Techstrong | Research

IN PARTNERSHIP WITH docker

Leadership Guide to Application Development

2024 AND BEYOND

SOFTWARE DEVELOPMENT is undergoing profound changes, from business conditions to the technology stacks and tools used to build software. For much of the past ten years, money was cheap, jobs were plentiful, developers were in high demand, and investors demonstrated a greater tolerance for bold strategies, even when they did not yield immediate returns.

Today, economic pressures demand that businesses and their software organizations deliver value quickly. The tech job market is highly competitive, while tech layoffs are occurring. Businesses have fewer funds to invest in technology. Leaders need to design highly agile, adaptable, and resilient organizations. That's where Docker and developer experience comes in.

The 2024 Docker State of Application Development Report provides a deep-focus snapshot of the rapidly evolving world of software development today. Highlights include the aggressive shift from monolith to containers and microservices architecture, rapid adoption of AI-assisted development, the incorporation of AI and machine learning (ML) within applications, heavy reliance on remote environments, and the increased focus on application security.

This white paper offers software leaders a view into the insights and challenges of software development. This paper highlights insights from the 2024 Docker State of Application Development Report, practical guidance, and actionable recommendations for software leaders, from engineering managers to senior and C-level leaders of software organizations.

Embracing Microservices and Containers:

The adoption of cloud-native software architecture has hit its stride and is growing rapidly. In the 2024 Docker State of Application Development Report, 51% of developers say they work on microservices-based applications today, and 80% utilize containers like Docker in their development environment.

THE CALL TO SOFTWARE LEADERSHIP

Leadership in software development organizations is multi-faceted. It requires a deep passion for learning technology and understanding people, a dedication to creating environments where teams can thrive, and tenacity to find and remove roadblocks inhibiting progress by their teams. Leaders are expected to inspire their teams, transform organizations, and equip them to execute in an environment where changes in the company and technology happen continuously.

Collaboration and Learning

Software teams highly value teamwork and collaboration. While a great deal of individual work occurs, creating software is a team sport. Developers may rely on architects, software testers, and platform engineers for designs, testing, tools, and configurations. Projects may have a security champion to improve application security and the underlying toolchain supporting software pipelines and workflows. Docker Business facilitates collaboration, streamlining workflows, shared image repositories, rapid development environment creation, security controls, and overall team visibility.

It's About The People:

Establishing the work environment and culture of software teams is the most important role of any software leader. In evaluating the operational efficiency of their organizations, survey respondents place a great deal of importance on collaboration among team members and stakeholders (48%), as well as learning and improving as a team (46%)

2024 Docker State of Application Development Report, pg. 23

Continuous learning is critical for software developers to grow and develop their technical skills, and developers have many individual preferences for learning. Developers value having time and the availability of computing resources like remote environments, containerized software, and sandboxes where they can experiment and learn.

Software leaders recognize that developers rely on a large variety of resources, including their team members and external resources, to learn, discover, and utilize new tools, programming languages, and other technologies, which is essential. For many developers, developing software is a big part of their personal interests, and they spend time away from work on their software projects, contributing to open-source projects, interacting with online communities, and continuous learning.

Continuous Learning:

Resources for self-learning are plentiful online, and software professionals rely on a number of them. Survey respondents rely heavily on online resources (54%), videos, blogs, forums, online courses and certifications (45%), and reading source code (45%). Additionally, software professionals read the documentation (55%) and take formal online courses and certifications (45%)

2024 Docker State of Application Development Report, pgs. 8, 10

Software leaders may not truly appreciate how much developers spend interacting on online developer sites, messaging friends and colleagues, watching technical videos, and reading blogs. Based on data from 250K+ developers in our global community, developers code 52 minutes per day — about 4 hours and 21 minutes during

a normal workweek from Monday to Friday - Global Code Time Report. In fact, developers must know their leaders value efforts to tap into these resources and encourage sharing information among teammates rather than seeing these efforts as wasteful, non-productive "time away from coding." See the daily.dev article Community for Developers: Benefits and Opportunities for additional information.

