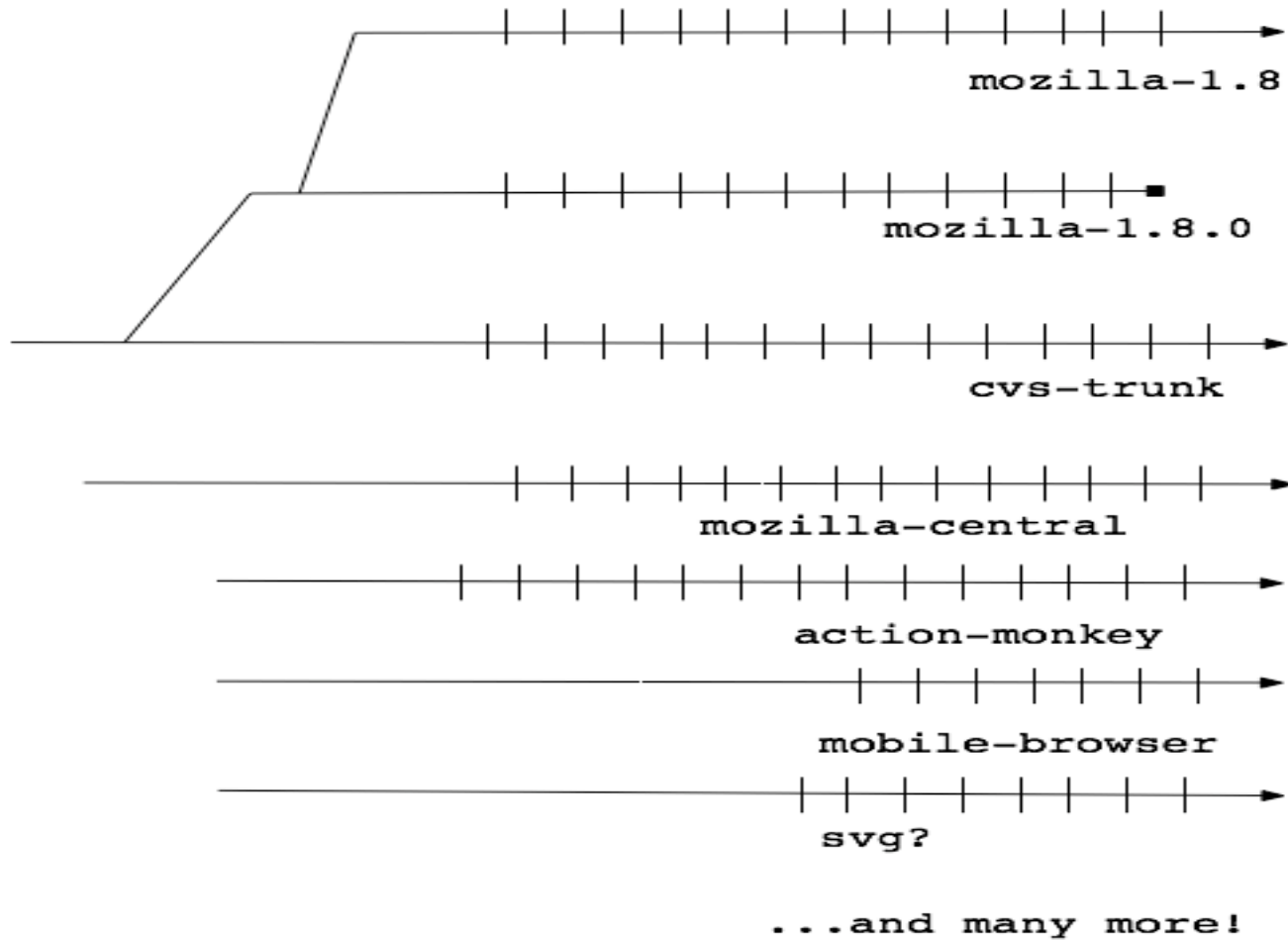


# Release Engineering Infrastructure

John O'Duinn

([joduinn@mozilla.com](mailto:joduinn@mozilla.com))

# Where we are today?



# ...on how many machines?

- mozilla1.8: 31 machines
- Mozilla1.9: 86 machines
- Mozilla2.0: 90->120 Machines (approx)
- What are all those machines doing?
  - Production builds (various debug, opt, depend, nightly, release), l10n, unittest, talos, fast talos, tryserver
  - x3 o.s. (sometimes 5 o.s.; linux64, mobile)
  - Staging machines

