As part of our issue triaging pipeline, a collection of scripts running as GitHub Actions aid in managing issue state. The source for these actions is available in its own repo, and the configuration is available in the vscode repo.

💡 The github triage extension for Edge/Chrome can be used to assist with triggering these workflows — it provides a "Command Palette"-style list of triaging actions like assignment, labeling, and triggers for various bot actions.

Issue Classification

Source | Example

Assigning

On a half-hourly basis, all recent issues are passed through two ML models. The first attempts to map the issue to a feature area, then consults the classifier config to determine the assignee corresponding to that feature area. If that model is unable to produce results at the desired confidence, the second model attempts to map the issue directly to an assignee.

In addition to mapping feature areas to owners, the classifier config also supports setting a threshold confidence on a per-area or per-assignee basis. The default value for all entries is 0.75, roughly corresponding to a minimum of 75% of issues being correctly assigned for a particular category. If the issue does not reach the target threshold, it will be left untouched for the inbox tracker to assign. If a team member finds that they are receiving a surplus of misclassified events, increasing the threshold for themselves or their feature areas may help.

Training

On a monthly basis, a dump of all issue data is automatically collected and given to a beefy Azure instance to train a new pair of models. This is manually verified on a test issue stream, then deployed to vscode proper by copying some bata around in Azure Blob Storage.

Author Verification

Source | Example

In cases where an issue is particularly difficult to verify, for instance those which only reproduce in a specific environment, the bot is able to help out by asking the original issue author to verify and automatically marking the issue as `verified` once they respond. It works as follows:

1. To initialize the workflow, label an issue `author-verification-requested` and either ensure it is closed with a commit, or manually add the `insiders-released` when you know it has been released in the latest insiders.
2. Upon the issue becoming both `insiders-released` and `author-verification-requested`, the bot will ask the issue author to verify it by commenting `\verified`.
3. Once the author comments, the bot will label the issue `verified` and it will be removed from our endgame queries.

Commands

Source | Config | Example Close | Example Comment

The bot supports a set of general "commands", declared in the commands config. Commands can be made to run upon either an issue being labeled with a specific label, or an issue being commented on by a particular set of users using a `\command` syntax. Commands can close issues, add labels, remove labels, and add comments. Further, commands can be made to only run when an issue either has or does not have particular labels.

Most commonly, these are used to close issues for various "wont fix" reasons. Adding labels `*question`, `*dev-question`, `*extension-candidate`, `*not-reproducible`, `*out-of-scope`, `*caused-by-extension`, `*as-designed`, `*duplicate`, `*off-topic`, or commenting `\extPython`, `\extJupyter`, `\extC`, `\extC++`, `\extCpp`, `\extTS`, `\extJS`, `\extC#`, `\extGo`, `\extPowershell`, `\extLiveShare`, `\extDocker`, `\extJava`, or `\extJavaDebug`, will close the issue and leave a comment explaining why the issue was closed and any applicable next steps for the user.

Some more examples of commands include:

```
// Upon adding the `~needs more info` label, remove the label, add the `needs more
info` label, and add a comment.
{
  "type": "label",
  "name": "~needs more info",
  "action": "updateLabels",
  "addLabel": "needs more info",
  "removeLabel": "~needs more info",
  "comment": "Thanks for creating this issue! We figured it's missing some basic
information or in some other way doesn't follow our [issue reporting]
(https://aka.ms/vscodeissuereporting) guidelines. Please take the time to review
these and update the issue.\n\nHappy Coding!"
},
```

```
// upon a team member or the listed users commenting \closedWith, close the issue
and add the label `unreleased`
{
  "type": "comment",
  "name": "closedWith",
  "allowUsers": [
    "cleidigh",
    "usernamehw",
    "gjsjohnmurray",
    "IllusionMH"
  ],
  "action": "close",
  "addLabel": "unreleased"
},
```

```
// Upon a team member or the listed users commenting `\gifPlease`, post a comment
instructing users on how to attach a screen recording
{
  "type": "comment",
  "name": "gifPlease",
  "allowUsers": [
    "cleidigh",
    "usernamehw",
    "gjsjohnmurray",
    "IllusionMH"
  ],
  "action": "comment",
  "comment": "Thanks for reporting this issue! Unfortunately, it's hard for us to understand what issue you're seeing. Please help us out by providing a screen recording showing exactly what isn't working as expected. While we can work with most standard formats, `.gif` files are preferred as they are displayed inline on GitHub. You may find https://gifcap.dev helpful as a browser-based gif recording tool.\n\nIf the issue depends on keyboard input, you can help us by enabling screencast mode for the recording (`Developer: Toggle Screencast Mode` in the command palette).\n\nHappy coding!"
},
```

```
// Upon a team member commenting `\jsDebugLogs`, add a comment and apply the `needs more info` label.
{
  "type": "comment",
  "name": "jsDebugLogs",
  "action": "updateLabels",
  "addLabel": "needs more info",
  "comment": "Please collect trace logs using the following instructions:\n\n> If you're able to, add `\"trace\": true` to your `launch.json` and reproduce the issue. The location of the log file on your disk will be written to the Debug Console. Share that with us.\n>\n> ⚠️ This log file will not contain source code, but will contain file paths. You can drop it into https://microsoft.github.io/vscode-pwa-analyzer/index.html to see what it contains. If you'd rather not share the log publicly, you can email it to connor@xbox.com"
},
```

