Shipping Features Responsibly

Geoff Evans
2019

Agenda

"With great power comes great responsibility"
-Voltaire / Spider-Man / Stan Lee

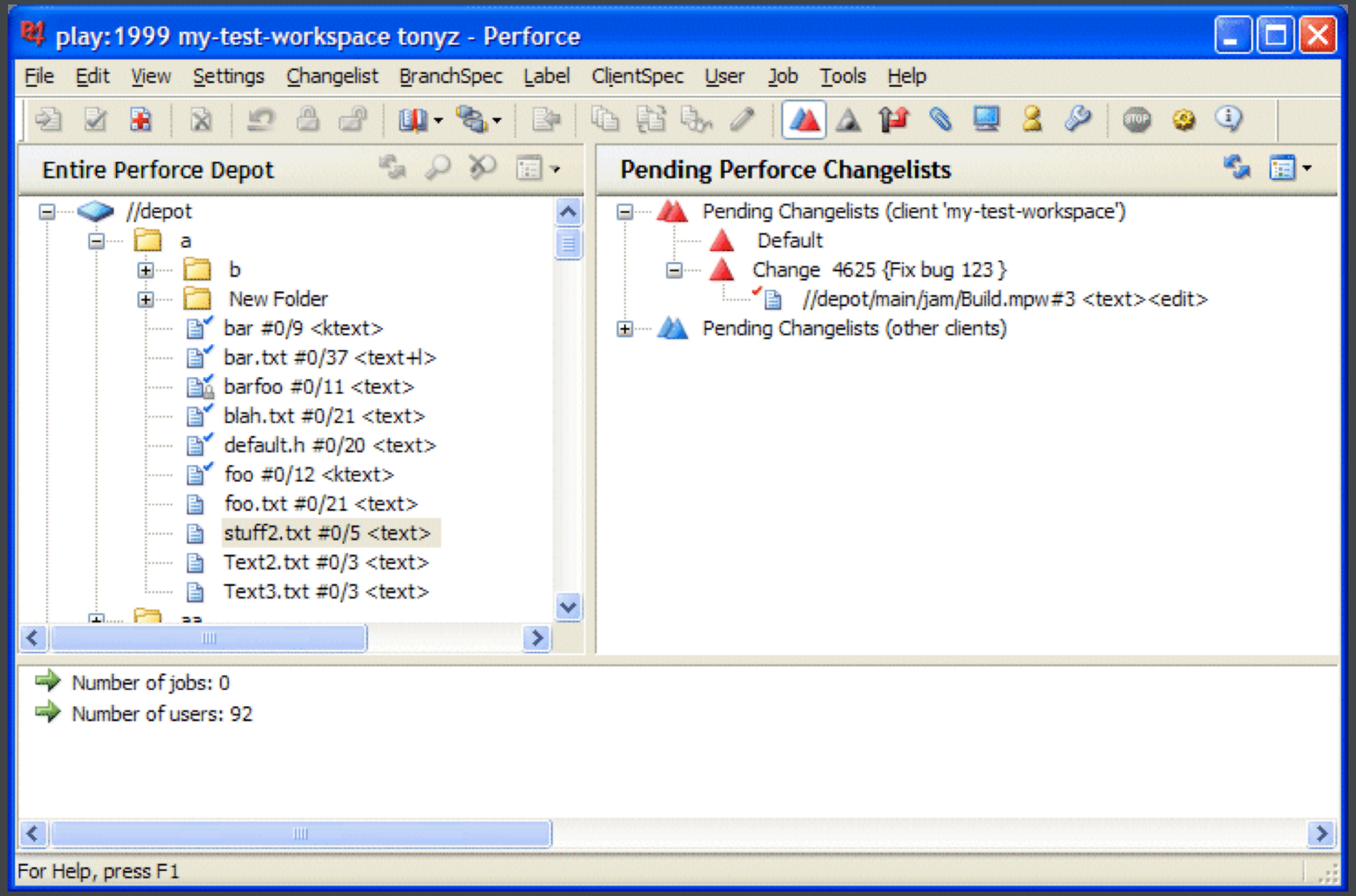
- This talk gives high level best practices of releasing new features
- All features don't necessarily need to follow all guidelines
- Any feature needs some combination of these

Me

- Gamedev Tools Engineer since 2003
- Infinity Ward, Kojima Productions, Insomniac
- Leading a tools team at IW
  - We are hiring, and having a party tonight!
  - *tinyurl.com/toolshappyhour2019* for details
- History in revision control, branching, file formats
- Built asset editors, level editors, tools frameworks

