Effortless Enterprise Deployments: Navigating Docker for Maximum ROI

Following its introduction on March 15, 2013, Docker immediately received considerable media attention, capturing the software industry's imagination with breathtaking speed. This happened because Docker containers offered an elegant solution to the common enterprise problem of deploying applications at scale.

Before Docker came along, software packaging and deployment was a slow, expensive and often painful process, involving a complex mixture of library dependencies, package management systems, configuration management tools and virtual machines, all looked after by multiple specialist engineers.

A side effect was that most enterprises updated their systems infrequently, perhaps between one and four times a year. This, in turn, meant it took at least three months, usually much longer, to get new ideas in front of customers, making it almost impossible to rapidly respond to a sudden opportunity or unexpected change in market conditions.

Nicole Forsgren, Jez Humble and Gene Kim examined what high-performing technology organizations have in common and published the findings in their book, Accelerate. They found that these organizations have a higher deployment frequency, a shorter lead time for changes, a lower change failure rate, and a shorter time to restore service when something goes wrong.

These processes are often referred to as the DORA metrics because they were first identified by the DevOps Research and Assessment (DORA) group through its State of DevOps reports. Docker can help with all four of these measures so that a high performing IT team can deploy code as often as needed, multiple times per day.

The effect of speeding up software delivery and the corresponding cost reduction as recommended by the authors of Accelerate is profound; it means that organizations can quickly get new ideas in front of their customers to see which ones gain traction.

> **"There's huge variation between companies in the elapsed time for a new product idea to appear in front of users. For some folk, it's months. For others, it's hours or minutes. Some marketing teams can have a wild thought at 9 am and see it in production that afternoon. Others have given up having wild thoughts."**
>
> **– Anne Currie and Chales Humble, The Cloud Native Attitude**

One of the case studies in The Cloud Native Attitude is The Financial Times. Writing in her own book Enabling Microservices Success, Sarah Wells, who spent over a decade at the news organization, notes that "The Financial Times did around 100 releases every day in 2021. This delivers real business value: you can quickly implement and get real feedback on your ideas."

Moving from infrequent to release on demand requires changes at the organizational level as well as technologically, but from a technology point of view, Docker containers are an essential enabler of this change. They provide a way to easily create a distributable artifact for any application and deploy it at scale into any environment, without requiring you to change your existing software stack.

Research conducted on our behalf by Forrester shows that organizations adopting Docker Desktop can expect improvements to developer efficiency of around 6%; a faster time to market for new applications, features, and enhancements; and reduced downtime due to reliable service delivery. Our customers, including CARIAD and The Warehouse Group, can attest to Docker's ease of use while speeding up and scaling their app build SLAs.

In this eBook we offer a concise but definitive guide for enterprise-level Docker deployment, including a strategic roadmap to seamlessly integrate Docker into various IT infrastructures.

Drawing on case studies and our own real-world experience, we focus on practical, high-impact strategies for adopting Docker products within large organizations. This will equip you to make informed decisions in order to improve your company's release velocity, quality, reliability and efficiency.

What is Docker?

Docker is a platform designed to help developers build, share, and run applications within containers. There are two basic elements to it:

A **container image** is a lightweight, stand-alone, executable software package that includes everything an application needs to run: code, runtime, systems, tools, libraries and settings.

Containerization involves a standard packaging format and an engine for running containers. The engine facilitates the deployment and execution of containers on any supported operating system.

Container images become **containers** at runtime when the container engine unpacks the application and runs it in a way that isolates the software from any other containers running on the same infrastructure. Docker containers run on the Docker Engine. This is conceptually similar to, but much more lightweight than, a Virtual Machine (VM) such as that offered by VMWare.

