Developing for Continuous Deployment

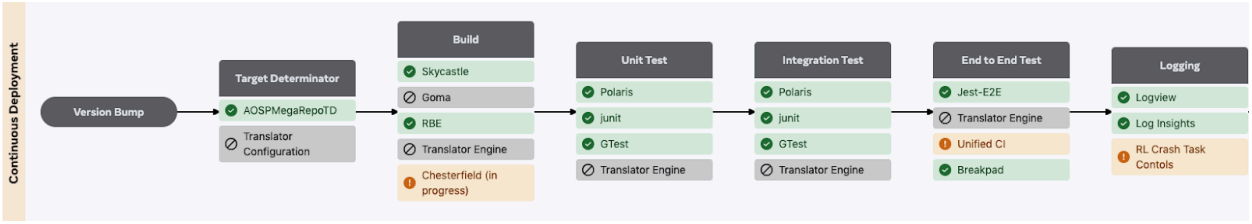
Executive Summary

This document provides an overview of the Continuous Deployment (CD) workflow and its various stages, including building, testing, and deploying code changes. It highlights the importance of automated tests, version control systems and deployment pipelines in ensuring that code changes are delivered to users quickly and reliably. The document also discusses signal aggregation and visibility, which involves collecting and presenting signals from multiple sources to provide a comprehensive view of the codebase's health and status. Additionally, it introduces two tools used for signal aggregation at Meta: Flight Deck and WhereIsSTU. Finally, the document touches on release candidate testing, developer preview channels, public test channels, and release operation and monitoring, emphasizing the importance of these processes in ensuring the quality and stability of software releases.

Understanding the CD Workflow Swim Lane

This page describes the stage you need to follow to complete the CD integration, based on the choices you selected and installed with the Supported Workflow Tool.

When working with the Supported Workflow Tool for continuous integration, you are dealing with the Continuous Deployment swim lane, as partially shown in the following image:



Continuous deployment (CD) is a software development practice that involves automatically building, testing, and deploying code changes to production. This allows teams to quickly and reliably deliver new features and bug fixes to users. To develop for continuous deployment, it's important to write automated tests that cover your codebase. These tests help ensure that your code changes don't break existing functionality and provide a safety net for continuous deployment.

Another key aspect of developing for CD is using version control systems like Git. Version control allows you to track changes to your codebase over time, making it easier to identify which changes may have caused issues in production. Implementing continuous integration (CI) is also crucial for CD. CI involves automatically building and testing code changes as soon as they are committed. This helps catch issues early and ensures that your code is always in a deployable state.

In addition to these practices, it's important to use a deployment pipeline. A deployment pipeline is a series of steps that your code changes must go through before they reach production. This can include building, testing, and deploying your code changes.

Finally, monitoring your deployments is essential for ensuring that your code changes are delivered to users quickly and reliably. This can involve monitoring logs, metrics, and user feedback to identify and fix any issues that arise in production.

By following these best practices, you can develop software that is ready for continuous deployment and ensure that your code changes are delivered to users quickly and reliably.

Shared Technologies Across Swim Lanes

This section describes the stages when developing for continuous deployment. Note that the following technologies are shared across the testing and debugging, continuous integration, and continuous deployment swim lanes.

- Unit testing
- Integration testing
- Logging

For more information about each of these technologies, refer to Testing and Debugging.

Signal Aggregation and Visibility

In the context of software development, signals are pieces of information that provide insights into the quality, reliability, and performance of a codebase. **Signal aggregation and visibility** refer to the process of collecting, analyzing, and presenting these signals in a way that is easily understandable and actionable for developers, managers, and other stakeholders.

The goal of signal aggregation and visibility is to provide a comprehensive view of the health and status of a codebase, enabling teams to identify issues, track progress, and make informed decisions about where to focus their efforts.

There are many different types of signals that can be collected and aggregated, including:

- Code quality metrics (e.g., code coverage, code complexity, code duplication)
- Test results (e.g., test failures, test duration, test flakiness)
- Build results (e.g., build failures, build duration, build artifacts)
- Deployment results (e.g., deployment failures, deployment duration, deployment status)
- User feedback (e.g., user reports, user satisfaction surveys)

To aggregate signals, teams can use various tools and techniques, such as:

- Signal hubs: Platforms that collect and display signals from multiple sources, providing a centralized view of the codebase's health.
- Dashboards: Customizable interfaces that display key metrics and signals in real-time, allowing teams to monitor progress and identify trends.
- Reports: Regular summaries of signal data, highlighting areas of improvement and providing insights into the codebase's overall health.

