aws marketplace

AWS Marketplace DevOps Workshop Series

Module 5: Continuous Testing

DevOps Institute

aws marketplace

Mark Peters

Ambassador, DevOps Institute

@tinysyber

tinycyber

Dilip Rajan

Sr. Partner Solution Architect at AWS

dilip-rajan-65643919

aws marketplace

Mark Peters

Ambassador, DevOps Institute

@tinysyber

tinycyber

Mark Peters

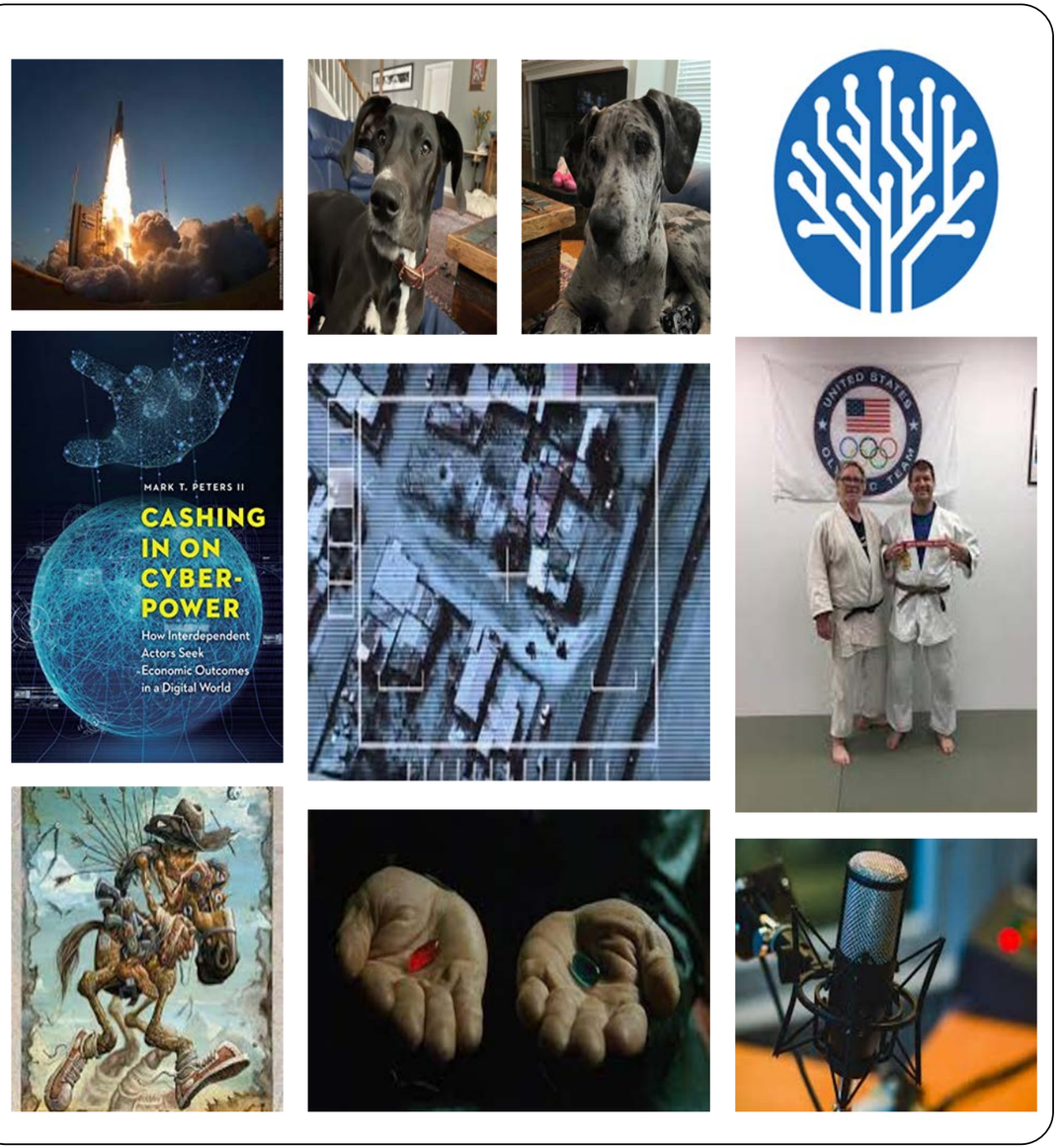
Technical Lead, Novetta

@tinycyber

Mark is Technical Lead for Novetta, working on integrating DevOps for a US Department of Defense program. A recently converted DevOps devotee, he is a DevOps Institute Ambassador, and the US chapter chair.

He holds a Doctorate in Strategic Security, is pursuing a PhD in cybersecurity and several industry certifications including CISSP and PMP. He authored “Cashing in on Cyberpower” analyzing ten years of cyber attacks.

After a career in Air Force Intelligence, he specializes in operationalizing security issues, finding ways to bring teams together and deliver accelerated value. In his spare time, he enjoys drawing, reading, speaking and judo as well as his two dogs.



About DevOps Institute

DevOps Institute's mission is to advance the human elements of DevOps by creating a safe and interactive environment where our members can network, gain knowledge, grow their careers, support enterprise transformation and celebrate professional achievements.

We connect and enable the global DevOps community to drive change in the digital age.