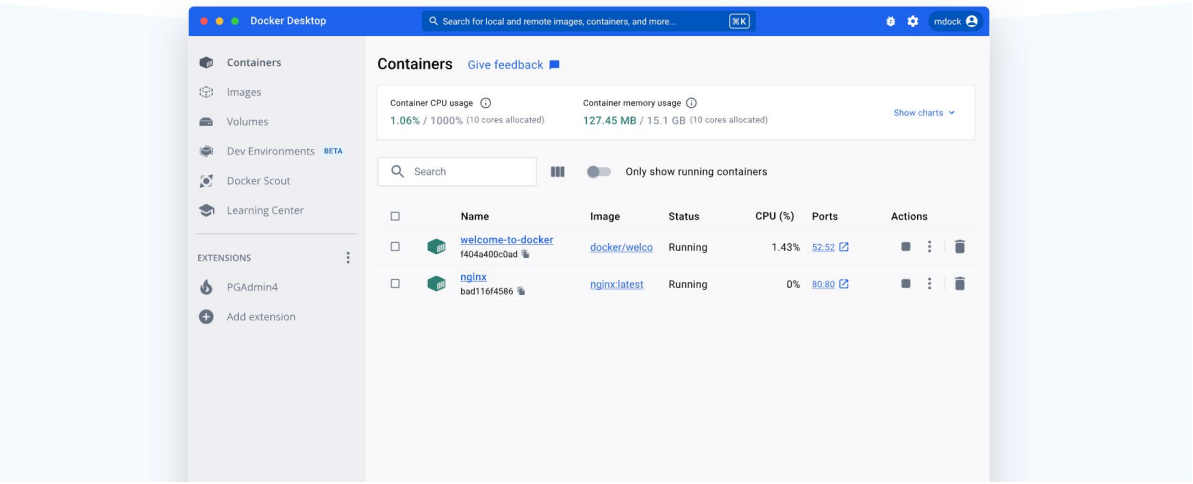
Whatever your teams are building—monolithic applications, microservices or AI/ML systems—Docker is invaluable. The lightweight nature of containers means you can get better hardware utilization through tighter bin packing. This reduces both your environmental impact and costs due to the effects of energy proportionality. Containerized applications are also faster to start up, and as a result have become synonymous with microservices architectures where we start up and tear down systems with alacrity.

Containers are **immutable**, so if you need to make a change you have to create a new container. This might not sound like a good thing, but it is. Historically, computers and software systems have been treated as mutable infrastructure, with changes applied to an existing system either all at once or over a period of time. This typically meant that the exact order in which software and patches were installed mattered, since one piece of software would replace older binaries and make incremental updates to configuration files. It was absurdly time-consuming and error-prone to get a computer into the correct state, as demonstrated by the common phrase "But it works on my machine!" Containers eliminated this entire category of issues.

Alongside other industry leaders, just two years after launching the Docker engine we created an open governance structure with the express purpose of creating open industry standards around container formats and runtimes. Called the Open Container Initiative or OCI, this currently maintains standards for Runtime, Image and Distribution.

Since 2019, Docker has focused on products for developers, particularly in the enterprise. Docker's vision is to increase the time developer teams spend innovating and writing code, and decrease the time they spend on everything else.

Our main product, Docker Desktop, is a one-click-install application which makes it easy for developers to work with containers, regardless of whether their desktop is running macOS, Linux, or Windows. Docker Desktop isn't solely about local development. It is a launchpad for container development. As part of the download you get Docker Engine, Docker CLI client, Docker Scout and Docker Build Cloud. By default these last two come with limited compute capacity but you can always purchase more.



Docker Desktop increases the speed and efficiency of the developer's task loop of code, build and test, which in turn makes the developer more productive. There are many features in Docker Desktop that help developers work with containers, such as Docker build history and analysis, Docker Debug, volume backup and share, and synchronized file-shares.

Docker Desktop offers enhanced security for enterprises. This includes SOCKS5 proxy support for seamless network connectivity; integration with NTLM and Kerberos for smoother authentication processes; Enhanced Container Isolation (ECI), which provides the most secure developer container environment without impacting developer tasks; and Hardened Docker Desktop.

We also offer Docker Hub, the world's largest and most active registry of container images. Docker Hub offers Docker Official Images and Docker Verified Publisher Images that are created by Docker and its partners, and are used by millions daily as the basis for their containers. Docker subscribers can also use Docker Hub to store private images for their company's applications.

