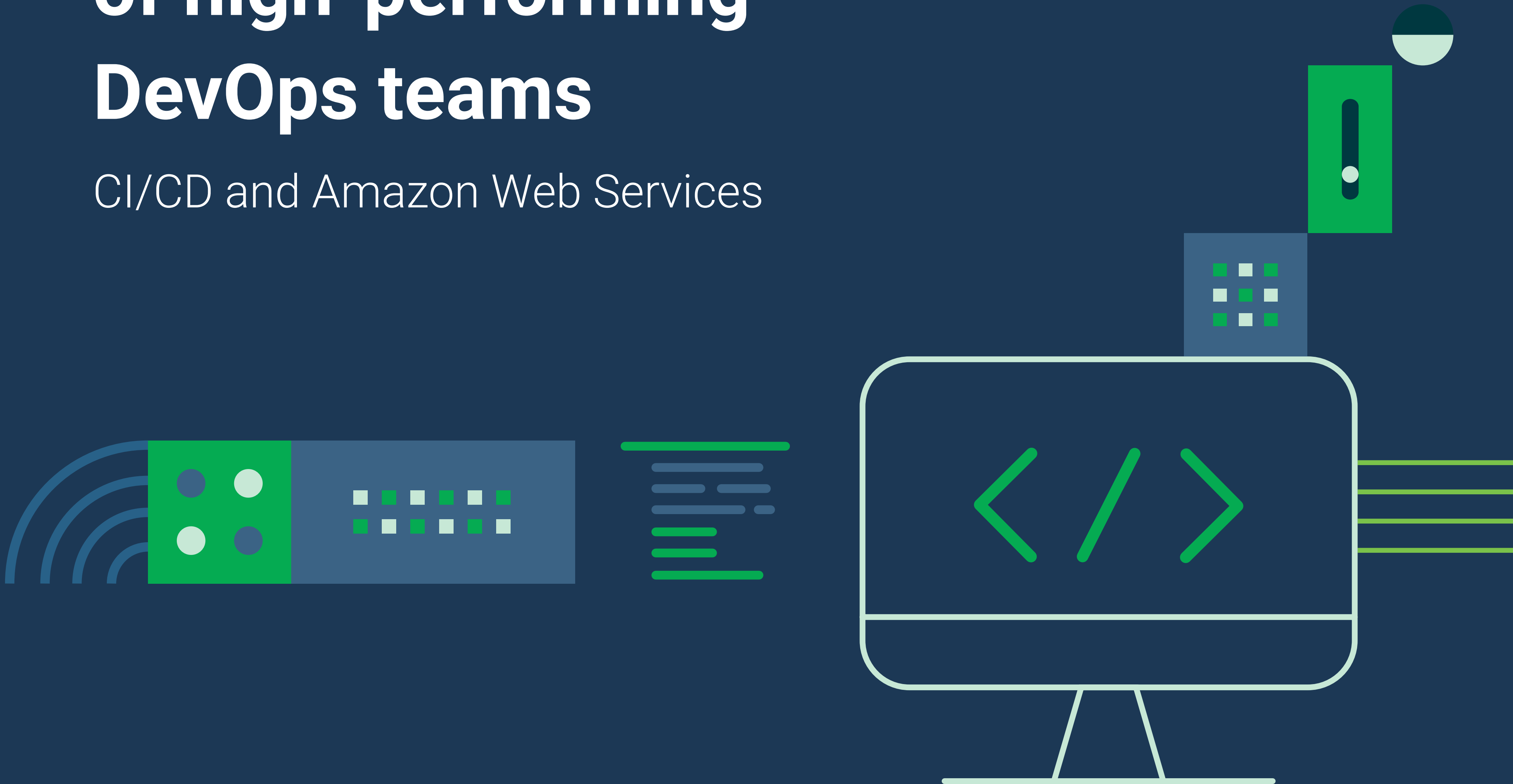
circleci

The secret weapon of high-performing DevOps teams

CI/CD and Amazon Web Services



Executive summary

It’s no secret that high-performing teams are constantly adapting to modern software development practices to stay ahead of the industry. Teams need to remain on the cutting edge, practicing techniques that enable faster development time to get to market faster, confidently. The ability to release modern applications is supported by three pillars: continuous integration, continuous deployment, and infrastructure as code. In our last ebook, we covered evaluation criteria for choosing your modern DevOps toolset, but once you’ve chosen a CI tool, what happens next?