Join the community at
www.devopsinstitute.com

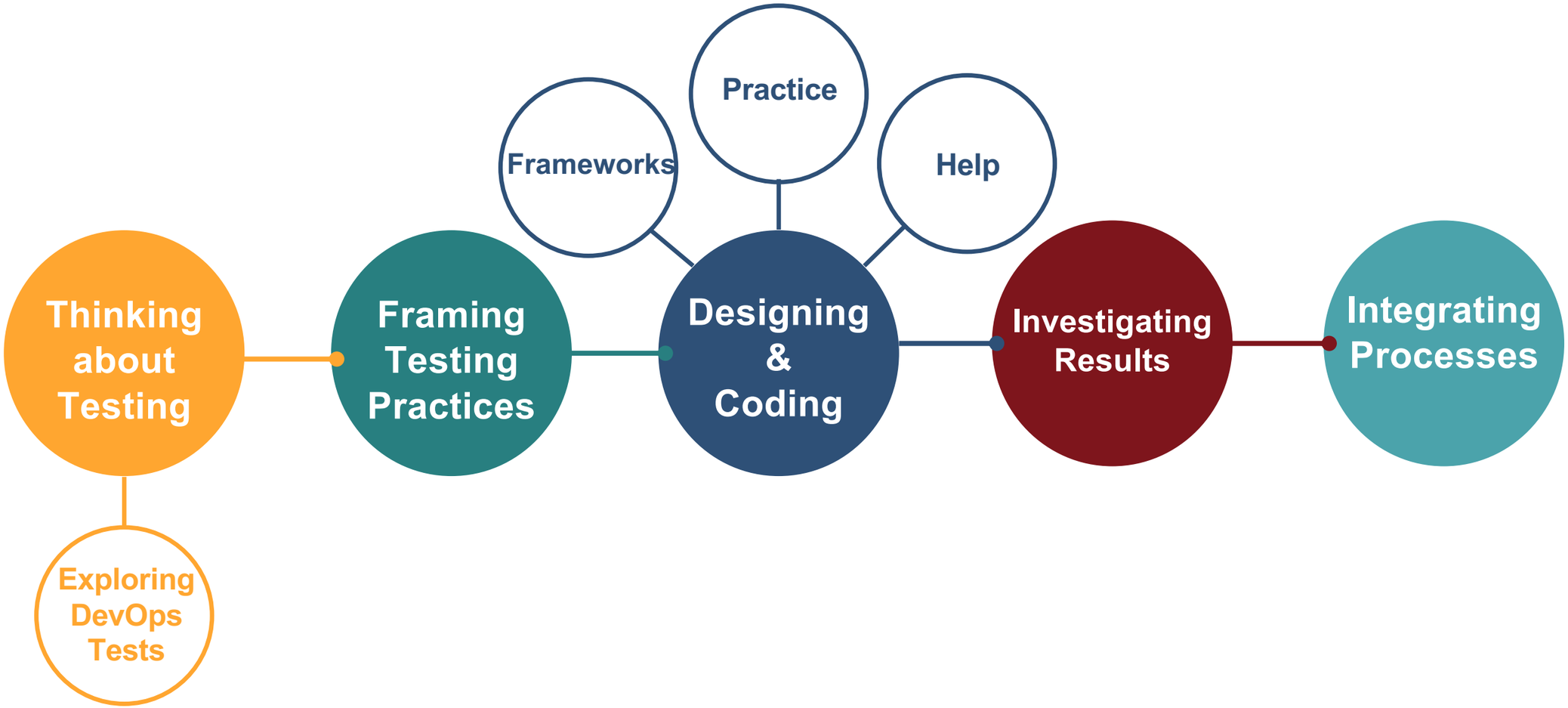
Recapping Modules 1-4

- In Module One, we discussed practicing DevOps through principles, design, and implementation
- Module Two began the process of developing your CI/CD pipeline, finding ways to accelerate value
- Module Three started looking at continuous deployment practices for integrated structures
- Module Four covered infrastructure as code and the importance of building a sound foundation for pipelines

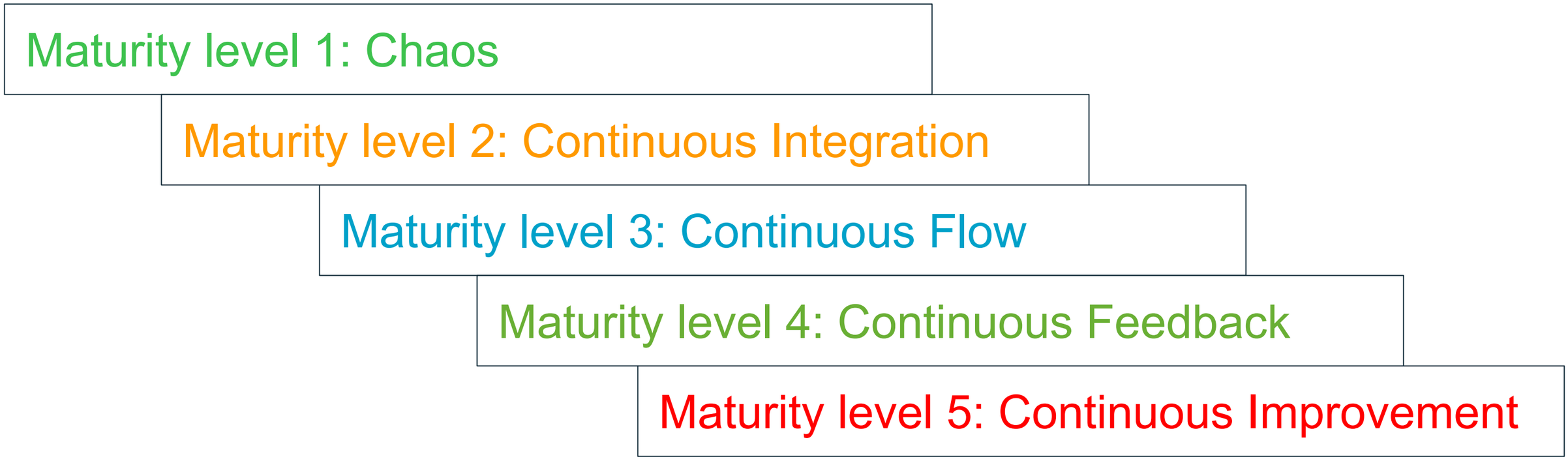
Agenda

You will learn:

- How to think about testing from a DevOps perspective
- Framing test practices for tasks
- Designing and coding to achieve test
- Investigating results for feedback
- Integrating processes



