infinity ward

Shipping Call of Duty at Infinity Ward

Paul Haile – Production Manager – paulh@infinityward.com
@Tyrael

Call of Duty - 2016

CALL OF DUTY®
INFINITE WARFARE

infinity ward

Agenda

- What is ‘Compass’?
  - **Automated Testing and Profiling for 'Call of Duty’** by Jan van Valburg
  - **Thursday 10:00 am @ Room 2006, West Hall**
- Code/Feature Releases
  - The smaller, internal case.
- Shipping Milestones
  - The larger, public case.
- Shipping Patches
  - The tiny, yet surprisingly complex case.

Code Releases

- Protect developers from volatility
- Cadence of feature drops
- Goals:
  - Well tested
  - Atomic
  - Includes all prior changes / never go backwards

Evolution of Code Releases

CALL OF DUTY MW3

- ‘Dibs’ system
- Simple / Easy
- Scaled poorly

CALL OF DUTY GHOSTS

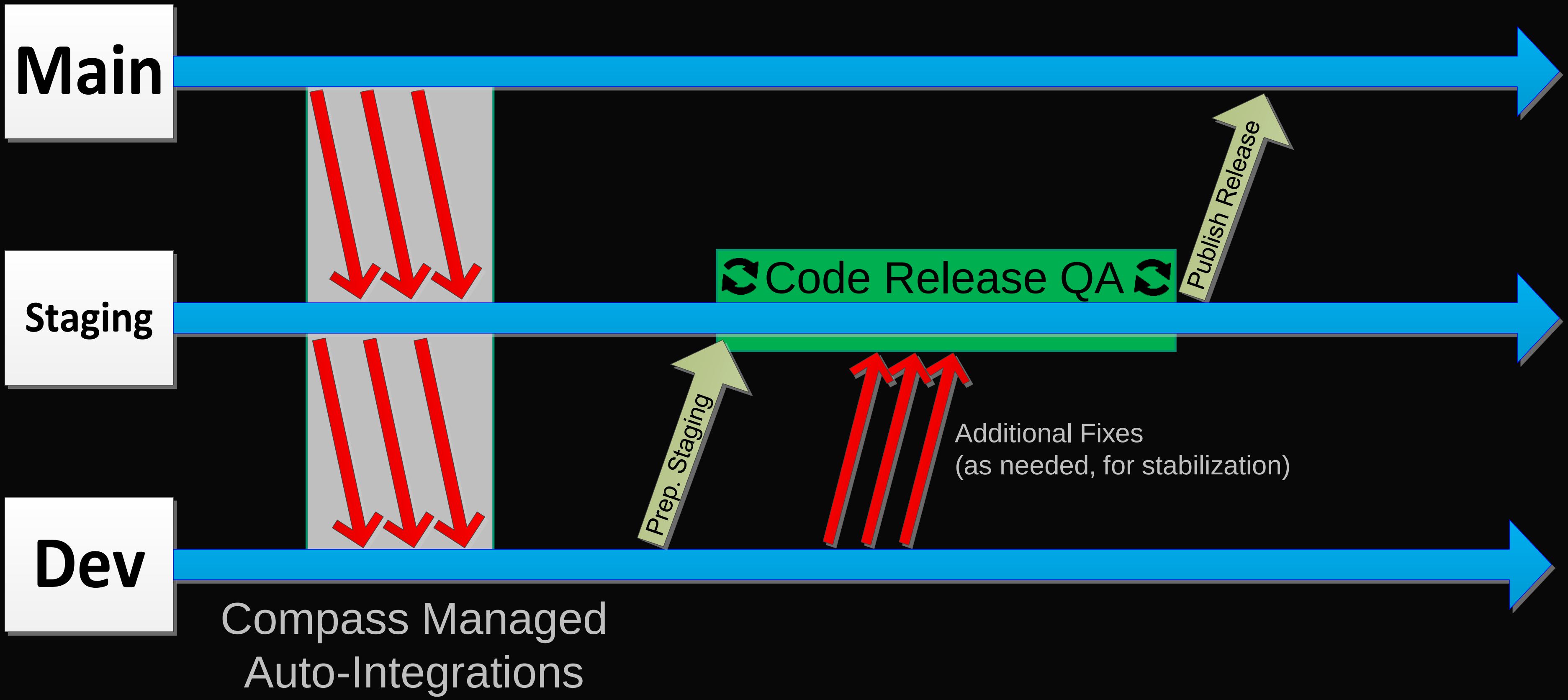
- Hybrid approach
- Added formal QA
- Half solution

