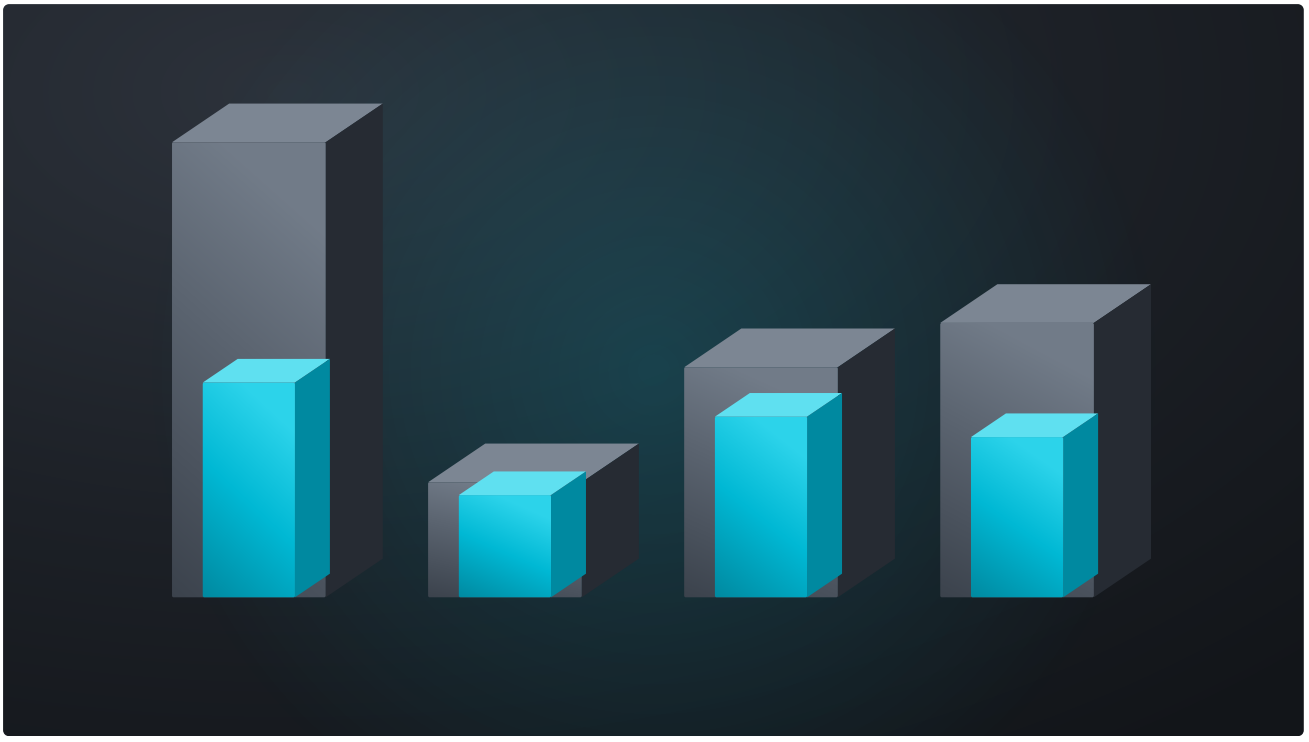
Black Shard

RESEARCH REPORT

How a small firm ships: deploy and release runs, May to August 2026

A buyer of engineering work wants to know how often a firm puts change in front of its users, how much of that is routine, and what the firm counts as a release. Four months of our own run record, counted by a rule anyone can apply to theirs.



Entity	Black Shard Pty Ltd · ABN 66 696 910 773
Document ID	BS-RPT-13
Classification	Public
Version	1.1
First published	8 September 2026
This edition	29 October 2026
Figures	Runs of 1 May to 31 Aug 2026, read 8 Sep 2026

Executive summary

Our deploy and release workflows ran 1,261 times from 1 May to 31 August 2026, and 772 of those runs succeeded.

May had the most successful runs of the four months, 252; June had 120, July 212 and August 188.

