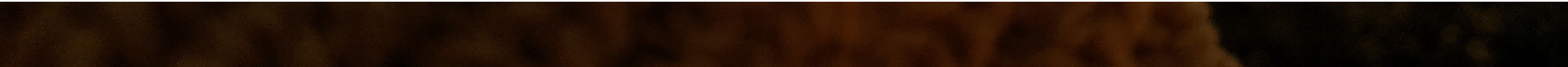
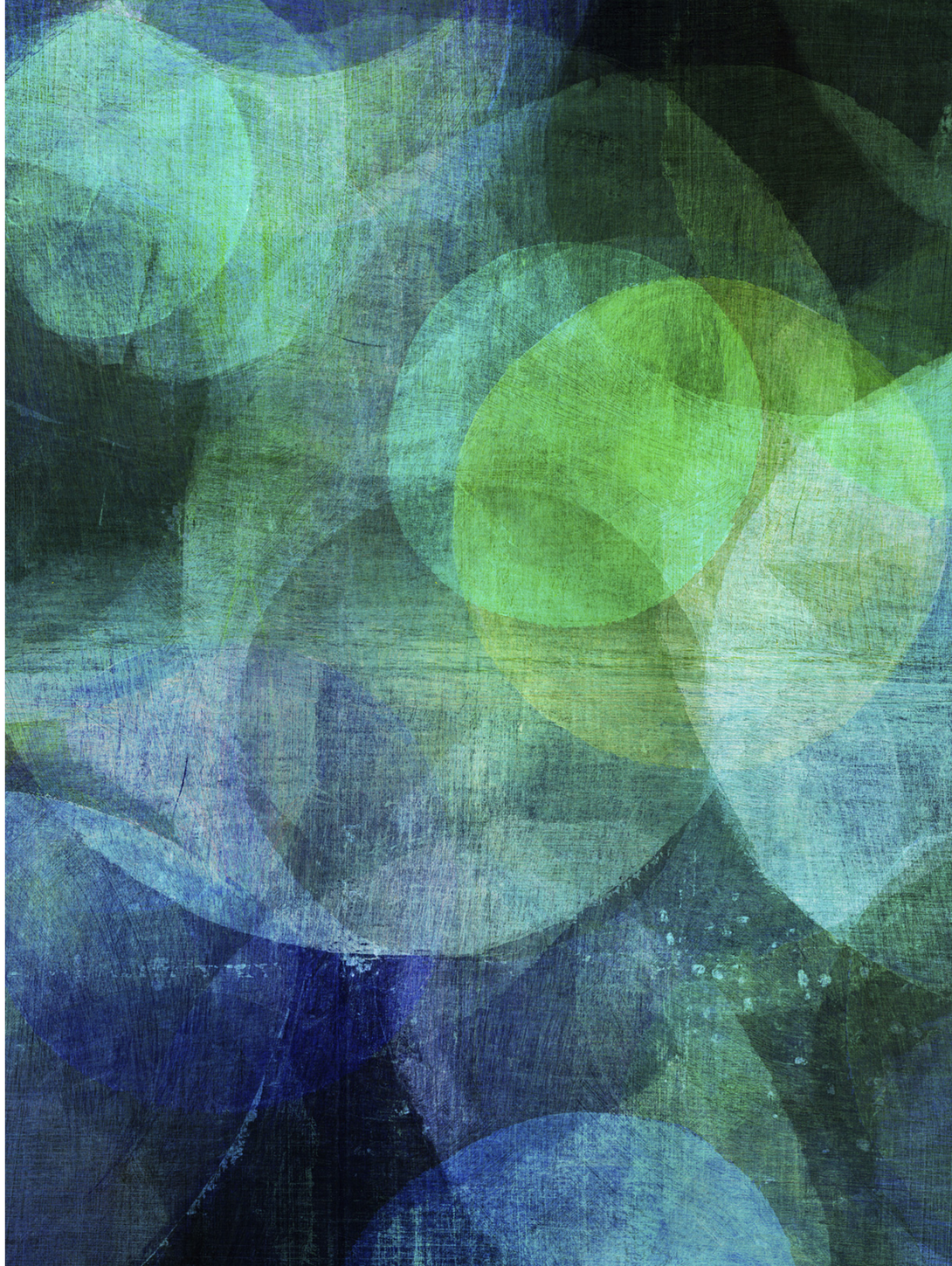
INTRODUCTION TO DOCKER