Today, we are focusing on modern architectures and their effect on developer agility.

CHAPTER ONE

Recap of the basics

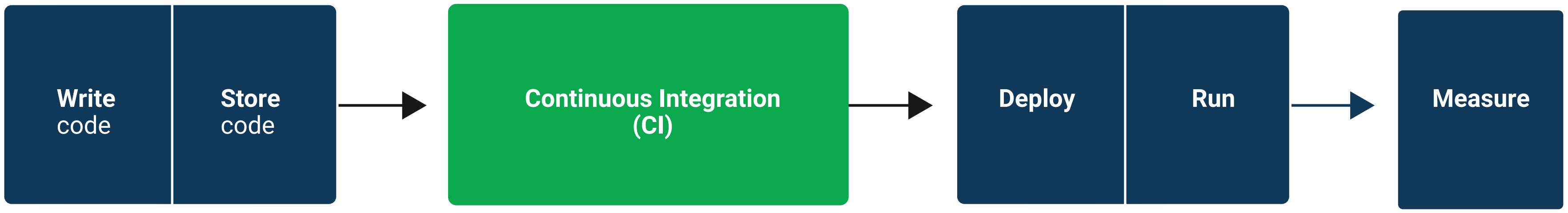
To begin, it’s important to recap the basics. What is continuous integration? Where does it sit in the development pipeline? What are some evaluation criteria teams should use when choosing a CI tool?

Continuous integration and delivery (CI/CD) is a software development strategy that increases the speed of development while ensuring the quality of the code that teams deploy.

Let’s take a look at where a CI platform would fit in the development pipeline and why it can have a dramatic impact on the productivity and efficiency of development teams.

All projects start at the source code. CI platforms pull in new and existing code from the VCS and build that code in clean containers or on virtual machines. Then, tests are run to evaluate whether to add the new code to the existing code, sending a passed/failed build status. Once the code has been built and tested, it is ready to be deployed.

CI in the development pipeline



CI/CD selection criteria

AWS provides an extensive selection of tools for software development. Although many teams use AWS services, their specific toolchain may vary greatly given their needs. As more players have entered the CI/CD market over the last few years, it's become increasingly important to determine which tool will best suit your company's needs. There are several high-level distinctions between platforms on the market. Here are our recommended evaluation criteria and questions to ask when choosing a provider:

Toolchain compatibility

- Can the platform support my VCS?
- Does the platform integrate with the other tools I use?
    - Monitoring
    - Notifications
    - Reporting
    - Security
    - Testing

Security

- Is the platform secure?
- Is the platform FedRAMP certified and SOC 2 compliant?
- Does the platform have security for my hosting method? (e.g., cloud, on-premise)
- Is there a simple way to manage secrets and environment variables?