Docker Scout is another example of our developer focus in action. The product provides insights on software supply chain vulnerabilities and company policy violations from development to production. Developers have direct access to this information as part of their normal workflow, which means they can quickly address issues. This matters because resolving security issues earlier in the development process can cost considerably less than resolving them later in the integration/testing process or in production. Docker Scout also provides dashboards and policies for understanding your container security posture across the software development lifecycle. The Docker subscription offers three free repositories for Scout analysis.

The business subscription also includes free minutes for Build Cloud, which can dramatically speed up the building of containers, and Docker Debug, which provides a built-in toolbox for developers working with containers.

Building the Business Case for Docker

A mistake we make repeatedly in IT stems from the idea that programming is reducible to typing, and that productivity gains are best created by making developers type less or faster. This idea led us to thinking that counting the number of lines of code a programmer wrote, the bugs they fixed, or their GitHub commit stats were somehow useful measures of their productivity. They weren't, and never were, because typing isn't the challenge with programming.

The DORA metrics demonstrate that a far more effective focus is on individual developer experience or DevEx. You should aim to provide rapid onboarding for new developers to the team, reduce the amount of time it takes to find and integrate different components, and then have the shortest possible inner loop of development, enabling each individual developer to code, run and test while getting rapid feedback as they work.

Using Docker will undoubtedly save you time. But, more importantly, it enhances overall developer productivity by creating consistency in the way you roll out environments. This is true regardless of whether they are local or remote.

Our customers have repeatedly seen measurable improvements as their teams adopt Docker Desktop, whether they are using the DORA or SPACE metrics, and/or running qualitative surveys of their development teams. Taking the DORA metrics as our example, we've already seen how Docker Desktop can increase deployment frequency. But, Forrester suggests, container techniques "such as automated deployments, easy rollbacks, and high availability allow for much more reliable services", thereby improving Mean Time to Restore (MTTR) figures and reducing Change Fail percentages.

Using Docker Desktop will also save you money, even if you are running on a public cloud. Forrester assessed the three-year impact of Docker at a 126% Return on Investment (ROI). This is based on the feedback from interviewed customers and a thorough analysis of costs, benefits and risks in the context of a composite organization.

They calculated a total investment of $53.1 million over three years and a total three-year net present value of close to $67 million.

Some of our customers have seen an even faster ROI.

The Warehouse Group is New Zealand's largest retail chain, with over 12,000 employees in more than 300 stores. Their legacy development process, based around VMware virtualization, saw developers waiting weeks for environments.

In addition, a lack of standardization and automation hindered collaboration between developers and operations teams, causing delays in deploying new features. By modernizing their development process and transitioning from VMware VMs to Docker they are now saving 52,000 developer hours annually, and have reduced their deployment time from weeks to 60 seconds.

The Warehouse Group realized that the time and cost savings allowed them to achieve a ROI in just eight months, and they continue to yield substantial savings year after year. This transformation underscores the impact of modernizing development processes, illustrating how their strategic shift to Docker has not only drastically improved operational efficiency but also significantly boosted their bottom line.

At a higher level, this example illustrates that investing in the right developer experience translates into competitive gains. Such investments have the potential to make your organization more agile and nimble, and your products more innovative. Realizing these gains, however, requires a concerted effort from an organizational leader to ensure that the right tools are in place for their team, and the right projects and business goals are prioritized for resource investment.

Strategic Adoption Framework

Companies will likely see different parts of their organizations adopt containers and Docker at different rates. The early adopters blaze a trail of productivity, realizing gains in their DevOps processes. Other teams hear about these successes, but are unsure how to replicate them. Some teams are so consumed by their day-to-day work that they don't see how Docker can help them or don't have the time to investigate. The question then becomes how to enable a broader adoption of Docker across the organization in order to reap the benefits.

A strategic adoption framework for Docker might include the following steps:

Discovery

What level of understanding does each team have about Docker and the benefits of containerization? Do they understand all the tools that Docker offers and the problems they solve?

Education

A broad-based education program for teams to get everyone to the same level of knowledge is critical. To help with this, Docker offers a number of educational programs as part of the subscription.