INTRODUCTION TO DOCKER

Docker in Continuous Integration

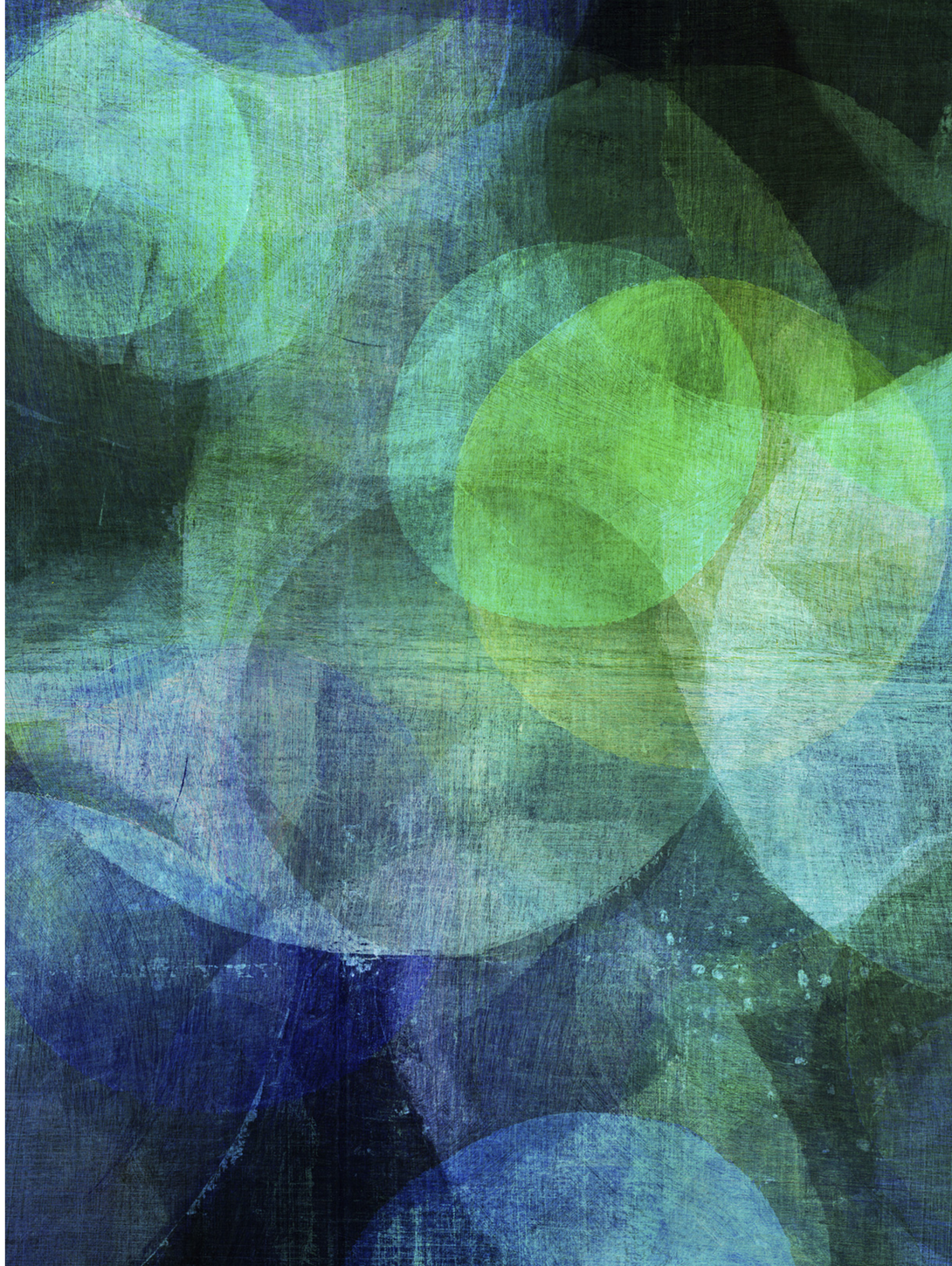


AGENDA



AGENDA

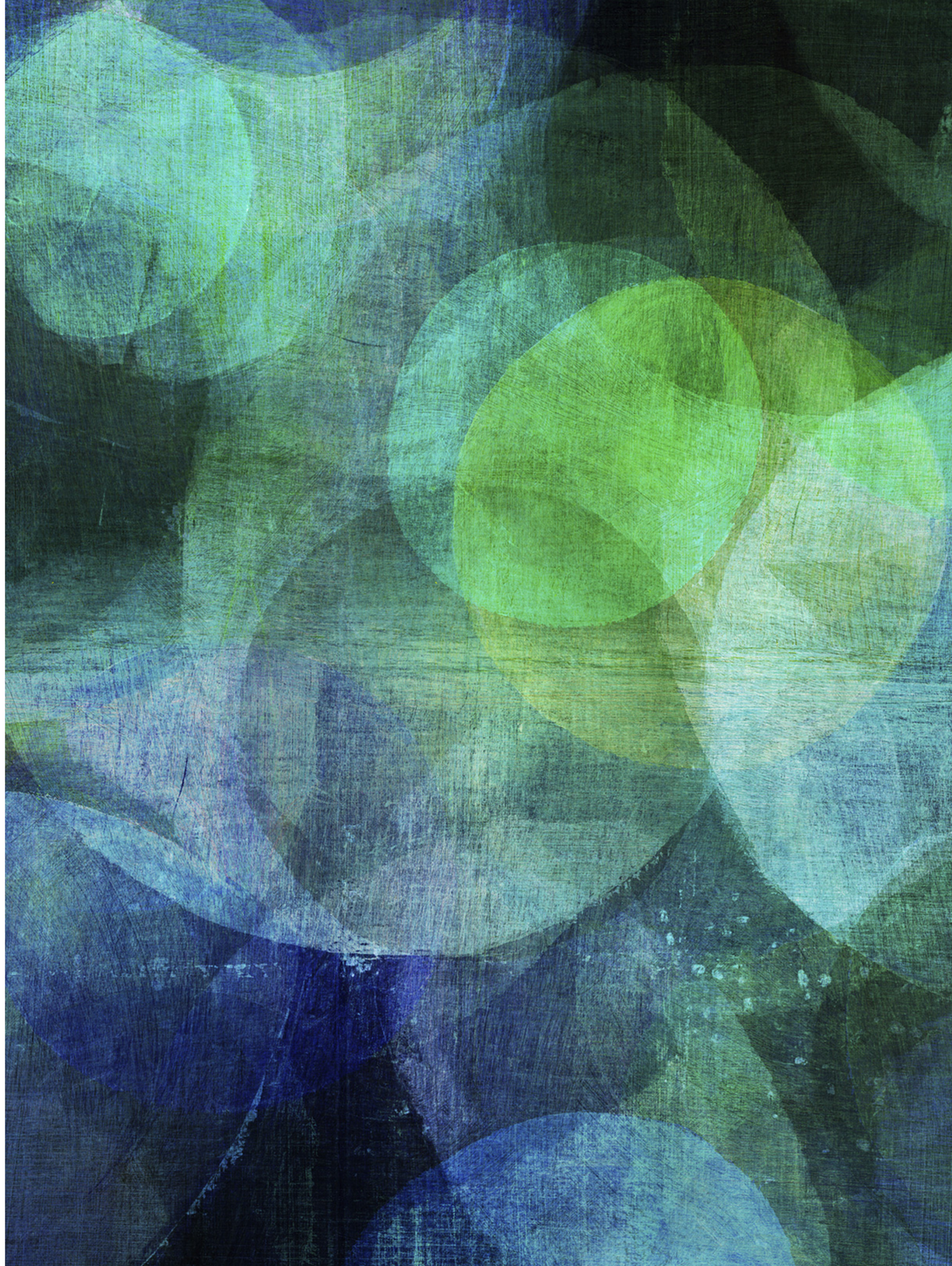
- Continuous Integration



AGENDA

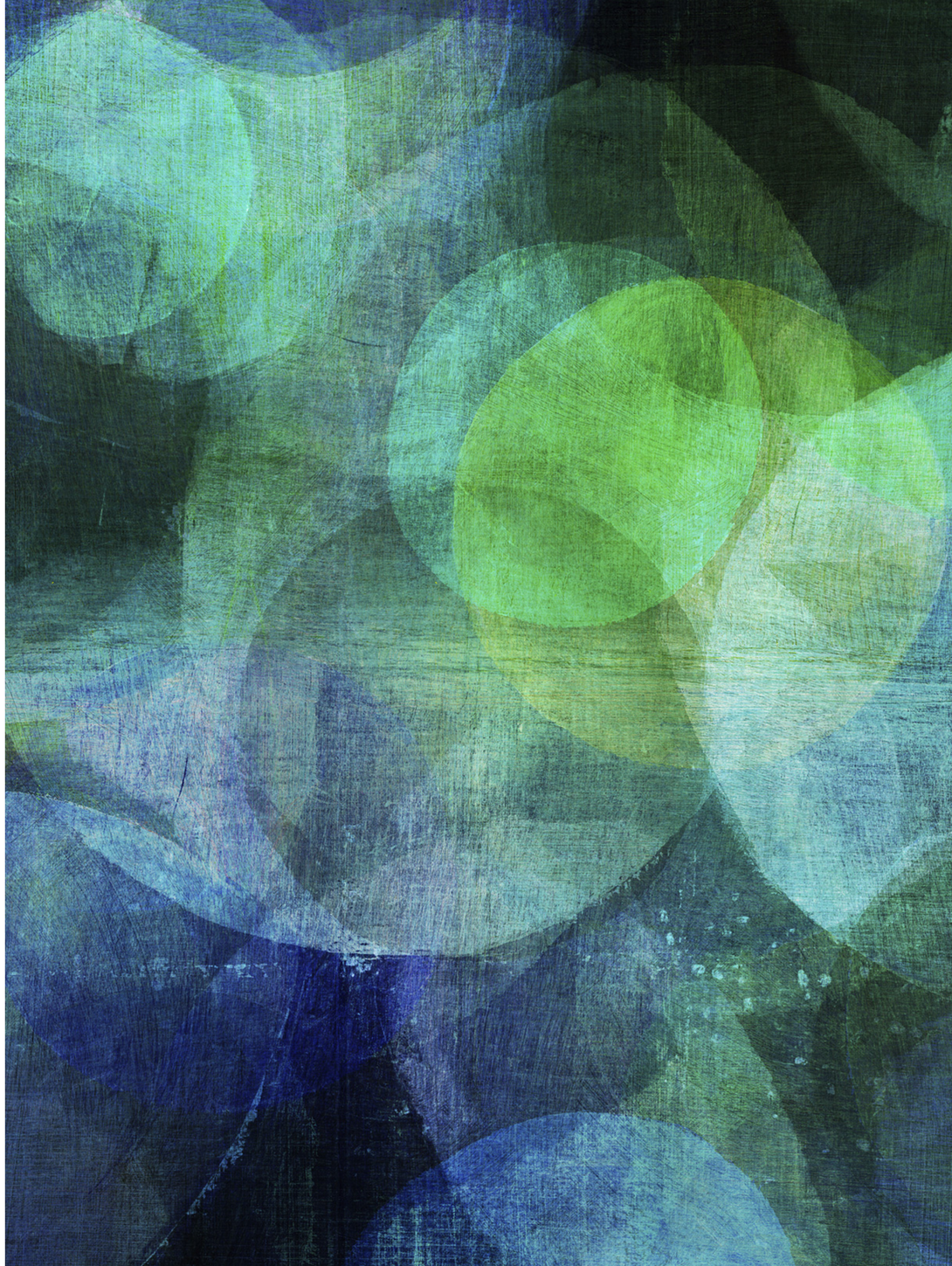
- Continuous Integration

- Build Pipeline



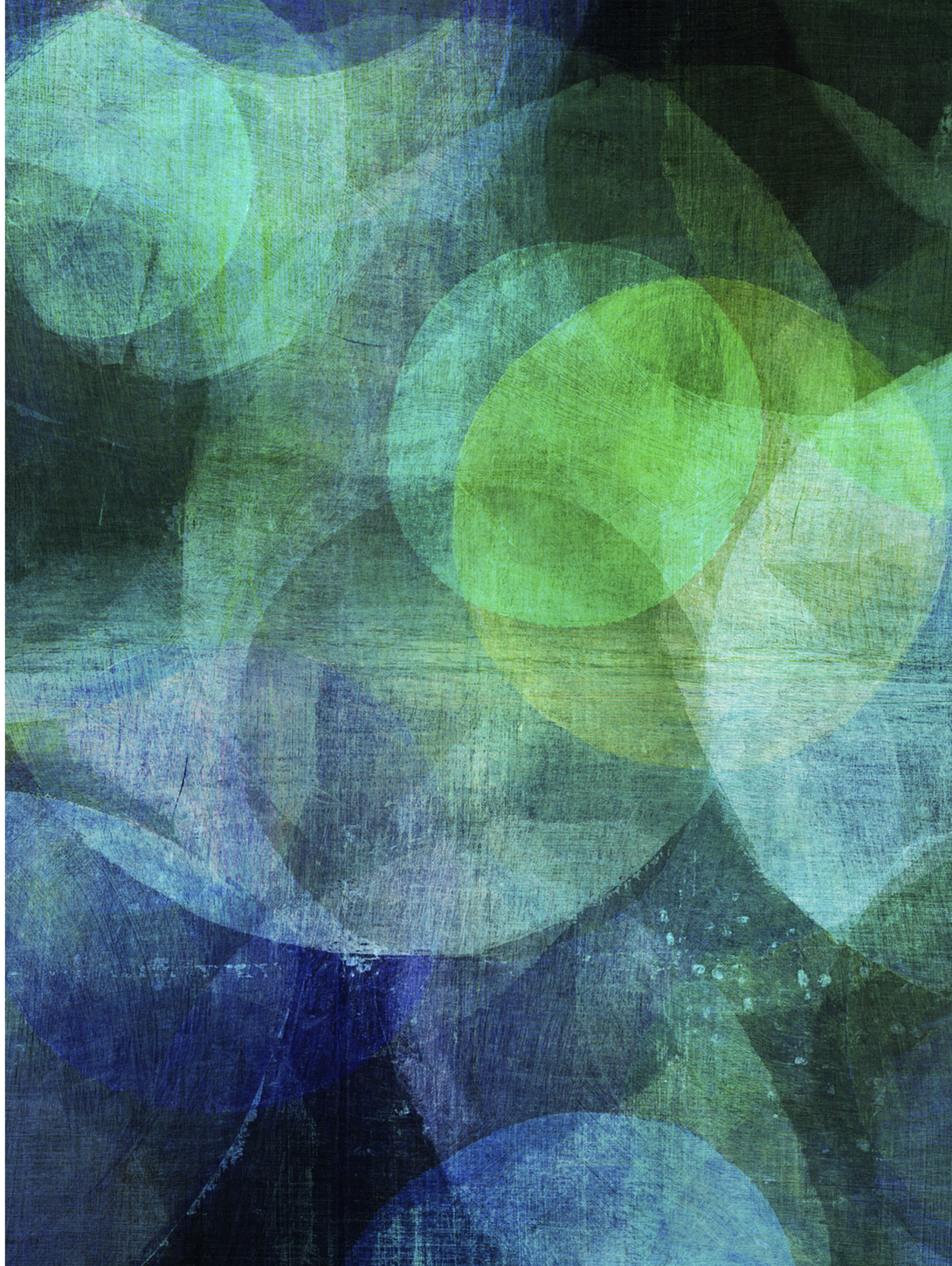
AGENDA

- Continuous Integration
- Build Pipeline
- Test Pipeline



AGENDA

- Continuous Integration
- Build Pipeline
- Test Pipeline
- Deploy application with docker

