Engineer

Change lead time

⧗ elapsed time from commit to deployment in production

What it is

Change lead time is the elapsed time from a change being committed to version control to that change being deployed in production. DORA uses this interval to describe software delivery throughput. It measures both work and waiting after the start event. If a change is committed on Monday morning and its first successful production deployment finishes on Tuesday afternoon, the elapsed interval includes the night and any time spent waiting for review or release.

The interval starts with a recorded commit. Lead time from a customer request, time spent writing code and time until a feature is enabled for users answer different questions. If code is deployed while its feature flag remains disabled, the code may already have a production deployment timestamp. The later exposure can be recorded separately. Define what your organization means by successful production deployment, especially if production has regional waves or several distinct targets.

A chart value is meaningful when its population is clear. Choose a service or application, define the production targets for it and say which source changes enter the calculation. Attribute each change to the first qualifying production deployment that includes it. Some histories record a squash or merge instead of the original commits. A metric based on the merge commit uses a different start and leaves the earlier review time outside the interval. Document the choice so a reader can understand what is being compared.

Use the concept in a decision

Put “What is change lead time?” to work in a real decision. We can help relate the concept to the system you are building or buying. Tested, released and reversible, so changes reach customers on a Tuesday afternoon rather than late on a Friday night.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Automated GIS SaaS deployment with GitHub Actions — deployment, data and reporting, all off the hands

Automated GIS SaaS deployment with GitHub Actions Get in touch →

Why it matters

Waiting is often easier to miss than work. If compilation takes six minutes and the accepted change waits two days for a deployment slot, a faster compiler changes only a small part of the measured interval. Record the handoffs that matter, such as the end of review, artifact creation, staging checks, approval and production. A simple timeline of a few changes can reveal a queue before an elaborate reporting tool is needed.

Use the measure to ask about the delivery system shared by a team. A small production fix and a change that crosses a platform boundary have different constraints. Comparing individual engineers may reward smaller tasks or encourage them to commit late. Comparing a service with its own earlier history after an explicit process change helps the team interpret what moved. Look at slow cases as well as typical ones and pair the finding with evidence about delivery stability.

Separate the metric from the objective. A team may decide that reducing wait at an approval step is worthwhile because a released fix can help customers sooner. It still needs evidence that the faster process catches defects and handles failures. DORA reports deployment recovery and failure-related measures alongside throughput. Changes in several measurements together can help determine whether a shorter path became safer, riskier or merely different.

What it looks like in practice

The case on automating GIS software deployment documents a manual test, build and deploy cycle that moved into GitHub Actions. It reports a shorter release cycle and repeatable configuration. Its public description does not publish commit-level times. A team building on that pattern can measure change lead time by connecting version-control records to deployment events.

Consider a service with two commits, one at 09:00 and the other at 10:30. A build containing both is successfully deployed to production at 14:00. The elapsed values are five hours and three and a half hours. If the same build goes to staging

at noon, the staging event helps explain the path, while the production event ends the interval. If the build fails earlier, keep that event for diagnosis. The measure ends when the change reaches the qualifying production state.

Start with records a human can verify. For a few deployments, retain the source revision range or manifest, artifact identifier, environment and completion time. Reconcile those records against the commit history. For a monorepo, record which service each deployment affected. Then add each measurement to a chart with its service and convention visible. Recheck a sample by hand after changing jobs, migrating tooling or introducing a new release strategy.

Where it goes wrong

Deployment records sometimes describe a job finishing rather than a service being available. A job can finish after submitting work to another system, before the new instances are healthy. Establish whether the end event is the accepted production deployment, verified replacement at a defined proportion of traffic or another deliberate convention. Make the convention consistent for the same service and keep transitions visible when it changes.

A report can miss the work that never reached production. An abandoned change has no production endpoint, and an active change can still be aging. Keep the count and current age of unreleased work separately so improved numbers cannot come only from leaving difficult changes out. Investigate missing links between artifacts and commits. Exclude unknown measurements explicitly, with a count, instead of treating them as zero.

A single average can hide long queues. Show a typical result, a slow-end percentile and the number of observations. For a small sample, review the actual timelines rather than reading a precise-looking percentile as a stable fact. If one long wait accounts for most of the change, inspect the stage responsible and test an improvement there. Follow the same service and the same measurement definition after the change.

Further reading

- DORA’s guide to software delivery performance metrics defines change lead time and its place among the delivery measures.
- DORA’s value stream mapping guide explains how to trace work and waiting through handoffs.

- Git documentation for commit formats distinguishes author and committer date fields available for source records.
- DORA's deployment automation capability describes how to make deployment events repeatable.

Glossary

Choose for your actual constraints

Bring the deployment path, the operational risks and the people on call. We can scope a change with checks, a recovery route and clear ownership.

See what it costs Get in touch →

Make the terminology practical

Ask how "What is change lead time?" applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/change-lead-time/