A deploy count answers the question a buyer of engineering work should ask first: whether the firm ships as a habit or as an event. A firm that releases a few times a year treats each release as a project, with a freeze before it and a scramble after it. A firm that releases many times a week has made release the ordinary end of a change, which is what makes small changes safe to make and faults quick to reverse.

Of the 772 successful runs, 771 started from a push to a branch or from a person dispatching the workflow by hand: 628 on a push and 143 on a dispatch. A deploy that starts from a push is a merge going live on its own, with the review and the tests already behind it. That is the property the count is for.

Runs that did not succeed stay in the total. Of the 489 runs that did not succeed, 288 failed, 146 were cancelled when a later push replaced them and 55 were skipped by their own conditions. A failed run is a deploy that stopped itself before the change reached a user, and the count stays in the total because a buyer should be able to look at it.

- Count your own deploys by a written rule, from the run record, and keep the runs that did not succeed in the count.
- Make the push the trigger for every ordinary release, and be able to say which releases still need a person to start them.
- Ask every supplier that builds or hosts software for you for the same count, by the same rule.

Method

The count comes from the run record every modern build service keeps: one row per workflow run, with the name of the workflow, what started it, when it started and how it ended. We read it for the four full months from 1 May to 31 August 2026 and applied one rule to every row.

A run counts when the workflow behind it is named for deploy, release, ship or publish and for none of check, status or preview, and it counts as successful when the build service recorded its conclusion as success; runs in forks and mirrors of other people’s repositories are left out.

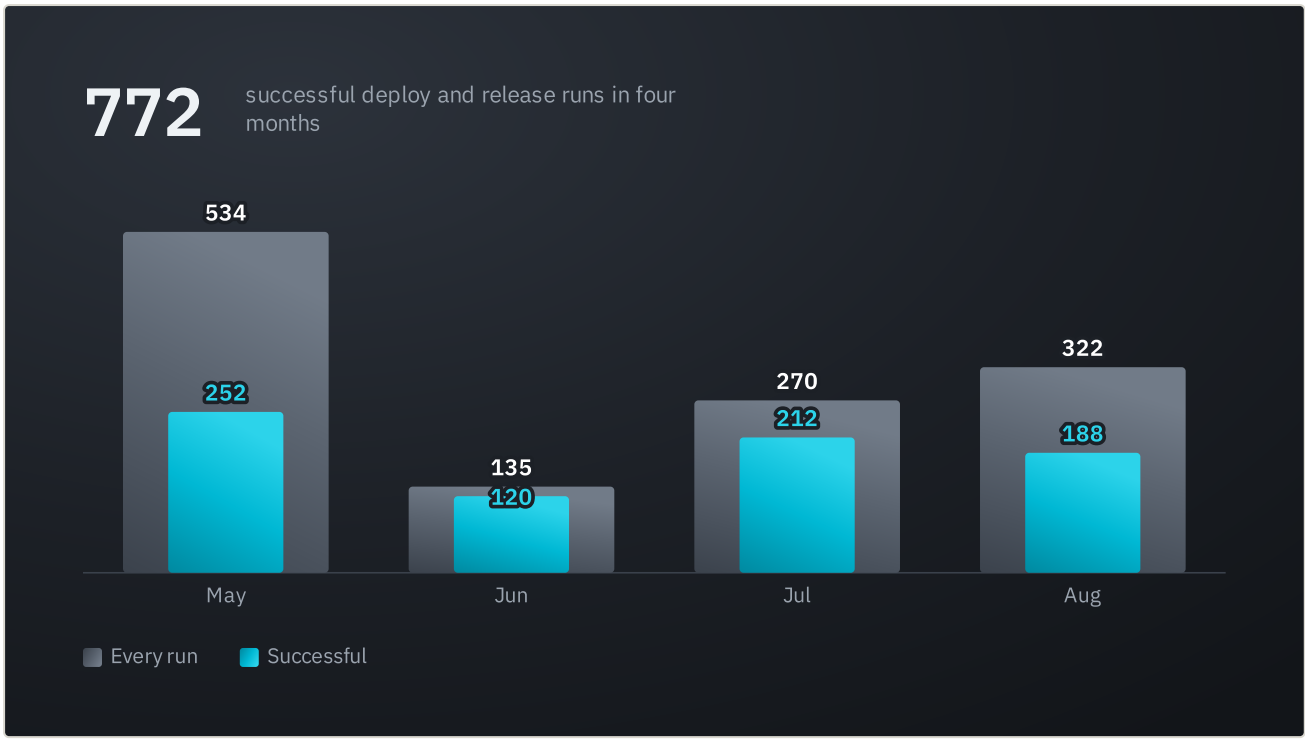
A run the service raised on its own, with no workflow file behind it, is left out. Runs that failed, runs cancelled because a later push replaced them, and runs skipped by their own conditions stay in the total and nowhere else. Months are calendar months in Coordinated Universal Time, the clock the record keeps, and a run belongs to the month it started.

