YANYAN WANG

AUTOMATE DYNAMIC APPLICATION SECURITY TESTING (DAST)

AN APPROACH IN DOCKERIZED CI/CD PIPELINES

DISCLAIMER

- This disclaimer informs readers that the views, thoughts, and opinions expressed in the text belong solely to the author, and not necessarily to the author's employer, organization, committee or other group or individual.

OUTLINE

- Who Am I
- What is a CI/CD pipeline
- Characteristics of CI/CD
- Challenges of Existing DAST tools in Dockerized CI/CD Pipelines
- An Approach to solve the problem
- Integrating With Selenium Tests

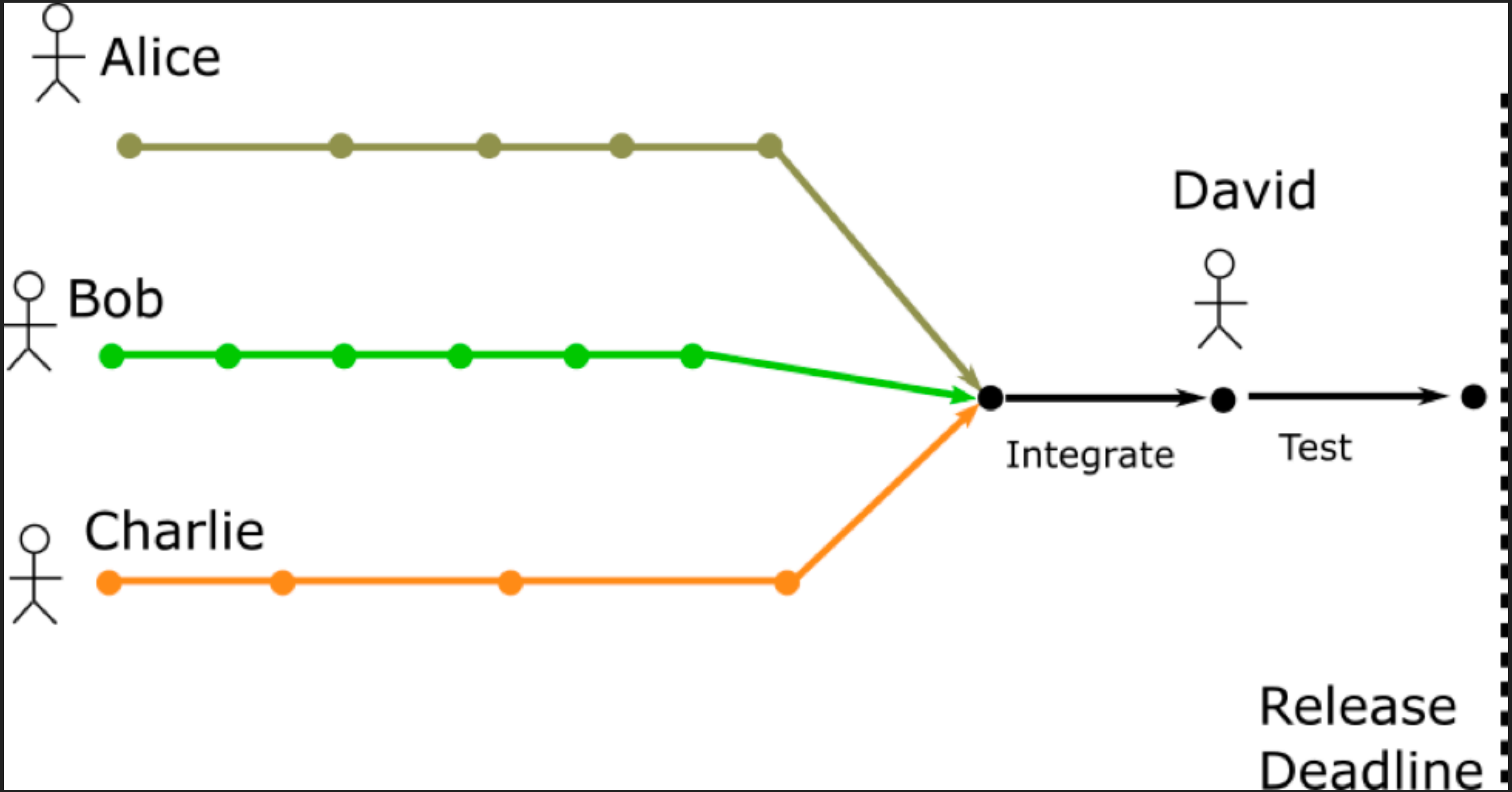
WHO AM I

- Cloud Application Security Architect

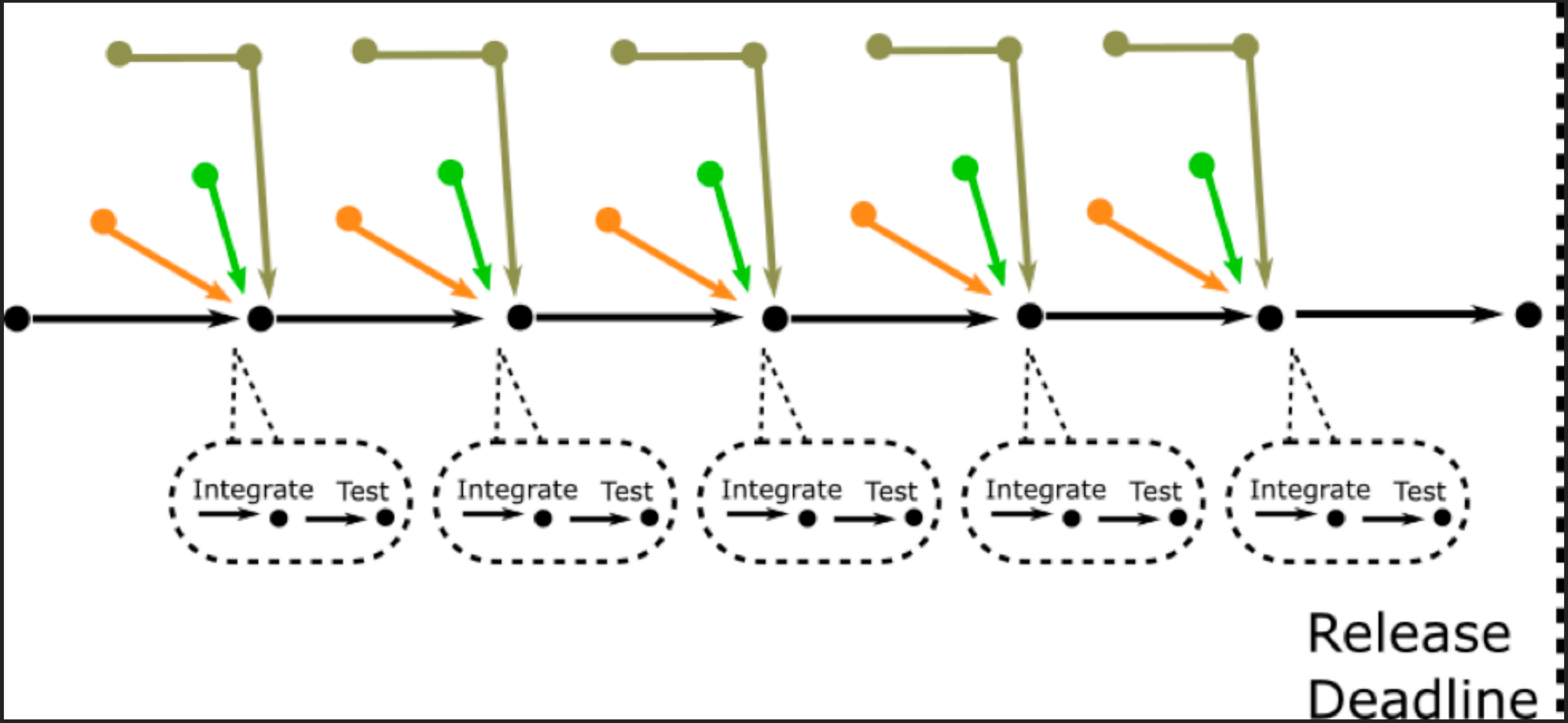
WHAT IS CI/CD

- Continuous Integration (CI):
  commit code into a shared repository several times a day. [1]
- Continuous Delivery (CD):
  every code change is releasable. [1]
- CI/CD Pipeline
  Automated steps in the software delivery process, e. g. builds, tests, and deployment. [2]

