K8-S14

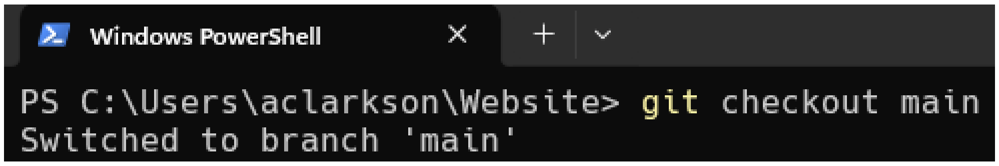
Organizational and Procedures At Labman For Source Control

Arthur Clarkson

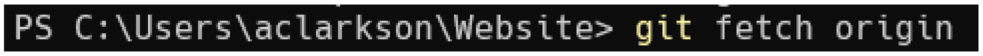
**5th September 2024

At work, our code is source controlled using git, with our remote branches being located on Azure DevOps. We have a few organisational procedures to make sure pushed into the main branch is of high quality.

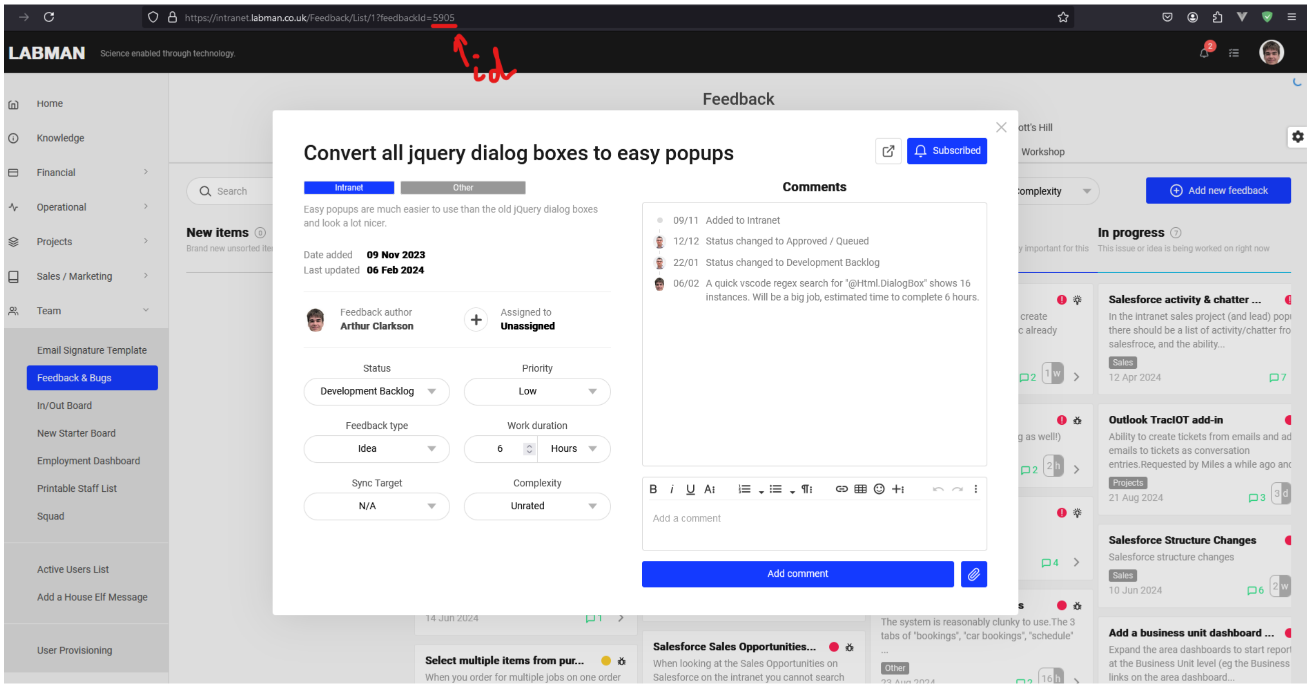
For starters, our version control management practice is trunk-based development, this is where developers merge small, frequent updates to a core "trunk" or main branch. Labman's version of this is pull from main, create a branch using a naming convention, work on what you're doing, and create a pull request. My workflow for creating a branch is:

1. When assigned to a feedback item, that I'm going to work on at present, I'll start off by running `git checkout main`, this will switch my current git working branch to main.


```
PS C:\Users\aclarkson\Website> git checkout main
Switched to branch 'main'
```

2. I would then pull all changes from the remote branch, to make sure local branch is up-to-date. I'd do this with the command `git fetch origin`


```
PS C:\Users\aclarkson\Website> git fetch origin
```

3. Our naming convention for branches is: `<username>_<intranet module>_<feedback item id>_<small imperative sentence of changes being made>`. With this feedback item for example:


Convert all jquery dialog boxes to easy popups

Intranet · Other

Easy popups are much easier to use than the old jQuery dialog boxes and look a lot nicer.