Continuous Testing Maturity Assessment



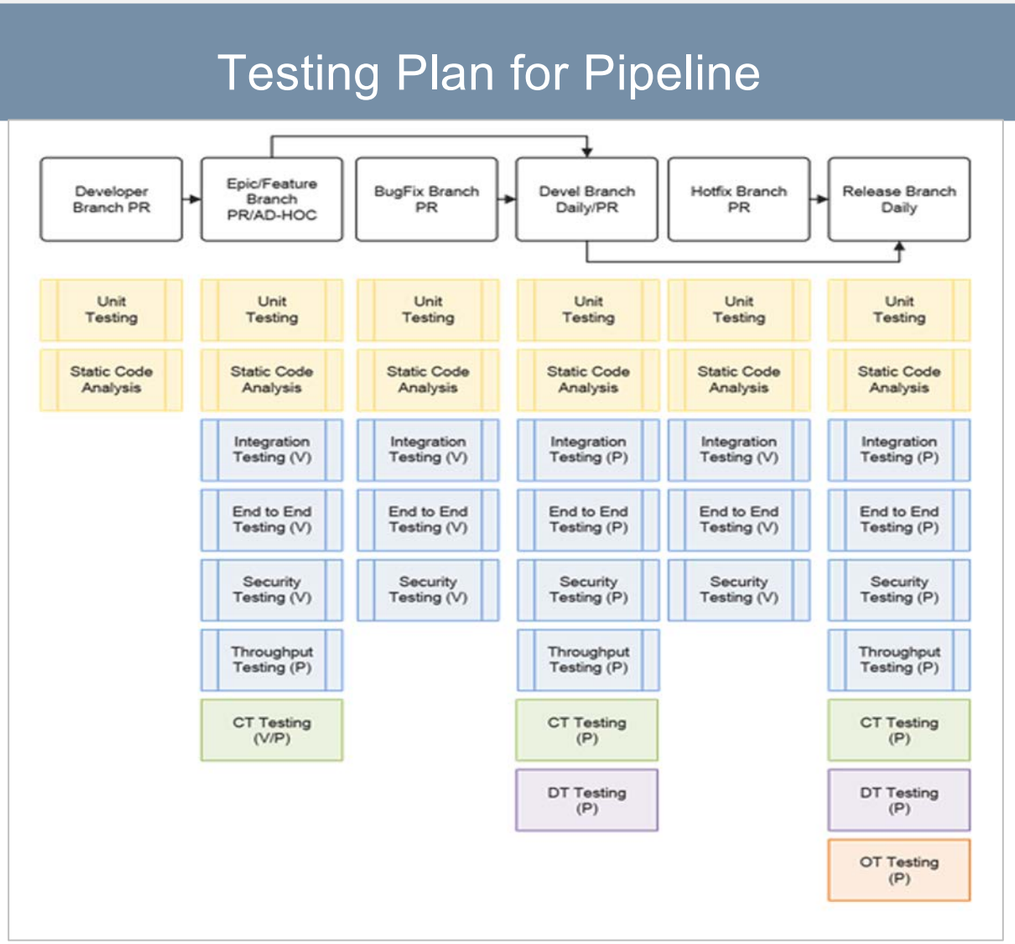
Thinking about testing

Where do I start testing

Good DevOps practices are sparked by Test-Driven Development

Previous modules discussed many different types of testing and employment

Today's focus is on designing employing your own testing practices

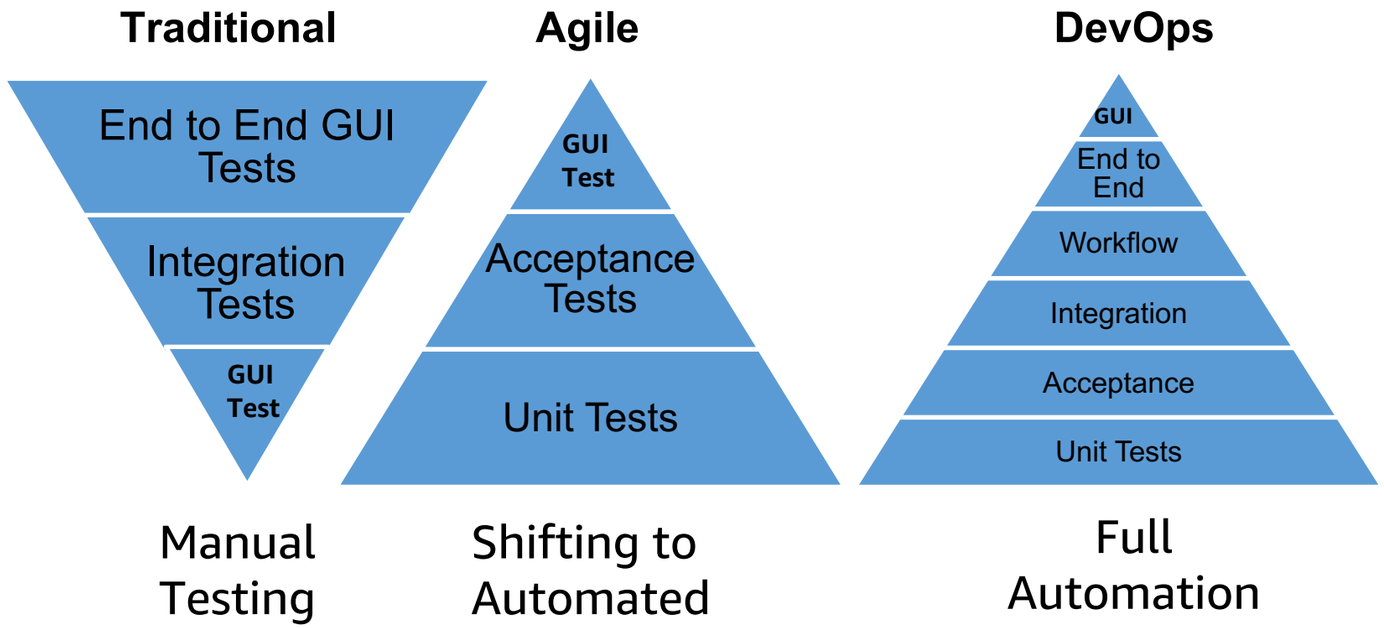


DevOps and testing

Flow	• Decrease lead time for Delivery • Build quality in • Prevent defects from being passed to work centers
Feedback	• High quality flow to fixers • Frequent information about processes • Fixes occur when smaller, cheaper, and easier
Continuous Experimentation	• Culture of learning about operations • Injection of stress to force improvement • Not simply testing the known good
Continuous Testing	• Check everything • Create process with known feedback at dedicated intervals • Constantly look for ways to verify improvement

