USE CASE

CI/CD Pipeline Automation

Infrastructure Insight & Management

Orchestral.ai
AI-Driven Orchestration

Situation: CI/CD Pipeline with Manual Steps

- Manual handling of code throughout the development life-cycle: Coding, QA/Test, Build, Deployment.
- Lack of visibility throughout the release process life-cycle and manual test intervention risks extending time to deploy.
- Releases process becomes stressful when excessive manual intervention introduces the risk of human error that forces unnecessary and costly rework and time delays.
- A DevOps approach to release process management is challenged to achieve the true potential of CI/CD if significant manual intervention remains part of the process.

Composer Benefits

- **Higher quality code**
  Composer-automated CI/CD pipelines eliminate human errors and results in higher quality through a more efficient release process.
- **Faster release cycles**
  Eliminating manual intervention also results in dramatically faster release cycles as each pipeline stage completes the next initiates automatically.
- **Improved performance**
  Composer's event-driven architecture supports the entire CI/CD pipeline with real-time insight into performance or resource utilization issues.

Manual CI/CD Release Process Management

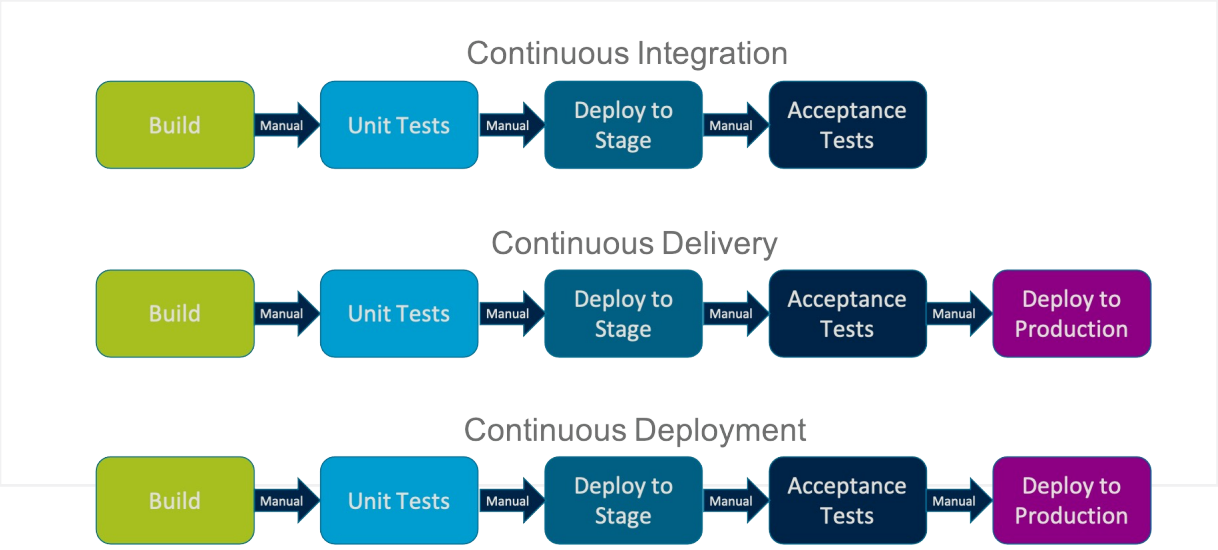

FIGURE 1: Manual CI/CD Release Process Management

Orchestral.ai's Composer Solution

Composer provides out-of-the-box integration with hundreds of tools, apps devices and services used in DevOps and/or CI/CD pipeline environments. Composer provides an over-arching orchestration engine that integrates and automates the entire end-to-end CI/CD pipeline without the need to "rip and replace". Composer's event-driven architecture ensures the appropriate actions, notifications and alerts are triggered in real-time for fluid, reliable and efficient CI/CD release process management.

Automated CI/CD Release Process Management

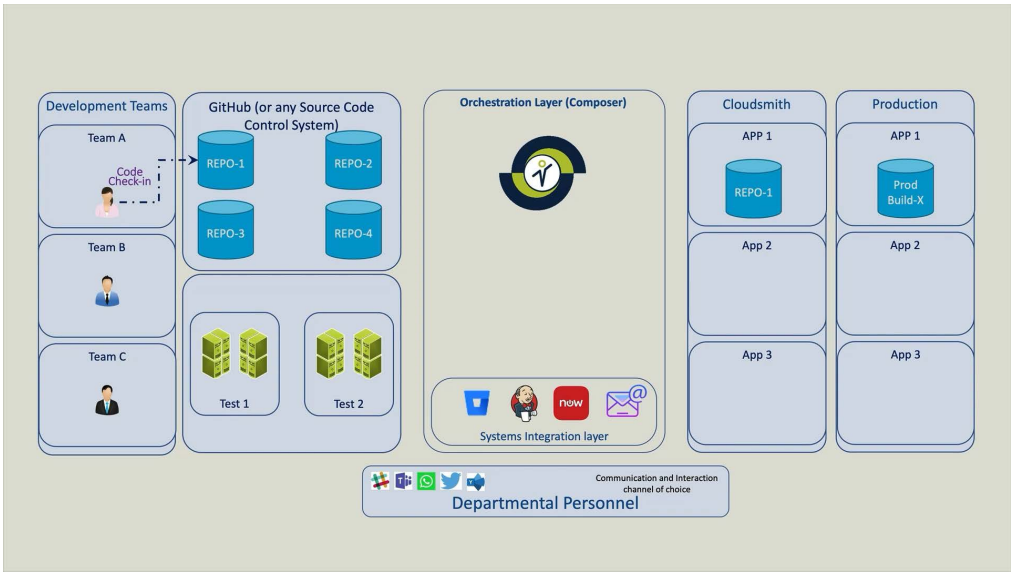

FIGURE 2: Automated CI/CD Release Process Management with Orchestral Composer

Composer CI/CD Pipeline Automation

1. Everything starts when a developer checks in code, which could be via a merge or a pull request.
2. That code check-in will be acknowledged by a Composer sensor at the integration layer of the code repository.
3. This acknowledgment of a code check in will send a trigger to Composer to start the testing stage.
4. Composer will reserve a test bed and pull the Branch to the test bed.
5. The test bed will then request the that was attached to the specified branch from the code repository.
6. The branch will be fetched to the test bed.
7. Upon successful preparation of the test bed a webhook will be kicked to Composer specifying the branch pull was a success.
8. Composer will then tell a test server such as Jenkins to run predefined tests against the new branch in the test bed.
9. The test server such as Jenkins will start the specified tests by installing the branch and running the tests.
10. The CI testing will then initiate.
11. Whether successful or not, Composer will trigger an email to the developer team to ensure they are aware of the status and progress of the code.
12. If successful, Composer will then start the build process and tell the code repository to move the branch to the next stage whether that be pre-production or production
13. The code repository such as GitHub will then move the branch to a package management tool such as Cloudsmith or PackageCloud.
14. Finally, if the repository passes all dependency tests and build tests, then Composer will move the new branch to Production.

Orchestral.ai
AI-Driven Orchestration

Orchestral.ai is a team of like-minded technology professionals possessing a combined experience of over 100 years in the IT industry.

Contact Us

For more information, please contact our Client Development Team at **info@orchestral.ai**