Ease of use

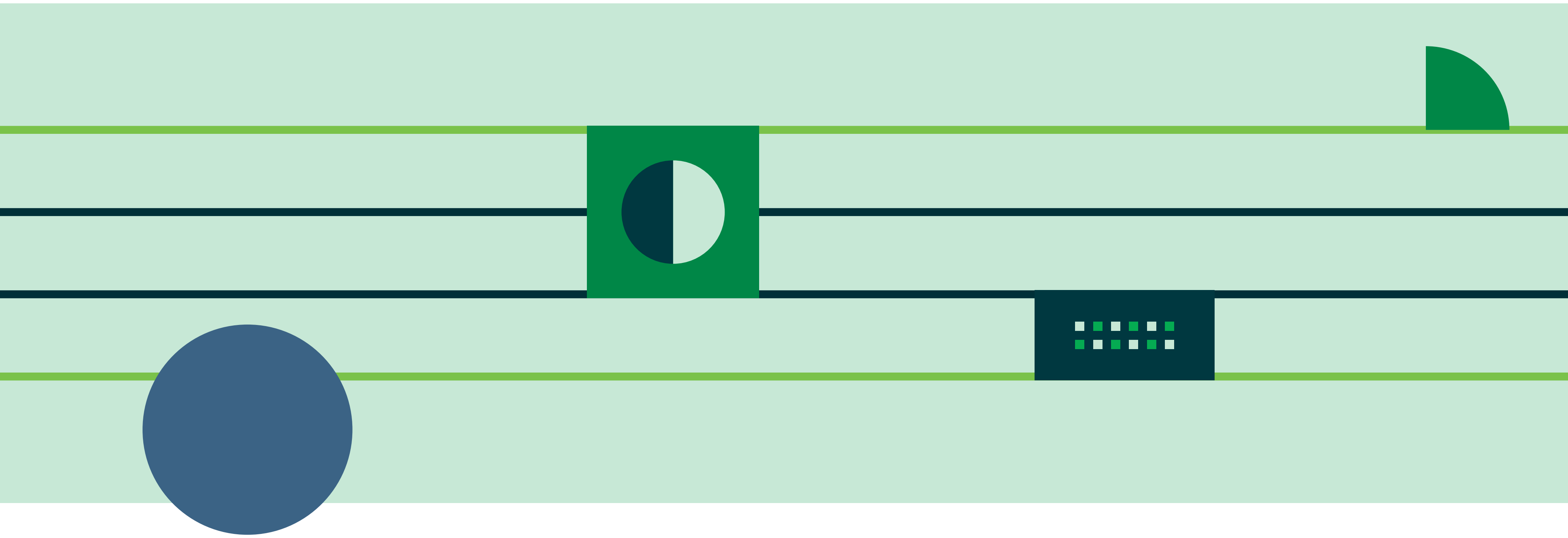
- How quickly and easily can I set the system up?
- What is the onboarding process like?
- How is the ongoing support after I've onboarded?

Community

- Are many engineers using the platform?
- Is there a community I can reach out to for support or collaboration?

Deployment

- Does the platform allow for seamless deploy to the most popular hosting solutions?
- Does the platform deploy serverless applications to your favorite cloud provider?
- Does the platform let you manage your microservices with Kubernetes? Terraform?



Migrate to modernize

Your team doesn't have to be the best at DevOps to be one of the best dev teams. You don't have to deploy a thousand times a day or spend a million dollars a month on infrastructure. Simply starting and committing to the practice of CI/CD is a reliable path to engineering success.

Once you've chosen a CI tool, what happens next? How do you optimize your usage to ensure modern development practices? What architectural patterns help you automate quicker? How do you measure the success of your team?

CHAPTER TWO

Modern Architectural Patterns

Research shows that high-performing DevOps teams continue to outperform their organizational counterparts with: 200x more frequent deployments, 24x faster recovery from failure, a 3x lower change failure rate, and 2,555x shorter lead time*.

It is important to emphasize here that speed alone is not the goal. The combination of automation, removing manual tasks from the process, and shifting testing left will enable speed and quality. Consistency of delivery is the unsung hero of automation: speed without reliable, consistent quality is not helpful.

Digital transformation and cloud migration have become buzzwords in software development. What does a cloud-native organization with CircleCI and AWS look like? What architectural patterns help teams build better code with confidence?

Microservices
Historically, applications were built with monolithic, three-tiered web architectures. Modern architectures can look a lot like a three-tier web architecture but with some very important differences.

The first major difference with this architecture is that what we see here is not the entire application — it's not a monolith, it's a single microservice.