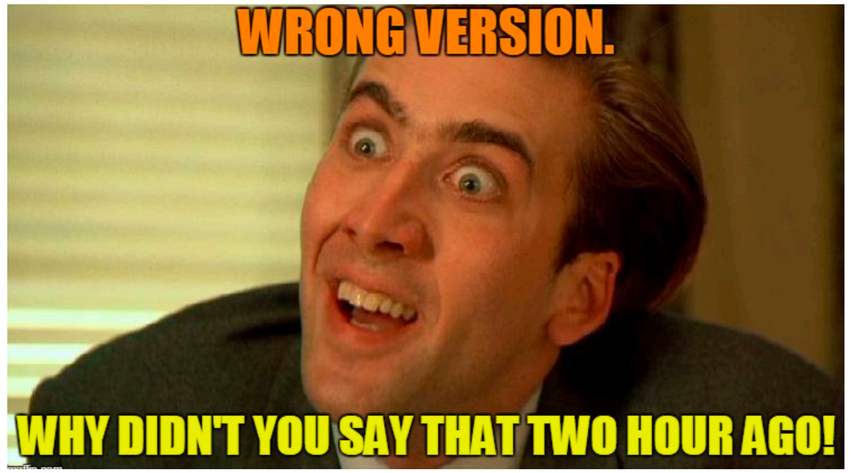


CONTINUOUS INTEGRATION

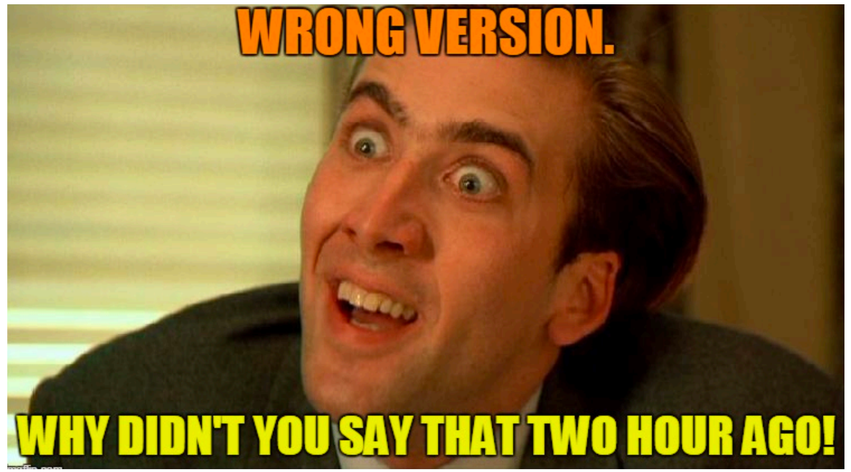
COMMON PROBLEM IN SOFTWARE DEVELOPMENT

COMMON PROBLEM IN SOFTWARE DEVELOPMENT

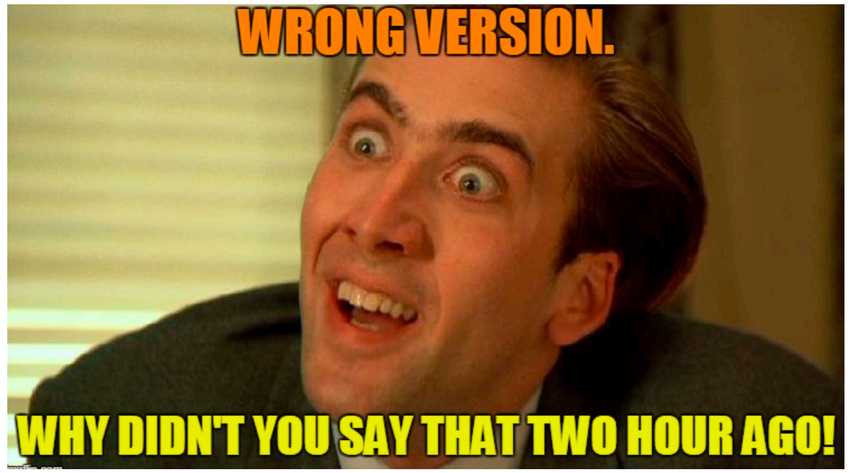
COMMON PROBLEM IN SOFTWARE DEVELOPMENT

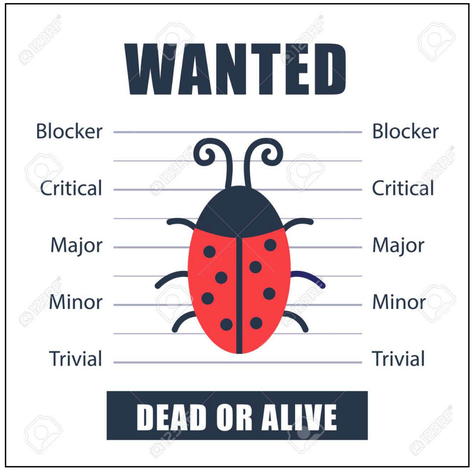


COMMON PROBLEM IN SOFTWARE DEVELOPMENT

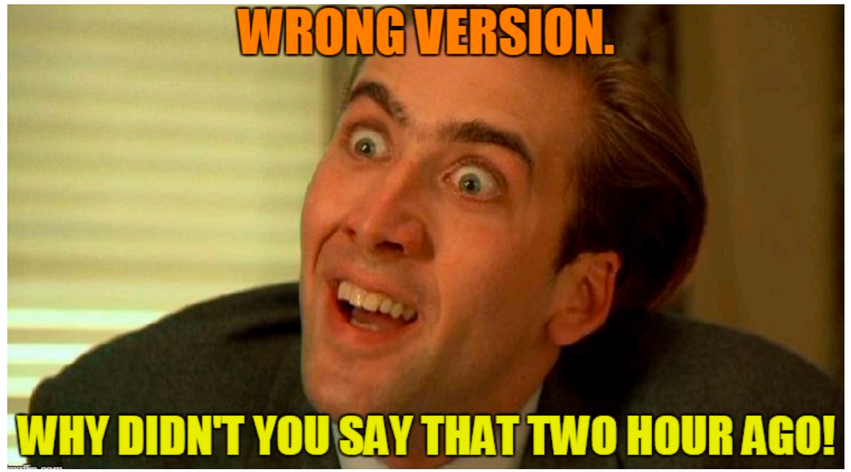


COMMON PROBLEM IN SOFTWARE DEVELOPMENT

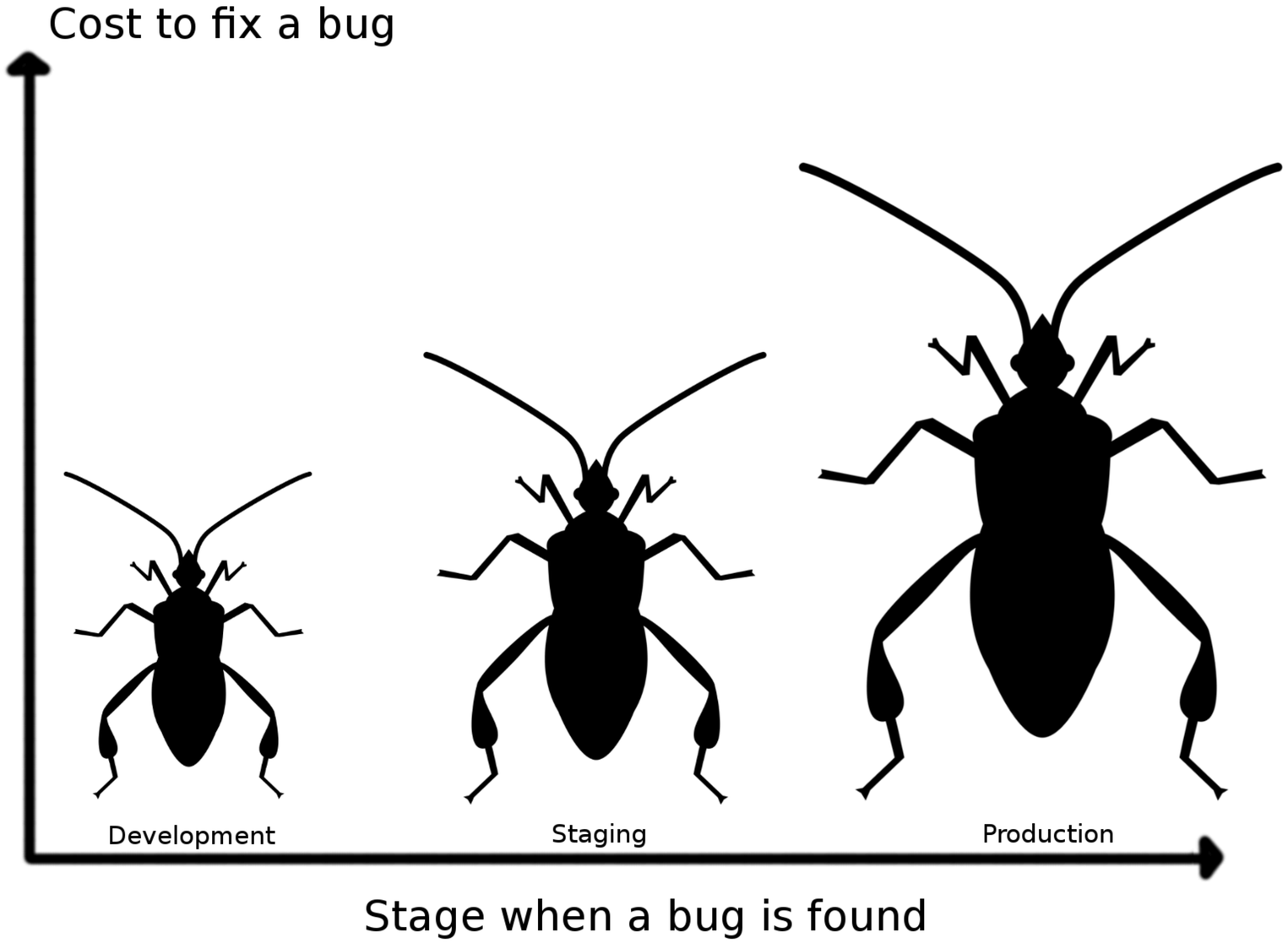




COMMON PROBLEM IN SOFTWARE DEVELOPMENT



COST OF INTEGRATION



KEY OF CONTINUOUS INTEGRATION

KEY OF CONTINUOUS INTEGRATION

Discipline to **integrate frequently**

