This page describes the Build Champion role. This is a weekly rotating role.

Responsibilities

- Daily: Triage all failed and partially succeeded builds
- At least once during the week: Review and triage error telemetry
- On the following Monday: Hand over the role to the next person

Triage non-green builds

👉 The main role of the build champ is to ensure our build remains trusted by driving down flakiness and to reduce team randomization.

We have an internal `#build` channel that a bot posts to with the results of all builds of the `main` branch on ADO. It's important to review all these failures and create issues because ignoring non-green builds ends up causing the build quality to get worse and worse over time, resulting in a loss of trust of tests, multiple retries, waste of engineering resources, etc.

It's expected that the build champion reviews all `failed` and `partiallySucceeded` builds and actions them appropriately at least once per day. The main focus is to triage and route the failure to the right owner and/or create an issue, so that the build continues to be healthy and run smoothly. **It is *not* your job to fix the problem unless you own the area.**

Follow this as a rough guide for how to review a build:

```
1. Open the "Build" link which will go to GH Actions or ADO
2. Click into the failed step and review the failure
3. If an issue is already created for this failure, mark the thread with a ✅
4. If not, here are some common failure types and how to handle them:
```

Test failure: If it looks like the test failed because of the linked change, ping the area owner. If it looks like the test flaked, search GH issues for the failure (useful queries: smoke-test-failure, integration-test-failure, unit-test-failure) and comment on it if found, otherwise create an issue for the area owner. If you end up seeing the test flake multiple times, you can skip the test and comment on the associated issue.

Compile failure: If this was a recent failure and the "Changes" seems relevant, ping the committer if they have not yet commented.

Triage error telemetry

Our error telemetry captures any uncaught errors thrown in VS Code and presents them in https://vscode-errors.azurewebsites.net/. This website allows us to view errors by each release and they also contain stats on the number of hits and machines that particular error had. Errors typically represent a case that wasn't considered in code, a broken feature and/or a bad error notification presented to the user.

At least one time during the week of being the Build Champion you should triage the errors in **the most recent ~3 pages** for a recent insiders build as well as the stable build.

Here are some examples of the types of errors you will encounter and how to handle them:

e is not iterable

Deminified Stack

```
TypeError: e is not iterable
at s in extensions/github/src/util.ts:12:28
at /extensions/github/dist/extension.js:2:97385
at u.fire in src/vs/base/common/event.ts:577:16
at i.deltaExtensions in
src/vs/workbench/services/extensions/common/extensionDescriptionRegistry.ts:88:21
at v.$deltaExtensions in src/vs/workbench/api/common/extHostExtensionService.ts:792:18
at processTicksAndRejections in internal/process/task_queues.js:93:5
```

This error looks actionable, you can tell from the paths in the stack trace that the error is within the core github extension, it also looks easily actionable. An issue should be created here (which will mark the error as triaged), and then duplicates that are encountered can also be checked off as triaged.

XHR failed

Deminified Stack

```
Error: XHR failed
at XMLHttpRequest.C.onerror in src/vs/base/parts/request/browser/request.ts:26:29
```

This error does not look actionable, while we can find where the error is thrown in code, it's unclear why this error is thrown so we would not be able to handle it properly.

Appendix

Area owners

If you're unsure who owns an area, you can roughly determine who an owner is by checking the history (git log) of the file it's contained within. Note that sometimes unrelated people perform refactoring or clean ups so look for the most active person and/or the one that is making changes related to that feature/test.

Useful links

- Error telemetry website
- ADO VS Code build
- ADO VS Code build analytics
- Dealing with test flakiness wiki page