The Deployment Pre-Flight Pack

Your deploy said "Succeeded" — then the app broke. These are the 28 dependencies that pass validation and fail in production, each with the pre-flight check that catches it. Plus ready package.xml + destructiveChanges templates and a printable checklist.

Missing dependencies | FLS & permissions | Metadata vs data | destructiveChanges | Validate + Quick Deploy

THE 28 SILENT DEPLOYMENT KILLERS

01	Apex references a field or object not in the package	02	Permission Set / Profile grants FLS on a field not deployed
03	Lightning record page set as org default	04	Picklist value referenced but not deployed
05	Record Type references values missing in the target	06	LWC / Aura references an undeployed controller, label, or resource
07	Flow references an undeployed Apex, subflow, or field	08	Report / Dashboard references a missing Report Type or field
09	Email alert / template references a missing merge field or letterhead	10	Custom Setting DATA (the records)
11	Reference / seed data and queue members	12	Named Credential & Connected App secrets
13	Remote Site / CSP Trusted Site for a new callout	14	Custom Metadata records vs Custom Setting data
15	Deploying user lacks Modify All Data / Author Apex	16	FLS not granted — the field is invisible
17	Object / tab / app visibility missing	18	Sharing rules / OWD mismatch between orgs
19	Org-wide Apex coverage below 75%	20	Failing or SeeAllData tests
21	Wrong deployment order	22	Flow deployed but never activated
23	Managed package version mismatch	24	API version mismatch
25	Deleting something still referenced	26	No Validate-Only run first
27	No rollback package	28	Post-deploy steps forgotten

"Succeeded" is not "done." A Salesforce deploy can pass, tests can go green, and the feature can still be dead on arrival — because a field is invisible, a flow never activated, a custom setting is empty, or a secret didn't travel. Change Sets barely track dependencies, and a deploy that worked in sandbox routinely fails in prod. Here are the 28 traps and the pre-flight check for each.

REFERENCES

01 Apex references a field or object not in the package

REFERENCES

SILENTLY FAILS A class or trigger uses a field/object you didn't include → "Missing dependency".

PRE-FLIGHT Deploy every field, object, and class the code touches. Let a dependency analyzer build the manifest.

02 Permission Set / Profile grants FLS on a field not deployed

REFERENCES

SILENTLY FAILS The perm set references a field that isn't in the package.

PRE-FLIGHT Deploy the field and the permission set together — never the access without the field.

03 Lightning record page set as org default

REFERENCES

SILENTLY FAILS A FlexiPage default assignment needs supporting components or it fails.

PRE-FLIGHT Bundle the Object, Record Type, FlexiPage, AND Profile to deploy the assignment.

04 Picklist value referenced but not deployed

REFERENCES

SILENTLY FAILS One value is used by a flow, validation rule, or record type that ships without it.

PRE-FLIGHT Deploy the picklist value / StandardValueSet before the things that reference it.

05 Record Type references values missing in the target

REFERENCES

SILENTLY FAILS The RT points at picklist values the target org doesn't have yet.

PRE-FLIGHT Deploy the field and its picklist values before the Record Type.

06 LWC / Aura references an undeployed controller, label, or resource

REFERENCES

SILENTLY FAILS The component ships without its Apex controller, custom label, or static resource.

PRE-FLIGHT Bundle the controller, labels, and static resources with the component.

07 Flow references an undeployed Apex, subflow, or field

REFERENCES

SILENTLY FAILS The flow calls an invocable, subflow, template, or field that isn't in the package.

PRE-FLIGHT Include every referenced component — and the flow's own active version.

08 Report / Dashboard references a missing Report Type or field

REFERENCES

SILENTLY FAILS The report depends on a custom report type or field not in the deploy.

PRE-FLIGHT Include the report type and every referenced field.

09 Email alert / template references a missing merge field or letterhead

REFERENCES

SILENTLY FAILS The template references a field or letterhead the target lacks.

