Web CI/CD

This is a work in progress - Omar Miskinyar is working on options for a complex release engineering process that can accommodate features across different squads / sprints that need to be combined for release purposes

Release Manager

- [redacted]
- Web team has started to move over to trunk based development style (re: webapp) due to requirements/expectations of product and QA
  - Implementation of feature flagging technique
  - Newer repos will be more trunk based or GitHub flow style, Next.js app will make use of feature/release flags
- If you are the 'Release Manager' and will be OOTO please make sure to notify everyone who your Back-Up will be while you are out

Current Coverage

Platform	Source Code	Automated Build Software	Build Failure Mechanism	Unit Test Coverage	Integrated Tests	Functional Tests/Automation Software	Automation Source Code	Automation Failure Mechanism	Re Ma
Web	Git=Web	• Jenkins=webapi • Jenkins=widget • Jenkins=webapp	N/A	• Web=All Files in web-app repo • 11.23% Statements 1482/13192 • 11.97% Branches 945/7898 • 19.16% Functions 458/2390 • 11.5% Lines 1452/12625	N/A	Jenkins=QA-AG-WebApp	QA-AG-WebApp-Automation	• Reports • Applitools • Cypress	Je

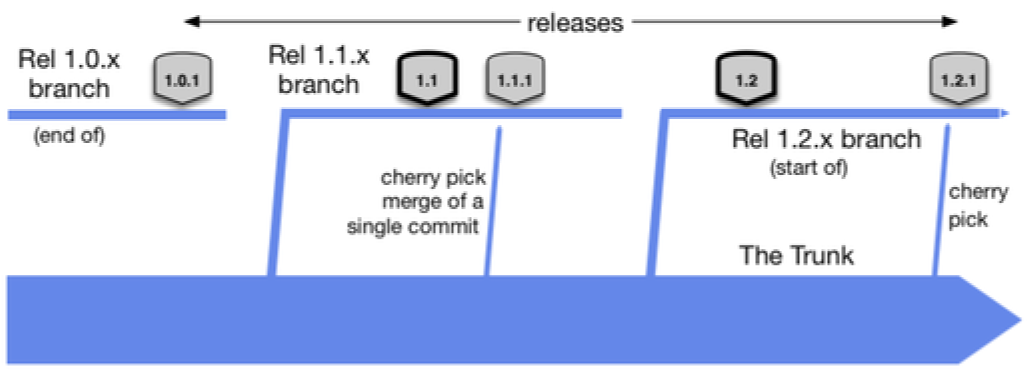
Suggested Release Pipeline

1. Release Pipeline Trunk Based Development

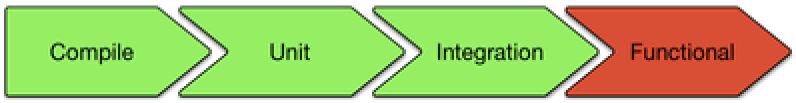
Summary:

- Web Branching Strategies Branching Strategy (Deprecated)
- There is only one branch, there are no other branches, so there is no merging. No merging equals no merge conflicts
- Trunk branch is always considered "`releasable`"
- Code merged to the trunk branch should never break the build
- Pull Requests are short-lived by design, typically living 24 - 48 hours
- Don't merge/fork off another feature or fix branch, we are trying to keep the "distance" of development code to the trunk short
- Granular commits, small PR's
- Smart and focused use of Feature and Release flags See Feature Flags
- Rely on automation and PR's to act as mechanisms to catch early warnings on breakages and incompatibilities
- There should be no concept of a code freeze, this is antithetical to Trunk-based development

Branch Strategy

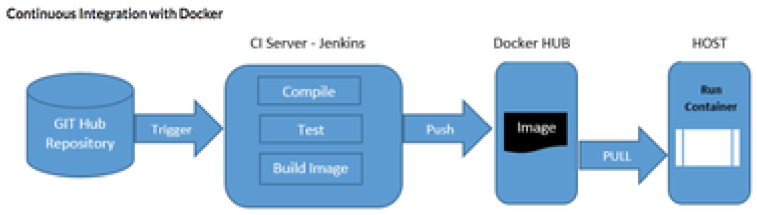


https://trunkbaseddevelopment.com/branch-for-release/

- **Code Compile**
  1. Create branch
     a. Pull request temporary feature branch
  2. Dependency Analysis - Developer checks for dependancies
     a. Microservices
     b. Feature Flag On/Off
        i. Feature Flag Documented Web Feature Flag
        ii. Release Flags
     c. Library dependency analysis Tools Example: NPM, OSSIndex, Bundler Audit
  3. Code analysis (Peer Review) **APPROVED** `master(trunk)`
     a. Compile + Test + Build: Merge to `master(trunk)` Jenkins triggers a build Build is Tagged
        i. Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests **PASSES**
        ii. If Test **FAILS**
            1. 