KEY OF CONTINUOUS INTEGRATION

Discipline to **integrate frequently**

Strive to make **small change**

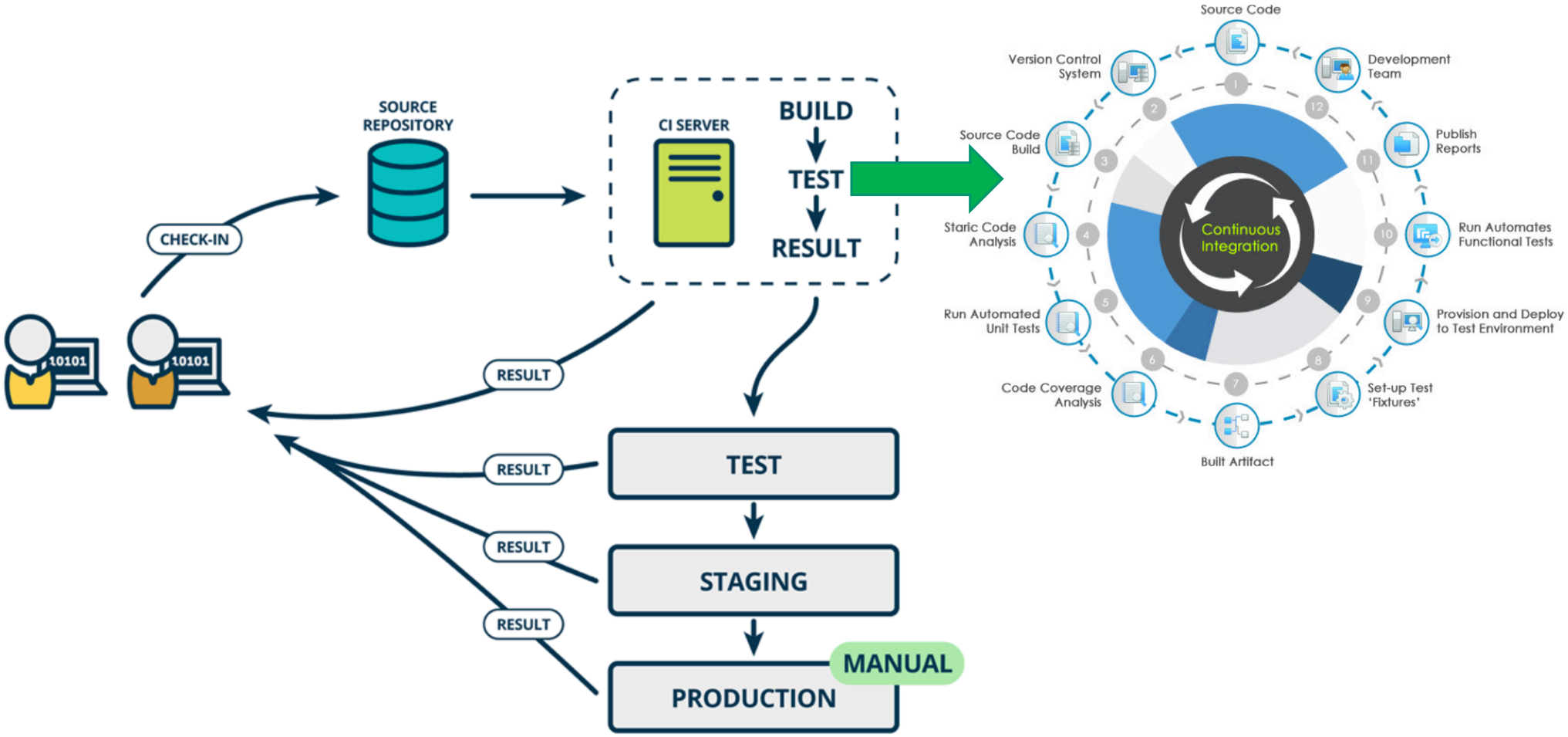
KEY OF CONTINUOUS INTEGRATION

Discipline to **integrate frequently**

Strive to make **small change**

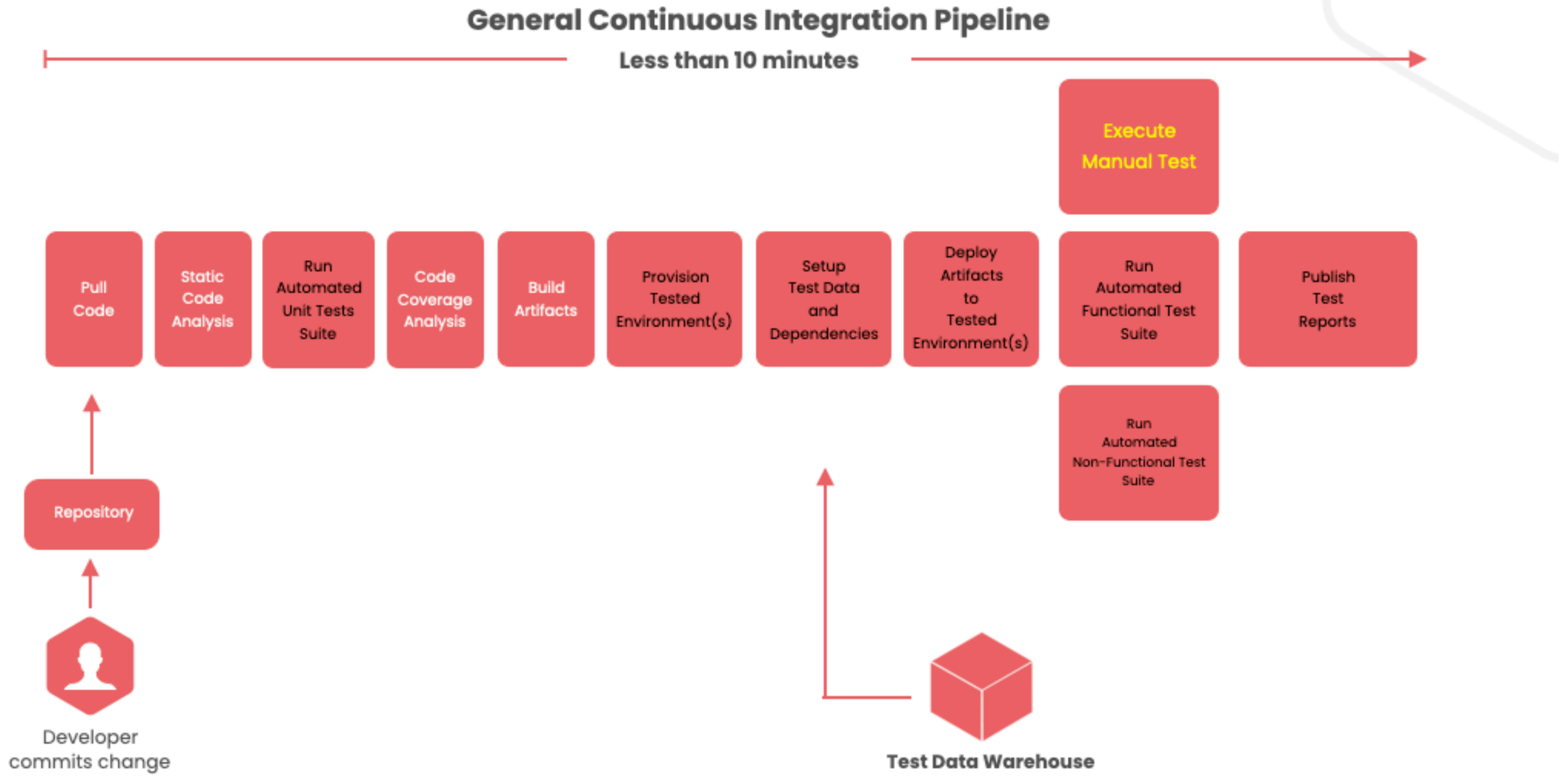
Strive for **fast feedback**

CONTINUOUS INTEGRATION AND CONTINUOUS DELIVERY



DEVELOPMENT CYCLE

THE CONTINUOUS INTEGRATION



VISUALISING YOUR PIPELINE – INCLUDING THE MANUAL STAGES

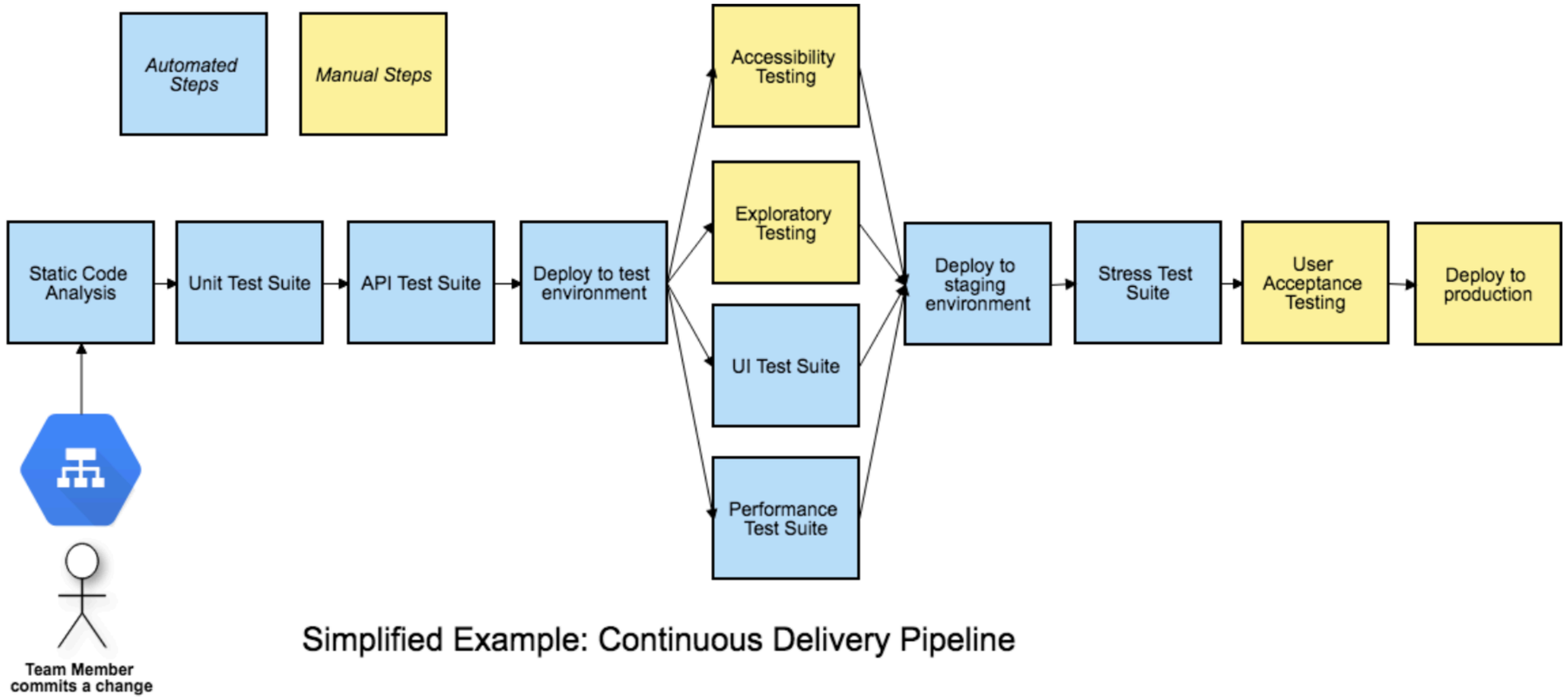


Figure 1: A Continuous Delivery Pipeline (my diagrams are inspired by Katrina Clokie's pipeline diagrams in "A Practical Guide to Testing in DevOps")