Testing and flow

- Where do production-like environments exist in your process?
  - Less difference = More effective tests
  - More environments = More tests needed
- Manual tests= More code = More money
- Automated tests run constantly
- Goal: Test, Test and Re-Test without intervention



Flipping the Testing Pyramid

Don’t forget….

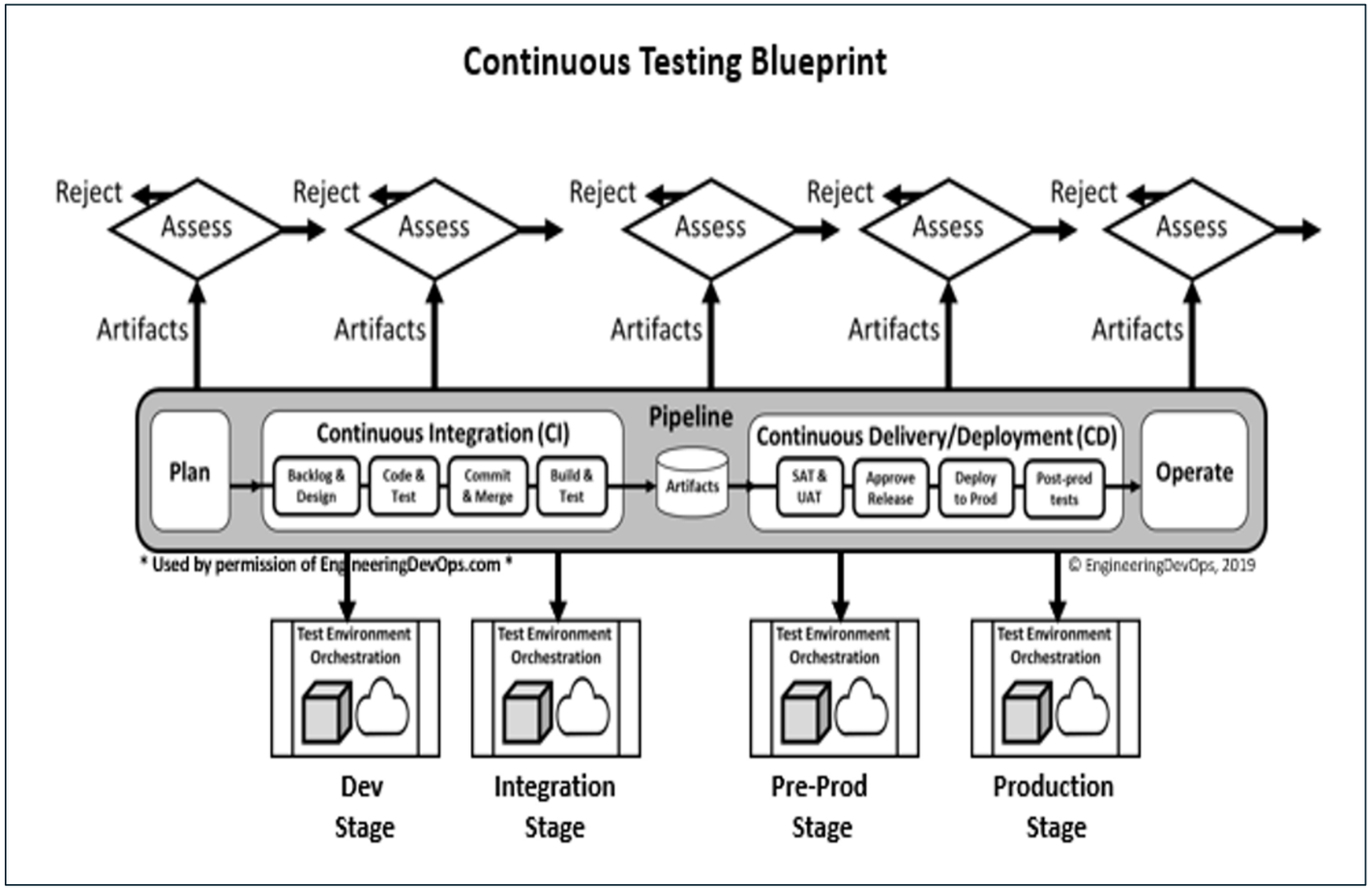
Conway’s Law -- Organization design systems tend to mirror their own communication structure

Stages for testing

DEVELOPMENT	INTEGRATION	DELIVERY	DEPLOYMENT
Guarantee code quality Catch issues early	Validate service	Final gate before production	Testing in Production
Unit testing	Data driven testing	Functional Tests	Runtime security tests
Code linting	End-to-End	Dynamic Security Tests	Metrics, traces and log analysis
Static analysis	Regression	Blue/Red Teaming	Reversion Tests
Code reviews	Performance		

Testing and feedback

- Feedback requires monitoring
  - Monitoring implies testing
  - Testing for feedback creates artifacts
- Testing looks for set barriers
  - Telemetry captures status
  - Metrics, logs, traces might not be testing
- How do you report test results?
  - Automated gating?
  - Dashboard
  - Weekly staff meetings?