Our customers run tens, hundreds, or even thousands of microservices, which improves their scalability and fault tolerance. The way you operate an application that releases multiple times a day is very different than one that releases 1 to 2 times per year.

Compute
The logic you write is important in differentiating your organization. We're building compute technologies that make focusing on this logic easier than ever.

CircleCI offers a wide range of machine types and class sizes. Selecting larger machines to run your workflow can reduce the time it takes for that workflow to run.

Test-splitting
Intelligent test-splitting and parallelization options allow for robust test suites to run in significantly shorter times. Because of this, increasing your testing does not cause a proportional increase in workflow duration.

Caching
Advanced caching options significantly reduce the duration of a workflow when optimized. Many packages used to build your application can be cached and reused, saving you the time involved with downloading these packages on every run. Docker layer caching, a premium feature on CircleCI, can allow for an even greater reduction in workflow duration.

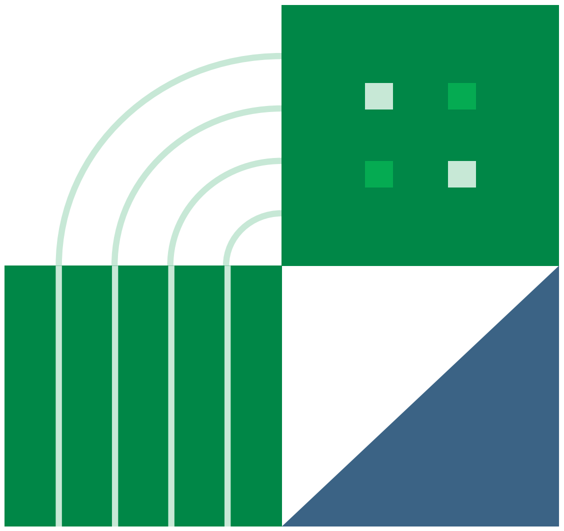
Debugging
Debugging failed builds is best when you have access to the machine where the workflow failed. CircleCI offers the ability to rerun a failed workflow and to use SSH to gain access to the machine that fails. Getting a signal fast is only one side of the CI feedback loop. The other side is the ability to quickly recover.

CHAPTER THREE

Developer Agility

Naturally, modern DevOps lends itself to developer agility. By optimizing and automating your processes, teams will now have the time to learn new skills. In this chapter, we'll cover tips for automating, abstracting, and standardizing developer agility.

Continuous integration can be viewed as an agility creator. CI puts the 'fail fast' mentality into practice: break things and then fix them quickly. Event-driven applications enable not just the decoupling of services, but also of the teams that built them. This facilitates a dramatic increase in team agility.



Orbs

CircleCI offers orbs, which are reusable packages of configuration. Abstracting layers of code from CI configuration files into open source, community-created and -validated components allows for adding and replacing services without the risk of failure. Using well-tested configuration components reduces the sources of errors, and using orbs to integrate testing suites into your CI pipeline means getting more information from your runs. We observed that recovery time for workflows decreases with increased orb usage (from 0 to 1 orb and from 1 orb to many).

Orbs make for an incredible amount of agility. Abstracting boilerplate config makes learning the ropes on a new project much easier since all of the code that is not specific to the project has been packaged in the orb.

Reusable config also helps in updating common dev paradigms across projects. If all of your projects use the same deployment (and you've encapsulated that deployment into an orb), then a decision to change deploy means updating once and then rolling all projects to the new orb version.

CircleCI has a suite of AWS integrations that allow users to easily execute pre-configured AWS operations in their CI/CD pipelines. Our out-of-the-box solutions allow users to build and test code, create and push artifacts, and deploy and update applications to their AWS account. CircleCI's AWS integrations are among our most popular orbs. As of now, they have been used more than 35 million times.

CLI

Install and configure the AWS command-line interface (awscli).

ECS

Deploy to and update Amazon Elastic Container Service (Amazon ECS)

S3

Use this set of tools for working with Amazon S3. Requirements: bash.