Adoption

Plans for each team to adopt Docker into their processes, taking into account company priorities, technical considerations and schedules.

Governance

Docker provides a suite of tools for helping to manage how Docker products and containers are used, to ensure enterprise standards and security requirements are met.

Community of Practice

Creating and maintaining a community of practice around Docker will pay dividends over time. Sharing best practices, learning new features, and resolving issues in a community format creates organizational strength.

The discovery component is necessary because in large enterprises, senior staff are not always up to speed with exactly what is being used in production and how. When evaluating your existing IT setup, and considering whether Docker might be able to help with any problems your teams are navigating, there are a number of questions you can ask to aid discovery.

As a manager of managers within IT (IT director or CTO for example) you might ask your direct reports:

1. How do you manage development environments across your team? Are there challenges in ensuring consistency and efficiency?
2. Do your developers test locally as part of their development process?
3. What is your security posture around containerized applications? How do you manage vulnerabilities?
4. How do you ensure that base images and open-source components comply with your security policies?
5. Which applications are you currently containerizing? Are there specific types that are prioritized (e.g. web, microservices)?
6. How would you describe your team's maturity with containers? What's holding back wider adoption?

Once you have identified your current state of maturity, you have the option to consider where our suite of products might be most useful. For example, if you are not using Docker at all you might want to try starting a new project directly with Docker to see what the impact is.

Performance issues and security risks are the common issues we see with companies who are not using Docker products. The complexity and corresponding cost of maintaining a homegrown stack in a secure manner across large teams of developers with various levels of expertise, is often a major challenge. In this case, your organization might consider replacing its homegrown stack with Docker, a shift that is likely to be driven by a central in-house team aided by local technical champions who see the potential of the technology, and are willing to go the extra mile to promote its usage internally.

These central teams are what Matthew Skelton and Manuel Pais refer to in their book, Team Topologies, as "platform teams", whose job is to provide a non-mandatory "paved road"—a secure set of options that make life easy for the product development teams.

Finally, if you are using Docker without a subscription, it is worth assessing whether the add-ons that come as part of a business subscription, particularly with regards to security and governance, would be useful for you and worth the cost. These include enhanced container isolation, settings management and usage information. As we'll explore in the next section, a large number of our customers have adopted these as part of a subscription and found that doing so paid dividends.

Implementation Roadmap

If we look at how our customers adopt containers and Docker products, there are typically two main paths. Both tend to be driven bottom-up by developers. In the first scenario, you have one or more developers who know about containers and want to move away from the issues of configuration and "this works on my machine." Since your development teams are able to choose the right tool for the job, they will likely reach for Docker, and quickly find that using Docker Desktop is the easiest way to get things done, particularly if their desktops are running on Windows or MacOS.

Because containers work for any software, a second path is where developers want to use containers for dependencies. Kevin Barfield, Director of Solutions Architects at Docker, says, "You can start off with, 'I'm building my application, I'm fine running it on my local machine and I don't need a container for that. But I need to run this database or this message queue', or 'I need to run AWS locally, and I do need containers there. How can I do that if I'm on Mac or Windows?' I install Docker Desktop and life is now good."

Your developers' next step is to integrate Docker with their existing SDLC toolchain, in order to get the most value out of it. Out of the box, Docker works with the vast majority of development and operations tools available. We have plug-ins for popular IDEs such as VS Code and IntelliJ, for example. These make the IDE container aware, providing syntax highlighting on Docker files and Docker Compose files, and allow developers to interact with containers directly, from their IDE.

Docker also integrates cleanly with major code repositories such as BitBucket and Git via GitHub actions. The GitHub actions allow teams to automatically pull from Docker Hub, build images, and then push out to Docker Hub for further testing in response to a Git push.

Beyond this, Docker provides a variety of other hooks that allow your team to connect whichever tools they use, to make it easier to work with containers throughout the whole SDLC. In addition:

- Docker Compose makes it simple to create fully reproducible development environments.
- Docker works with all the testing frameworks for unit testing, integration testing, and so on.
- Docker Scout provides software supply chain tooling directly to the developer, allowing them to see vulnerability and policy issues immediately.
- Docker plugs into all the major continuous integration tools to ensure seamless builds and testing.
- Docker containers are deployable anywhere, from any public cloud to any private server or even smaller devices like IoT.