VISUALISING YOUR PIPELINE – INCLUDING THE MANUAL STAGES

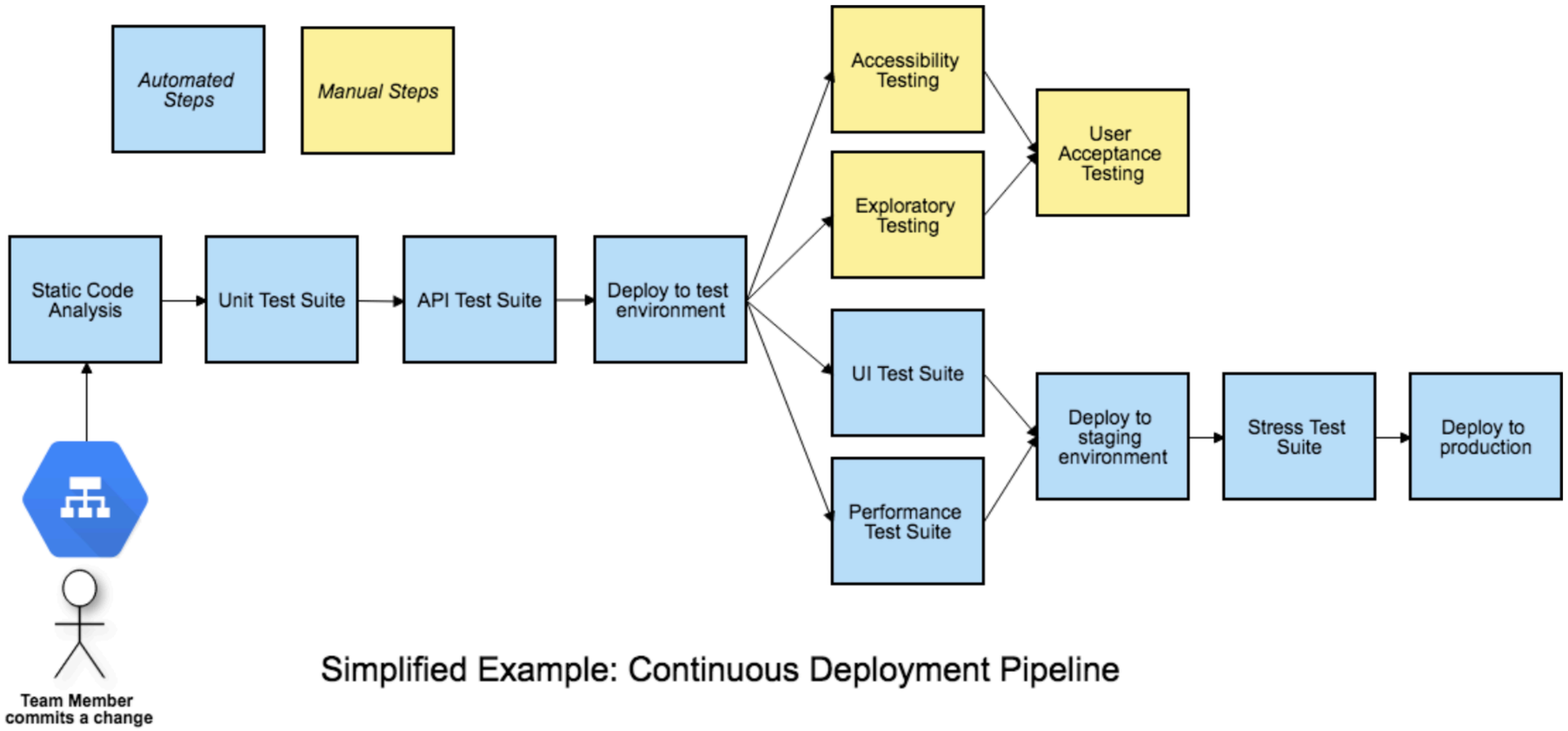
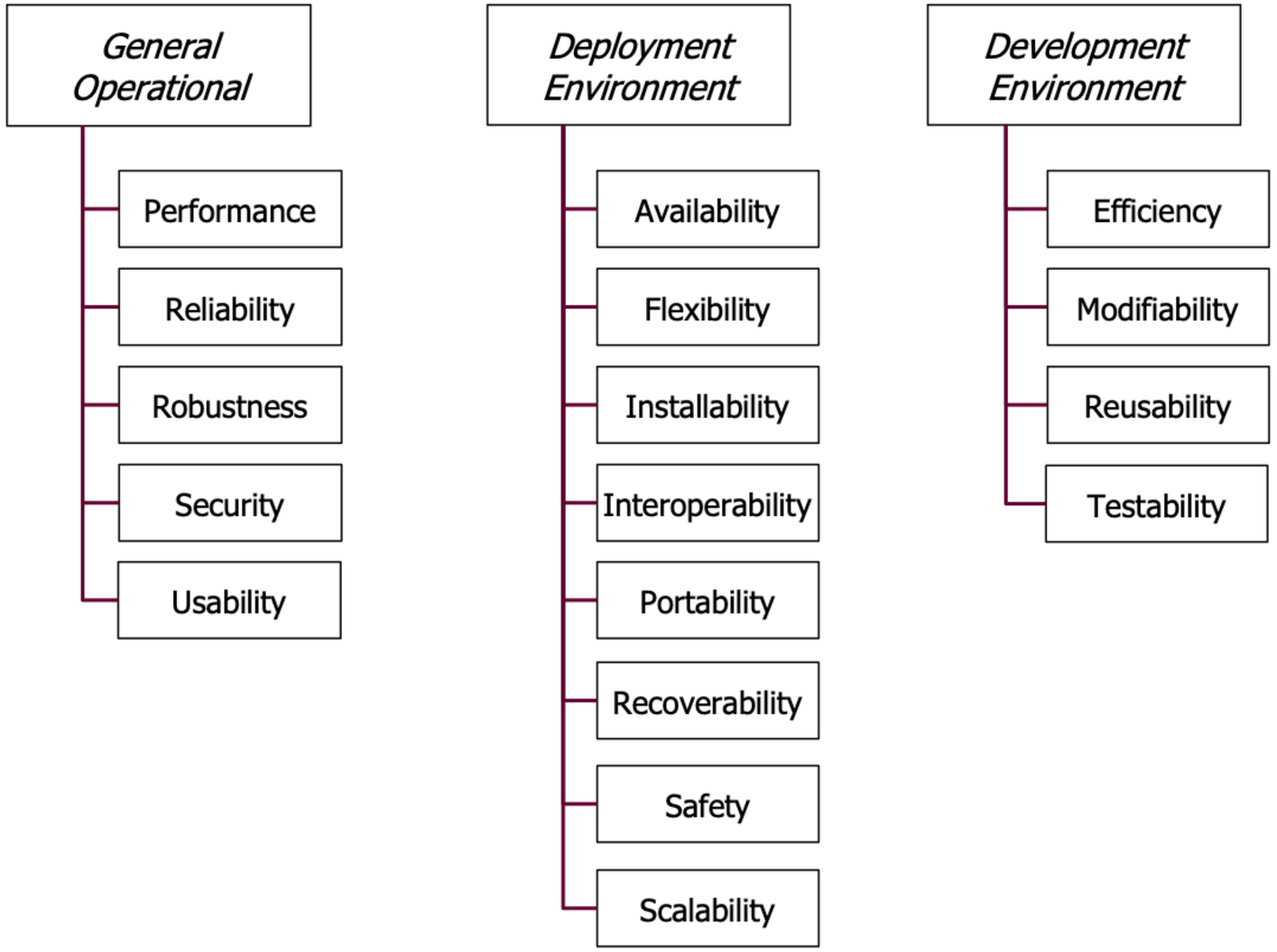


Figure 2: Asynchronous stages in a deployment pipeline

SOFTWARE QUALITY ATTRIBUTES



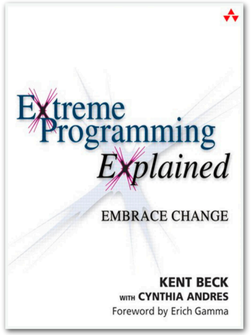
Original Source: The Whole Team Approach to Agile Testing by Janet Gregory and Lisa Crispin

EXTREME PROGRAMMING

Ten-Minute Build

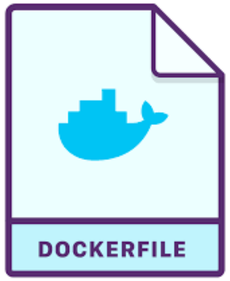
Automatically build the whole system and run all of the tests in ten minutes. A build that takes longer than ten minutes will be used much less often, missing the opportunity for feedback. A shorter build doesn't give you time to drink your coffee.

The ten-minute build is an ideal. What do you do on your way to that ideal? The statement of the practice gives three clues: *automatically* build the *whole* system and run *all* of the tests in ten minutes. If your process isn't automated, that's the first place to start. Then you may be able to build only the part of the system you have changed. Finally, you may be able to run only tests covering the part of the system at risk because of the changes you made.