To improve visibility, teams can take several steps, such as:

- Establishing clear goals and metrics for each signal, ensuring that everyone understands what is being measured and why it is important.
- Providing training and support for developers, managers, and other stakeholders, ensuring that they know how to interpret and act on signal data.
- Encouraging collaboration and communication across teams, fostering a culture of transparency and accountability.

Signal aggregation and visibility are critical components of modern software development, enabling teams to make data-driven decisions and deliver high-quality products. By collecting, analyzing, and presenting signals in a clear and actionable way, teams can improve code quality, reduce risk, and increase efficiency.

Signal Aggregator

A **signal aggregator** is a tool or platform that collects and consolidates signals from multiple sources, providing a unified view of the health and status of a codebase. Signal aggregators can help teams to:

- Monitor code quality and identify areas for improvement
- Track test coverage and ensure that all critical paths are tested

- Identify build and deployment issues and optimize the release process
- Collect user feedback and sentiment analysis to inform product decisions

A signal aggregator is a valuable tool for any software development team, providing a unified view of the codebase's health and status. By collecting and consolidating signals from multiple sources, teams can improve visibility, increase efficiency, and make better decisions. When selecting a signal aggregator, teams should consider factors such as multi-source integration, data normalization, visualization, alerting, and analytics capabilities.

When it comes to selecting a signal aggregator at Meta, you have two choices, FlightDeck and WhereIsStu, both of which are described next.

Flight Deck

Flight Deck is a comprehensive platform developed by Meta for managing app releases. It integrates various functionalities to streamline the process of starting and monitoring builds, tests, and shipments. This tool is particularly useful for release managers and operators, providing a centralized hub for overseeing the release process.

Key Features of Flight Deck

- **Build and CI Management**: Flight Deck supports the management of build processes and continuous integration, ensuring that builds are executed efficiently and consistently.
- **Testing and Signal Tracking**: It facilitates the tracking of test results and other signals, which are crucial for assessing the health of the release.
- **Deployment and Shipment**: Flight Deck allows for the configuration and execution of shipments to various platforms, making it easier to manage releases across different environments.
- **Configurability and Logging**: The platform offers a high degree of configurability and detailed logging, which helps in troubleshooting and fine-tuning the release processes.

Operational Use

- **Monitoring Launch Blockers**: It provides tools to identify and manage launch-blocking issues that could affect the release schedule.
- **Version Management**: Flight Deck includes mechanisms for version bumping and managing different release versions effectively.
- **Manual and Automated Shipping**: Users can manually ship releases or configure auto-shipments based on specific criteria, enhancing flexibility in release management.
- **Rollout Management**: The platform supports phased rollouts, allowing for gradual release to users which can be adjusted based on real-time feedback and metrics.

To use Flight Deck, you will need to have the appropriate permissions and access to the platform. Once you have access, you can follow these steps:

1. Navigate to the Flight Deck page for your app.
2. Select the release version you want to manage from the list of available releases.
3. View the details of the selected release, including build status, test results, and shipment information.
4. Use the available actions to manage the release, such as starting a new build, running tests, or shipping the release to a specific destination.

It is important to note that the exact steps and options may vary depending on the specific configuration and settings of your app in Flight Deck. If you have any questions or issues with using the platform, you can reach out to the Flight Deck support team for assistance.

Integration with Meta's Ecosystem

Flight Deck is integrated with various Meta tools and platforms, ensuring seamless operation within Meta's ecosystem. It supports a wide range of apps, making it a versatile tool for release management across the company.

For detailed information on working with Flight Deck, refer to the Flight Deck wiki.

WhereIsSTU

WhereIsSTU is a tool used to monitor and manage the Reality Labs Safe-To-Use (STU) CI system. It provides various pages to track the status of OS builds, APK builds, and the ingestion process that bundles STU APK builds into new OS builds. Additionally, it includes a Test Explorer Page for browsing all the Gauntlet E2E tests run during OS and APK CI runs. The tool can be accessed at https://www.internalfb.com/whereisstu and its various subpages for different functionalities [source].