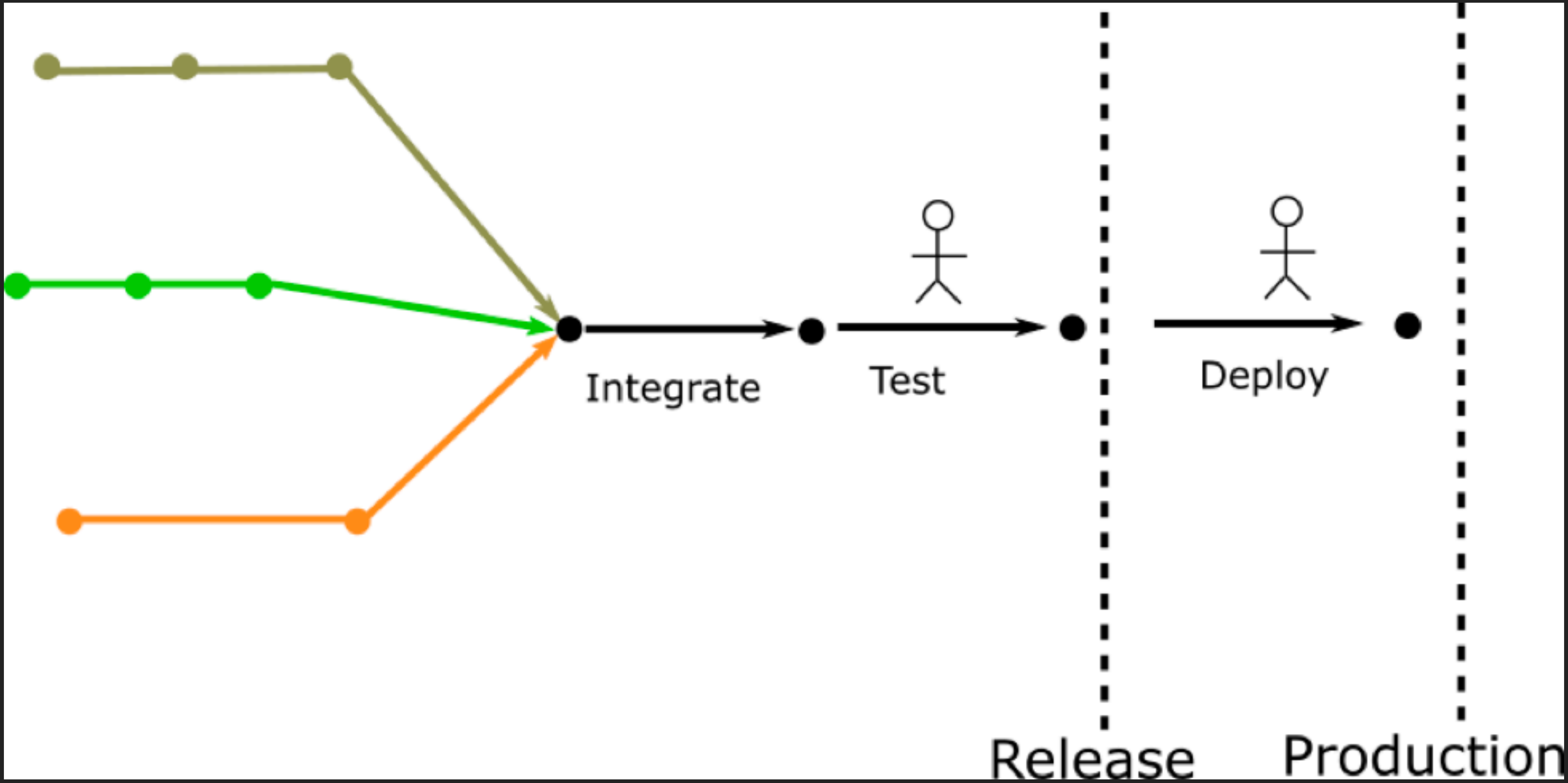
THE DARK AGES OF SOFTWARE INTEGRATION



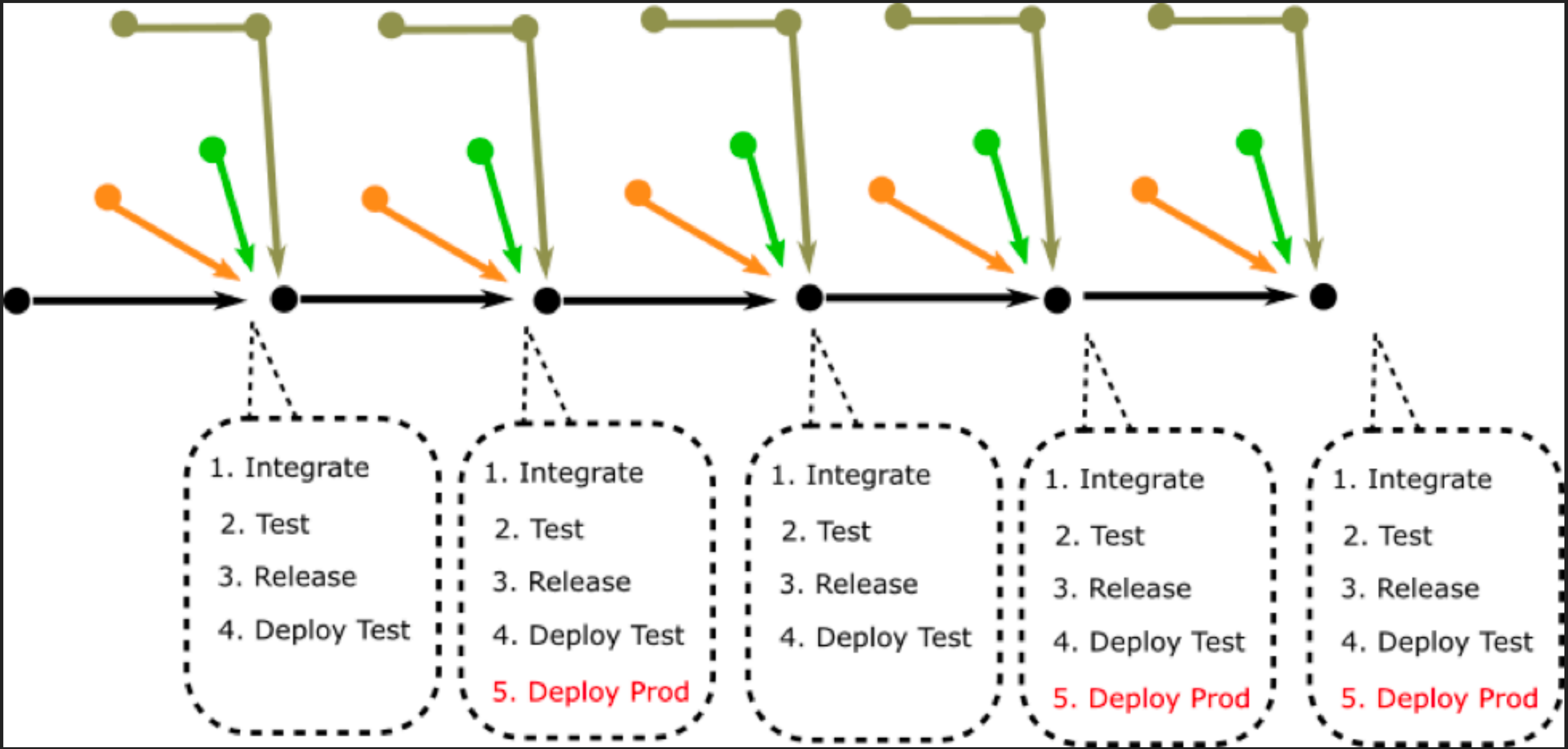
ADD CONTINUOUS TO THE SOFTWARE INTEGRATION