The rule is narrow on purpose. A build that compiles and tests a change is work, and a status check on a pull request is work, but neither puts anything in front of a user. Counting them would double the figure and tell a buyer nothing about how often the product changes. The rule also hides nothing that failed: a count of successes with the attempts left out is a figure a buyer should distrust.

The same rule runs on any run record. Export the runs for four full months, keep the rows whose workflow name passes the word test, drop the forks, bucket by the month each run started, and count the total and the successes in each bucket.

Four months, month by month

DEPLOY AND RELEASE RUNS BY MONTH, MAY TO AUGUST 2026



Every month of the four carried more than a hundred successful runs; May carried the most. Source: Our deploy and release workflow run record, 1 May to 31 August 2026.

May had the most successful runs of the four months, 252; June had 120, July 212 and August 188.

Counting every run, successful or not, May had 534, June 135, July 270 and August 322.

The gap between runs and successes holds failed, cancelled and skipped runs, in that order of size. Of the 489 runs that did not succeed, 288 failed, 146 were cancelled when a later push replaced them and 55 were skipped by their own conditions. A failed run is a deploy that stopped itself: a test or a smoke check on the path found a fault and the change never reached a user. The fix is a commit and the re-run is minutes away, so the record keeps the failure and the firm pays for it in minutes. In a firm that releases quarterly the same failure costs a week. A cancelled run is a push the next push overtook before its deploy finished, and a skipped run is a release job that found nothing to release. All three stay visible in the count, because a supplier that hides them is the one to worry about.

A month with fewer runs and a narrow gap, as June was, is a month of fewer and steadier changes. A month with many runs and a wide gap, as May was, is a month of fast iteration on something new. Both are healthy. The month to worry about is the one with no runs at all, because it means the product stopped changing, or the changes went out by some path the record does not see.

What starts a deploy

Of the 772 successful runs, 771 started from a push to a branch or from a person dispatching the workflow by hand: 628 on a push and 143 on a dispatch.

A push-triggered deploy is a merge that goes live on its own. The review happened before the merge, the tests ran on the branch, and the deploy is the ordinary consequence of the change landing. Nobody decides to release;

releasing is what a merge does. That is the property a buyer is looking for, because it means every change is small enough to go out alone and the path from a fix to production is measured in minutes.

A dispatched deploy is a person pressing a button: a rollback, a re-run after an infrastructure change, a release to a client environment on a date the client set. There are cases that need a hand on the button, and a firm should be able to say what they are. What it should never have is a release that only one person knows how to start.

The one remaining run started when another workflow finished, a chain in which one job hands to the next. Chains are useful and easy to lose track of, so the count names them.

What a buyer reads from the count

A firm buying custom software, or a platform a supplier hosts for it, is buying the supplier's habit of change along with the code. Three questions are worth asking before the contract is signed: how often the supplier ships to production, what it counts as a ship, and what happens when one fails.

The first answer should be a number from a run record, by a rule the supplier can state. The second should leave out builds and checks. The third should be a re-run measured in minutes, with a rollback that is itself a deploy of the previous version. A supplier that cannot answer from its own run record is a supplier whose releases are events, and events are where outages come from.