PRE-FLIGHT Include the letterhead and referenced fields with the template.

NOT CARRIED

10 Custom Setting DATA (the records)

NOT CARRIED

SILENTLY FAILS Metadata deploys the Custom Setting object — NOT the rows inside it. Apex then reads empty settings.

PRE-FLIGHT Seed the data post-deploy via a script or Data Loader. Metadata ≠ data.

11 Reference / seed data and queue members

NOT CARRIED

SILENTLY FAILS Sample records, reference data, and queue membership aren't metadata.

PRE-FLIGHT Load them post-deploy as a scripted step in your runbook.

12 Named Credential & Connected App secrets

NOT CARRIED

SILENTLY FAILS The credential/app deploys; the secret, key, and token do not.

PRE-FLIGHT Re-enter secrets and reconfigure OAuth in the target after deploy.

13 Remote Site / CSP Trusted Site for a new callout

NOT CARRIED

SILENTLY FAILS Without it, the code deploys but the callout fails at runtime.

PRE-FLIGHT Include Remote Site Settings / CSP Trusted Sites for any new endpoint.

14 Custom Metadata records vs Custom Setting data

NOT CARRIED

SILENTLY FAILS CMDT records ARE deployable metadata; Custom Setting rows are not — teams confuse the two.

PRE-FLIGHT Deploy CMDT records in the package; seed Custom Setting data separately.

PERMISSIONS

15 Deploying user lacks Modify All Data / Author Apex

PERMISSIONS

SILENTLY FAILS The deploy fails on permission-gated components.

PRE-FLIGHT Grant Modify All Data and Author Apex to the deploying user (or use a CI user that has them).

16 FLS not granted — the field is invisible

PERMISSIONS

SILENTLY FAILS The field deploys but no one can see it. The app 'works' and shows nothing.

PRE-FLIGHT Deploy field-level security via permission sets alongside the field.

17 Object / tab / app visibility missing

PERMISSIONS

SILENTLY FAILS Users can't reach the new object or tab on their profile.

PRE-FLIGHT Include object, tab, and app access in a permission set.

18 Sharing rules / OWD mismatch between orgs

PERMISSIONS

SILENTLY FAILS Records exist but are invisible because sharing differs from sandbox.

PRE-FLIGHT Align OWD and sharing rules as part of the release.

TESTS & ORDER

19 Org-wide Apex coverage below 75%

TESTS & ORDER

SILENTLY FAILS Production blocks the deploy if total coverage is under 75%.

PRE-FLIGHT Check org-wide coverage before you deploy — not just the classes you changed.

20 Failing or SeeAllData tests

TESTS & ORDER

SILENTLY FAILS Tests that pass on sandbox data fail in production.

PRE-FLIGHT Fix failing tests; never rely on SeeAllData(true). Create your own test data.

21 Wrong deployment order

TESTS & ORDER

SILENTLY FAILS A validation rule / perm set is deployed before the field it references.

PRE-FLIGHT Sequence the deploy, or combine dependent components into one package.

22 Flow deployed but never activated

TESTS & ORDER

SILENTLY FAILS The new flow lands inactive and the OLD version keeps running.

PRE-FLIGHT Set the flow status to Active in the deploy, or activate it as a post-step.

23 Managed package version mismatch

TESTS & ORDER

SILENTLY FAILS The target org doesn't have the package (or version) the components need.

PRE-FLIGHT Install / upgrade the managed package in the target BEFORE deploying.

24 API version mismatch

TESTS & ORDER

SILENTLY FAILS Components built on a newer API version than the target expects.

PRE-FLIGHT Align component API versions with the target org's supported version.

DELETES & ROLLBACK

25 Deleting something still referenced

DELETES & ROLLBACK

SILENTLY FAILS A destructive change removes a field/object still used by a flow, layout, or report.

PRE-FLIGHT Remove every reference first, then run destructiveChanges.

26 No Validate-Only run first

DELETES & ROLLBACK

SILENTLY FAILS You deploy straight to prod and discover the failure live.