My Evil Twin

- Extensive experience in *breaking* artists & designers
- I've halted entire studios' progress for hours
- I've degraded artists' productivity for days
- I've released tools that are:
  - undertested (buggy)
  - underdesigned (don't work in practice)
  - overdesigned (confusing/too many features)


play:1999 my-test-workspace tonyz - Perforce
File Edit View Settings Changelist BranchSpec Label ClientSpec User Job Tools Help
Entire Perforce Depot
//depot
a
b
New Folder
bar #0/9 <ktext>
bar.txt #0/37 <text+l>
barfoo #0/11 <text>
blah.txt #0/21 <text>
default.h #0/20 <text>
foo #0/12 <ktext>
foo.txt #0/21 <text>
stuff2.txt #0/5 <text>
Text2.txt #0/3 <text>
Text3.txt #0/3 <text>
Pending Perforce Changelists
Pending Changelists (client 'my-test-workspace')
Default
Change 4625 {Fix bug 123 }
//depot/main/jam/Build.mpw#3 <text><edit>
Pending Changelists (other clients)
Number of jobs: 0
Number of users: 92
For Help, press F1


STOP

1 - Question Your Design

- Have you *actually* done all the design work to ship this feature?
  - Remember: *impact* usually surpasses *intent*
  - Take a step back and make *intent* crystal clear
- What problem are you trying to solve?
  - Is this something that only bothers (or makes sense to) *you*?
  - Imagine giving an "elevator pitch" of explaining this design
  - Do you sound like a crazy person?
  - Is this a waste of time/effort?

2 - Estimate The Impact

- Have you considered how it will impact:
  - Every workflow permutation (use case)?
  - How do you know? Have you searched the code (GREP)?
  - How will offsite staff and outsource vendors work?
  - Downstream projects (teams offset in time)?
  - Fellow engineers (different from users)?

3 - Perform And Document Your Testing

- The bigger the impact, the more testing you need
- Be greedy with automated tests
  - No cheating: actually wait for them to finish!
- Critical code paths should be stepped through in the debugger
  - Don't end up saying "How did this *ever work*?"
- Document testing you have done in change comment
  - Adds value to the code review process

4 - Measure And Document Your Performance

- Is it faster? Is it slower? By how much?
- What is the bottleneck? Did you change it?
- Have you measured the impact on resources?
  - CPU: are you wasting CPU cycles needlessly?
  - Memory: what about going wider on many-core CPUs?
- Disk Space
  - Have you considered how disk caches will be evicted?
  - How will you know if eviction has a bug?

5 - Prepare For Failure

- What does the worst possible failure look like?
  - How (and when) will you know if this is happening?
- How easy is this change to roll back?
  - Are there costs that make roll back painful?
  - If so, what is the commensurate change in testing?
  - Should you launch this feature as a features toggle (setting)?
- Are you *actually* prepared to roll back if it causes a problem?
  - What time is it *right now*?

6 - Update User Documentation

- Have you searched for documentation that needs updating?
- There may be many places where "documentation" exists...
  - Wiki
  - Code comments
  - Documents in revision control
  - Recent message threads about an issue

7 - Send Good Notifications

- Now the feature is release, it's time to tell people!
  - Always TLDR in your emails
  - Provide details for technical stakeholders (maybe separately)
  - If the performance wins/costs are substantial, tell people
  - Include images, GIFs, videos: eye candy helps!
- Consider and advise about issues you may expect
  - Include some advice for fixing potential issues
  - Be on call to answer replies within a time window

8 - Do The Follow Up

- Contact users that are meant to benefit from the change
  - Did they actually pay attention to your notification?
  - Did they actually benefit as you intended?
  - Did they experience a bug, and forget to tell someone?

9 - Post Mortem Yourself

- How many times did you have to roll back?
  - How are roll backs trending?
  - What can you do to decrease it?
- Verify with people randomly:
  - Your TLDR was short enough
  - Your details are pertinent

Review:

1. Question Your Design
2. Estimate The Impact
3. Perform And Document Your Testing
4. Measure And Document Your Performance
5. Prepare For Failure
6. Update User Documentation
7. Send Rich Notifications
8. Do The Follow Up
9. Post Mortem Yourself

Questions?

Twitter: @gorlak

Community:

- Twitter: @thetoolsmiths
- Website: thetoolsmiths.com
- Chat: thetoolsmiths.slack.com

Call of Duty Happy Hour: *tinyurl.com/toolshappyhour2019*

See you later tonight!