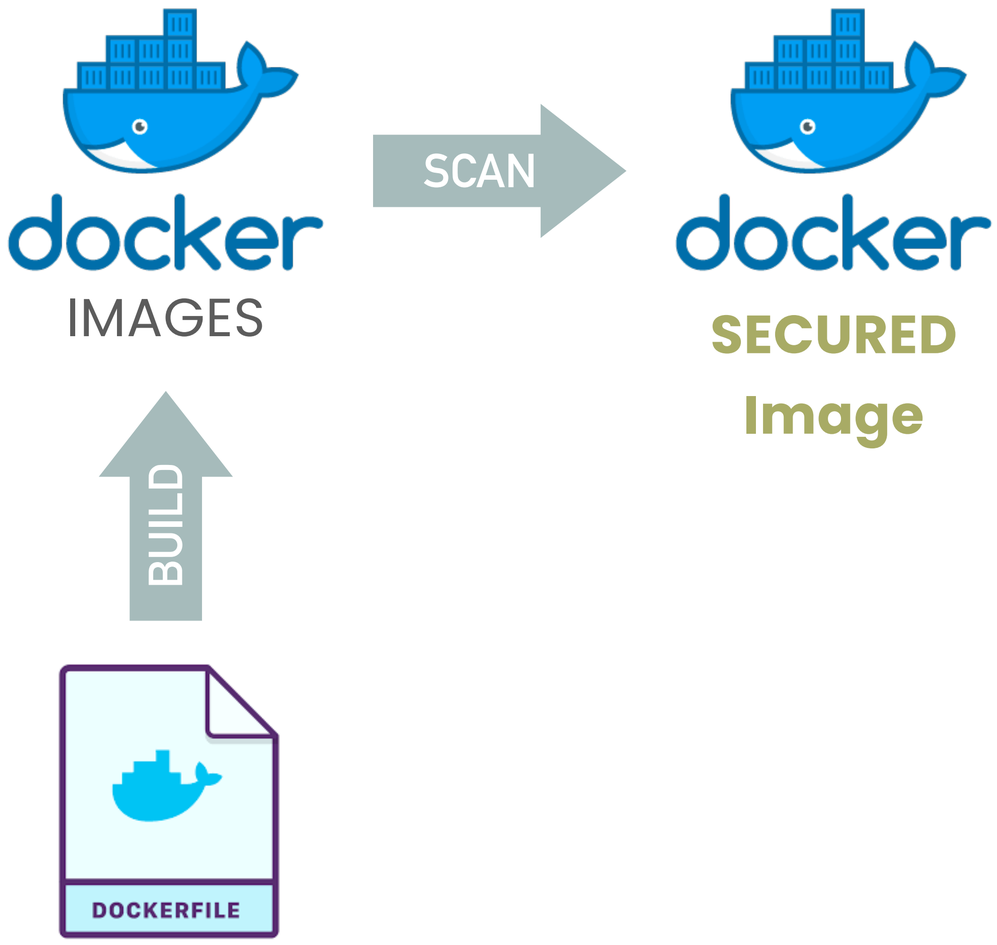


DEVELOPMENT WORKFLOW WITH DOCKER

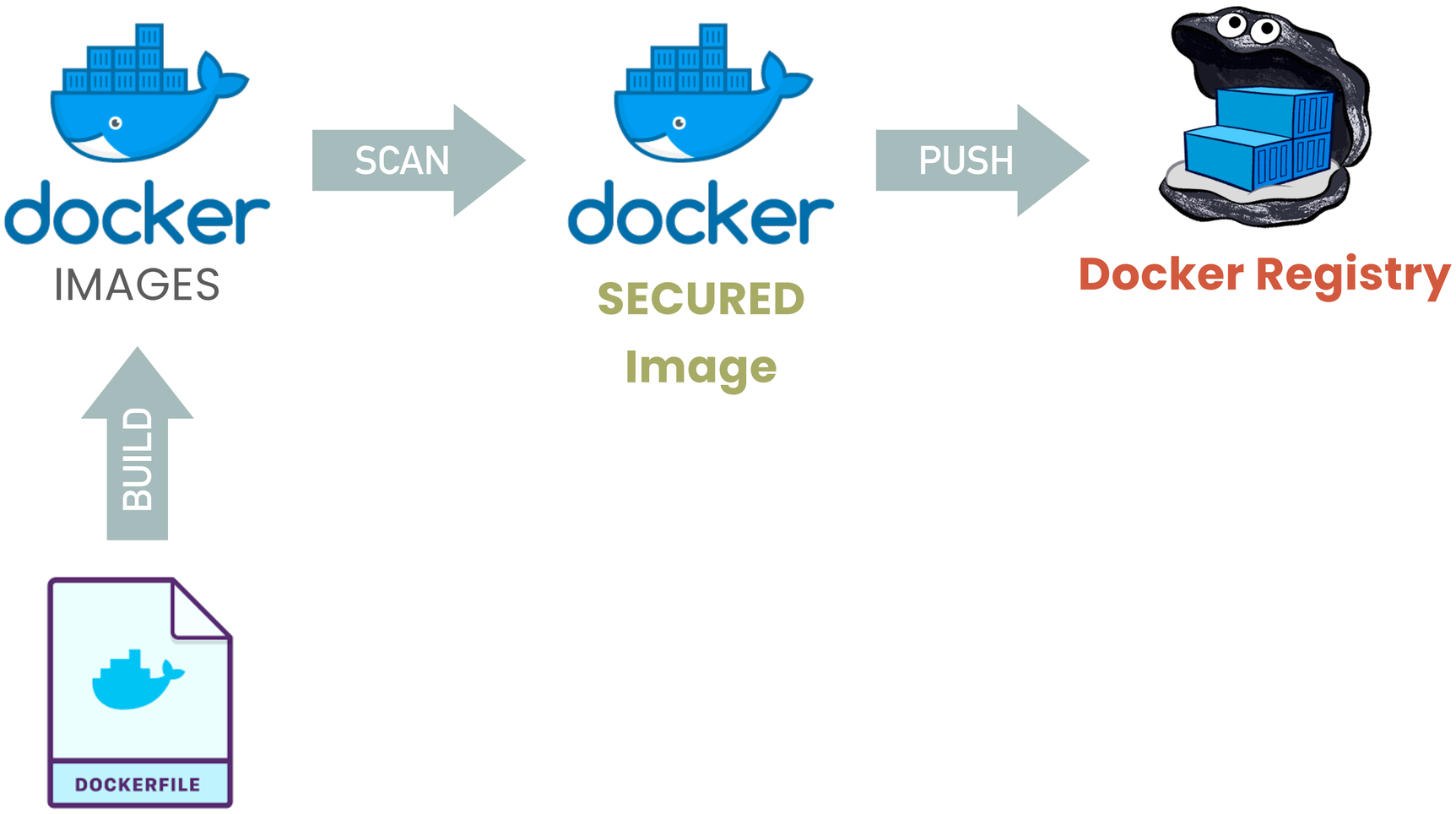
DEVELOPMENT WORKFLOW WITH DOCKER



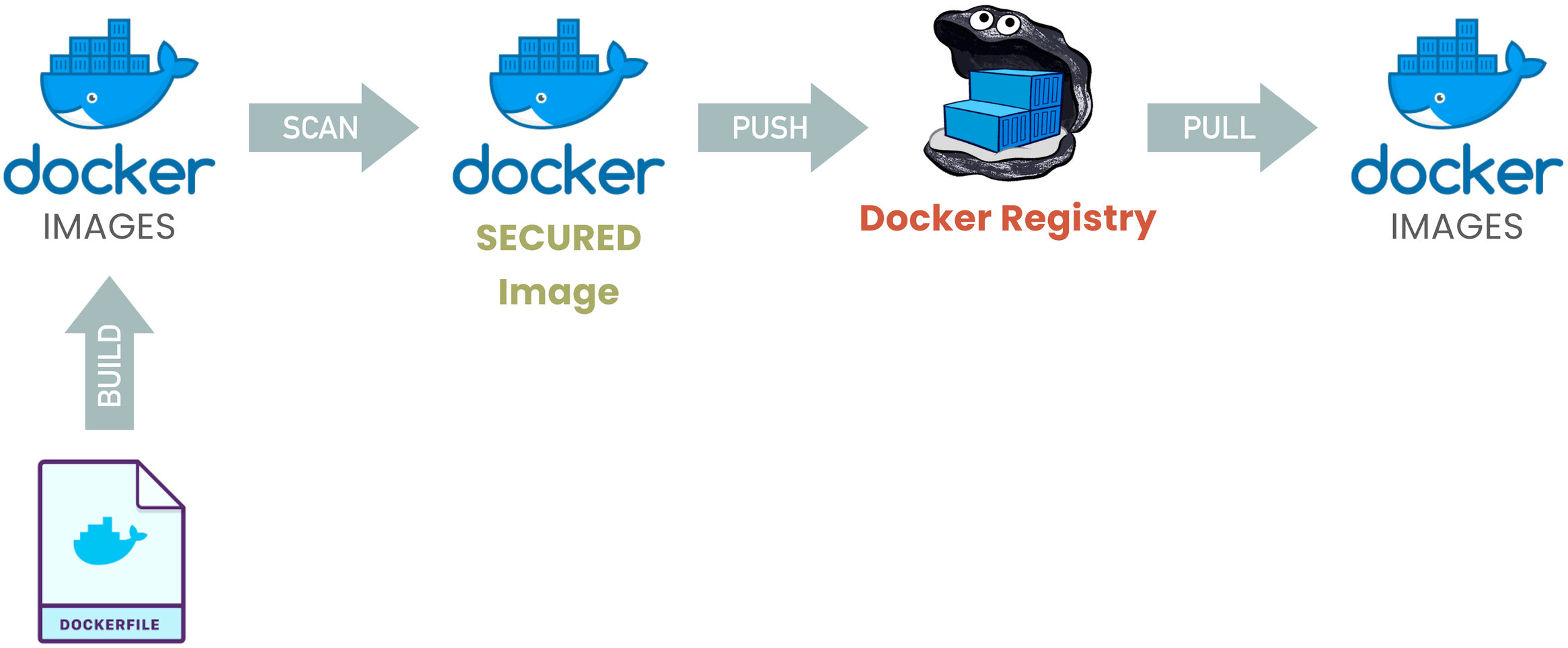
DEVELOPMENT WORKFLOW WITH DOCKER



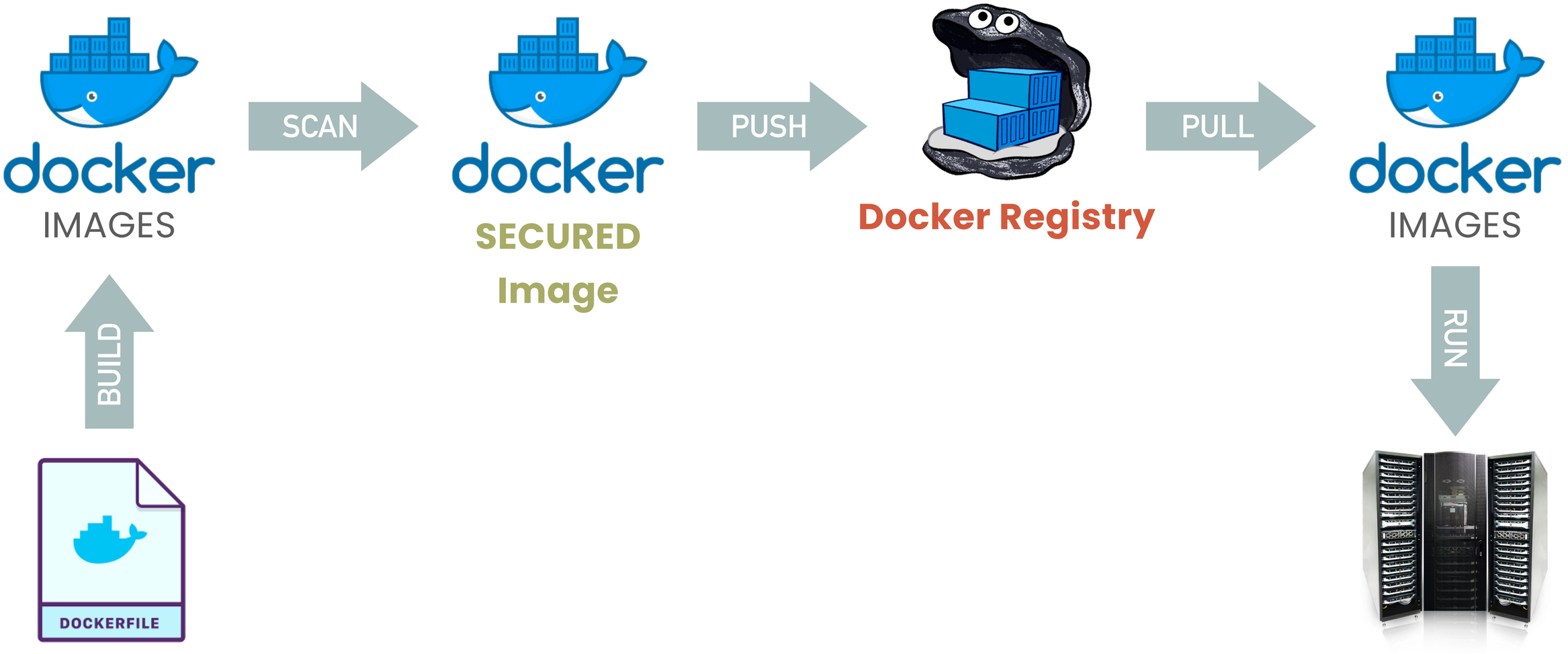
DEVELOPMENT WORKFLOW WITH DOCKER