Feature Requests

Source | Example Promotion | Example Closing

The feature requests bot serves to implement our feature request triaging pipeline. To that end, it:

1. Upon labeling an issue `feature-request`, it waits a few minutes. If a milestone has not been manually assigned, it assigns the issue to the `"Backlog Candidates"` milestone.
2. Upon an issue being assigned to `"Backlog Candidates"`, it comments explaining the feature request triaging pipeline to the issue author.
3. If at any point in the next 60 days the issue receives enough upvotes (20), it promotes the issue to `"Backlog"` and comments explaining what happened. (This may take 24h to occur)
4. If 60 days pass without the issue accruing enough upvotes, the bot will close the issue and comment explaining why. A warning comment is posted 10 days before this happens.

> *Note: If the issue receives a threshold number of comments (20), the issue is considered to have "hot discussion" and the bot will not automatically close the issue - the assigned team member should evaluate the issue's merit based on the "hot discussion" and decide to either promote or close out the issue themselves.*

Needs More Info

Source | Example

All issues which have the `needs more info` label, haven't been interacted with 7 days, and were last interacted with by a team member are closed.

If an issue has the `needs more info` , hasn't been interacted with in 60 days, and was last interacted with by a non-team member, the bot will comment pinging the issue assignee to take a look at the issue to ensure it doesn't "slip though the cracks".

Locker

Source

Issue which have been closed for 45 days and have not been interacted with in 3 days are locked. If the issue has the label `author-verification-requested` and does not have the label `verified` , it will not be locked. If the issue has the label `*out-of-scope` , it will not be locked.

English Please

Source | Examples

Upon a new issue being created, check if the issue is probably-not-english using a sophisticated AI. If so, comment a translation of a generic message instructing the user to translate the issue (preferably without using online tools as they often fail to provide useful translations of technical documents). Additionally, apply `english-please` , `needs more info` , and a language specific `translation-requested-{LANG_ID}` label to help community translators in translating issues.

Despite the sophistication of the language detection AI, it can still fail to correctly identify non-English issues when the language glyphs look similar to English. In these cases, a team member can apply the `english-please` label to trigger the above flow.

> *Note: In general, the translated comment is created by Azure Cognitive Services, however human translators can add better language-specific translations by contributing to the translation data file.*

New Release

Source

Upon an issue being created, if the issue contains a VS code version reference for a Stable release created in the last 5 days, adds the `new release` label.

Upon 5 days passing from a Stable release the bot deletes the label, removing it from all existing issues.

Insiders Released

Source | Unreleased Query | Insiders Released Query

The `insiders-released` pipeline runs automatically to:

- Mark all issues newly closed with a commit as `unreleased`
- Periodically (~daily) scan through all `unreleased` issues and promote them to `insiders-released` if their closing patch is present in the latest insiders.

Topic Subscriber

Source | Config | Example

Upon adding a label to an issue, the bot comments a list of usernames to "subscribe" to the issue by means of GitHub notifications. This is configured in the subscribers configuration file

Regex Labeler

Source | Example English | Example Chinese

Applies labels to issues that either do or do not match a particular regex. For example:

```
- name: Run Clipboard Labeler
  uses: ./actions/regex-labeler
  with:
    appInsightsKey: ${{secrets.TRIAGE_ACTIONS_APP_INSIGHTS}}
    label: "invalid"
    mustNotMatch: "^We have written the needed data into your clipboard because it was too large to send\\. Please paste\\.$"
    comment: "It looks like you're using the VS Code Issue Reporter but did not paste the text generated into the created issue. We've closed this issue, please open a new one containing the text we placed in your clipboard.\n\nHappy Coding!"

- name: Run Clipboard Labeler (Chinese)
  uses: ./actions/regex-labeler
  with:
    appInsightsKey: ${{secrets.TRIAGE_ACTIONS_APP_INSIGHTS}}
    label: "invalid"
    mustNotMatch: "^所需的数据太大，无法直接发送。我们已经将其写入剪贴板，请粘贴。$"
    comment: "看起来您正在使用 VS Code 问题报告程序，但是没有将生成的文本粘贴到创建的问题中。我们将关闭这个问题，请使用剪贴板中的内容创建一个新的问题。\n\n祝您使用愉快！"
```

Closing Issues "With a Commit"

Source | Example "\closedWith" Various pipelines work best when an issue is closed "with a commit" and we can perform operations using the SHA associated with the closing of the issue. The SHA used for the operation in is the timeline's last most:

- Commit with the `Closes/Fixes #NUM` GitHub syntax put on `main`
- PR marked as `Closes/Fixes #NUM` merged into `main`
- Comment with `\closedWith {SHA}` made by a team member

> *Note: If an issue is reopened, prior closing events will be ignored.*