Date added 09 Nov 2023
Last updated 06 Feb 2024

Feedback author Arthur Clarkson
Assigned to Unassigned

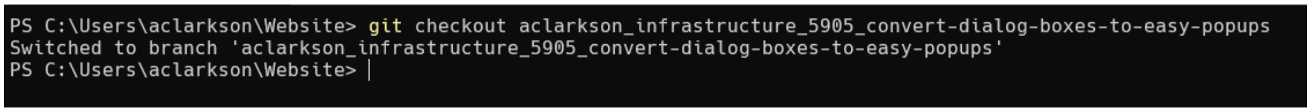
Status	Priority
Development Backlog	Low
Feedback type	**Work duration**
Idea	6 Hours
Sync Target	**Complexity**
N/A	Unrated

Comments

09/11 Added to Intranet
12/12 Status changed to Approved / Queued
22/01 Status changed to Development Backlog
06/02 A quick vscode regex search for "@Html.DialogBox" shows 16 instances. Will be a big job, estimated time to complete 6 hours.

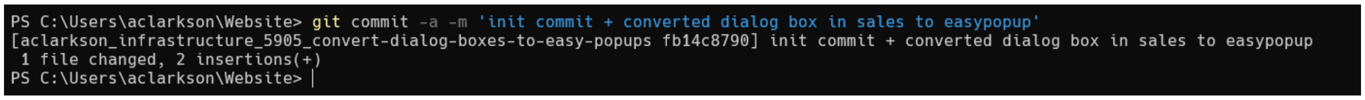
My branch name would be `aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups`. I would create this branch with the `git branch aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups`.

4. Then to change my local branch to the one I've just created I'd use the command: `git checkout aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups`



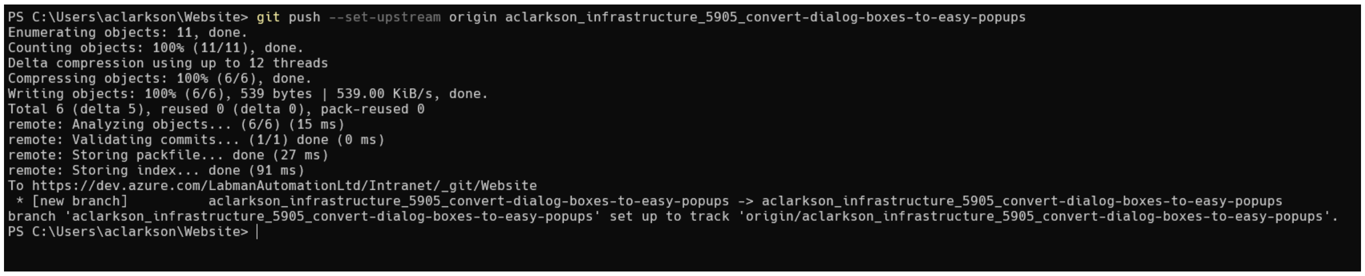
```
PS C:\Users\aclarkson\Website> git checkout aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups
Switched to branch 'aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups'
PS C:\Users\aclarkson\Website> |
```

To create a remote version of my branch, once I've made a few changes, I'd use the commands:
`git commit -a -m 'init commit + <short description of changes made>'` to create the initial commit to the remote branch.



```
PS C:\Users\aclarkson\Website> git commit -a -m 'init commit + converted dialog box in sales to easypopup'
[aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups fb14c8790] init commit + converted dialog box in sales to easypopup
 1 file changed, 2 insertions(+)
PS C:\Users\aclarkson\Website> |
```