THE DARK AGES OF SOFTWARE DELIVERY



ADDING CONTINUOUS TO THE SOFTWARE DELIVERY



1. Integrate
2. Test
3. Release
4. Deploy Test

1. Integrate
2. Test
3. Release
4. Deploy Test
5. Deploy Prod

1. Integrate
2. Test
3. Release
4. Deploy Test

1. Integrate
2. Test
3. Release
4. Deploy Test
5. Deploy Prod

1. Integrate
2. Test
3. Release
4. Deploy Test
5. Deploy Prod

WHAT INSTAGRAM IS SAYING ABOUT CI/CD

- "At Instagram, we deploy our backend code 30-50 times a day…
  - With no *human* involvement in most cases.
- It lets our engineers move really fast.
- It makes it much easier to identify bad commits.
- Bad commits get detected very quickly and dealt with." [4]

CHARACTERISTICS OF CI/CD PIPELINES

That are relevant to DAST

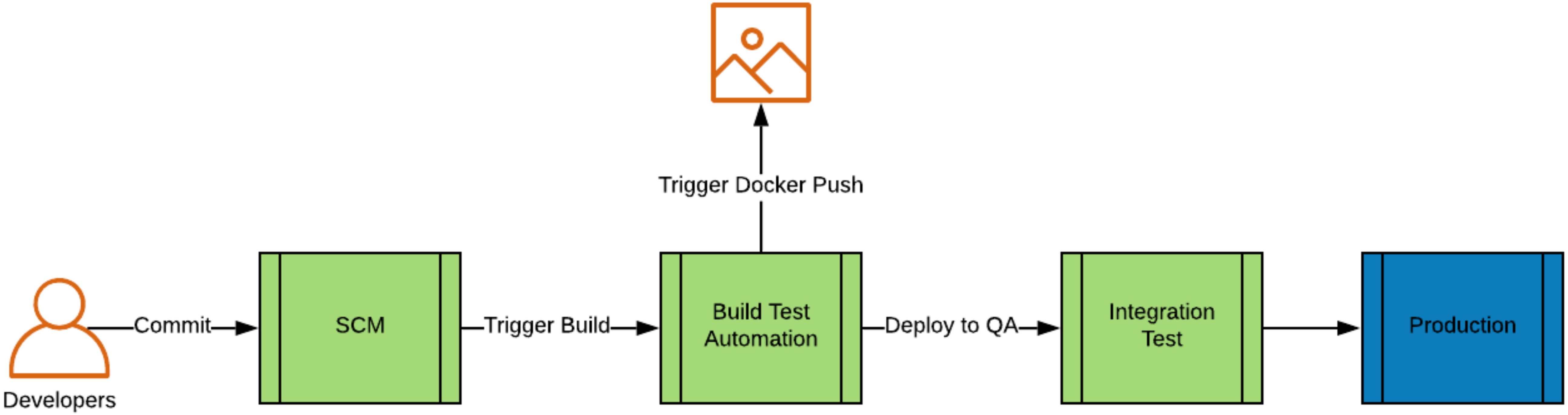
- All release steps are automated.
- Each feature/release is tested in its own test environment.
- All test suites are automated and relatively fast.
- Builds are repeatable and deterministic.