CALL OF DUTY INFINITE WARFARE

- Fully branched
- Compass assisted
- Full solution

infinity ward

Code Release Diagram



infinity ward

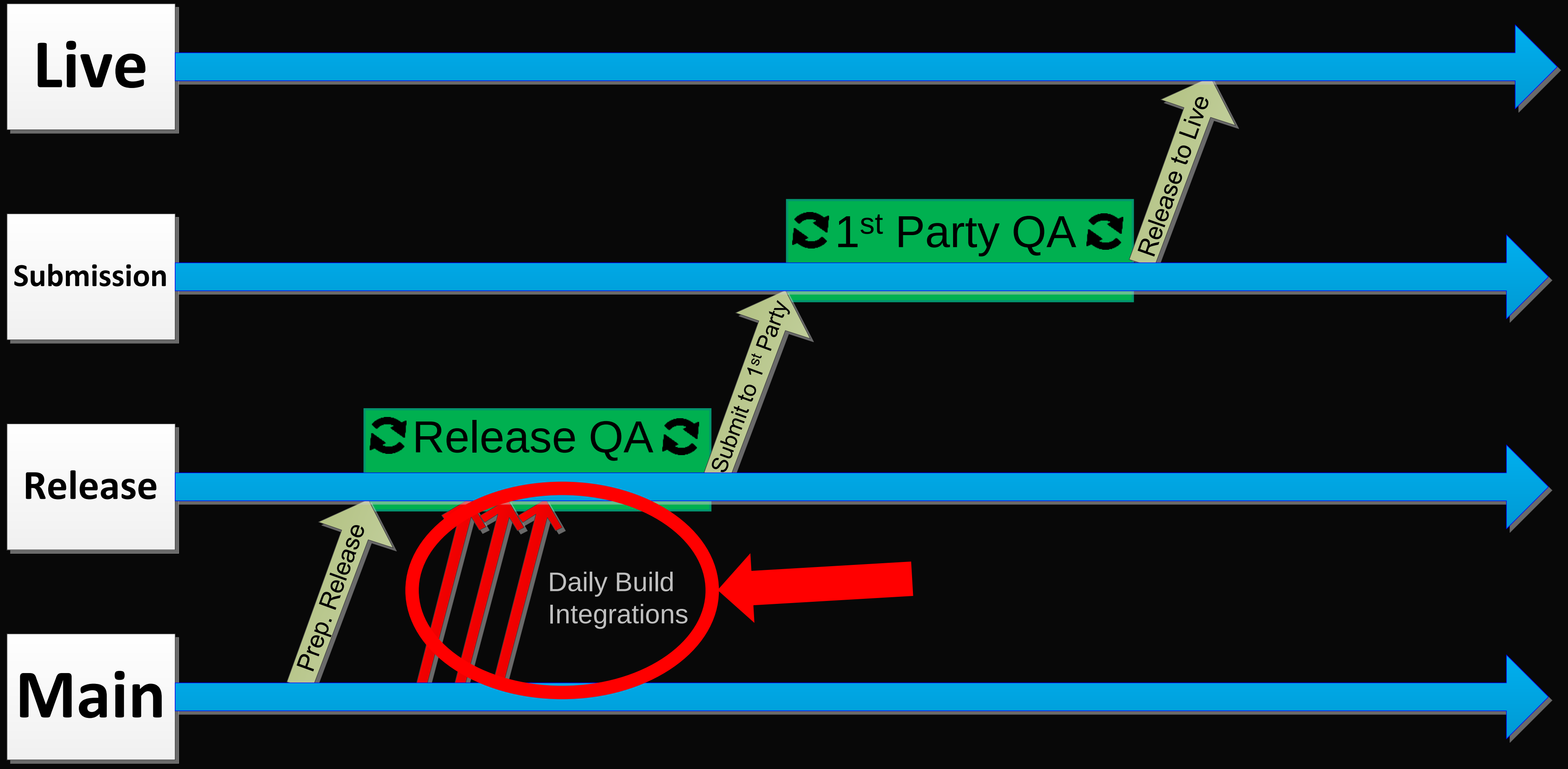
Summary on Code Releases

- Reliably release code minimum of 3 times per week
- Instability/breakages still happen but are generally minor
  - “Hotfix” pipeline is key to get single fixes out quickly
- Review test plan often to add problem areas
  - Remove unnecessary tests to keep QA time down and focused on most important areas
- Increased velocity of feature releases, and also improved stability

Shipping Milestones

- Building on the foundation of our code release process
- Created a reusable stream structure that represents the flow of a milestone from development to live
- Entirely production driven
- Goals
  - Parallel Development
  - Enable developers to ‘always be submitting’
  - Be able to recreate or modify any milestone build

Release Stream Diagram



Nightly Build Process

- Developers tag their changelists with a milestone when checking in to P4
  - For example “[Title Update 5]” or “[DLC 2]”
- Determine what needs integration (‘p4 interchanges’ report)
- Production does integrations and triggers the build when the state of the branch is correct
- Compass handles everything else