Used with permission from EngineeringDevOps.com

Testing and continuous experimentation

- What happens after testing?
- Picking a stage to learn about more testing
- Abstracting to incorporate multiple layers

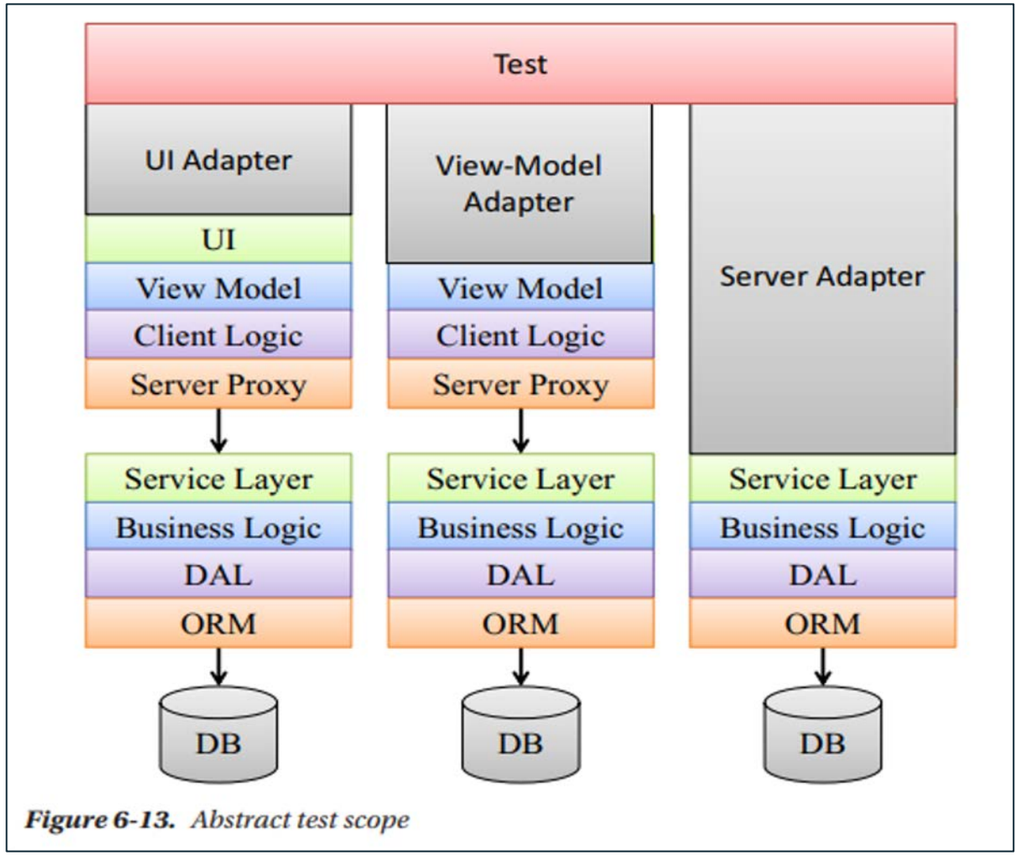


Figure 6-13. Abstract test scope

Continuous Testing vs everybody

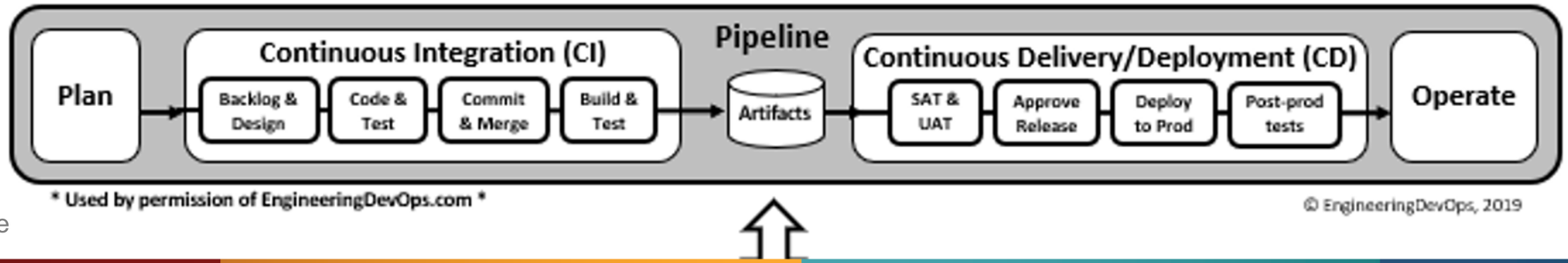
Testing should form the base of the DevOps Pipeline Pyramid

Continuous Testing

Designed to cover functional and non-functional applications. Ensure as many relevant test are executed as early as possible in the pipeline. Tests should be selected automatically, and results should be critical to promotion to next stage

