



The Continuous Integration Approach to Engineering Leadership

Lena Reinhard | @lrrrd
VP, Product Engineering, CircleCI



Change is the work

The image displays the CircleCI web interface for a specific pipeline run. The interface is divided into a dark sidebar on the left and a main content area on the right. The sidebar contains navigation links: Pipelines, Add Projects, Organization Settings, Plan, and Old Experience. At the bottom of the sidebar, it shows the status as 'OPERATIONAL' and a 'Help' link. The main content area shows the details of a pipeline run named 'build-test-and-deploy' on the 'master' branch, which has completed successfully. The run details include the duration (8m 30s / 9m ago), the commit hash (02b91a4), and the author (Metadata and labels visual tweaks & commit message on workflow page (#2239)). Below this, a workflow diagram shows the sequence of steps: dependencies (1m 20s), test (2m 41s), test-production-dock... (3m 21s), build-and-deploy-static (2m 49s), e2e (2m 25s), storybook-build (1m 36s), web-ui-orb/storybook-de... (36s), chromatic-snapshot (1m 29s), and deploy (3m 36s). Each step is marked with a green checkmark, indicating successful completion.

circleci
yvonne-ko

Pipelines

Add Projects

Organization Settings

Plan

Old Experience

Status OPERATIONAL

Help

web-ui > master > build-test-and-deploy

build-test-and-deploy SUCCESS

Rerun

Duration / Finished 8m 30s / 9m ago

Branch master

Commit 02b91a4

Author & Message Metadata and labels visual tweaks & commit message on workflow page (#2239)

dependencies 1m 20s

test 2m 41s

test-production-dock... 3m 21s

build-and-deploy-static 2m 49s

e2e 2m 25s

storybook-build 1m 36s

web-ui-orb/storybook-de... 36s

chromatic-snapshot 1m 29s

deploy 3m 36s

Case study:

How CircleCI Engineering **doubled** our team
and increased our customer facing
deliverables by 2.5x

The spirit of CI/CD:

*Software teams quickly and confidently validating and
seamlessly shipping change to end users.*

The spirit of CI/CD:

Software teams quickly and confidently validating and seamlessly shipping change to end users.

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

The spirit of CI/CD:

*Software teams quickly and **confidently** validating and seamlessly shipping change to end users.*

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

Con·fi·dence

/'kän-fə-dəns/

noun

1. a feeling or consciousness of one's powers or of reliance on one's circumstances
2. faith or belief that one will act in a right, proper, or effective way
 - i. *have confidence in a leader*

Source: [Merriam-Webster](#)

Change isn't a detractor
for confidence, it's a
prerequisite for it

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

VALUE

Accountability & ownership

Empower teams & build confidence through ownership of metrics and learning



VALUE

Accountability & ownership

Empower teams & build
confidence through
ownership of metrics
and learning

*Examples: Clear service ownership, roles & responsibilities,
E2E ownership, SLIs/SLOs, business KPIs*

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

VALUE

Leadership at all levels

Invest in growing
confidence in technical-
and people leaders
around you



VALUE

Leadership at all levels

Invest in growing confidence in technical- and people leaders around you

Examples:
Lead through context, delegate outcome-focussed, use growth frameworks

Key area (5)	Attribute / value (15)	Theme (27) (Competency)	Competency			
Engineering Competency Matrix Guidelines & FAQ			E1	E2	E3	E4
			Associate Engineer	Engineer	Senior Engineer	Staff Engineer
			Focus on writing of code	Focus on execution of work	Focus on team	Focus on team & with team's business
			Writing code	Writes code with testability, readability, edge cases, and errors in mind.	Consistently writes functions that are easily testable, easily understood by other developers, and accounts for edge cases and errors. Uses abstractions effectively.	Consistently writes production-ready code that is easily testable, easily understood by other developers, and accounts for edge cases and errors. Understands when it is appropriate to leave comments, but focuses towards self-documenting code.
			Quality & testing	Knows the testing pyramid. Writes unit tests, sometimes with help from more senior engineers.	Understands the testing pyramid, writes unit tests in accordance with it, as well as higher level tests with help from more senior engineers. Always tests expected edge cases and errors as well as the happy path.	Understands the testing pyramid, and writes unit tests as well as higher level tests in accordance with it. Always writes tests to handle expected edge cases and errors gracefully, as well as happy paths.
			Debugging	Understands the basics of debugging and the tools used for it.	Uses a systematic approach to debug issues located within a single service.	Proficient at using systematic debugging to diagnose all issues located to a single service. Uses systematic debugging to diagnose cross-service issues, sometimes with help from more senior engineers.
			Debugging & observability	Is aware of the organization's monitoring philosophy and the operational data for their team's domain.	Is aware of the organization's monitoring philosophy. Helps tune and change the monitoring on their team accordingly. Is aware of the operational data for their team's domain and uses it as a basis for suggesting stability and performance improvements.	Is aware of the organization's monitoring philosophy. Helps tune and change the monitoring on their team accordingly. Is aware of the operational data for their team's domain and uses it as a basis for suggesting stability and performance improvements.
			Observability	Is able to gain context within team's domain with help from more senior engineers.	Understands a portion of the team's domain, can gain sufficient context to work productively in that portion.	Understands their team's domain at a high level and can gather sufficient context to work productively within it. Has expertise in a portion of their team's domain.
			Understanding code	Is able to gain context within team's domain with help from more senior engineers.	Understands a portion of the team's domain, can gain sufficient context to work productively in that portion.	Understands their team's domain at a high level and can gather sufficient context to work productively within it. Has expertise in a portion of their team's domain.
			Software design architecture	Is aware of overall service architecture. Designs basic functions with an awareness of overall service architecture, avoiding duplication across codebases and interface-breaking changes.	Designs functions that are aligned with the overall service architecture.	Consistently designs code that is aligned with the overall service architecture. Utilizes abstractions and code isolation effectively.
			Security	Understands the importance of security.	Understands the importance of security. Utilizes this knowledge to ask more senior engineers for help on making decisions that may have security implications.	Approaches all engineering work with a security lens. Actively looks for security vulnerabilities both in the code and when providing peer reviews.