Online and In-Person Networks:

The report shows that respondents reach out to developer communities (50%), check social media (46%), use search engines (45%), and talk with colleagues and friends (40%). Additionally, interacting and engaging with other software professionals at conferences (28%) and via live conference talks (25%) are also important sources for learning about new tools and development techniques.

2024 Docker State of Application Development Report, pg. 9

Removing Inefficiencies

An essential responsibility of a software leader is to continuously examine the software development lifecycle (SDLC) for roadblocks that hinder progress and stop work. Roadblocks can arise at any stage of the SDLC, from requirements gathering and design to coding, testing, and deployment. Software projects often experience ambiguous and shifting business requirements, knowledge sharing and communication challenges across the team, and challenges estimating work when too many variables are unknown.

Another important consideration is bottlenecks, points in software development workflows where work backs up and inhibits downstream progress. Much of the work on understanding bottlenecks, which are most important to address, comes from the lean manufacturing movement and Eliyahu M. Goldratt's work on The Theory of Constraints. Additional information is available at The Theory of Constraints and Lean Manufacturing.

Removing Roadblocks and Paving Roads:

The report shows that problems in the planning (31%), estimation (24%), and designing (22%) processes frequently impede work. Better planning and testing tools (28%) were also indicated.

2024 Docker State of Application Development Report, pg. 19

By looking for and anticipating inefficiencies like roadblocks and bottlenecks, software leaders can aid in getting questions answered, closing communication and knowledge-sharing gaps, and identifying where tools and process improvements can increase the flow of work.

Technology Choices

Software teams are adopting cloud-native architecture, a design approach that utilizes scalable and resilient cloud technologies, often built around containers, microservices, and dynamic orchestration, to optimize systems for cloud environments. Containers and microservices help developer teams enhance scalability, flexibility, and efficiency in application development and deployment. Containers allow developers to package applications with all their dependencies, ensuring consistency across various environments from development to production. This portability makes it easier to deploy and manage applications in different environments, whether on-premises or in the cloud.

Embracing Microservices and Containers:

The adoption of cloud-native software architecture has hit its stride and is growing rapidly. According to the report, 51% of developers say they work on microservices-based applications today, and 80% utilize containers like Docker in their development. Respondents said they are transitioning from monolithic to microservices (29%) or use a hybrid monolith/microservices approach (48%).

2024 Docker State of Application Development Report, pgs. 13,14

Microservices complement containers by breaking down large, monolithic applications into smaller, independent services that can be developed, deployed, and scaled separately. This architectural approach allows teams to work on different services simultaneously, reducing dependencies and resulting in faster development cycles. Microservices also improve fault isolation; if one service fails, it doesn't necessarily bring down the entire system. By using containers to manage microservices, software teams can more easily update, scale, or redeploy individual services without disrupting the entire application. Docker's widespread use among developers is evidenced by Stack Overflow's developer surveys. In the 2024 Developer Survey, 59% of professional developers reported using Docker in their work, and it was also recognized as the most-admired tool, with 78% of respondents expressing their admiration.

Software Development Domain Knowledge

The software leadership mindset goes beyond overseeing schedules, setting priorities, meeting deadlines, and providing performance feedback. It requires a deeper understanding of creating quality and secure software, making decisions to help software creators be more productive, and balancing the variety of demands on developers, all while meeting the needs of the business.

Leaders must have the ability to recognize great or inspired work when they see it, even when they didn't build it themselves. It is also important to support learning from failures, as negative consequences from prior failures can stifle innovation and hinder problem-solving. Why are experimentation and risk-taking so important?

The ability to experiment allows teams to explore new ideas, test novel solutions, and discover more efficient ways of building software. By experimenting, developers can validate assumptions early, identify what works, and learn from what doesn't, leading to better decision-making and more effective solutions. Embracing calculated risks helps teams make decisions in challenging situations or break from the status quo and create innovations. When encouraging risk-taking, software leaders should set boundaries in what areas of experimentation are encouraged and when developers should exercise caution should be exercised. Software leaders can also look to other developers for input and review when assessing risks versus the potential rewards.