# Headache 1

- Keeping all these machines up and running
  - more machines, more risk of failure
- Each machine is a unique specialized machine
  - Any machine failure closes the tree
  - repairs stressful because of tree closures
  - 24 x 7

# Headache 2

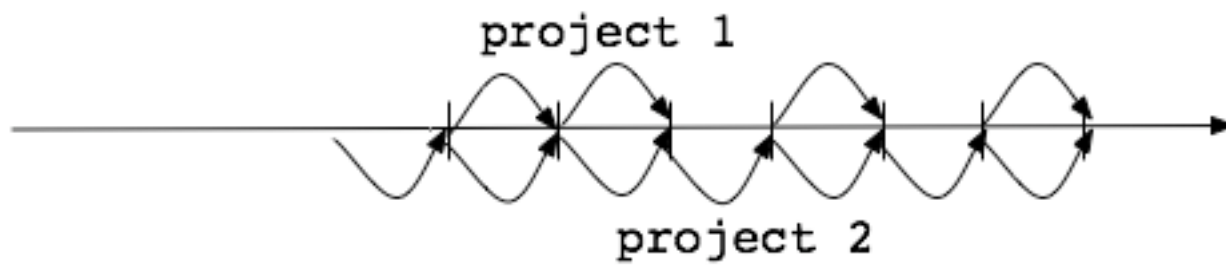
- How to open up new branches?
  - Add another 86 machines per line?
    - Months of custom setup work per line
    - More work to keep them all running
  - How many active code lines is “enough”?
  - Each branch contains 86 machines, so adding 3 lines of active development means  $3 \times 86 = 258$  additional machines.

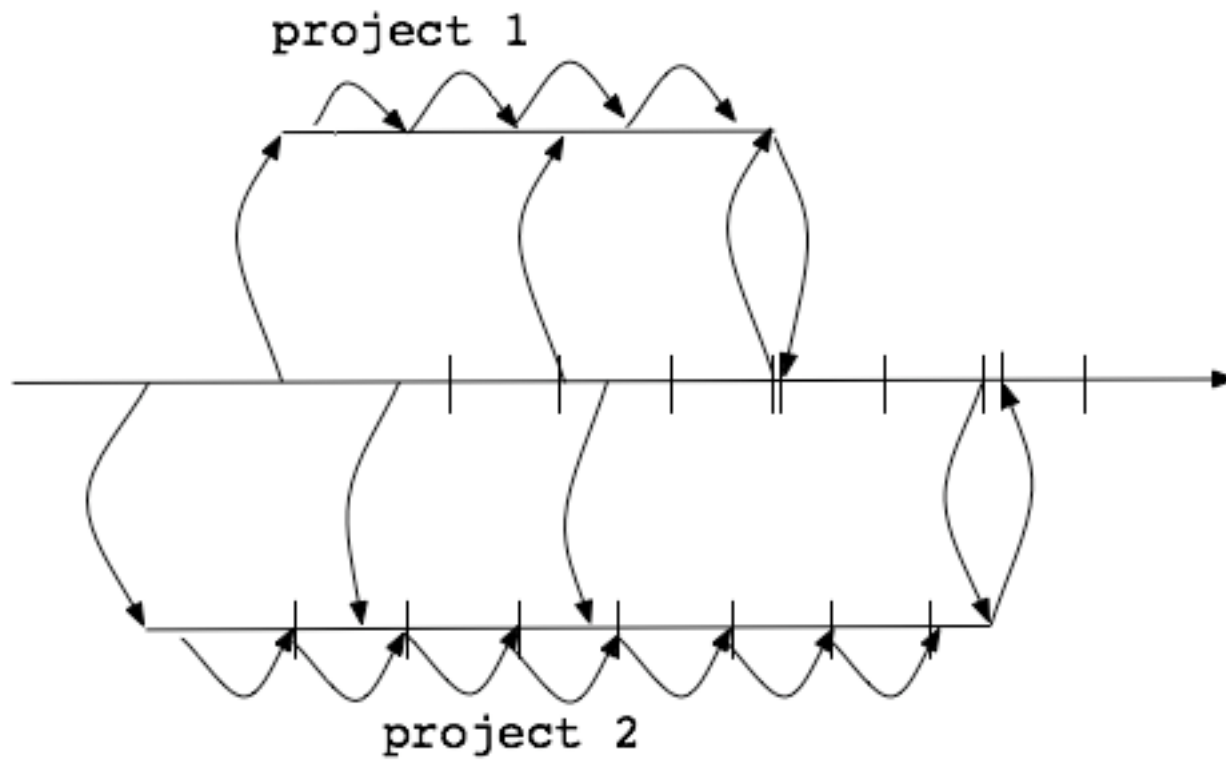
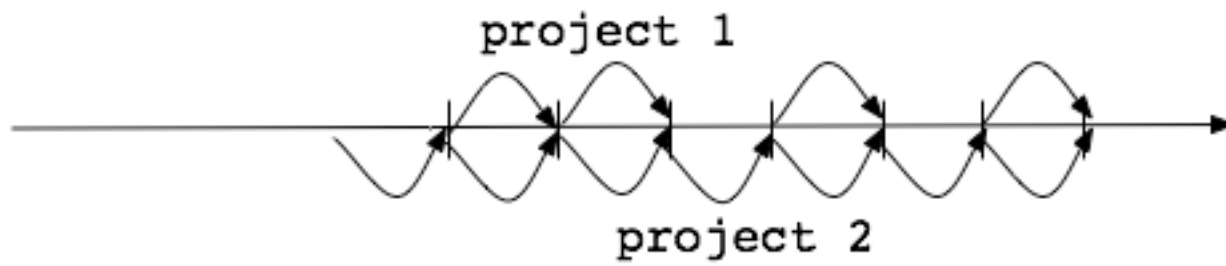
# Headache 3