Whichever path you start down, usage permeates out via what are effectively pilot projects, whether officially or otherwise. Over time you'll find more groups of developers start to adopt Docker Desktop and find value in it.

As your organization gains maturity working with containers, the next step typically requires some level of standardization and input from a security team. Obviously the exact nature of this will vary depending on the nature of the organization. In highly regulated industries where developers may be restricted in terms of what they can install on their machines, this implementation route gets reversed—a platform team starts with configuring what is allowed, integrates that with the existing enterprise SLDC, then provides it to developer teams at the end. Whichever route you take, the end state is the same.

Docker offers a number of administrative features which come to the fore at this point, including enhanced container isolation and settings management. An important first step is ensuring that the users are signed into your Docker organization so that they can receive the subscription benefits. Docker offers the ability to enforce a centralized sign in for users, and simplify the login process and provisioning/deprovisioning of users through support for single sign-on (SSO) and SCIM.

Enhanced container isolation helps secure the container on the developer's machine against malware and other risks. Settings Management offers standard settings for Docker Desktop to all your developers, including the ability to configure HTTP proxies, network settings, Kubernetes settings, which Windows hypervisor to use, and more.

For the wider software supply chain, our Scout product can alert the developer to problems. Registry Access Management controls which registries a developer can pull containers from or push containers to. And the Enhanced Container Isolation prevents supply chain—by making the containers rootless and restricting access to the VM and other Docker internals, thereby blocking anything malicious from breaking out of an individual developer's machine.

Finally, Image Access Management within Docker Hub controls the types of images a developer can pull from—community, Docker official images, Docker Verified publisher images or their own organizational images—and the types they can't.

Combined, these features can help an organization comply with governmental standards such as the European Union's Network and Information Security (NIS2) Directive and Executive Order 14028. Coming into force in 2023, these init to occur at the application level.

Most competing solutions do not scan at the container level, whereas Docker Scout provides a GitHub application that can be embedded in your CI/CD pipeline to identify and prevent vulnerabilities in images from ever reaching production. Companies can use Scout to monitor their images for vulnerabilities, identify if fixes are available and provide the most up-to-date information to create more secure products.

We provide a number of tools to help:

- SSO/SCIM to integrate logins to your identity provider.
- Tooling to help regulate where developers can pull container images from or push them to.
- Tooling to standardize Docker settings across development teams.
- Tooling to secure the developer environment, ensuring that the enterprise software supply chain is protected.

Docker tools facilitate the deployment of a secure software development life cycle that aligns with compliance frameworks and regulations such as, but not limited to, SOC 2, ISO 27001, and FedRAMP. By incorporating these tools, companies can ensure their software development practices meet rigorous security and compliance standards, enhancing overall trust and reliability.

Docker grows with you as your usage grows. As your teams implement Docker into their processes, there will need to be governance incorporated into the process to ensure enterprise standards and security are being maintained.

Leveraging Docker Business for Maximum Impact

Docker Business enables enterprise-scale management and security for businesses. As we just discussed, it allows large teams to:

- Efficiently manage all Docker development environments with centralized management and Single Sign-On
- Maintain security with Registry and Image Access Management, limiting which registries and container images developers can use

Having looked at how our customers typically roll out Docker, we're going to do a brief survey of several more real-world examples where organizations across government, the automotive industry, video publishing and retail have taken advantage of Docker as part of their IT modernization programs. These examples will function as models for you as you plan your next strategic move.

Canadian Digital Service (CDS) is a group of over 200 public servants including around 30 developers. It was created to help change the way the Canadian government designs and delivers services for people in Canada. As they sought to integrate new technologies within existing bureaucratic frameworks, they faced a number of challenges—ensuring data security and privacy, fostering a culture of agility and openness within traditionally risk-averse institutions, and dealing with a number of legacy systems and processes.