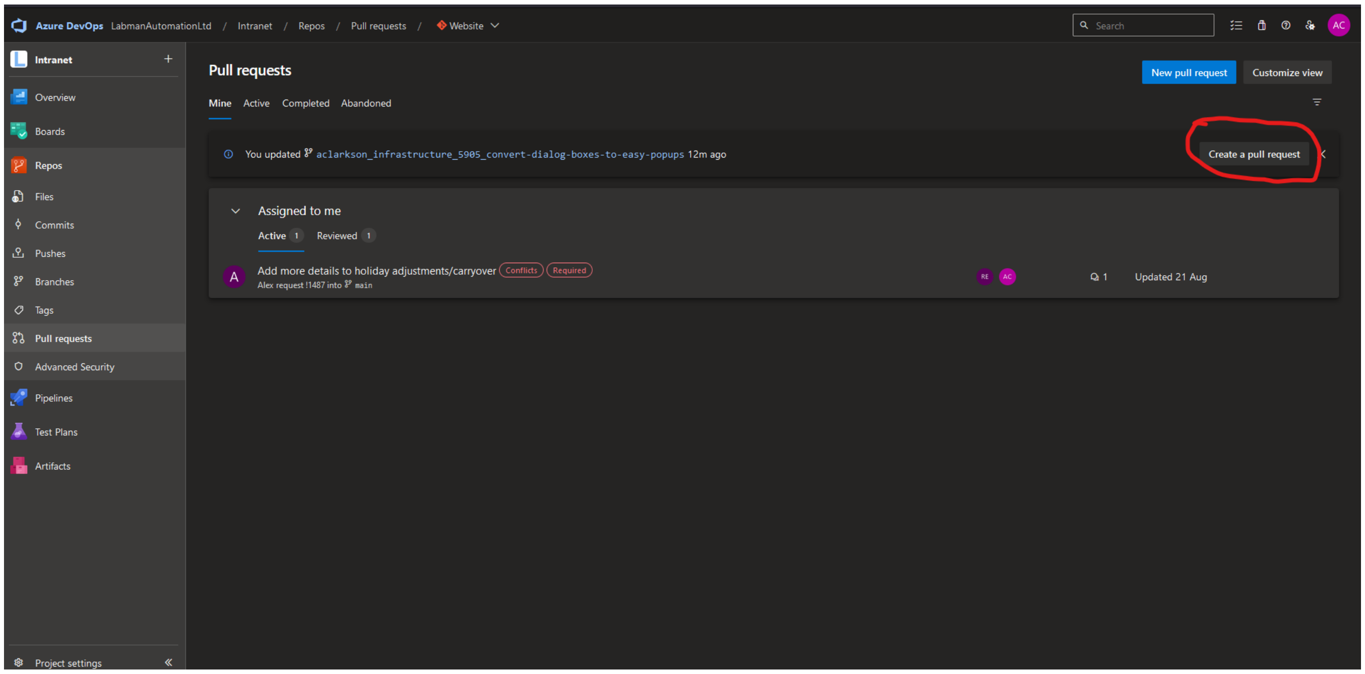
I'd then run the command:
`git push --set-upstream origin aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups` to create the remote branch.



```
PS C:\Users\aclarkson\Website> git push --set-upstream origin aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 12 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 539 bytes | 539.00 KiB/s, done.
Total 6 (delta 5), reused 0 (delta 0), pack-reused 0
remote: Analyzing objects... (6/6) (15 ms)
remote: Validating commits... (1/1) done (0 ms)
remote: Storing packfile... done (27 ms)
remote: Storing index... done (91 ms)
To https://dev.azure.com/LabmanAutomationLtd/Intranet/_git/Website
 * [new branch]          aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups -> aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups
branch 'aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups' set up to track 'origin/aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups'.
PS C:\Users\aclarkson\Website> |
```

Every time I make a milestone within my code, it is a work policy to make a commit within the branch with a suitable message. For example `git commit -a -m 'converted all dialog boxes in feedback module to popups'`