DEVELOPMENT WORKFLOW WITH DOCKER



DEVELOPMENT WORKFLOW WITH DOCKER



BUILD PIPELINE

BUILD PIPELINE

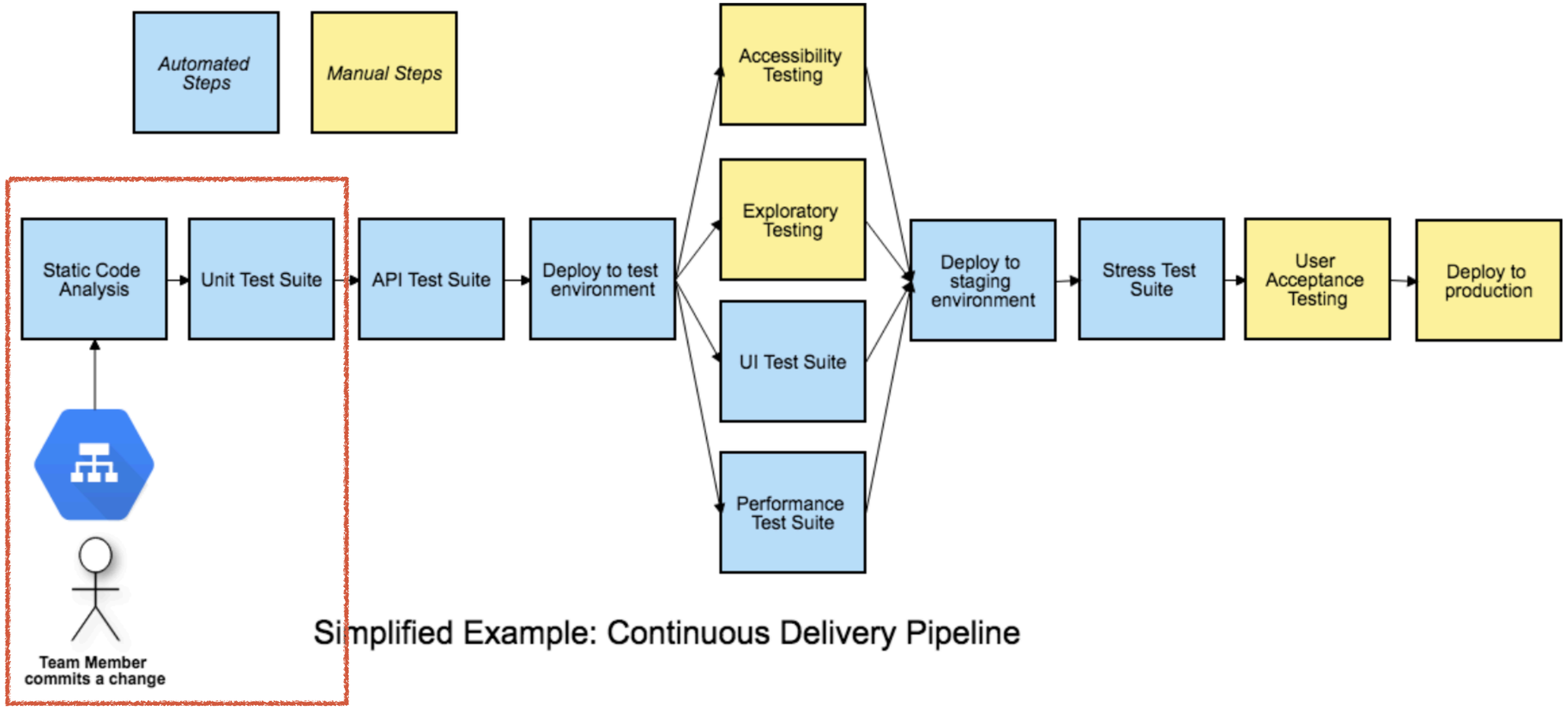
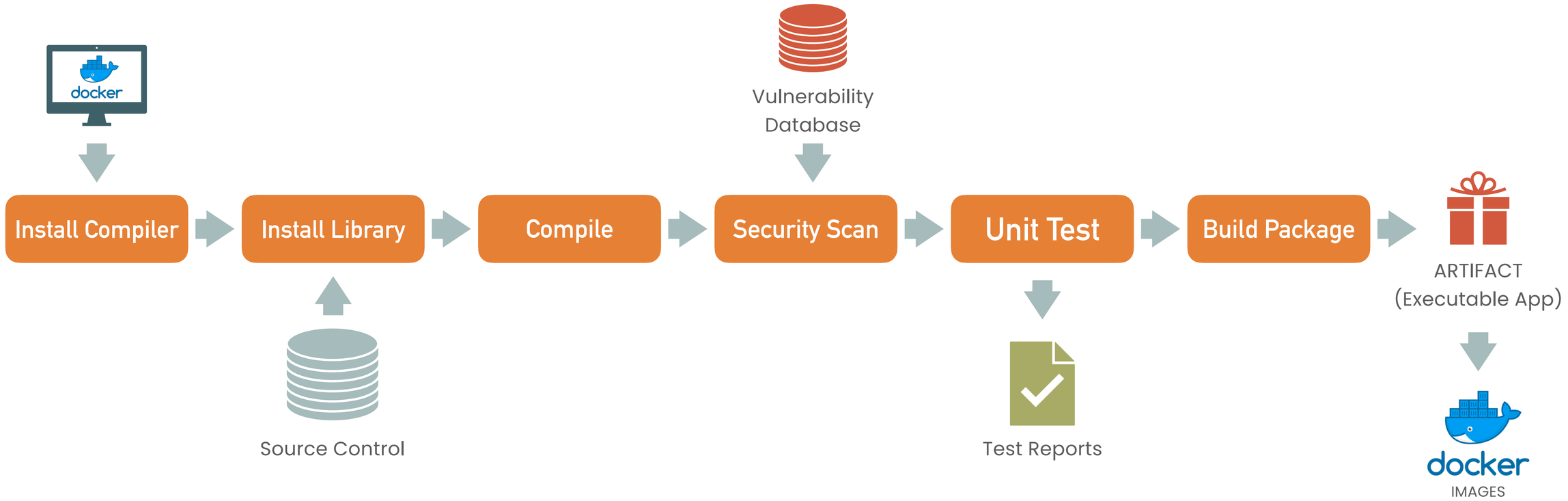


Figure 1: A Continuous Delivery Pipeline (my diagrams are inspired by Katrina Clokie's pipeline diagrams in "A Practical Guide to Testing in DevOps")