VALUE

Leadership: shared goals

Highly connected &
visible goals create a
clear line of impact for
everyone



VALUE

Leadership: shared goals

Highly connected &
visible goals create a
clear line of impact for
everyone

*Examples: Regular strategy sharing, OKRs (quarterly),
individual goals aligned*

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

VALUE

Visibility: Defining metrics that work

Utilise key metrics to
measure incremental
impact

Measure what matters.

Learn quickly.

Focus on impact.

Use a good baseline.



VALUE

Visibility: Defining metrics that work

Utilise key metrics to
measure incremental
impact

Measure what matters.

Learn quickly.

Focus on impact.

Use a good baseline.



VALUE

Visibility: Metrics used by successful teams

Defining metrics that work
for *your* team & making
them visible



CI/CD benchmarks for high performance

	Median CircleCI Developer	Suggested Benchmark
Throughput The average number of workflow runs per day	0.7 times/day	Merge on any pull request
Duration The average length of time for a workflow to run	< 4 minutes	5 - 10 minutes *
Mean time to recovery The average time between failures and their next success	< 56 minutes	Under 1 hour
Success rate The number of successful runs divided by the total number of runs over a period of time	80% for default branch	90% or better on the default branch

* Whether or not this length makes sense for your project depends entirely on what your workflow is accomplishing. Knowing your baseline Duration and striving to improve it is more important than hitting an arbitrary number.

High-performing IT organizations
report experiencing:



Deployment frequency

200x more frequent
deployments



Mean time to recovery

24x faster recovery from
failure



Success rate

3x lower change fail rate



Workflow duration

2,555x shorter lead time

**High-performing organizations
are decisively outperforming
their lower-performing peers
in terms of throughput**

VALUE

Visibility: Metrics used by successful teams

Defining metrics that work
for *your* team & making
them visible

Engineering quality: CI/CD benchmarks, availability

Delivery metrics: lead time, cycle time, investments

Goal progress: OKR status & progress, projection & confidence

Hiring metrics: time to \$stage, pass-through rates

V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

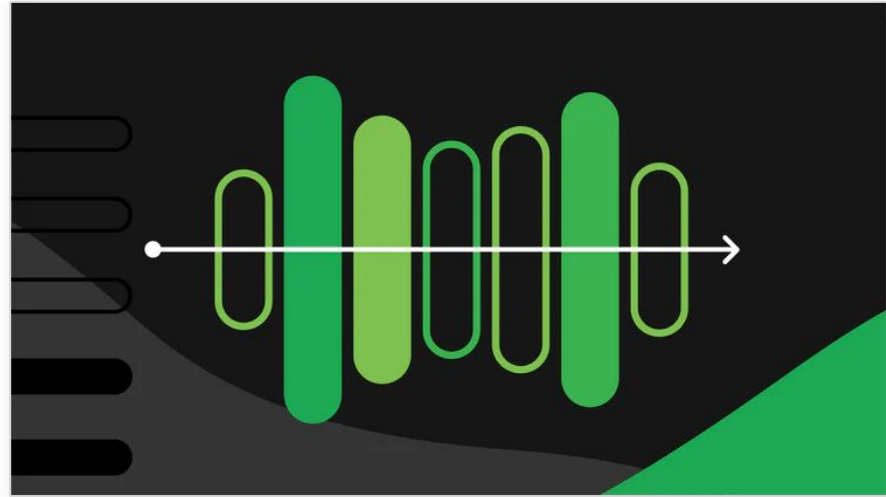
E

Enhancement

VALUE

User value delivery: external users

Confident teams drive incremental changes for continuous value delivery



V

Visibility

A

Accountability

L

Leadership

U

User value
delivery

E

Enhancement

VALUE

Enhancement: the real failure is not learning from failure

Confident teams learn continuously and further increase their impact



VALUE

Enhancement: the real failure is not learning from failure

Confident teams learn continuously and further increase their impact

Examples: Incident reviews & retrospectives, training, ensuring right visibility & accountability & learning, are we continuously improving?



The Continuous Integration approach to Engineering Leadership: Confidently & continuously delivering

V

Visibility

A

Accountability

L


Leadership

U

User value delivery

E

Enhancement



Join us, we're hiring!

circleci.com/careers

Learn more about software delivery metrics

[CircleCI's State of Software Delivery Report](#)



Thank you.

Lena Reinhard, VP Product Engineering, CircleCI