EXISTING DAST CHALLENGES IN CI/CD PIPELINES

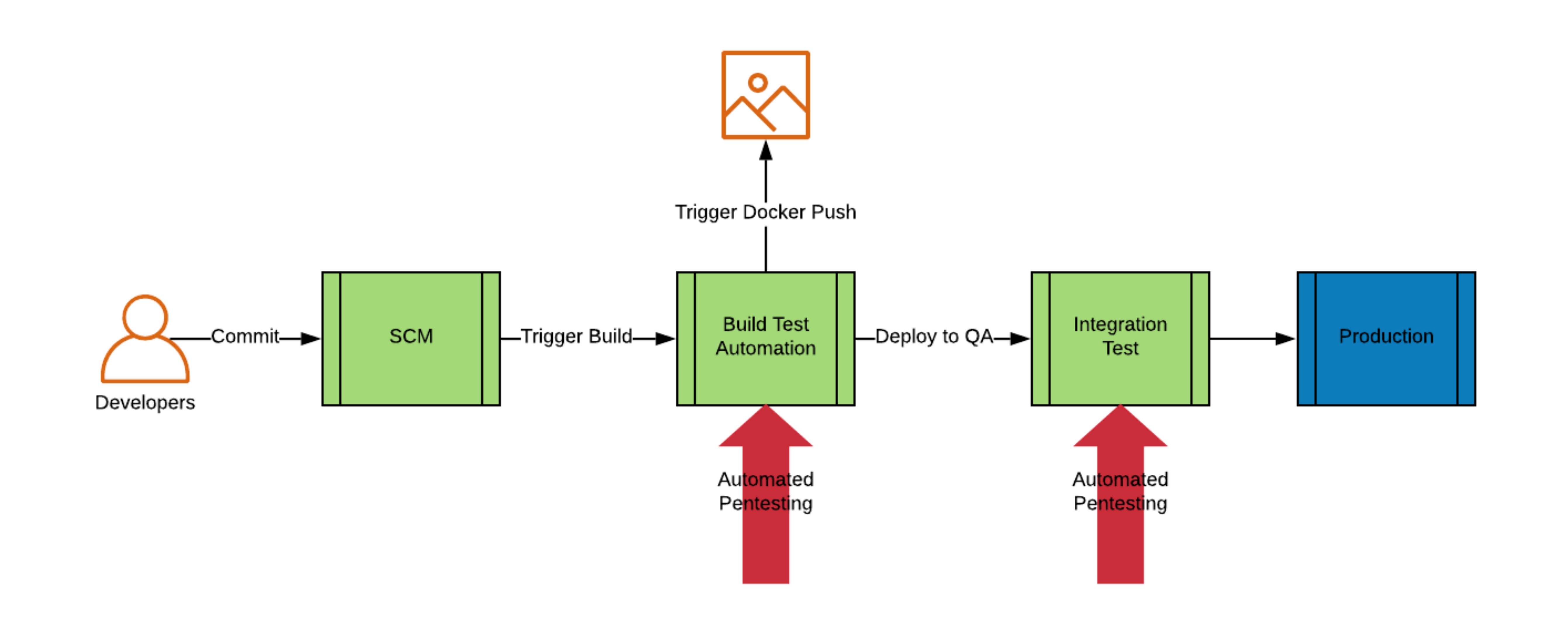
- Most DAST tools are stuck in dark ages
- Need human intervention to set up a scan
    - e. g. URL, credentials and other information
- Need to keep a running application for DAST testing
- Often have separate reporting systems from the CI/CD tools

IS IT TIME TO PANIC?

COMMON STEPS IN CI/CD PIPELINE



HOW CAN DAST BE INTEGRATED WITH CI/CD



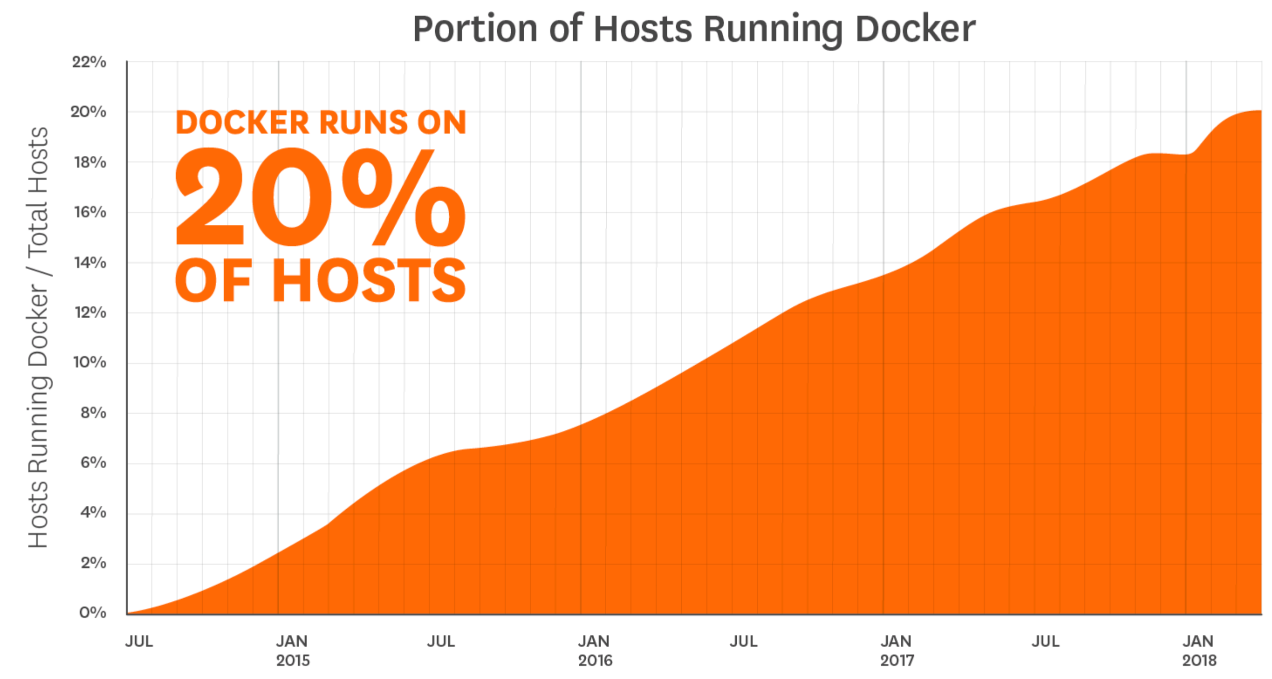
DAST TOOLS MUST-HAVES IN CI/CD PIPELINES

- Can be automated completely
- Produce reports within the CI/CD pipeline tools
- Relatively fast
- Need to be scalable