The same count is a security figure. A firm that deploys most days can ship a patch the day it is published, because shipping is routine. A firm that deploys quarterly has a patch window as long as its release cycle, whatever its policy says. A patch window measured in days is only within reach of a firm whose release path is measured in minutes.

The count also says something about the firm's security posture that a questionnaire cannot. A deploy that starts from a push runs through the same gates every time: the review, the tests, the secret scan, the dependency check. A release a person assembles by hand on the day runs through whichever of those the person remembers.

Count your own

The rule, as a worksheet for the run record your build service keeps. Four full calendar months is enough to show the pattern, and recent enough to describe the team as it is now.

FIELD IN THE RUN RECORD	KEEP THE RUN WHEN	WHY
Workflow name or file	It says deploy, release, ship or publish, and none of check, status or preview	Only a run that puts something in front of a user counts
Trigger	A push, a manual dispatch or a workflow chain; never the service's own housekeeping	The trigger says whether release is a habit or a button
Repository	The repositories you own; never a fork or a mirror of someone else's	A fork's runs are someone else's cadence
Conclusion	Count every run in the total, and the successes separately	The gap between the two is the cost of shipping often, and it should be visible

FIELD IN THE RUN RECORD	KEEP THE RUN WHEN	WHY
Start time	Bucket by the calendar month the run started, in one time zone	A month with no runs is the finding

The rule as a worksheet for your run record.

Then read the result as a buyer would read it of you. Where releases start from a push, the path is a habit. Where most start by hand, find out why, because the reason is usually a step nobody has automated or a fear nobody has written down. Where a month is empty, find the path the changes took instead.

What to do

1. Write the rule down: which workflow names count as a release, which triggers count, and which repositories are yours.
2. Count the last four full months from the run record, every run and the successes, and bucket them by month.
3. Make the push the trigger for every ordinary release, and list the cases that still need a hand on the button.
4. Put a smoke test at the end of every deploy so a failed release stops itself, and make a rollback a deploy of the previous version.
5. Keep the runs that did not succeed in the count, and look hard at any month where the gap widens.
6. Put the same gates in the path of every push: review, tests, a secret scan and a dependency check, so a release by push is also the most checked release.
7. Ask every supplier that builds or hosts software for you for the same count, by the same rule, and read the months with the fewest runs first.

We build and run platforms for Australian organisations on this cadence: small changes, reviewed before the merge, tested on the branch and deployed by the push, with security designed in from the first commit. An engineering engagement with us starts with the same count run on your repositories and ends with a release path your team runs on its own. Scope and price are at blackshard.com.au/services/engineering, and the secure build offer at blackshard.com.au/solutions/secure-build.

Our record

Black Shard builds and defends the systems Australian organisations run on. Every figure here is counted from the run record of our own platforms, and that record is held.

The record is as held at 29 October 2026, the date this edition was revised.

- SMB1001:2026 Gold, issued by CyberCert, valid 16 Jun 2026 to 17 Jun 2027.
- Essential Eight Maturity Level 3, assessed September 2026.
- Prequalified supplier to Queensland councils under Local Buy arrangement LB308, ICT Solutions, Products, Services and New Technologies.
- Prequalified supplier under the NSW Government ICT Services Scheme (SCM0020).
- Holder of Victorian Fair Jobs Code pre-assessment certificate FJC-261023-11119, required for Victorian Government threshold and high-value procurement.

- Our CISO holds the Exemplar Global ISO/IEC 27001:2022 Lead Auditor credential.
- Professional indemnity, public liability, cyber and workers' compensation insurance, all current.
- More than 65,000 tests across our repositories, counted on 26 October 2026.
- More than 1,500 deploys and releases shipped since 1 January 2026.

> 24/7 incident line: 1800 370 270.

Sources

- Our deploy and release workflow run record, every run from 1 May to 31 August 2026, counted by the rule in the method
- The running tally of our scale figures behind the record section, measured 26 October 2026 at 00:45 AEST, read at the 29 October 2026 edition