To use WhereIsSTU, follow these steps:

1. Go to the WhereIsSTU page: https://www.internalfb.com/whereisstu
2. Select the product you want to check from the drop-down menu at the top of the page.
3. You will see a list of recent STU builds, with the latest build at the top.
4. Click on a build number to see more details about that build, including its date, test results, and any known issues.

By using WhereIsSTU, you can quickly find the latest STU build for your product and ensure that you are testing on a stable and reliable build.

For detailed information on working with WhereIsStu, refer to the WhereIsSTU wiki.

Choosing a Signal Aggregator

Given the broader range of functionalities and the integration with other tools, Flight Deck would be the recommended choice if you are looking for a comprehensive tool to manage releases, monitor

builds, and handle shipments. It provides a centralized platform that can cater to various needs, making it more versatile for different tasks related to app release and management.

However, if your specific requirement is only to track and locate specific builds or versions, WhereIsStu might be sufficient for that narrower purpose.

Release Candidate Testing

Release Candidate (RC) Testing is a crucial phase in the software development lifecycle, aimed at ensuring that new software versions are robust and ready for production. The main goal of RC testing is to conduct end-to-end (E2E) testing of the software to identify and fix any potential issues before the software is released to the public.

Here are some key aspects of release candidate testing:

1. **Selection of Release Candidates**:
    - The QA team identifies potential release candidates based on previous testing results. For instance, specific software (SW) and operating system (OS) builds are selected for extended testing if they show promise as good release candidates [source].
2. **Testing Process**:
    - RC testing involves rigorous E2E testing. This includes running the software through all its functionalities to ensure it behaves as expected in a production-like environment.
    - Automated and manual testing methods are employed to cover various aspects of the software.
    - Continuous testing frameworks and tools like BOLT are used to ensure the software's trunk is stable before moving to release candidate stages [source].
3. **Monitoring and Feedback**:
    - Tools and dashboards are used to monitor the software's performance during the RC phase. Issues such as crashes are tracked to assess the stability of the release candidate.
    - Feedback mechanisms are in place to quickly address any critical issues found during testing [source].
4. **Deployment and Rollback**:
    - Once a release candidate passes all tests, it is scheduled for deployment. Deployment can be staged to roll out the software gradually to monitor its performance in the live environment.
    - Rollback procedures are established to revert to previous versions if significant issues arise after the release [source].
5. **Developer and Community Involvement**:
    - Developers are involved in the RC testing phase to ensure that the software meets all technical standards and requirements.
    - In some cases, a broader community of users may also participate in testing release candidates to provide additional feedback and identify issues that may not have been caught during internal testing [source].

In summary, Release Candidate Testing is a comprehensive process that involves selecting potential release candidates, conducting thorough testing, monitoring software performance, and preparing for deployment and potential rollback. This phase is critical to ensure that the software is stable, functional, and ready for wider release.

Developer Preview Channel

The **Developer Preview Channel (DPC)** is a system designed to provide developers with early access to over-the-air (OTA) firmware updates for Quest devices, allowing them to receive updates earlier than the general public. This early access enables developers to test and report issues before the updates are rolled out to the public, creating a symbiotic relationship where developers help improve the stability and quality of the firmware, and Oculus gains valuable insights and feedback on potential issues [source].

The DPC has been managed manually by collecting device serial numbers from developers and changing their OTA channel using the Firmware OTA Device Admin Tool. However, due to the increasing scale of managed developers, this manual process has become unsustainable, prompting the need for an automated or self-service solution [source].

The release calendar for the DPC has been adjusted to provide developers with a sufficient preview of OS updates before the public release. This adjustment aims to aid in finding regressions before the release, with the branch cut now scheduled for the 12th of every month, adding an extra week between the branch cut and the release [source].

Overall, the Developer Preview Channel is a critical component in ensuring the quality and stability of firmware updates for Quest devices by involving developers in the testing process early on.

To access the Developer Preview Channel (DPC) for Oculus software, you need to follow these steps:

1. Contact Developer Relations Engineering: According to the information provided, you should talk to your Developer Relations Engineering contact to request access to the DPC. This is mentioned as a way for developers to get early access to preview OS releases to test their applications against new OS releases and provide feedback prior to the public launch [source].
2. Provide Headset Serials: For developers to be opted into the DPC, the Content team or developers need to provide headset serials (Quest 1 / Quest 2). This process is managed by the team, and currently, there is a list of several thousand headsets that are part of this program [source].
3. Wait for Approval and Setup: Once you have contacted the appropriate contacts and provided the necessary information (like headset serials), you will need to wait for your request to be processed and for your devices to be set up to receive DPC builds.