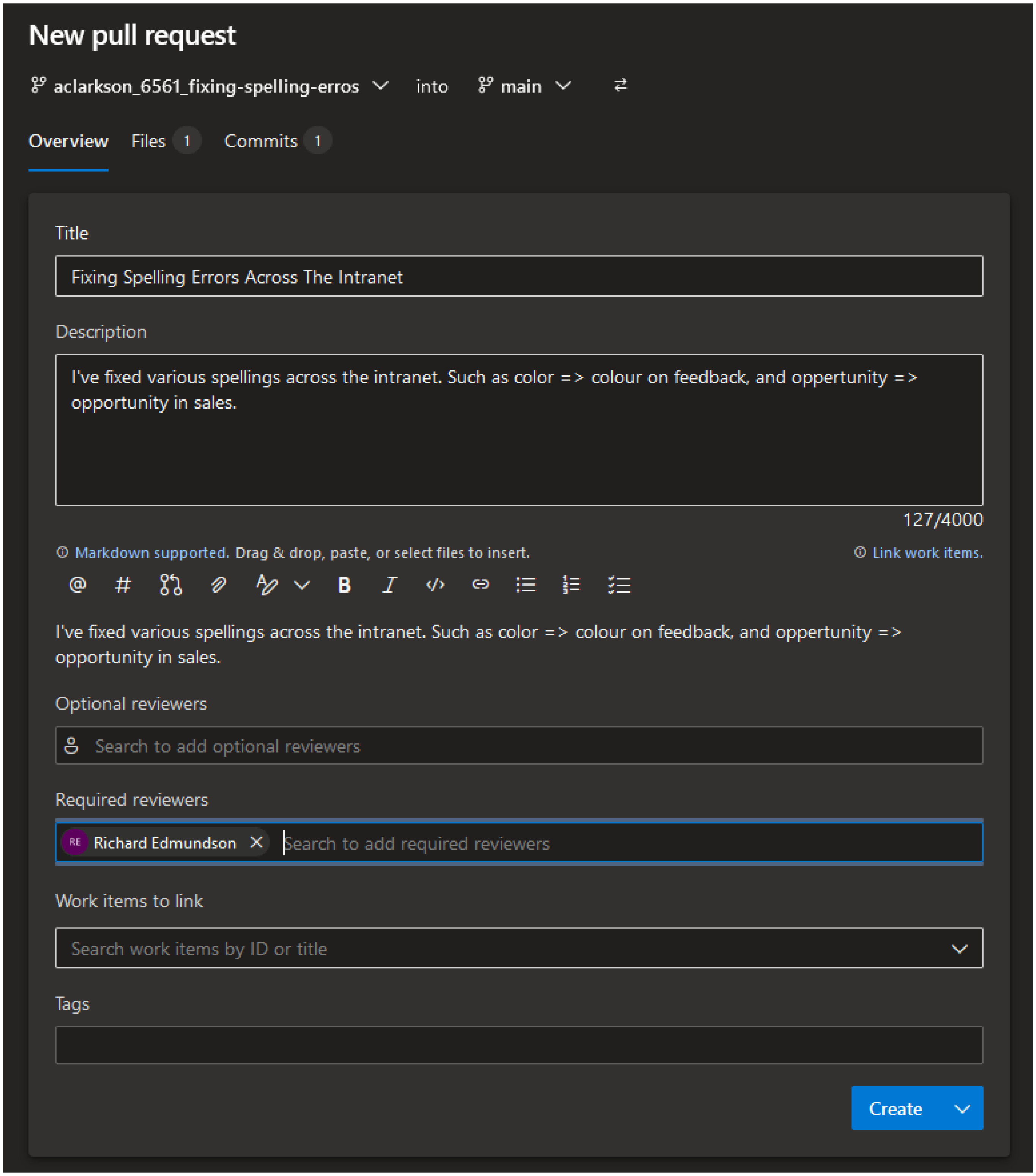
Once the code is tested and I'm happy with the code, it's policy that all branches must be pushed into main via a pull request, which has to be approved and code reviewed by a senior developer. To do this, I'd go onto Azure DevOps pull requests page, it usually has the most recently changed branch at the top:



You updated aclarkson_infrastructure_5905_convert-dialog-boxes-to-easy-popups 12m ago

Add more details to holiday adjustments/carryover
Alex request !1487 into main

I'd then give the pull request a valid title, description and choose a required reviewer (senior dev):


New pull request

aclarkson_6561_fixing-spelling-erros into main

Overview | Files 1 | Commits 1

Title

Fixing Spelling Errors Across The Intranet

Description

I've fixed various spellings across the intranet. Such as color => colour on feedback, and oppertunity => opportunity in sales.

127/4000

Markdown supported. Drag & drop, paste, or select files to insert. Link work items.

I've fixed various spellings across the intranet. Such as color => colour on feedback, and oppertunity => opportunity in sales.

Optional reviewers

Search to add optional reviewers

Required reviewers

Richard Edmundson

Search to add required reviewers

Work items to link

Search work items by ID or title

Tags

Create

Why are organisational policies important?

Organizational policies and procedures, such as those governing source control, are vital in software development because they ensure that tasks are completed consistently and efficiently. By following these guidelines, developers can maintain code integrity, facilitate collaboration, and streamline the integration process. Adhering to company, team, or client approaches to continuous integration, versioning, and source control not only helps in tracking changes and preventing conflicts but also aligns individual contributions with the overall objectives of the project. This adherence demonstrates professionalism and a commitment to quality, ensuring that the software developed meets the required standards and expectations.