            2. Event Notification
               a. Slack to notify events/status
               b. This should go up on Screen/Board if developers are co-located and email notification should be sent out

- **Deploy Acceptance Release Candidate**



1. Front End Development Passed Docker Staging
   a. Deploy **release candidate** to Staging
      i. Tests: Unit Tests Integration Acceptance Testing:E2E Cypress Tests Manual (Feature based testing) **PASSES**
      ii. Event Notification
          1. Slack to notify events/status
          2. This should go up on Screen/Board if developers are co-located and email notification should be sent out
   b. ~~Deploy **release candidate** to Preprod *(Optional)*~~
      i. ~~Tests: Unit Tests Integration Acceptance Testing:E2E Cypress Tests Manual (Feature based testing) **PASSES**~~
      ii. ~~Event Notification~~
          1. ~~Slack to notify events/status~~
          2. ~~This should go up on Screen/Board if developers are co-located and email notification should be sent out~~

- **Production Check List**

1. Dependency Analysis
    a. Microservices
        i. Release Manager checks for dependencies
    b. Feature Flag On/Off
        i. Feature Flag Documented Web Feature Flag
        ii. Release Flags
    c. Library dependency analysis
2. Frontend Stakeholders Approval Production

- **Release Notes**

1. ~~Promote Bumped Version Release.YYYYMMDD.[SequenceNumber], HotFix.YYYYMMDD.[SequenceNumber] Semantic~~
    a. ~~The sequence number is the production build sequence number + 1~~
2. When we name our branches, we want to aim for clear, concise names.
    a. Don't include any information that is not useful.
    b. Typically, you just want to include Jira issue number and brief summary/title:
        i. {branch-prefix}/{jira-id}-{title-summary}
        ii. For example, a feature branch for a Jira ticket that drops in a vendor script in the document head might look like this:
        iii. feature/pbr-98-optimizly-include
        iv. Similarly, bugfix branch could look the same:
        v. bugfix/bac-101-card-layout-fix
3. Scrum Master or Release Manager Sends out *Release Notes*

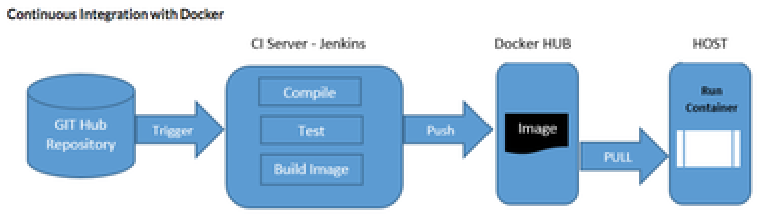
Hotfix and Unplanned Releases

- **Code Compile**

1. Branch off `master` Verify Fix Cherry-Pick back to Release Branch/Production
2. Code analysis (Peer Review) **APPROVED**
3. Jenkins triggers a build `master`

- **Deploy Acceptance Hotfix**

1. Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix **PASSES**
2. Event Notification
    a. Slack to notify events/status
    b. This should go up on Screen/Board if developers are co-located and email notification should be sent out
1. Deploy fix to Staging   Docker



    a. Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix **PASSES**
    b. Event Notification
        i. Slack to notify events/status
        ii. This should go up on Screen/Board if developers are co-located and email notification should be sent out

- **Deploy to Production Cherry-Pick**

1. Dependency Analysis
    a. Microservices
        i. Release Manager checks for dependencies
    b. Feature Flag On/Off
    c. Library dependency analysis

1. Frontend Stakeholders Approval Production
2. Release Notes

    a. Promote Bumped Version Release.YYYYMMDD.[SequenceNumber], HotFix.YYYYMMDD.[SequenceNumber] Semantic
        i. The sequence number is the production build sequence number + 1
    b. Scrum Master or Release Manager Sends out *Release Notes*

