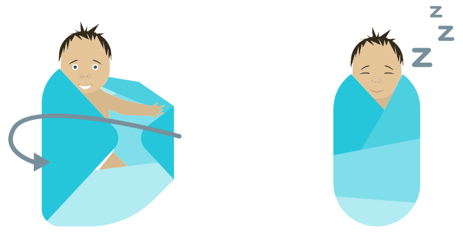
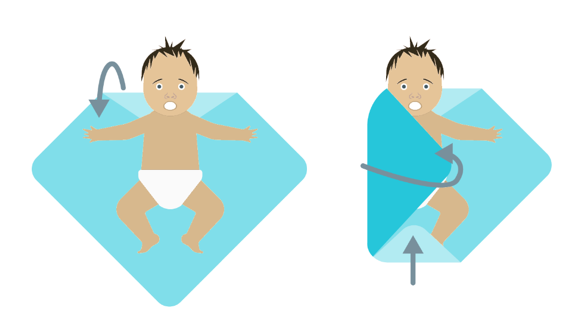
What’s a Docker Image?

Picture the neonatal care unit of a hospital, with rows of incubators lined up to receive infants. IV poles, ECG machines, and respirators are ready to go. The medication cart is stocked, and a stack of tiny beanies is on hand. Now imagine having to pull together all these tools and supplies for each new arrival. It would be inefficient, if not impossible. The unit’s productivity would slow, and its performance would suffer.

A Docker image is like a NICU for containerized microservices. It’s a sweet little platform that holds the monitoring, logging, security, and storage resources necessary to package, run, and maintain code. Developers can stop, start, add, or move tasks and services to match demand.

By design, a Docker image can’t be changed. This read-only container environment imposes consistency, so that each service works the same way in development and testing as it does in production. The containers themselves, though, are scalable and flexible. They sit atop the static image framework, rather than inside it. This arrangement allows the containers to draw on the Docker infrastructure yet operate independently of it.

Docker architecture has other advantages, too. Companies pay only for the resources they use. During periods of heavy demand, engineers can spin up new containers. When less capacity is needed, they can scale down the number of instances. With their code safely swaddled



in dockerized containers, developers can focus on

optimizing the apps in each bundle, rather than

worry about deploying or scaling them. And that should help everyone

sleep through the night.