Key initiatives led by CDS include developing and implementing a suite of platform products; common, open source tools designed to be easily set up, used and built on by other governmental teams. Included in this suite of products is GC Notify, a notification service originally developed in the UK that runs on Docker.

Pat Heard, site reliability engineer at CDS, told us that, "Docker allows us to quickly push changes many times a day, test them in a sane way and roll them back when they don't work. The Docker side is the easiest part of running GC Notify because it gives us all the goodness of Docker." This has contributed to the platform's success in sending out over 135 million notifications and enhancing the directness and timeliness of governmental communication.

Docker also speeds up onboarding new developers for CDS using tools like dev containers and GitHub Codespaces, with Docker Compose files. At the same time, Docker's security aligns with the government's stringent data protection and privacy standards.

CARIAD is the automotive software company that bundles together the Volkswagen Group's software and expands it for all of the brands under Volkswagen's umbrella, including Volkswagen itself, Audi, Bentley, Lamborghini and Porsche.

CARIAD had expanded rapidly and as a result, the group had to contend with multiple technology stacks and disparate development methodologies. In addition, while the software in the car is running on top of embedded Linux, much of the development happens on Windows laptops due either to a lack of support for certain tools on alternative operating systems, or contractual obligations with vendors.

CARIAD took the decision to adopt Docker and containerization technology to create a standardized, scalable, efficient development and operational framework. This shift marked a pivotal transformation for CARIAD. It provided a more integrated approach to software development and management, which aligned with their strategic goals of innovation and efficiency.

The technical team at CARIAD noted significant improvements in deployment speeds and reduced time required to bring products to market.

> **"Docker improves our deployment processes, ensuring consistency and efficiency from development to deployment. It's been a tipping point in approaching software delivery, aligning closely with our security and reliability needs."**
>
> **– Julius Pravtchev, CARIAD's senior DevOps, CARIAD**

In addition to adopting Docker's container technology, CARIAD implemented a comprehensive strategy for managing Docker Desktop at scale. Taking advantage of our Registry Access Management and Image Access Management capabilities, CARIAD enforces strict policies to ensure all Docker images are sourced from verified publishers on Docker Hub.

This practice minimizes security risks by preventing the use of potentially unsafe or unauthorized images that could introduce vulnerabilities into the development environment. By insisting on verified images, CARIAD significantly enhances the security of its containerized applications, aligning with the company's high standards for software safety and compliance.

Our security story is also important to JW Player, although they took a different approach to CARIAD. With over 40,000 broadcasters, publishers and other video-driven brands, JW Player has built the SaaS video platform of choice for companies who rely on video, and have streamed over 860 billion videos to date.

In 2023, JW Player set out to implement an image vulnerability management program while preserving their core engineering values, including giving their software engineers full autonomy over their build and release processes. Speaking at DockerCon that year, Stewart Powell, engineering manager at JW Player, described how they used Docker Scout to meet their compliance goals while remaining true to their principles as an engineering organization:

> **"With Docker Scout, we were able to go into Docker Hub, check a box, and provide developers with a comprehensive software supply chain and image vulnerability program with no effort. In fact it was so easy, we didn't have to wait weeks or months for developers to go and update their development pipelines. We were able to get more than 300 repositories up and running on our first day in under an hour."**
>
> **– Stewart Powell, Engineering Manager JW Player**

Scout offers developers near real-time feedback of the security of their containers, Powell explained, but they've also been able to provide their security team with an overview that "allows them to prioritize and choose areas for remediation." This has allowed JW Player to strengthen protection across their streaming services without added complexity.

Our last case study shows how Docker is able to help with AI/ML projects. The Ingka Group, who operate IKEA home furnishing stores in 31 countries, has embarked on an ambitious journey to harness the potential of AI/ML to enhance operational efficiency and customer experience. Its uses include demand forecasting and lifelike design apps for consumers, while at the same time taking a strong stance on using AI in an ethical way.

Tasked with making AI/ML development easier, Ingka Group's MLOps team of 14 has a structured yet flexible approach, providing both automation and personalized support. Yasin Faizan Shaik Mohammed, tech lead for the MLOps Platform, explains the team's service model for internal developers, saying, "You concentrate on building your model. We will make sure that it is up and running all the time."