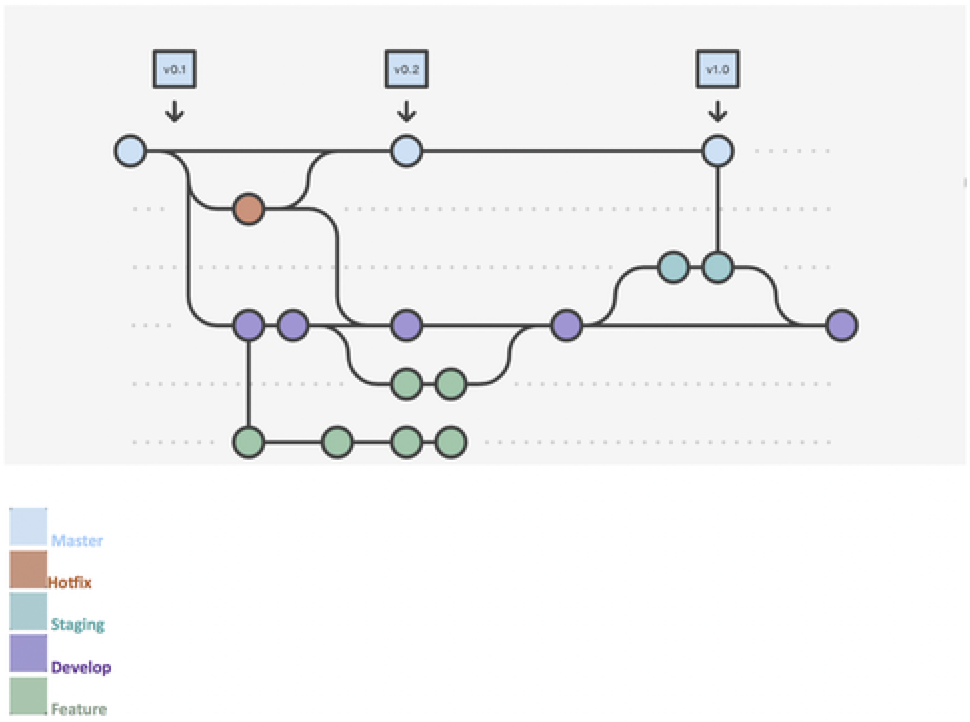
~~Before 8/27 Release Process Web~~

- ~~Web Deployment Guide~~

~~1. Web Team Release Pipeline GitFlow~~

- ~~Web Team is currently using GitFlow but transitioning over to Trunk~~
- ~~Team has started to move over to TBD~~

~~Branch Strategy~~



- ~~**Code**~~
  1. ~~Create feature branch~~
     a. ~~Pull request feature branch~~
  2. ~~Code Analysis (Peer Review)~~ **APPROVED**
  3. ~~Merge to Develop/Staging Jenkins triggers a build   Build is Tagged~~
- ~~**Compile**~~
  1. ~~Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests~~ **PASSES**
- ~~**Deploy Acceptance Release Candidate**~~
  1. ~~Front End~~
     a. ~~*Completion of Code Freeze (Release Branch – small and one off features)*~~
     b. ~~Deploy **release candidate** to Staging~~
        i. ~~Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual~~ **PASSES**
     c. ~~Deploy **release candidate** to Preprod~~
        i. ~~Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual~~ **PASSES**

- ~~**Production Check List**~~
    1. ~~Frontend Stakeholders Approval Production~~
- ~~**Release Notes**~~
    1. ~~Deployed Release/Master Production (Weekly Release Note sent out by PM)~~
- ~~**Hotfix and Unplanned Releases**~~
    1. ~~Branch off latest release tag and call it **hotfix/{app_name}**~~
        a. ~~Prod release tag http://deploy-tool.autogravity.com/~~
    2. ~~Push that branch to GitHub~~
    3. ~~Create a feature branch off of the branch **hotfix/{app_name}**~~
    4. ~~Pull Request into Master~~
    5. ~~Jenkins triggers a build~~
        a. ~~Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix~~ **PASSES**
    6. ~~Deploy to Preprod~~
        a. ~~Tests: Unit Tests Integration Tests Acceptance Testing:E2E Cypress Tests Manual/QA Verify Fix~~ **PASSES**
    7. ~~Deploy to Production~~
        a. ~~Frontend Stakeholders Approval Production~~
    8. ~~Release Notes~~
        a. ~~Deployed Release/Master Production (Weekly Release Note sent out by PM)~~