engine
- API to access that engine
- Friendly mechanism to access that API

Don't panic.

Someone else already did the work for you!

What is "Docker"?

It's a company that builds...

- A huge amount of open-source, low-level container technologies
- A way to build images and run containers locally (**Docker Desktop**)
- A way to host/find/share images (**Docker Hub**)
- A way to orchestrate containers (**Docker Compose**)

...now focused on containerized development.

How are images, containers, and orchestration used for deployment?

Images in Production

A tool for reproducibility

- Used to ship fully built and packaged versions of application components
- Provide a way to package dependencies
- Provide a way to package assets (HTML/CSS/JS)
- Provide consistent configuration

Containers in Production

A tool for isolation and resiliency

- Isolate application components
- Carefully control code capabilities
- Replicate services with fewer resources

Orchestration in Production

A tool for managing complex applications

- Load balance containers on multiple nodes
- Handle version migrations
- Define ingress points for services
- Provide detailed monitoring
- Automatically restart failed containers

How do we adapt these concepts for development?

Images for Development

A tool for consistency

- Ideally use the same images as in production, but:
    - Install additional tools (e.g. compilers or debuggers)
    - Override entrypoints (e.g. to do hot reloads or use different flags)
    - Dynamically (re-)build images
- Enables you to:
    - Avoid installing tools and dependencies on your local system
    - Ship a consistent set of tools/toolchains for developers
    - Ship a reproducible set of dependencies
    - Set consistent static configuration for components

Containers for Development

A tool for process automation

- In addition to production use cases, you might:
    - Start a shell inside a container (e.g. to run a debugger or REPL)
    - Have a container that runs a command and then exits (e.g. to perform a build)
    - Dynamically recreate containers when re-building images
- Enables you to:
    - Reproduce your production environment easily
    - Create consistent and reproducible environments for developers
    - Automate common workflows for development teams

Orchestration for Development

A tool for workflow automation

- Roughly model your project after production, but:
    - Potentially decompose services
    - Potentially replace/mock services
- Enables you to:
    - Approximate how application components will interact in production
    - Avoid a terminal-tab-per-component development workflow
    - Ship a common automated configuration for your application
    - Automate common development tasks

Takeaways

- There's a lot of documentation, opinion, and tooling around containers
    - A lot of this is geared towards *deployment*
- For development (and also most deployment scenarios), it really boils down to three basic concepts: **images**, **containers**, and **orchestration**
- Understanding these concepts will help you contextualize tools and determine how they can help your development workflows
- Each concept has a role in deployment that's modified or expanded in development, and you should experiment!

Thanks for your time!

Slides available at

`havoc.io/talks/repurposing-containers`

Ping me with questions or feedback

@havoc_io