Continuous Integration, Delivery/ Deployment

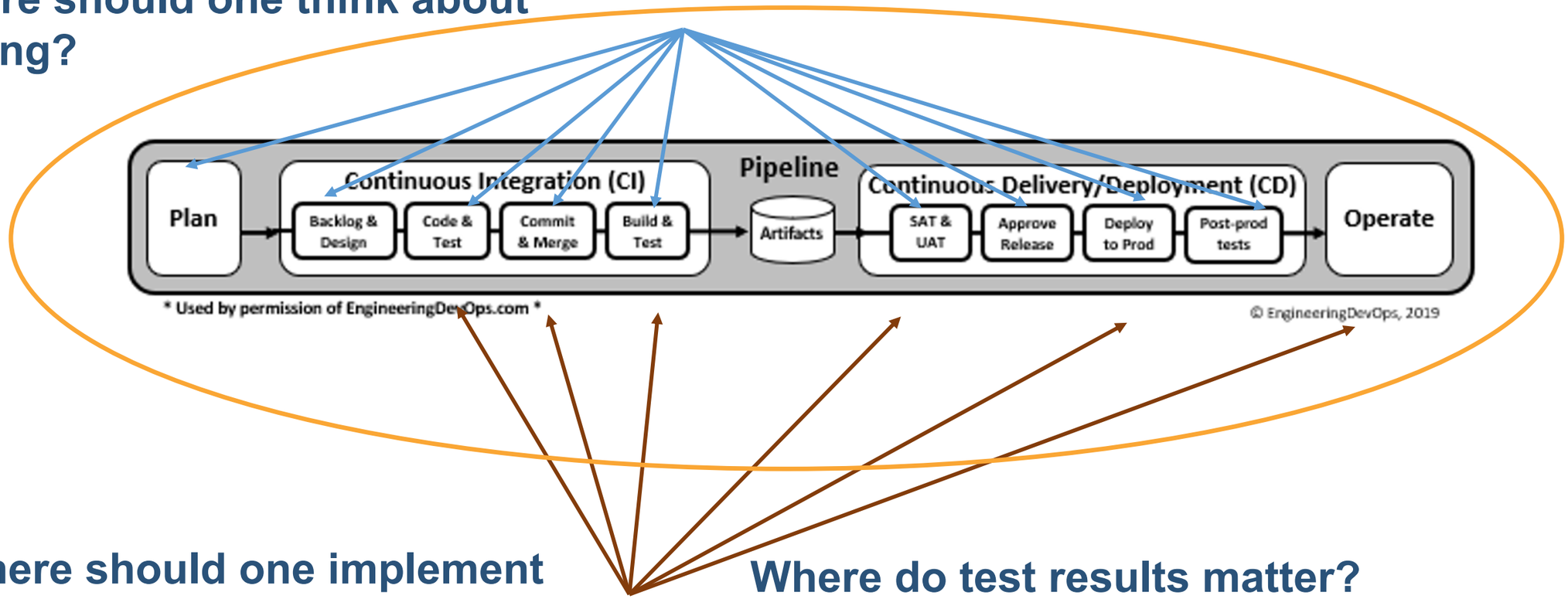
All these functions depend on testing. Automated staging and promotion can not happen without testing. If your automated pipeline requires manual testing



Continuous Testing in practice

Identify stages

Where should one think about testing?



Where should one implement testing?

Where do test results matter?

Framing testing practices

Planning for testing

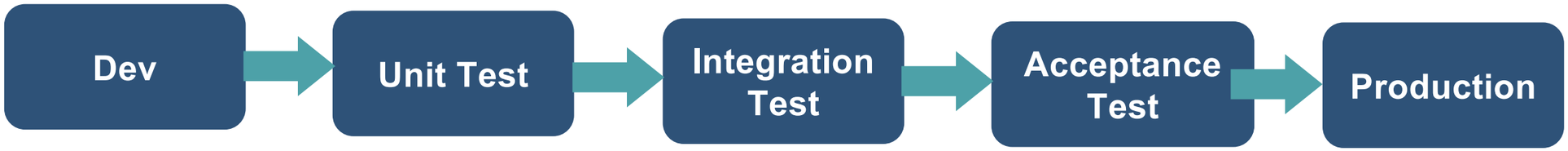
- Where should I have tests
- How do I decide what happens at each test stage?
- From Manual testing to automated testing

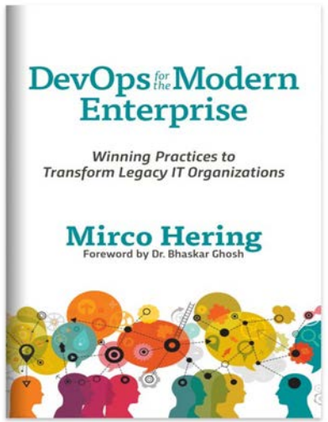
Continuous Delivery

Automatic trigger

Manual trigger

Continuous Deployment





From: Mirco Hering: notafactoryanymore.com, author of 'DevOps for the Modern Enterprise'

Tests in development

Unit Testing (Automated)	• Testing of the smallest piece of code that can be **logically isolated** • Function, subroutine, method or property
Code Quality (Parts Automated)	• Quality is subjective • Identify your metrics • Reliability, Maintainability, Testability, Portability, Reusability, Defects, Complexity
Static Analysis (Automated)	• Debugging before the program is run • Analyze code against coding rules • Sonarqube, Veracode (SAAS) , Fortify (HP)
Code Reviews (Largely Manual)	• Build a culture based on code review • Implement a process to review codes

Implementing development tests

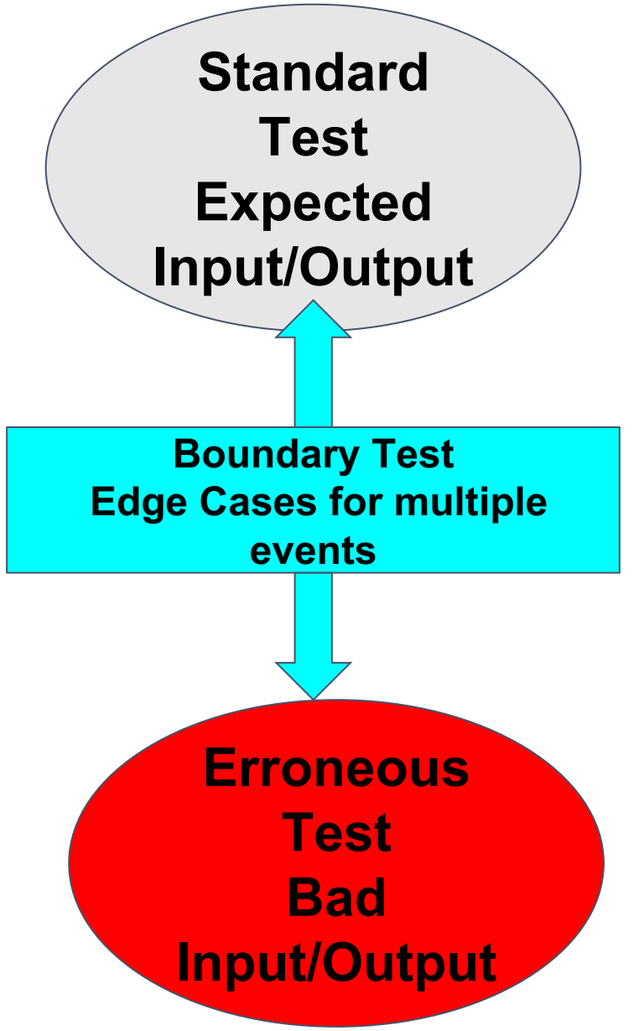
- Unit tests are unique to coding language
  - Find the best tool for you
  - Multiple tools can be a good thing
  - Move quickly from unit test to static testing
- Code quality
  - Establish metrics based on value stream
  - Know what matters most
- Code Reviews
  - Tips for code Review
    - Less than 400 lines at a time (200-400)
    - Inspection rate less than 500 lines an hour, and never for more than an hour
  - Tips to help code review
    - Establish a process to fix defects
    - Foster a positive code review culture
    - Practice lightweight reviews