AWS Systems Manager Parameter Store

Load AWS Systems Manager Parameter Store keys as environment variables.

EKS

Deploy to and update Amazon Elastic Container Service for Kubernetes (Amazon EKS).

AWS-SAM-Serverless

Build, test, and deploy your AWS SAM serverless applications on CircleCI utilizing the AWS Serverless Application Model.

ECR

Build images and push them to the Amazon Elastic Container Registry (Amazon ECR).

CodeDeploy

Deploy applications to AWS CodeDeploy.

Elastic Beanstalk

Deploy and scale web applications and services via AWS Elastic Beanstalk with CircleCI.

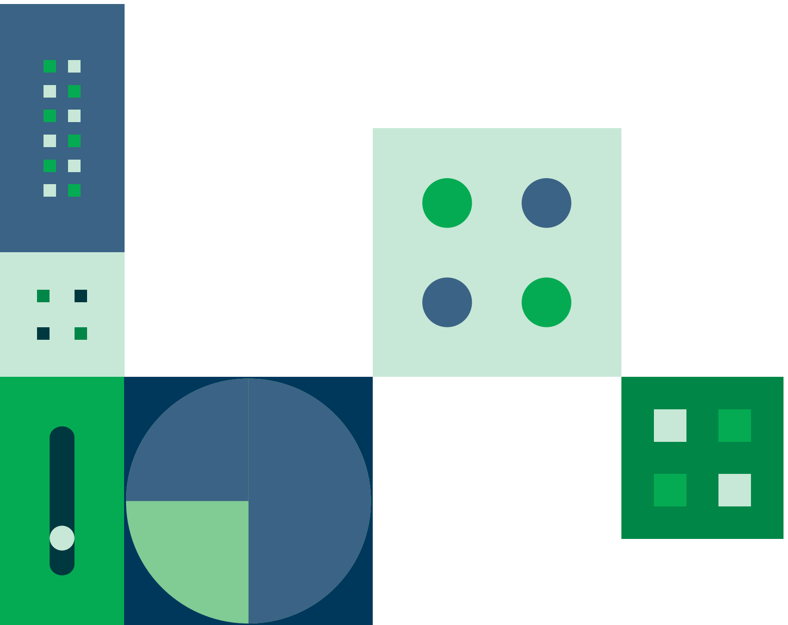
CHAPTER FOUR

Metrics that Matter

Every company is now a tech company, and many of them are already practicing continuous integration. Are they doing it well? Our data shows that you don't need to be an expert at CI to see a material increase in the metrics most important to your development teams.

Teams using CI are incredibly fast: 80% of all workflows finish in less than 10 minutes.

- Teams using CI stay in flow and keep work moving: 50% of all recovery happens in under an hour.
- 25% recover in 10 minutes.
- 50% of orgs recover in 1 try.



Our comprehensive data on engineering team performance has identified these four benchmarks:

- **Throughput:** the number of workflow runs matters less than being at a deploy-ready state most or all of the time
- **Duration:** teams want to aim for workflow durations in the range of five to ten minutes
- **Mean time to recovery:** teams should aim to recover from any failed runs by fixing or reverting in under an hour
- **Success rate:** success rates above 90% should be your standard for the default branch of an application

Optimizing these four key metrics leads to a tremendous advantage over organizations that have not yet begun to adopt continuous integration and delivery. These metrics drive digital transformation for software development and delivery, and show that it is possible to optimize for stability without sacrificing speed.

If you don't know how well your team is doing, it is impossible to set realistic targets for them. The ability to measure your engineering productivity in order to establish a baseline is absolutely necessary for staying competitive.

CHAPTER FIVE

Next Steps

circleci

Technical documentation

Review our getting started docs to get up and running on CircleCI. If you're interested in integrations, learn more about orbs or read about our GitHub and Bitbucket integrations.

Access your free trial in AWS Marketplace to explore how CircleCI can work for your team and support your AWS infrastructure requirements.

aws **Available in AWS Marketplace**