ARCHITECTURE NOTES · RESOURCE

Shipped a feature Friday. Woke up Saturday to three broken integrations.

Shipped a feature Friday. Woke up Saturday to three Slack messages. Three log fields catch what unit tests miss.

Companion to: Shipped a feature Friday. Woke up Saturday to three broken integrations..

What this deck covers

A late-week deploy. A refactor to the part of the stack that decides whether a draft is allowed to publish. The next morning, drafts that were supposed to ship had not shipped.

- What broke when we shipped Friday and woke up Saturday to broken integrations
- The framework: three log fields that turn silent loops into something a human can fix
- Runbook: the post-deploy health check that watches outputs
- When this is wrong: trade-offs and edge cases

What broke when we shipped Friday and woke up Saturday to broke

01 **The shape of a green-and-broken state**

Green-and-broken is the natural state of any system where the trigger and the work product live on different layers.

02 **Why try/catch on Friday is the worst kind of refactor**

A try/catch that swallows errors is the most common silent-failure pattern in autonomous code.

The framework: three log fields that turn silent loops into somethin

01 **Every refusal in an autonomous pipeline**

Every refusal in an autonomous pipeline needs three fields in the log.

02 **Without all three, you are operating**

Without all three, you are operating blind on the most failure-prone class of work in the stack.

03 **** Why did this artifact get**

** Why did this artifact get refused.

Runbook: the post-deploy health check that watches outputs

01 **Identify every autonomous loop the deploy**
Identify every autonomous loop the deploy touched.

02 **A loop is anything that runs**
A loop is anything that runs without a human in the request path.

03 **Crons, dispatchers, agent webhooks, scheduled flows**
Crons, dispatchers, agent webhooks, scheduled flows.

When this is wrong: trade-offs and edge cases

01 **The standard advice is "don't deploy**

The standard advice is "don't deploy on Friday.

02 **" That is not actually the**

" That is not actually the lesson here.

03 **The lesson is that any change**

The lesson is that any change that touches an autonomous loop needs a post-deploy health check that watches the output, not the deploy logs.

What success looks like

01 **When you build autonomous systems, you**

When you build autonomous systems, you stop being able to trust green checks.

02 **The system can be green and**

The system can be green and broken at the same time because the unit of work is no longer the request and response.

03 **The unit of work is whether**

The unit of work is whether something happened in the world that should have happened.

FAQ

01 ****Should I just stop deploying on**

**Should I just stop deploying on Friday?

02 **The calendar rule is a coping**

The calendar rule is a coping mechanism for missing instrumentation.

03 **Add the post-deploy health check that**

Add the post-deploy health check that watches outputs and any day of the week is the same risk.

NEXT STEP

Read the full architecture note

Shipped a feature Friday. Woke up Saturday to three Slack messages. Three log fields catch what unit tests miss.

arthea.ai/book ->