The Right Tools and Decisions:

The report provides several examples of highly-rated tools, including Visual Studio Code (71%) and JetBrains (62%) IDEs, GitHub (73%), and the automation and development workflow execution capabilities of GitHub Actions (62%). While a manager may see benefits in standardizing on one specific tool, such as an editor or integrated development environment (IDE), developers may see the flexibility of choosing their IDE as a positive. The potential disruption to what's working is not always worth the perceived benefits from a change.

Decisions about tools and technology stacks may not fall the way every person wants, but whenever possible, it is best to include and rely on the input of technical staff members when making decisions that impact them and their work.

2024 Docker State of Application Development Report, pg. 19

Experimentation and Taking Risks:

Participants in the survey rated empowerment to experiment and take risks (49%) as important. Themes like "failure is not an option" can instill fear and inhibit trying new ideas, taking reasonable risks, and learning from failures. "Fail early, fail fast," on the other hand, signals leaders don't expect every idea to succeed; experimentation and failures are part of learning and lead to solutions. *[Don't Be Afraid to Fail Because You Can Learn From It! How Intrinsic Motivation Leads to Enhanced Self-Development and Benevolent Leadership as a Boundary Condition, National Institutes of Health]*.

2024 Docker State of Application Development Report, pgs. 8, 23

UNDERSTANDING SOFTWARE DEVELOPMENT WORK

While not every task of a software developer requires their full attention (deep work), software development is a cognitively demanding job that requires periods of sustained concentration, creativity, and problem-solving skills. Focus is necessary when developing ideas and approaches to implement functionality, tracing detailed logic flows, finding errors in code, and refactoring code. Interruptions and switching between multiple tasks, projects, or parts of the codebase are mentally taxing and incur time penalties as developers struggle to return to the mental point where they can pick back up on their work."

Research has shown that interruptions have a material impact on software developers. In the 2024 study "*Breaking the Flow: A Study of Interruptions During Software Engineering Activities*," researchers that specific on-screen interruptions significantly increase the time spent on code comprehension tasks. According to the study, "The combined influence of in-person and on-screen interruptions significantly impact the time spent during the code review process, with various interruption combinations leading to different effects on task duration."

Ready access to tools, base images, configurations, security tools, software stacks, and technology environments developers can readily use to spin up new development and production-like test environments, knowing they are using supported technologies for development and testing. Containers are widely used by software developers across development, testing and production. Docker is used by over fifty percent of developers in the Stack Overflow 2024 Developer Study.

Time to Focus:

When survey participants rated areas of inefficiencies, the most significant were balancing technical debt with feature work (32%) and securing uninterrupted time for deep work (28%).

Software leaders can take steps to reduce distractions and jettison long and inefficient meetings. Many organizations implement "no meeting days" or scheduled quiet hours. By turning off messaging and email notifications, minimizing interruptions, and eliminating distractions, developers can focus and know they have dedicated time to work and not be interrupted.

2024 Docker State of Application Development Report, pg. 23

It Works On My Machine:

A surprising insight from the survey is the prevalence of cloud-based development environments, or remote environments, in use by developers. While nearly two-thirds of respondents (64%) use their laptops and desktops for software development, over one-third (36%) opt to develop in remote environments.

There are likely multiple reasons for this significant shift to remote environments in the cloud, ranging from faster onboarding and setup to increased consistency across environments to avoid the dreaded "it works on my machine" but not yours scenario. Additionally, remote environments can more closely mimic production setups, reducing deployment issues. The decreased maintenance and upkeep of individual developer environments is a benefit many software leaders will value.

