CTO,

I'd like to address the specific concerns you detailed in your recent CI/CD memo. I understand that you are detail-oriented, so I have provided some technical details on how CI/CD would help our organization. Thank you for your consideration of these points.

We have too many customers to configure CI/CD for - at Citrine every customer gets their own AWS account

The fact that all Citrine customers use AWS makes CI/CD a simple AWS configuration exercise. All AWS accounts have the AWS Code Deploy service at their disposal. AWS Code Deploy enables the deployment and update of running applications. All that is needed to enable AWS Code Deploy as part of a CI/CD solution is:

1) Configure all customer AWS accounts with a Citrine-only AWS IAM user. The Citrine-only IAM user possesses permissions only to deploy services with AWS Code Deploy and post updates to AWS S3.
2) Write a shell script to create a Code Deploy Application and Deployment Groups in all customer AWS environments. This script will run with the Citrine-only IAM user permissions.
3) Configure a CD server to deploy Citrine updates to each customer AWS environment using AWS Code Deploy

The AWS CodeDeploy service is very convenient, easy to use, and easily integrates into existing CD tools.

We should release our code at the end of every sprint

Limiting releases to specific timelines does not eliminate many of the benefits of CI/CD. AWS Code Deploy defines Deployment Groups to enable segregated deployments. Examples of segregated deployments are: 'testing', 'staging', 'pre-release', and 'production. Configuring different Code Deploy Deployment Groups allows us to reap the benefits of CI/CD while preserving the ability to release to a 'production' Deployment Group at the end of a sprint.

Each team uses different technologies

Modern development and deployment technologies largely alleviate this issue. For example, most modern applications leverage container technologies like Docker. Docker enables developers to *configure-as-code* the details of the technology used in their solutions. Code that is deployed with Docker runs in a container that is identical to the container in which the developer coded the solution.

• We can't afford to take engineers off of product work to work on it

The short-term cost of having engineers set up our CI/CD system will pay off in the very near-term. Here are a few reasons why.

- One of the biggest benefits of CI is reduced waste and increased productivity in the software delivery pipeline. A significant portion of the cost in releasing software comes from the progress of software through testing and operations. The ability to reduce waste and increase productivity makes CI very affordable process.
- A benefit of CD is the shortening of the feedback cycle between development, test and deployment. Without CD, development teams often receive bugs long after the team has moved on to new or different functionality. The reduced feedback cycle from CD decreases the need to take engineers off of current product work to address critical production bugs.

In summary, CI/CD is an efficient practice with cost-reduction and productivity-increasing benefits. Please consider funding a CI/CD budget for our organization.

Best Regards,

Travis Brown