CHALLENGING ABOUT BUILDING DOCKER IMAGE



TEST PIPELINE

TEST PIPELINE

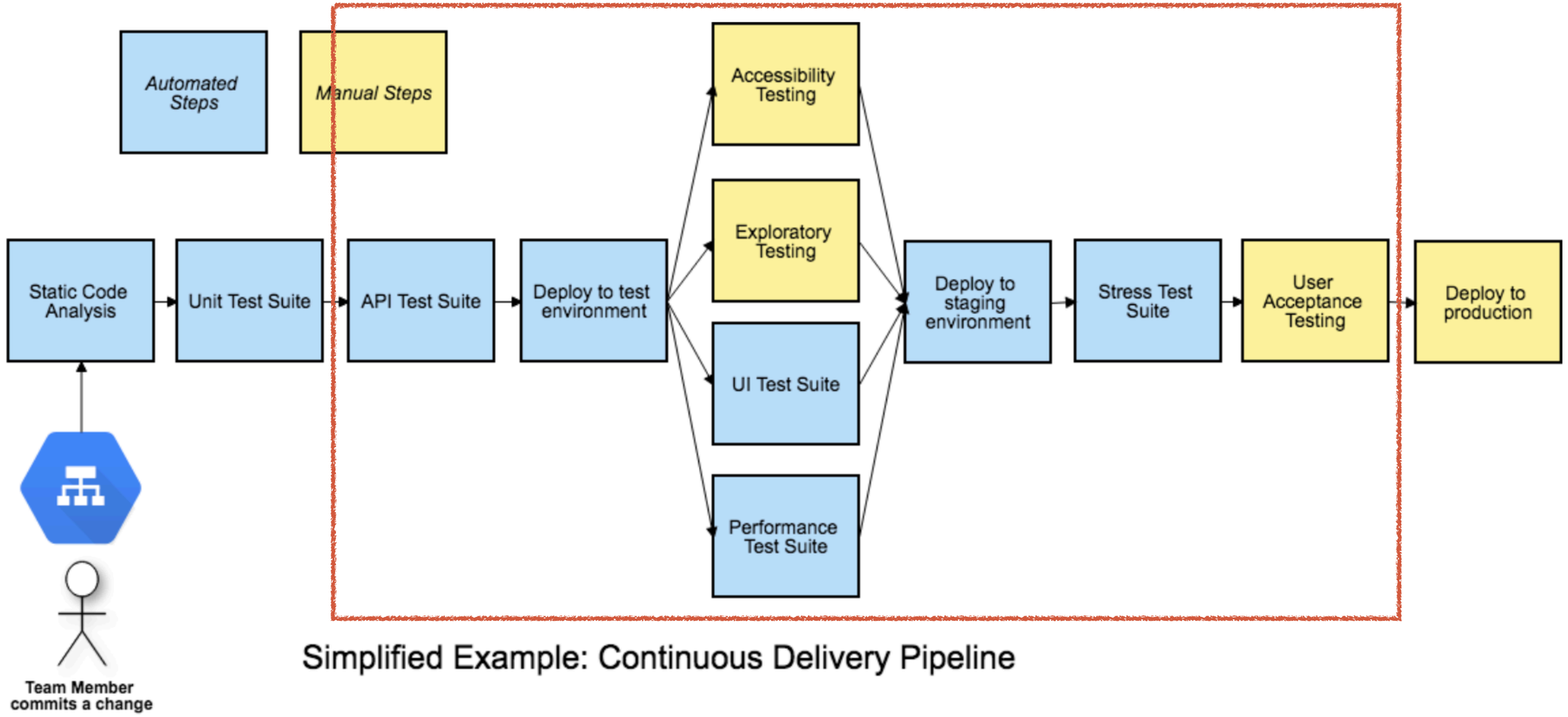
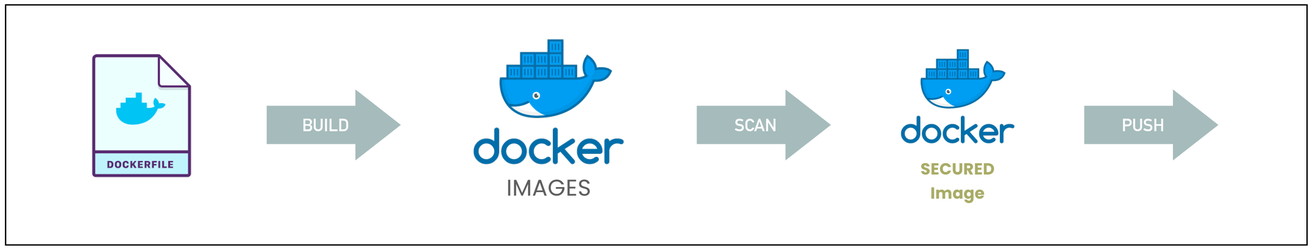


Figure 1: A Continuous Delivery Pipeline (my diagrams are inspired by Katrina Clokie's pipeline diagrams in "A Practical Guide to Testing in DevOps")

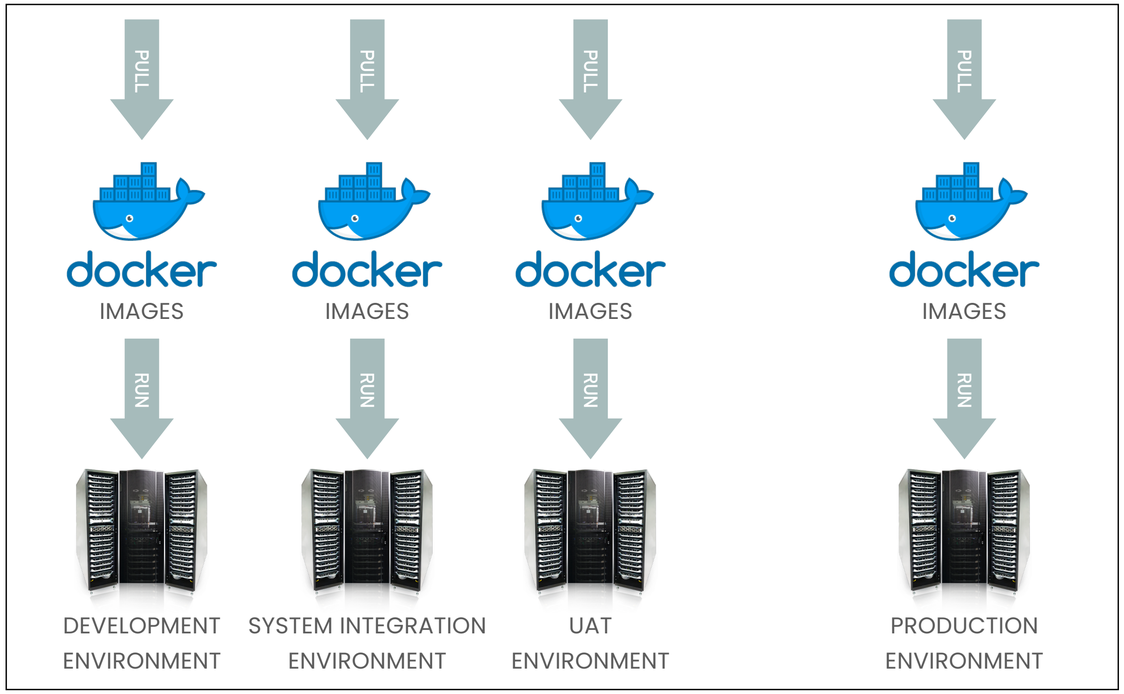
DEVELOPMENT APPLICATION WITH DOCKER

BUILD Pipeline



Docker Registry

DEPLOY + TEST Pipeline



DEPLOYMENT PIPELINE

DEPLOYMENT PIPELINE

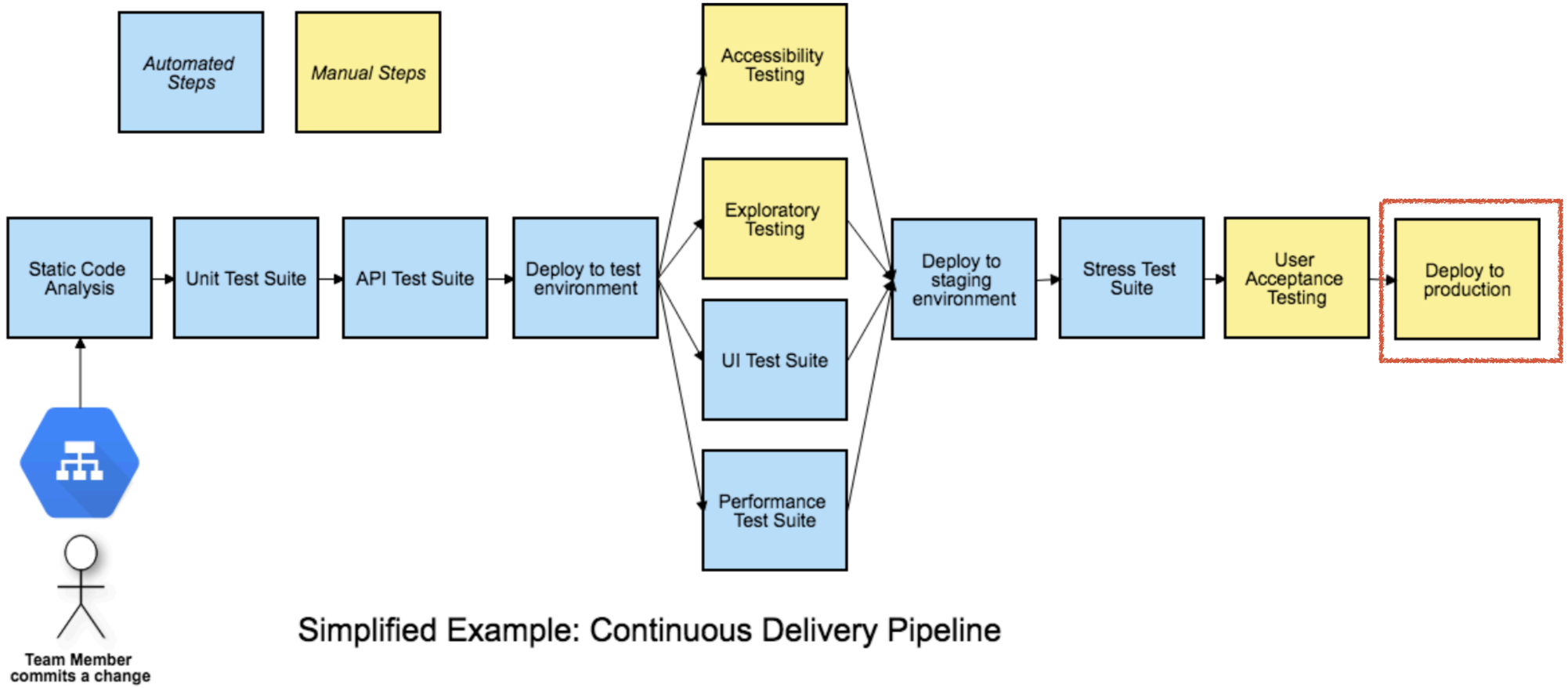


Figure 1: A Continuous Delivery Pipeline (my diagrams are inspired by Katrina Clokie's pipeline diagrams in "A Practical Guide to Testing in DevOps")

THANK YOU