2024 Docker State of Application Development Report, pg. 16

RESILIENCE IN CONTINUOUS CHANGE

Change is constant for virtually all leaders and software teams. Continuous change requires organizational agility and adaptability of leadership and teams. A resilient organization can effectively anticipate, prepare for, respond to, and adapt to incremental changes and sudden disruptions while maintaining its core functions and long-term viability.

The Covid era accelerated the transition to digital experiences, necessitating businesses and customers to quickly shift how they transact and conduct business to new digital experiences. If

investing in software gave businesses an edge ten years ago, today, that investment is mandatory to stay afloat. Digital transformation evolved accordingly, and projects focused on application modernization and cloud migration are mainstream.

Resilience is equally applicable to embracing and leveraging new technologies and trends. Utilizing microservices, containers, orchestration, and AI requires organizational resilience as teams learn and become proficient in the latest technologies and techniques. AI, machine learning, and generative AI (increasingly) are commonplace in the technologies software teams use, and in the software we create. Copilots, working within developers' integrated development environments (IDEs), now perform code completion, augmentation, and generation. Docker is aiding developers' adoption of AI/ML. Hugging Face Hub, a platform for creating and sharing machine learning models, recently added Docker Spaces support, enabling developers to quickly prototype and build ML-enabled applications.

AI and AI-Augmented Development:

Development teams use AI today in virtually all phases of the Software Development Lifecycle (SDLC). In the 2024 SOAD study, respondents identified GenAI as an important trend in software development (40%) and viewed AI assistants for software engineering as a key industry trend (38%).

Developers today report using AI for work (64%) and are using AI to write or augment code (33%), write software tests (23%) and documentation (29%), and troubleshoot or debug problems (21%). Many development teams incorporate AI and machine learning (46%) as part of their software work today or train and deploy ML models (54%).

2024 Docker State of Application Development Report, pgs. 26, 29, 32

CONCLUSION

Software development is rapidly evolving due to changing economic conditions, technological advancements, and shifting business priorities. Delivering value quickly is expected as businesses face tighter budgets and increasing competition. Many development teams are transitioning from monolithic architectures to microservices and containers. Additionally, AI and machine learning are increasingly being integrated into development processes, while remote environments and application security are becoming critical focuses for organizations.

To succeed, software leaders must build agile, adaptable, and resilient teams. This involves leveraging essential technologies like containers and microservices to improve scalability and efficiency while fostering collaboration through tools that streamline workflows and enhance security. Continuous learning is essential for developers, as they increasingly rely on online resources and developer communities to stay current. And, by addressing inefficiencies and empowering teams to experiment, leaders can ensure their organizations remain competitive and deliver high-quality software in an environment of continuous change. In many ways, software leadership is akin to servant leadership, adept at equipping and supporting people's needs, removing roadblocks, and leveraging today's and tomorrow's technologies.

5 KEY TAKEAWAYS FOR SOFTWARE LEADERS

1. **ENGAGE AND UNDERSTAND.** software leadership is a contact, participation sport.
2. **ALIGN SOFTWARE ORGANIZATION GOALS AND METRICS** to deliver value quickly.
3. **INCREASE THE PACE OF INNOVATION** by fostering learning, experimentation and acceptable risk-taking.
4. **LEVERAGE CLOUD-NATIVE** (containers, microservices, and orchestration) and AI/ML/GenAI technologies for faster, more flexible application delivery and new services.
5. **BUILD AGILITY AND RESILIENCE** in software organizations to thrive under continuous change.

ABOUT THE AUTHOR

MITCH ASHLEY is a technology executive and entrepreneur who is an advisor, analyst, product creator and tech leader, bringing 30+ years in cybersecurity, DevOps, cloud, AI, product development, software engineering and networking. Mitch is Chief Technology Advisor with The Futurum Group and CTO of Techstrong Group's tech media platforms covering DevOps, cloud native, cloud infrastructure, cybersecurity, AI, platforms and ITSM. A highly sought-after C-level advisor, analyst, CTO, CIO and head of engineering, Mitch's analyst research is available on TechstrongResearch.com and FuturumGroup.com.