Tests in integration

Data Testing (Automated)	• Check the information with which the code must interact • Bitwise QualDI, ICEDQ, RightdData
End to End (Automated)	• Test application workflow from beginning to end • Replicates real scenarios - Very valuable with data testing
Regression (Automated)	• Re-run functional and non-functional tests to ensure software performs after a change
Performance (Automated)	• Ensure software can perform under desired workload • Vroom, Vroom….

Implementing integration tests

- Data Testing
  - Three types of data - Standard, Erroneous, Boundary
- End to End Testing
  - Measure user functions, conditions, and test cases
  - Key to Test-driven Development
  - End state for automated testing
- Regression testing
  - Need to make sure one can go back to a previous state
  - When a bug is found and fixed, regression is essential
- Performance Testing
  - What is the expected standard?
  - Apache JMeter, WebLOAD, BlazeMeter
  - Most performance tests are not free

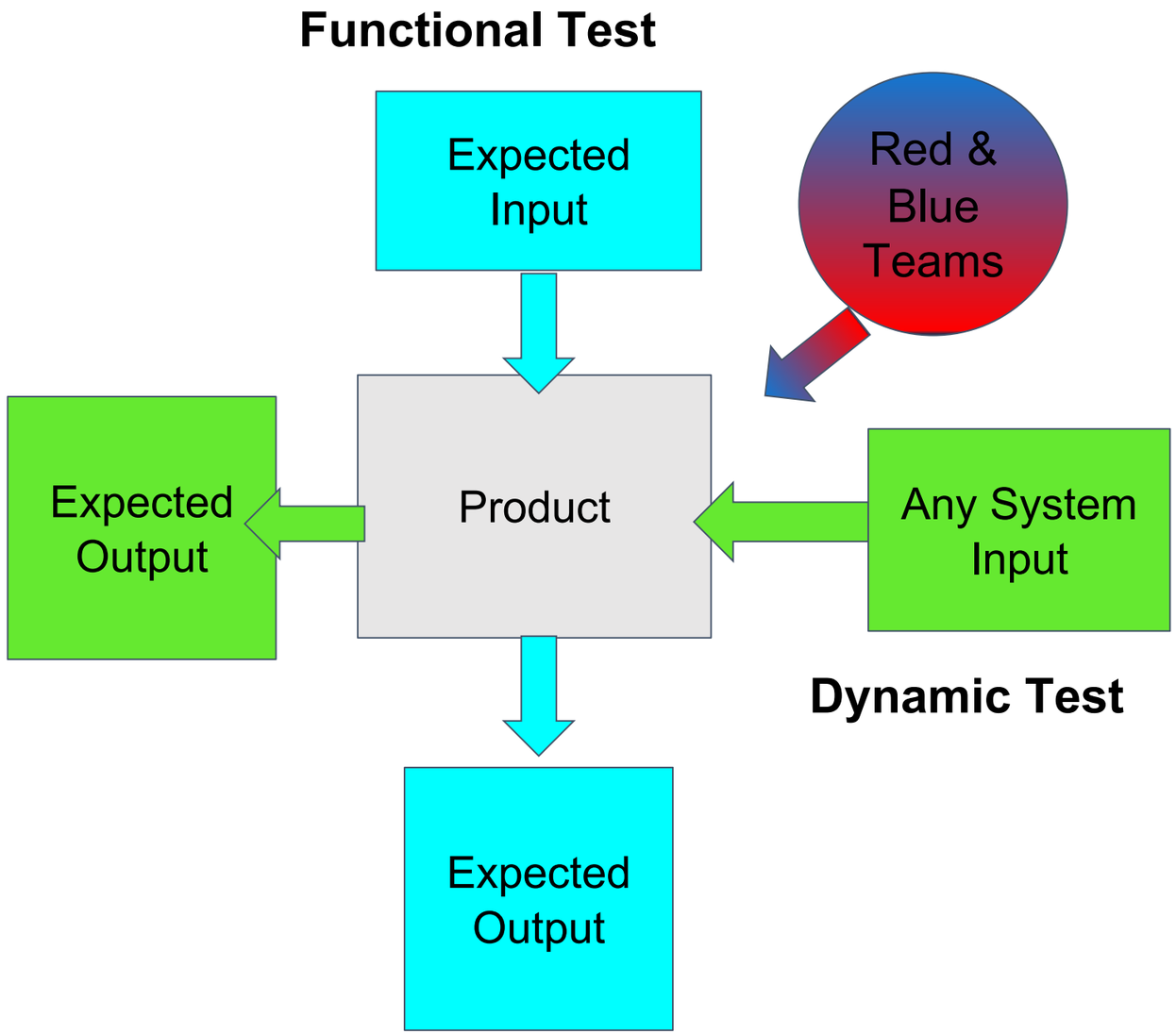


Tests in delivery

Functional (Partially Automated)	• Quality assurance testing based on blackbox formats • Input vs output, rarely considers internal
Dynamic (Partially Automated)	• Execute to find errors while program is running • Doesn’t look for code errors in program but in functions
Red/Blue Team (Blended Processes)	• Red teams are offensive professionals • Blue teams are defenders • Purple teams are integrated
Operational Testing (Manual)	• Brings in users to test additional capabilities

