Module owner and reviewer

Overview

As part of TF evolution and maturity we would need to establish a governance and administration model for code contribution, review and approval process for lock free, escalation and resolution pipelines. While we have identified a structure for contributors and leads for the modules, we have to define how the leads can nominate themselves or their team member with expertise in the module to be the owner / co-owner to approve a pull request.

While I studied some of the open source projects for the model, I liked what is being implemented in Kubernetes. They use the concept of owners and reviewers.

Reviewers - A group of experts for a components and/or module

Owners - A group of experts for module and/or overall architecture of the system

The nice thing about the way it is implemented is that the details of group alias for each module is also code driven by the OWNERS file.

OWNERS files are used to designate responsibility over different parts of the Kubernetes codebase. Today, it is used to assign the **reviewer** and **approver** roles that are used in our two-phase code review process.

The velocity of a project that uses code review is limited by the number of people capable of reviewing code. The quality of a person's code review is limited by their familiarity with the code under review. Kubernetes addresses both of these concerns through the prudent use and maintenance of OWNERS files.

OWNERS spec

The https://github.com/kubernetes/test-infra/blob/master/prow/repoowners/repoowners.go is the main consumer of OWNERS files.

OWNERS

Each directory that contains a unit of independent code or content may also contain an OWNERS file. This file applies to everything within the directory, including the OWNERS file itself, sibling files, and child directories.

OWNERS files are in YAML format and support the following keys:

- `approvers`: a list of GitHub usernames or aliases that can `/approve` a PR
- `labels`: a list of GitHub labels to automatically apply to a PR
- `options`: a map of options for how to interpret this OWNERS file, currently only one:
    - `no_parent_owners`: defaults to `false` if not present; if `true`, exclude parent OWNERS files. Allows the use case where `a/deep/nested/OWNERS` file prevents `a/OWNERS` file from having any effect on `a/deep/nested/bit/of/code`
- `reviewers`: a list of GitHub usernames or aliases that are good candidates to `/lgtm` a PR 

The above keys constitute a *simple OWNERS configuration*.

All users are expected to be assignable. In GitHub terms, this means they must be members of the organization to which the repo belongs.

A typical OWNERS file looks like:

```
approvers:
  - Ed
  - Xu
reviewers:
  - Peter
  - Chandra
  - config # this is an alias
```

Emeritus

It is inevitable, but there are times when someone may shift focuses, change jobs or step away from a specific area in the project for a time. These people may be domain experts over certain areas of the codebase, but can no longer dedicate the time needed to handle the responsibilities of reviewing and approving changes. They are encouraged to add themselves as an *"emeritus"* approver under the `emeritus_approvers` key.

GitHub usernames listed under the `emeritus_approvers` key can no longer approve code (use the `/approve` command) and will be ignored by prow for assignment. However, it can still be referenced by a person looking at the OWNERS file for a possible second or more informed opinion.

When a contributor returns to being more active in that area, they may be promoted back to a regular approver at the discretion of the current approvers.

```
emeritus_approvers:
- david    # 2018-05-02
- emily    # 2019-01-05
```

OWNERS_ALIASES

Each repo may contain at its root an OWNERS_ALIAS file.

OWNERS_ALIAS files are in YAML format and support the following keys:

- `aliases`: a mapping of alias name to a list of GitHub usernames

We use aliases for groups instead of GitHub Teams, because changes to GitHub Teams are not publicly auditable.

A sample OWNERS_ALISES file looks like:

```
aliases:
  config:
    - Ignatious Johnson
    - Nagendra
  vrouter:
    - Kiran KN
    - Anand
```

GitHub usernames and aliases listed in OWNERS files are case-insensitive.

Code Review using OWNERS files

This is a simplified description of our full PR testing and merge workflow that conveniently forgets about the existence of tests, to focus solely on the roles driven by OWNERS files. Please see below for details on how specific aspects of this process may be configured on a per-repo basis.

The Code Review Process

- The **author** submits a PR
- Phase 0: Automation suggests **reviewers** and **approvers** for the PR
  - Determine the set of OWNERS files nearest to the code being changed
  - Choose at least two suggested **reviewers**, trying to find a unique reviewer for every leaf OWNERS file, and request their reviews on the PR
  - Choose suggested **approvers**, one from each OWNERS file, and list them in a comment on the PR
- Phase 1: Humans review the PR
  - **Reviewers** look for general code quality, correctness, sane software engineering, style, etc.
  - Anyone in the organization can act as a **reviewer** with the exception of the individual who opened the PR
  - If the code changes look good to them, a **reviewer** types `/lgtm` in a PR comment or review; if they change their mind, they `/lgtm cancel`
  - Once a **reviewer** has `/lgtm`'ed, an `lgtm` label to the PR
- Phase 2: Humans approve the PR
  - The PR **author** `/assign`'s all suggested **approvers** to the PR, and optionally notifies them (eg: "pinging @foo for approval")
  - Only people listed in the relevant OWNERS files, either directly or through an alias, as described above, can act as **approvers**, including the individual who opened the PR.
  - **Approvers** look for holistic acceptance criteria, including dependencies with other features, forwards/backwards compatibility, API and flag definitions, etc
  - If the code changes look good to them, an **approver** types `/approve` in a PR comment or review; if they change their mind, they `/approve cancel`
  - Once all **approvers** (one from each of the previously identified OWNERS files) have approved, an `approved` label
- Phase 3: Automation merges the PR:
  - If all of the following are true:
    - All required labels are present (eg: `lgtm`, `approved`)
    - Any blocking labels are missing (eg: there is no `do-not-merge/hold`, `needs-rebase`)
  - And if any of the following are true:
    - there are no presubmit prow jobs configured for this repo
    - there are presubmit prow jobs configured for this repo, and they all pass after automatically being re-run one last time
  - Then the PR will automatically be merged

Maintaining OWNERS files

OWNERS files should be regularly maintained.

We should encourage people to self-nominate, self-remove or switch to emeritus from OWNERS files via PR's. Ideally in the future we could use metrics-driven automation to assist in this process.

We should strive to:

- grow the number of OWNERS files
- add new people to OWNERS files
- ensure OWNERS files only contain organization members
- ensure OWNERS files only contain people are actively contributing to or reviewing the code they own
- remove inactive people from OWNERS files

Escalation, Mediation and Resolution

Technical Steering committee should act as escalation point for mediating and ensure timely resolution of a blocked state on a PR.