It's important to note that the process might involve specific agreements or additional steps depending on your status as a developer and the nature of the projects you are working on. Make sure to maintain communication with your Developer Relations Engineering contact throughout this process.

Public Test Channel

The **Public Test Channel (PTC)** is a platform where updates and new features are rolled out to a subset of users who opt-in to test the latest versions before they are released to the general public. This allows developers to gather feedback and identify any potential issues or bugs. The PTC is part of a broader release process that includes several stages such as planning, development, feature complete, branch cut, public test channel, gold master candidate, and release. Users who participate in the PTC can experience the newest features and provide valuable feedback to improve the product [source].

The PTC is used not only for testing stability and performance but also for gathering telemetry data to ensure that new updates perform well under various conditions. This channel is crucial for catching issues that may not have been identified during earlier testing phases. For instance, in the context of Oculus software, opting out of the PTC has been recommended to resolve specific issues with third-party applications like Microsoft Flight Simulator, indicating the channel's role in identifying and addressing compatibility issues [source].

Overall, the Public Test Channel serves as an essential step in the development and release process, providing a controlled environment to validate changes and enhance the overall quality of the software before it reaches a broader audience.

To access the Public Test Channel (PTC) for Oculus software, follow these steps:

1. Make sure you have an Oculus account: If you don't have one, create an account on the Oculus website.
2. Join the Oculus PTC: Go to the Oculus PTC page and click "Join PTC" in the top-right corner. You'll need to sign in with your Oculus account if you're not already logged in.
3. Enable PTC on your headset: On your Oculus Quest or Quest 2 headset, go to Settings > System > Software Update, and toggle the switch next to "Public Test Channel" to enable it.
4. Restart your headset: Your headset will automatically download and install the latest PTC build.

Note: The PTC is only available for Oculus Quest and Quest 2 headsets. If you have an Oculus Rift or Rift S, you won't be able to access the PTC.

Once you've joined the PTC, you'll receive early access to new features and updates before they're released to the general public. Keep in mind that PTC builds may be less stable than the public release, so you might encounter bugs or issues. However, your feedback is crucial in helping the Oculus team improve the software before its official release.

Release Operation and Monitoring

Release Operation and Monitoring involves various tools and processes to ensure the smooth rollout and health of software releases. Here are some key aspects based on the provided resources:

1. **Release Operators**: These individuals are responsible for operating the release of new production packages and ensuring their health. They use developed tools to monitor

the health of the release and to triage issues that occur, collaborating with AI systems developers to apply necessary fixes [source].

2. **Tools and Frameworks**: Various tools like FlightDeck and Cockpit are used to manage operational tasks during gradual rollouts, track regressions, monitor releases, alert on issues, and provide recommendations for fixes. These tools help in reversing the interaction with release engineering tooling, pushing information and fixes to release operators [source].
3. **Release Beast Tool Set**: This is a collection of UI and automation tools that support release operations. It includes tools for tracking the high-level status of the release state, managing pick requests, and validating product health after each rollout stage [source].
4. **Monitoring and Alerting**: Monitoring the health of releases is crucial. For instance, the Messenger Desktop Release Dashboard is used to monitor key metrics of new releases, compare them with previous releases, and set guardrails for metrics to ensure quality [source].
5. **Automated and Manual Release Operations**: Systems like the Release Manager (RM) manage release operations with capabilities like rollout and rollback automation, manual operations (e.g., hotfix, package pin), and instrumentation and auditing with a release metadata database [source].

These components collectively ensure that release operations and monitoring are conducted efficiently, with tools and processes in place to handle various scenarios that might arise during the lifecycle of a software release.

When it comes to release operation and monitoring, you can use Release Beast, which is described next.

Release Beast

Release Beast is a comprehensive tool used by Reality Labs (RL) to manage the release process and information for pushing major products to production. It provides a full picture of the release process, broken down by major product and release version. Release Beast includes several bots that communicate information through posts, messages, and tasks, assisting release engineers, release captains, TPMs, stakeholders, and product teams in various aspects of the release process.

Key functionalities of Release Beast include:

- **Checklist for Release Engineers**: It helps release engineers to check list their way through all the steps needed for a release.
- **Assistance for Release Captains/TPMs/Stakeholders**: It aids in answering questions and automating some of their work.
- **Overview for Product Teams**: Gives a full picture of the status of the release.