- How to implement major features \*and\* do more frequent releases?
- Firefox3 took 2-3 years
  - Lots of major features
  - Lots of time for competitors to catch up after Firefox2
  - Long time for us to predict competitors and react to industry changes
- Major features take time to write
  - Major features can destabilize work in other major features
  - All features ship whenever the last feature ready
  - Un-landing a partially implemented feature is complex

# One possible solution?

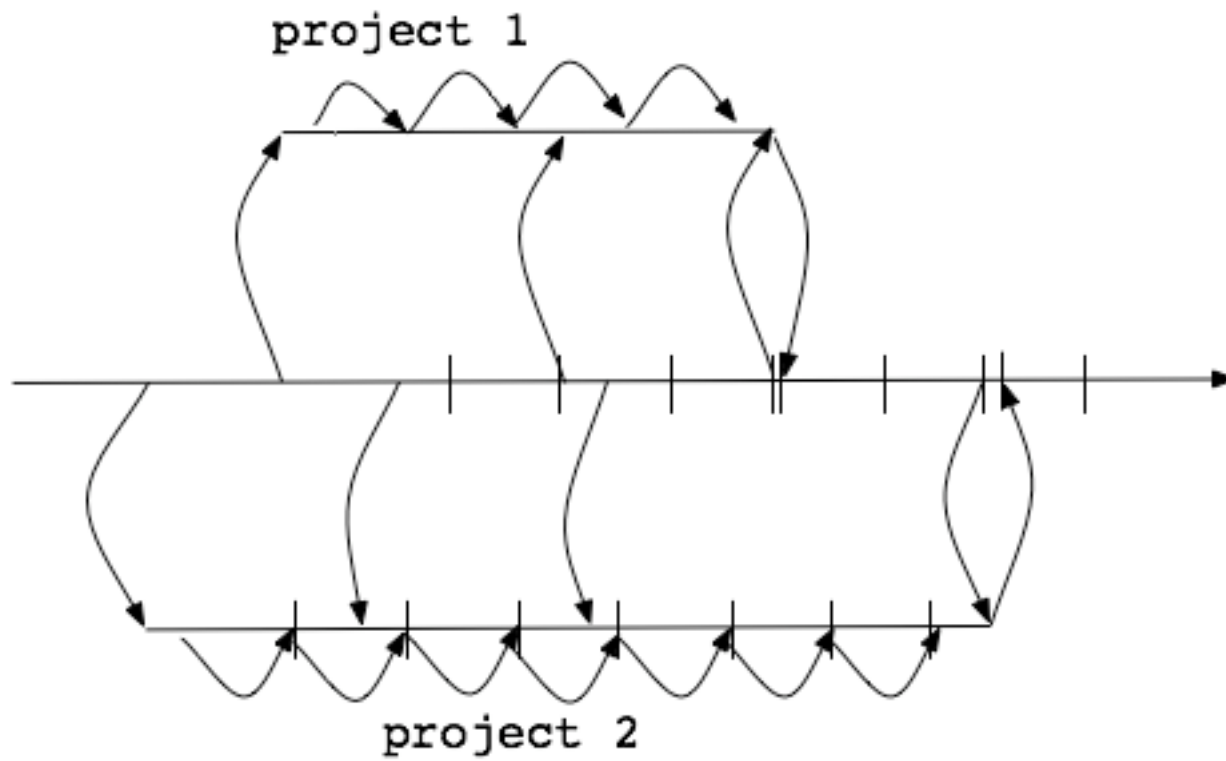
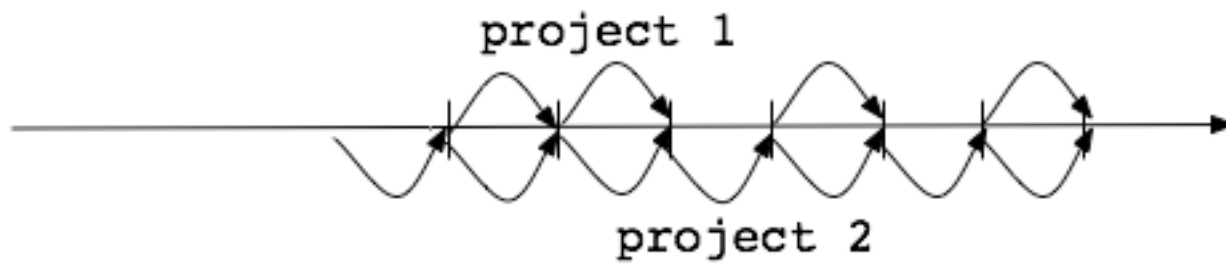
- Project branches
  - Terminology: feature/project/side branch/repo
- What would that look like?