CONTAINER AND ORCHESTRATION

- A container image is a lightweight, standalone, executable package of software. [5]
- Orchestration is managing the lifecycles of containers, especially in large, dynamic environments. [6]

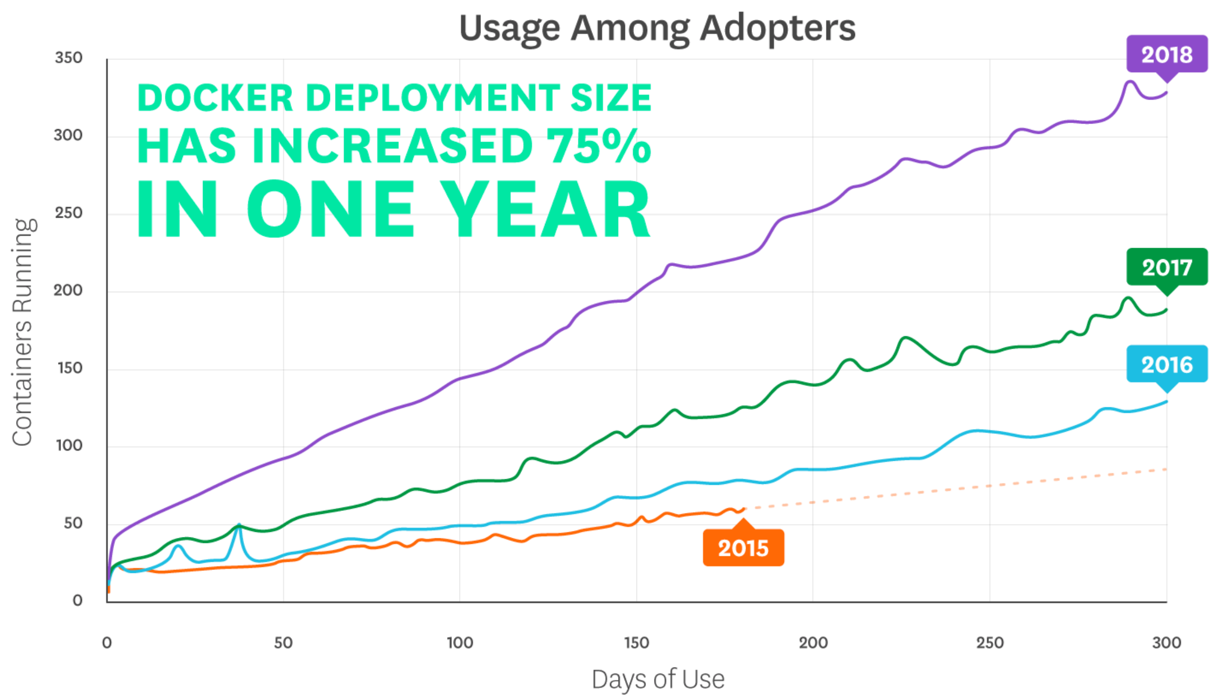
USAGE OF DOCKER AND ORCHESTRATION



Portion of Hosts Running Docker

DOCKER RUNS ON **20%** OF HOSTS

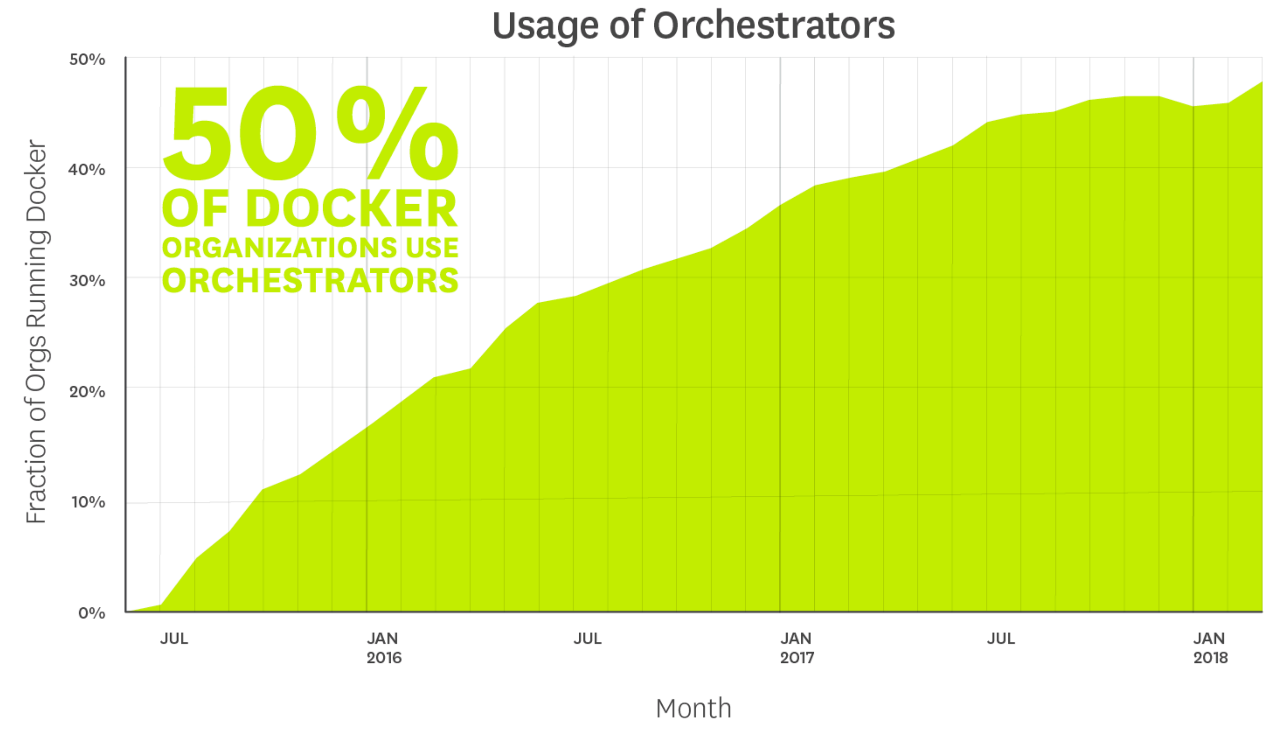
Source: Datadog



Usage Among Adopters

DOCKER DEPLOYMENT SIZE HAS INCREASED 75% **IN ONE YEAR**

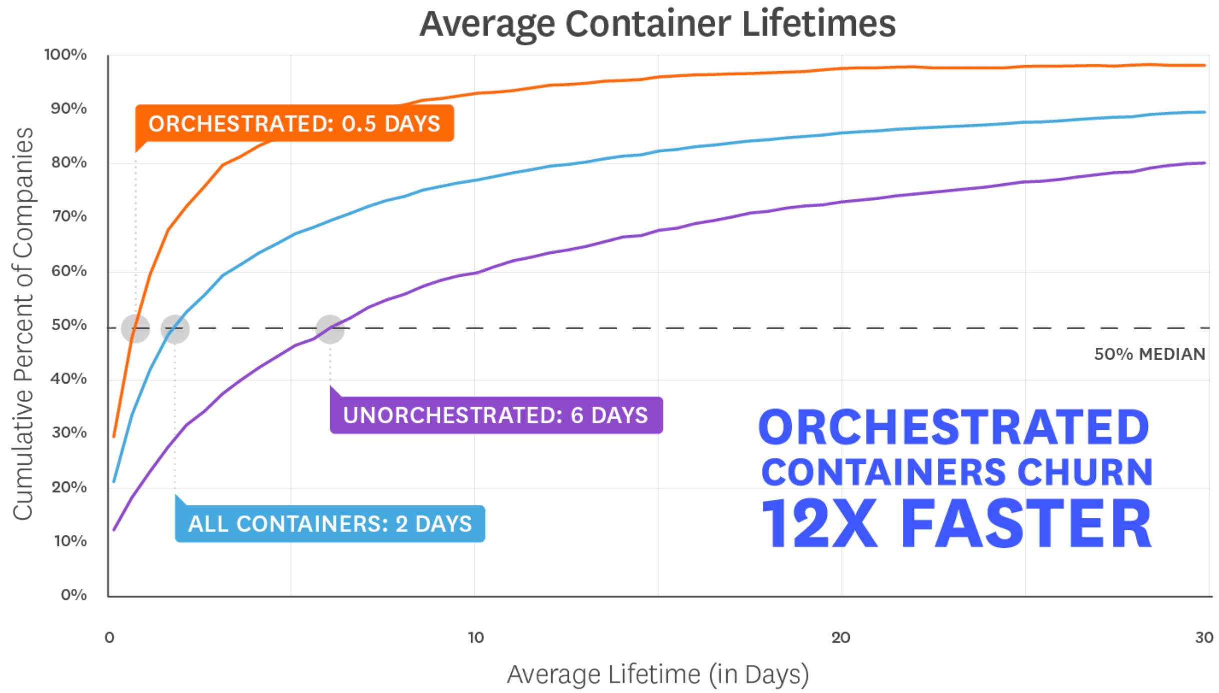
Source: Datadog



Usage of Orchestrators

50% OF DOCKER ORGANIZATIONS USE ORCHESTRATORS

Source: Datadog



Average Container Lifetimes

ORCHESTRATED: 0.5 DAYS

UNORCHESTRATED: 6 DAYS

ALL CONTAINERS: 2 DAYS

ORCHESTRATED CONTAINERS CHURN **12X FASTER**

Source: Datadog

AN APPROACH TO SOLVE THE PROBLEM

- Need a Dockerized solution
- Integrates with selenium test automation
- Results should be available to the engineers in Jenkins UI
  - Users: I don’t want to log in another tool!
- Finds meaningful vulnerabilities out of the box

BURP VS ZAP

- Burp
  - Existing docker images, but only suitable for interactive use through UI
  - Meaningful results (number and accuracy)
  - Integration with Selenium automation
- Owasp Zap
  - Existing docker image with junit/html/xml/markdown reports
  - Limited results compared to Burp

STEP1: DOCKERIZE HEADLESS BURP

- Headless Burp extension
  - Command-line interface
  - Used for automation
  - Selenium integration
    - No on boarding needed: Passes URLs, TLS cert and credentials from selenium to Burp
    - Incremental scans: Test exactly what's released, no more, no less
- junit/html report formats which can be added to Jenkins UI
- False Positive filtering features: actionable findings

