La primera ventaja ya la hemos comentado y es que Docker mejora enormemente la fiabilidad de la entrega continua, ya que las imágenes de Docker son portables inmutables y se generan una vez validadas contra la ejecución automática de pruebas.

La segunda ventaja es la facilidad que ofrece Docker para la automatización de tareas. El proceso de generación de imágenes de Docker y de su distribución es casi trivial gracias a la API que implementa el demonio de Docker. Pero también lo es automatizar la respuesta a eventos como aumentos de tráfico o caída de *datacenters*.

Por último, Docker es una herramienta ideal para implementar arquitecturas basadas en microservicios, y gracias al aislamiento entre contenedores, permite la combinación de microservicios en una misma máquina para la consecución de diversos objetivos. Por ejemplo, imagina una aplicación con una API pública que, entre otros, ofrece un servicio de autenticación. Imaginemos otro componente que hace uso de la API pública para autenticar peticiones. Además de servidores dedicados a ofrecer la API pública, Docker facilita enormemente correr siempre el segundo componente con un contenedor de API pública en la misma máquina (y no visible públicamente). De esta manera, aunque nuestra API pública sufra, por ejemplo, un ataque de denegación de servicio, el segundo componente podrá seguir funcionando. A esta combinación de contenedores, que tienen que correr conjuntamente, se les denomina *pods*.