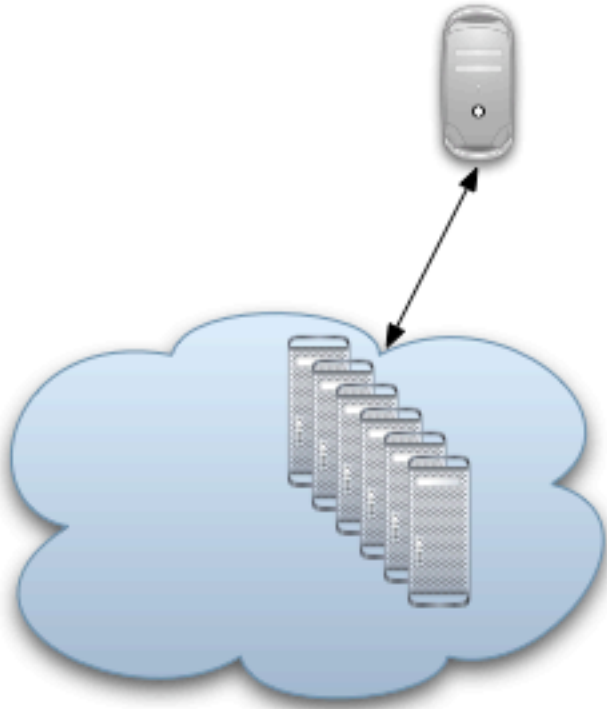
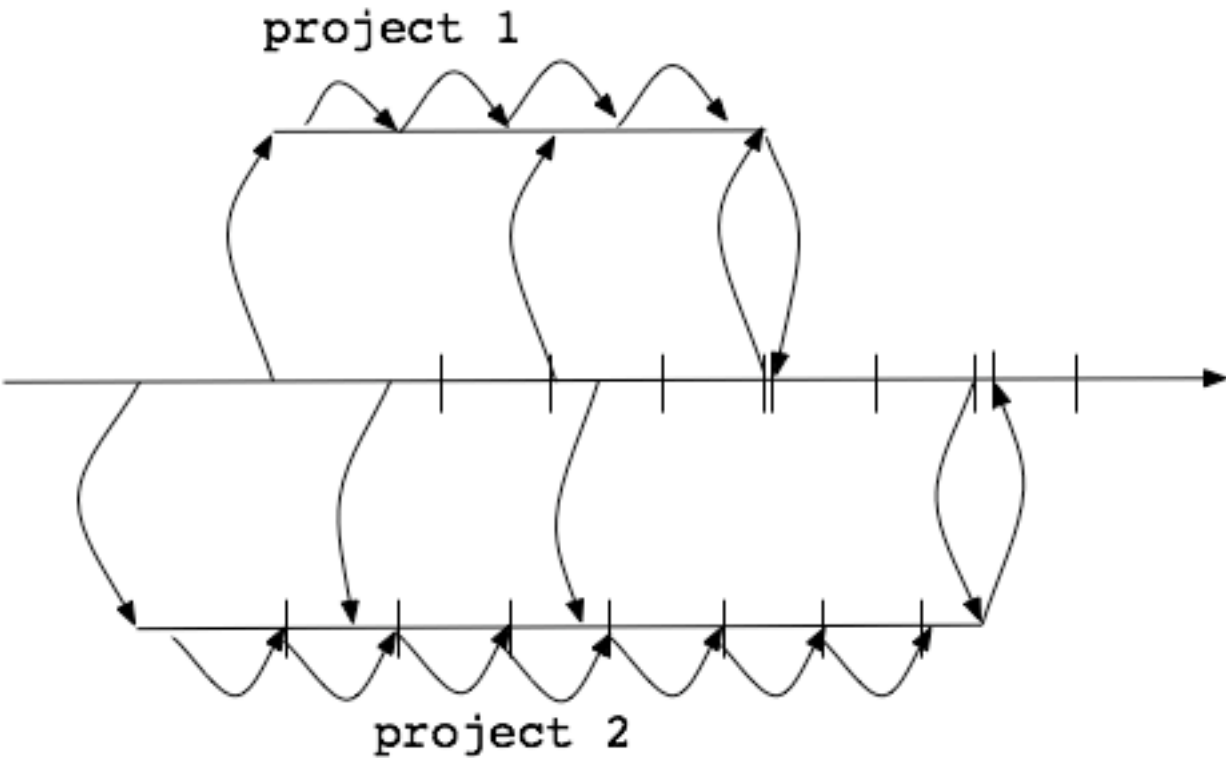