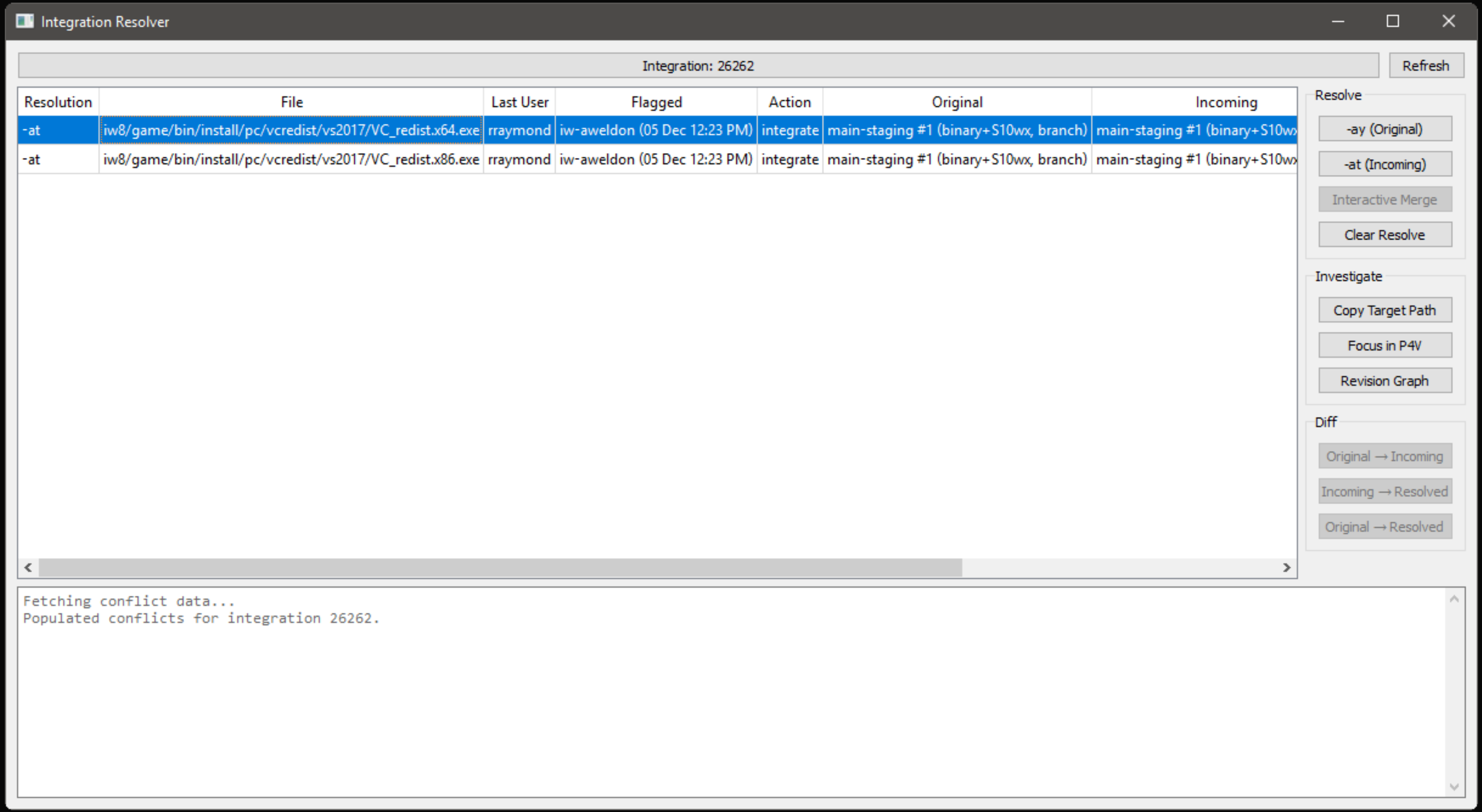
Things To Watch Out For

- Exclusive Lock files
- P4 copy is awkward
  - Ended up with the **mergeany** flag on all streams
- Production team education
- Merge conflicts
- P4 server scalability

P4 Server Specs

- Commit Server:
  - Virtual Machine
  - 16 CPUs @ 2.4 Ghz / 64 GB RAM
  - Nimble Array 5TB flash cache + 30TB Storage
- 2x Replicas (1 for users, 1 for compass):
  - 2x Xeon E5-2630s @ 2.2 Ghz / 256 GB RAM
  - 16TB SSD Storage RAID 10 (10 TB Usable Space)
- Networking
  - 40Gb Links between all internal servers + backing storage
- 7 full streams at peak production, 700,000+ files each - No performance issues

Conflict Resolution Tool



Resolution	File	Last User	Flagged	Action	Original	Incoming
-at	iw8/game/bin/install/pc/vcredist/vs2017/VC_redist.x64.exe	rraymond	iw-aweldon (05 Dec 12:23 PM)	integrate	main-staging #1 (binary+S10wx, branch)	main-staging #1 (binary+S10wx
-at	iw8/game/bin/install/pc/vcredist/vs2017/VC_redist.x86.exe	rraymond	iw-aweldon (05 Dec 12:23 PM)	integrate	main-staging #1 (binary+S10wx, branch)	main-staging #1 (binary+S10wx

```
Fetching conflict data...
Populated conflicts for integration 26262.
```

Conflict Resolution Tool

- Simple PyQt interface over the top of a database
- Allows developers to perform merges without impacting local state
- Developer’s intentions are communicated from the tool to a MySQL database
- On the next merge Compass reads from the database and performs the desired actions
- Developers can instantly resolve conflicts occurring in any stream anywhere

Patches – Old Way

- Manually curated asset whitelists
- Because of this devs needed intimate build process knowledge

Patches – New Way

- Determinism
- Checksum & manifest driven patching
- No changes to developer work flow
- Production driven

Parallel Development

Dev **Main**

- Title Update 7
- Bulk of development efforts.

Future

- DLC 1
- Heavy art content iteration, but isolated from TU development.

Release

- Title Update 6
- QA Iteration
- Cherry picking fixes as needed.

Submission

- Title Update 5
- In submission with 1st party.

Live

- Title Update 4
- Hot patch creation and deployment.

Takeaways

- Parallel development wins you back time when you need it most.
    - Achievable with a small amount of branching.
- Enable Production to manage the build process.

Thank You!

Paul Haile – Production Manager – paulh@infinityward.com
@Tyrael