Release Beast also integrates various tools and features to support release operations, such as:

- **Release Overview**: Tracks the high-level status of the release state.
- **Release Details**: Tracks the full release playbook and identifies critical roles and various stages to ship a release.

- **Release Notes**: Automatically generates release notes based on top of roadmap tasks.
- **Telemetry Signoff Automation**: Validates product health after each gradual rollout stage.
- **Release Assistant Bot**: Manages release-related notifications and updates.
- **Shiproom Requests Tool**: Provides a centralized platform that ties launch blockers to pick requests.

For more detailed operations, Release Beast is used to sign off on different types of releases, including Go releases, hotfixes, security patches, and enterprise releases. It is also involved in the process of picking diffs into release branches, where it automates the creation of shiproom posts and assists in managing the pick requests.

To use Release Beast, you can follow these steps:

1. Go to the Release Beast website: https://www.internalfb.com/intern/release_beast/
2. Select the product you want to view or manage from the list on the left-hand side of the page.
3. Click on the "Releases" tab to view information about upcoming and past releases.
4. Click on the "Launch Blockers" tab to view information about any issues that are blocking the release.
5. Click on the "Calendar" tab to view a calendar of upcoming releases and events.
6. Click on the "Details" tab to view detailed information about a specific release, including release notes, rollout plans, and more.

By using Release Beast, teams at Meta can more easily track and manage their releases, identify and resolve launch blockers, and communicate important information to stakeholders.

Overall, Release Beast is essential for streamlining and automating the release process within Reality Labs, ensuring efficient and coordinated product releases.

For more specific details, you can visit the Release Beast Wiki page directly through this link: Release Beast Wiki.

Release

When it comes to releasing at Meta, your primary tool is Firmware OTA, which is described next.

Firmware OTA

Firmware OTA (Over-The-Air) is a backend system used to manage and serve remote updates for devices, supporting updates of both OS images and application-level packages like APKs and custom language packs. This system allows product teams to push updates directly to devices quickly and seamlessly. Key features of Firmware OTA include:

- **Self-Service OTA Channels**: Product teams can create and manage their own OTA channels for different use cases such as dogfooding, external users, or testing.

- **Version Management**: Different upgrade path options can be defined to support various requirements, such as incremental releases or mandatory updates between specific versions.
- **Security**: Updates are served only if the client provides a properly authorized device token. If the device token is not valid, no data is returned to the client.
- **Client Event Reporting**: The system is capable of collecting OTA update events from the client for generating metrics or debugging issues seen during the update.
- **WebUSB Tools**: Provides tools for sending ADB commands to the device through the browser without additional drivers or programs, including utilities for users who prefer not to use ADB directly.

Users interact with the OTA system primarily through OTA channels that hold all their OTA releases. They can create OTA channels, OTA releases within those channels, and manage the OTA channel of devices they own or administer [source].

For more detailed management, OTA channels can be configured with options like testing, dogfooding, or production, each with specific settings for channel optimization and auto-cleanup times to manage the lifecycle of releases within the channel [source].

Overall, Firmware OTA is a comprehensive system designed to facilitate the efficient and secure distribution of firmware and software updates to devices over the air.

To use Firmware OTA, you can follow these general steps:

1. **Create an OTA Channel**: Go to the Firmware OTA page and click on "Create Channel" in the top right corner. Fill out the required information, such as the channel name, description, and organization.
2. **Create an OTA Release**: Once you have created a channel, you can create a new release by clicking on the "Create Release" button within the channel. You will need to provide information about the release, such as the build number, base version, and target version.
3. **Upload the Firmware**: After creating the release, you will need to upload the firmware file(s) for the update. This can be done using the "Upload File" button within the release details page.
4. **Configure the Release**: You can configure various settings for the release, such as the rollout percentage, release notes, and testing status.
5. **Assign Devices to the Channel**: To enable devices to receive updates from your channel, you will need to assign them to the channel. This can be done using the "Device Admin" tool or through the device's settings menu (if supported).
6. **Monitor and Manage the Release**: Once the release is live, you can monitor its progress and manage it through the Firmware OTA dashboard. You can view metrics such as update success rates, failure rates, and more.

For more detailed instructions and specific use cases, refer to the Firmware OTA wiki and its subpages.