# What does all that mean?

- Allows project1 to ship when ready
  - without having to wait for project2 to complete or be backed out.
- More stable development environment?
  - Faster?
- Automated tests important
  - make sure later landings don't break what used to work
- Assumes all are using same toolchain, will need to land in m-c at some point!
- When to do a project branch?
  - Not free, time to setup, time to resync
  - 1-2 for days/weeks, probably try server
  - multi-person work over weeks/months, probably project branch



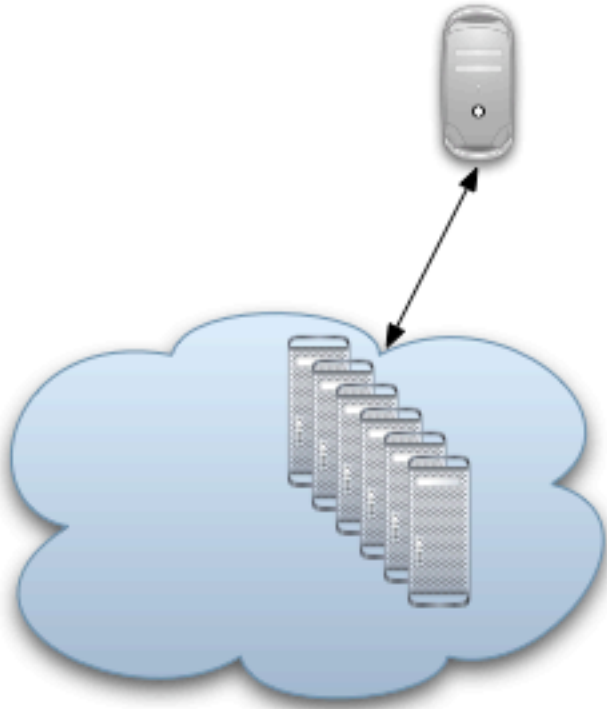
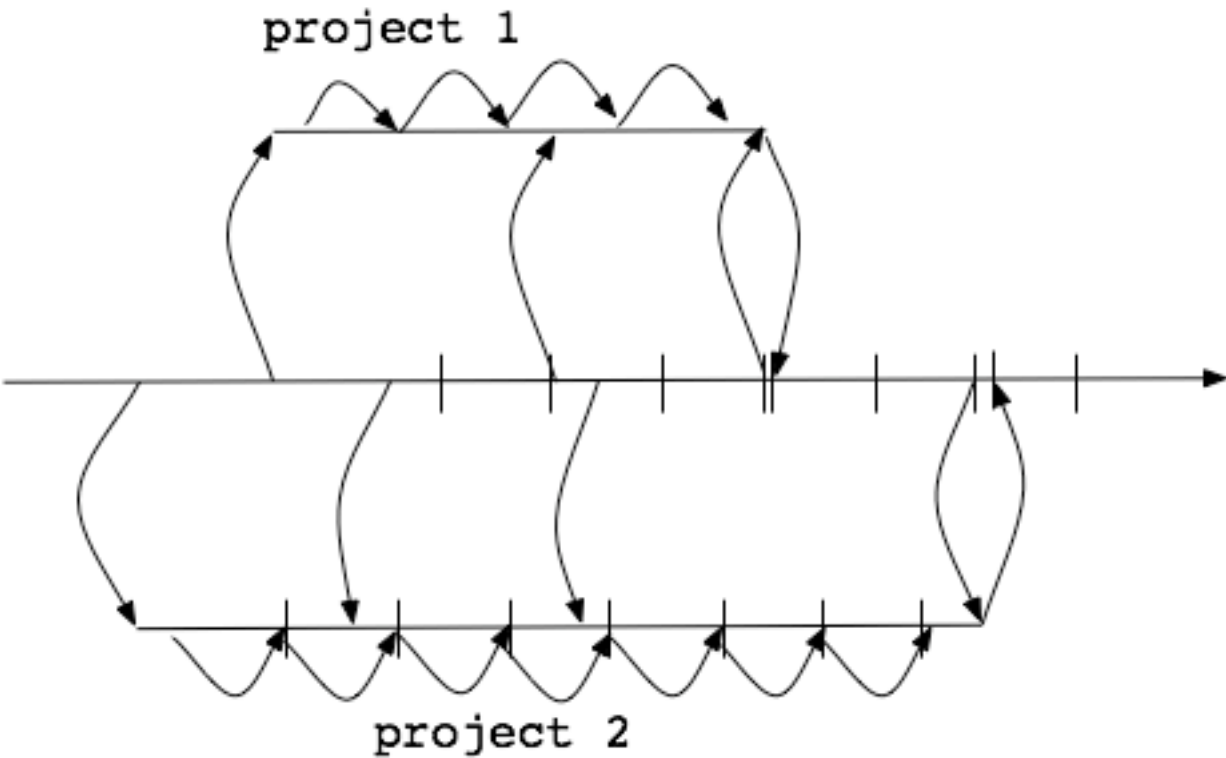


# How can we do that?

- Don't dedicate - design dynamic system that reassigns machines as needed.
- Create/destroy new branches quick & easy
  - Dynamic system that reassigns slaves as needed
  - Tracemonkey branch recently
- Easy to add slaves if needed (!= build out for peak capacity)
- Remove slaves for maintenance without closing the tree
- Machine failure != tree closure

# Continuous Build vs Build-on-checkin

- Historically, we build all the time, even if nothing changed.
- Project branches would flood shared pool of slaves
- Build only when something changes.
  - Builds start at same time on each o.s.
  - New builds trigger new unittest, talos runs
  - Periodic builds after long gaps to keep systems “warm”.



# Still To Do...

- Cleaning up machine-specific situations:
  - L10n builds
  - release update vs nightly updates
- Status info on Tinderbox or Buildbot or...?
- Unify Build and Unittest machines into the same pool of available machines.
- Talos machines need to be kept in a separate pool from build/unittest machines
  - Very specific toolchain requirements
  - Need easier way to setup Talos on project repos.

# Summary

- Project branches, try server, staging environments
  - Healthier work environment for Rel Eng, Dev, QA
- Write automated tests to ensure that no other subsequent landings break your “finished” project.
- Mozilla now able to ship major features sooner.

Q & A