The MLOps team recognized the importance of a unified platform for making large-scale AI/ML projects run smoothly, and chose Docker to organize software in containers and Kubernetes to coordinate those containers. The team incorporates essential tools like MLflow to keep track of experiments, and Seldon Core to put models and other customized tools into use to fit the Ingka Group's needs.

By containerizing their AI/ML applications, the Ingka Group could quickly prototype, test and deploy new models.

> **"Docker fosters collaboration both within and across teams"**
>
> **– Soufiane Benzaoui, Consultant Machine Learning Engineer, Ingka**

Although exact numbers are hard to pinpoint, there has been a noticeable reduction in the time to market for applications developed with the platform, Benzaoui told us. In addition, using Docker "affects the development cycle from end to end, the models are now scalable, secure, and observable," he said.

Docker's versatility makes it a crucial investment for improving an organization's software development lifecycle. Its immediate use of speeding up developer onboarding and boosting collaboration, as seen with CDS and CARIAD, strategically enables innovation.

Docker enhances efficiency by standardizing and securing deployments, which is essential for data-centric organizations like CDS and JWPlayer. This leads to tangible business benefits, such as application modernization and increased experimentation, and it also enables new offerings, like streamlined machine learning operations at IKEA and the Ingka Group. The result at the other end of the cycle: better products and customer experience.

Future Trends and Continuous Innovation

While Docker will continue to support its early adopters, the majority of whom are comfortable working with the command line interface (CLI), a major focus for us is to democratize working with containers, making it easier to onboard junior developers who haven't worked with them before. As part of this, we are developing new tutorials and prompts on how we present features.

Our 2024 Docker State Of Application Development Report highlights how far AI/ML has already penetrated the software development field, with 64% of respondents stating they use AI for tasks such as writing code, documentation and research, and 46% working on ML in some capacity. As an example of the former, at Microsoft Build this year we announced a private beta of a Docker co-pilot agent that helps developers transact within GitHub, build files, and execute. In terms of the latter, courtesy of the Ingka Group, we've already seen how Docker can help ML and AI projects, and we continue to invest heavily to support developers working in this area.

AI/ML projects will typically involve both data scientists and traditional software engineers. For software engineers, the intense and specialist compute demands these projects involve means that even though our laptops are getting faster and more powerful, it simply isn't possible to to run everything on a local machine.

As a consequence, we are seeing a renewed push to facilitate remote building, testing and running of environments, with 36% of respondents in our survey working mainly from a remote development environment. Despite this trend, we still need to keep the developer feedback loop fast and tight.

Docker is investing heavily to help, with Docker Desktop moving towards being truly hybrid. Docker Build Cloud already lets developers build in the cloud as easily as they can locally. Test Containers Cloud allows you to spin up environments to automated testing in a cloud environment so you don't have to wait. And a third tool, which will enable developers to run their applications and interact with them in a cloud environment, is currently in the works.

Later this year, we will be adding new insights dashboards to Docker Desktop, which will provide statistics on how fast you are building containers and your overall success rate. Meanwhile, we continue to invest in our overall ecosystem, with a growing number of partnerships and alliances.

As noted earlier, to truly get the most out of Docker requires a change in how your company works. If you look at the last few years of software development, a great deal of progress has been made around reducing time to value—how we get an idea in front of customers quickly in order that we can validate it. Docker can help you get there, but only if the rest of your organization is able to make the necessary supporting changes.

From a software delivery perspective, we need small, autonomous, cross-functional teams building independently deployable units of software. We also need developers who are able to focus, and work with short feedback loops.

In this eBook we've shown how Docker can help software delivery teams reach these goals, saving you money and helping your teams to innovate faster—with real-world examples from The Warehouse Group, Canadian Digital Services, CARIAD, JW Player and Ingka, alongside supporting research from sources like Forrester. We have also shown how Docker continues to invest and rapidly innovate in response to changing market conditions, making our product suite a safe, long-term investment for your IT modernization program.

Are you ready to roll out Docker in your enterprise?
Explore the Docker Business Subscription features today.