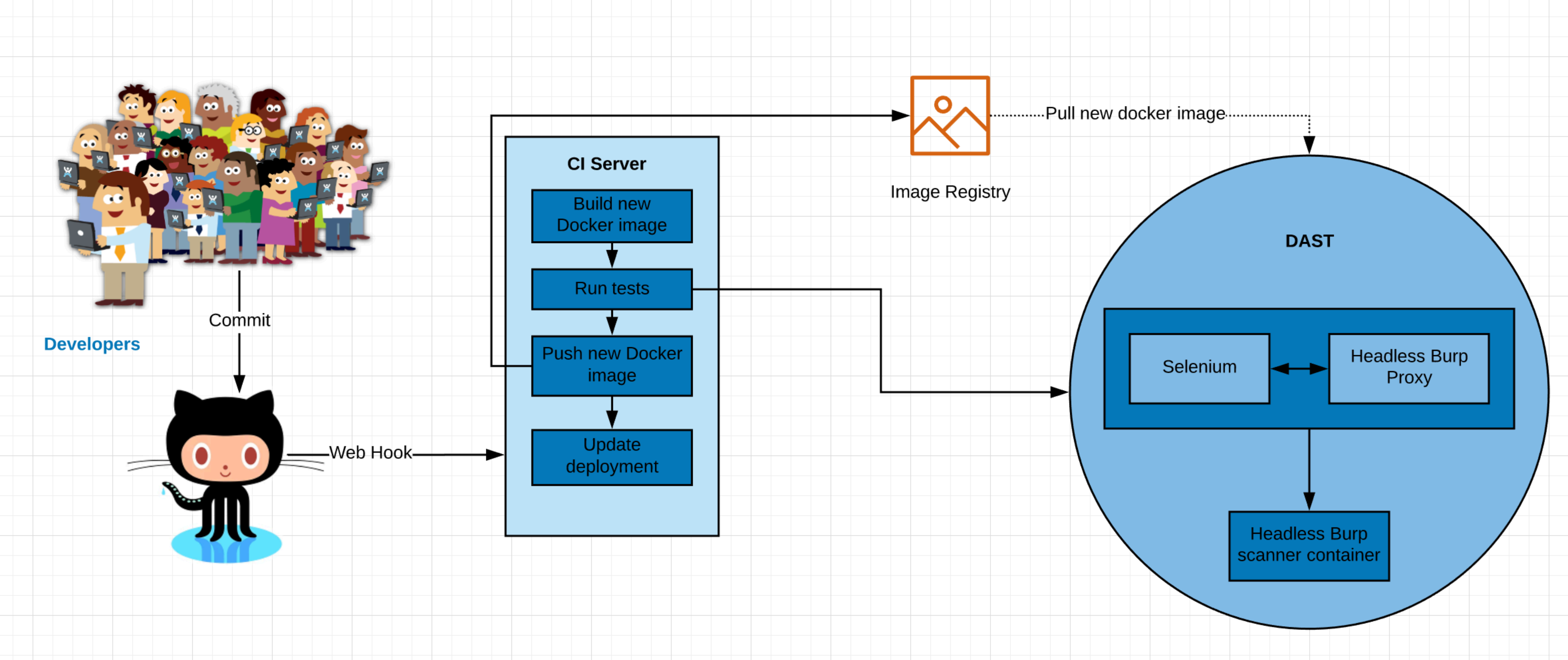
STEPS TO DOCKERIZE HEADLESS BURP

1. Clone docker-burp-suite-pro
2. Build and save headless proxy and scanner
3. Copy the headless proxy and scanner to your docker image in [1]

SPEED BUMPS

- The only way to activate the Burp license is through UI
- Use Docker --volume to persist data
- This reference helped
- Build the headless proxy and scanner yourself. The Burp BApp store app didn’t work.
- Set “listen_mode":"all_interfaces" for working with Selenium
- Follow instruction in docker-burp-suite-pro carefully, and make sure you restart the Mac after install XQuartz

STEP 2: INTEGRATE WITH THE CI/CD PIPELINE



AUTOMATE DAST IN CI/CD PIPELINES

SOURCE CODE

- Working with my organization to open source it
- Here is the project link to GitHub: https://github.com/pinkgladiator/headless-burp-docker-image

SUMMARY OF THE PRESENTATION

- Modern CI/CD pipelines is automated and fast.
- DAST tools need to automate and integrate with CI/CD natively, particularly dockerized environment, but they they are not.
- Dockerize headless burp and produce feedback in CI/CD pipelines is an approach to solve the problem.

AUTOMATE DAST IN CI/CD PIPELINES

REFERENCE

1. https://www.thoughtworks.com/continuous-delivery
2. https://semaphoreci.com/blog/cicd-pipeline
3. https://thenewstack.io/ci-cd-with-kubernetes-tools-and-practices/
4. https://instagram-engineering.com/continuous-deployment-at-instagram-1e18548f01d1
5. https://www.docker.com/resources/what-container
6. https://blog.newrelic.com/engineering/container-orchestration-explained/
7. https://support.portswigger.net/customer/portal/questions/17107769-is-that-possible-to-create-a-docker-image-of-burp-pro-
8. https://github.com/koenrh/docker-burp-suite-pro
9. https://github.com/NetsOSS/headless-burp