PRE-FLIGHT Run a Validate-Only (check-only) deploy with tests, then Quick Deploy the validated build.

27 No rollback package

DELETES & ROLLBACK

SILENTLY FAILS A partial deploy leaves the org in a broken half-state with no way back.

PRE-FLIGHT Keep a rollback package of the previous version before you deploy.

28 Post-deploy steps forgotten

DELETES & ROLLBACK

SILENTLY FAILS Activate flows, seed settings, re-enter secrets, assign perm sets, schedule jobs — all skipped.

PRE-FLIGHT Write and run a post-deploy runbook every time. The deploy isn't done when it says 'Succeeded'.

TEMPLATES — COPY, ADAPT, DEPLOY

package.xml (manifest)

```
<?xml version="1.0" encoding="UTF-8"?>
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.My_Field__c</members>
        <name>CustomField</name>
    </types>
    <types>
        <members>My_Feature_Access</members>
```

```
        <name>PermissionSet</name>
    </types>
    <types>
        <members>Opportunity_Screen_Flow</members>
        <name>Flow</name>
    </types>
    <types>
        <members>My_Handler</members>
        <members>My_HandlerTest</members>
        <name>ApexClass</name>
    </types>
    <version>64.0</version>
</Package>
```

destructiveChanges — delete safely

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- destructiveChangesPost.xml : deleted AFTER the deploy succeeds -->
<Package xmlns="http://soap.sforce.com/2006/04/metadata">
    <types>
        <members>Account.Legacy_Field__c</members>
        <name>CustomField</name>
    </types>
    <version>64.0</version>
</Package>
<!-- Pair with a valid package.xml (empty <types> is fine). Use
     destructiveChangesPre.xml to delete BEFORE the deploy instead. -->
```

Validate-Only → Quick Deploy (sf CLI)

Never deploy straight to prod. Validate first (runs tests, saves nothing); if it passes, Quick Deploy the exact validated build.

```
# 1) Validate only (check-only) with tests - NOTHING is saved yet
sf project deploy validate \
   --manifest manifest/package.xml \
   --test-level RunLocalTests \
   --target-org prod

# 2) If validation passes, Quick Deploy it - skips re-running the tests
sf project deploy quick --job-id <VALIDATED_JOB_ID> --target-org prod

# Deletes: pass the destructive manifests alongside the deploy
sf project deploy start --manifest package.xml \
   --post-destructive-changes destructiveChangesPost.xml --target-org prod
```

Run this before every production deploy

Any unchecked box is a post-deploy incident waiting to happen.

- ☐ Every referenced field, object, and class is in the package
- ☐ FLS + object/tab/app access included via permission sets
- ☐ Picklist values & record types deploy before what references them
- ☐ Lightning page defaults include Object + RecordType + FlexiPage + Profile
- ☐ Custom Setting data seeded (metadata carries the object, not the rows)
- ☐ Named Credential / Connected App secrets re-entered post-deploy
- ☐ Remote Site / CSP Trusted Site added for every new callout endpoint
- ☐ Deploying user has Modify All Data + Author Apex
- ☐ Org-wide Apex coverage ≥ 75%, all tests green, no SeeAllData
- ☐ Deployment order sequenced (or dependent components combined)
- ☐ Flows set to Active (or a post-deploy activation step exists)
- ☐ Managed packages installed / upgraded in the target first
- ☐ Ran a Validate-Only (check-only) with RunLocalTests
- ☐ Rollback package of the previous version saved
- ☐ destructiveChanges paired with a valid package.xml
- ☐ Post-deploy runbook written (activate, seed, assign, schedule)

Deploys that pass and then break in prod?

That's a pipeline problem, not bad luck. Genetrix sets up dependency-aware CI/CD, source-driven releases, and validate-then-quick-deploy across Salesforce core, Marketing Cloud, and Data Cloud — so "Succeeded" actually means shipped.

Book a 30-minute release-process review →