Live migration allows a user to move a running application from one machine (a source) to another (a destination) without restarting it. The technique has proven useful for diverse tasks including load balancing, managing system updates, improving data locality, and improving system resilience. Unfortunately, current live migration solutions fail to meet today's computing needs. First, most techniques do not support heterogeneous source and destination hosts, as they require the two machines have the same instruction set architecture (ISA) or use the same operating system (OS), which hampers numerous live migration use-cases. Second, many techniques are not transparent, as they require that applications be written in a specific high-level language or call specific library functions, which imposes barriers-to-entry for many users. We present a new lightweight abstraction, called an vessel, that supports transparent heterogeneous live migration. An vessel maintains a machine-independent encoding of a process's state, using WebAssembly abstractions, allowing it to be executed on nearly-arbitrary ISAs. An vessel virtualizes all of its OS state, using the WebAssembly System Interface (WASI), allowing it to execute on nearly-arbitrary OS. We introduce docks, software systems that execute and migrate vessels. docks face two key challenges: First, maintaining a machine-independent encoding at all points in a process is extremely expensive. So, docks instead ensure that an vessel is guaranteed to eventually reach a machine-independent point and delays the initiation of vessel migration until the vessel reaches such a point. Second, a dock may receive an vessel migration that originates from a dock executing on a different OS. Rather than attempting to restore the vessel's OS state, as is conventional for live migration, the dock instead recreates the OS state and updates the vessel's mapping accordingly. We implement a prototype and three live migration use cases for edge computing, batch processing, and scientific computing, and show order-of-magnitude benefits compared to existing state-of-the-art techniques