Implementing delivery tests

- Functional Testing
  - Set expectations for what should occur
  - Have clear inputs, outputs for each portion
  - Can use mocks and stubs to create desired results
- Dynamic Testing
  - More and more automated solutions
  - Tightly integrated with functional testing
- Red/Blue Team
  - Can be local
  - Can be professional
  - Can be expensive
- Operational testing
  - Might be unnecessary if other tests are sufficient
  - Sometimes a government requirement



Tests in production

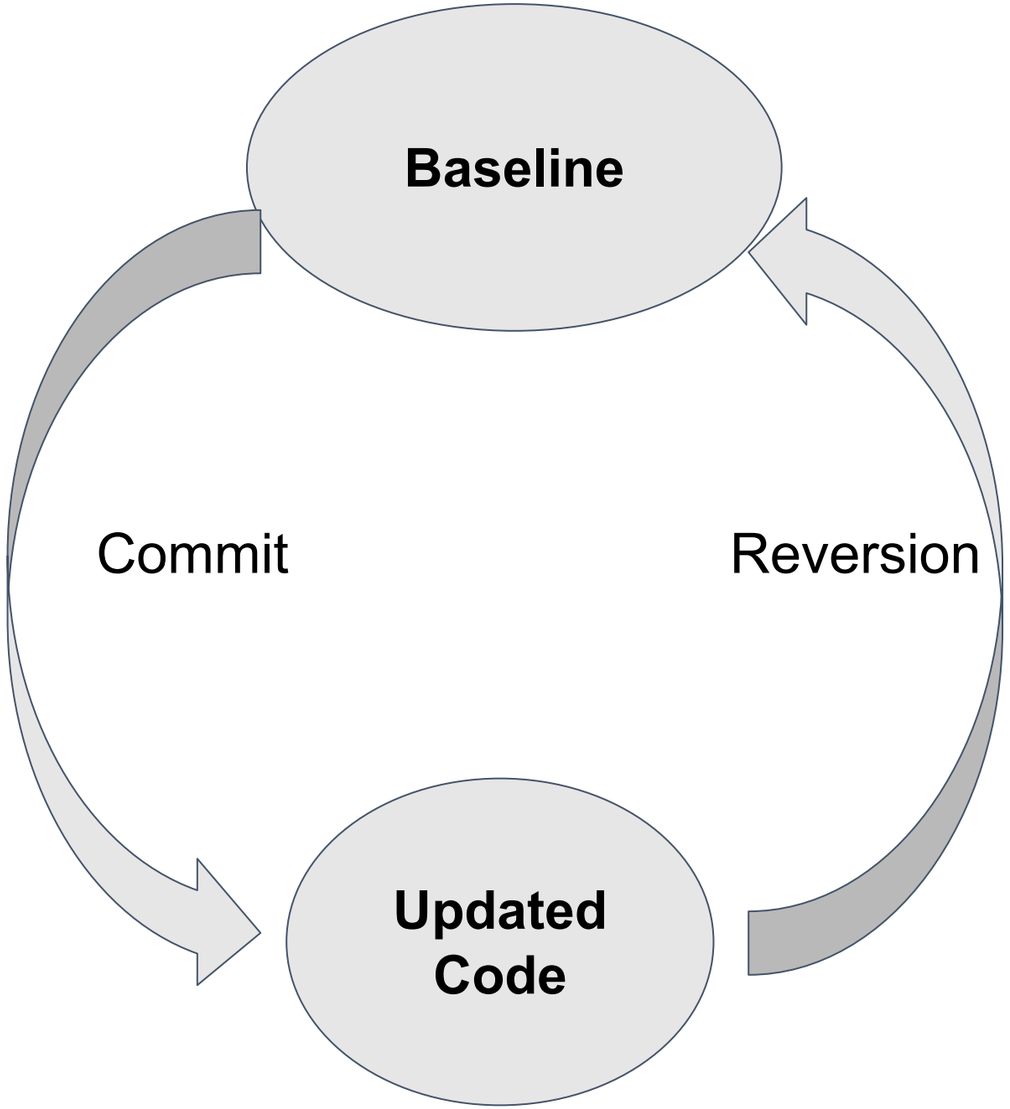
Run-Time Security (Automated)	• Uses runtime instrumentation to detect and block attacks inside running software • Observability and telemetry is key
Metrics, Logs, traces (Automated)	• Three ways of DevOps? Three components of observability
Reversion (Either)	• Similar to regression testing, typically occurs on full system

Implementing production tests

- Run-time security
  - Extends from dynamic testing frameworks
  - Best used for large enterprises with constantly running tools
  - Checks for items not found in first test

- Metrics, Logs, Traces
  - Metrics are a combined assessment
  - Logs are all system functions by time
  - Traces follow a single path/function over time
  - Each offers advantages

- Reversion
  - Can revert in A/B formats on an automated basis based on metrics
  - Can revert manually based on user/management request



DevOps INSTITUTE

Designing and coding

Testing frameworks in delivery

VSCode	• IDE environment, multiple plugins available • Most IDEs offer some range
CodeGuru	• Program analysis to find defects in Java and Python Code • Works with multiple source providers
Robot Framework	• Generic test automation framework for acceptance testing - keyword driven with tabular syntax • www.robotframework.org
Security Scans (Automated)	• Test of network for known vulnerabilities • OWASP offers range of free tools • Nessus, BurpSuite, NetSparker, Tenable, Snyk, AWS Inspector
Differential Testing (Automated)	• Checking outputs against other outputs • Uses for security to measure risk in changing vulnerabilities

Find a test case design

Find somewhere you can practice

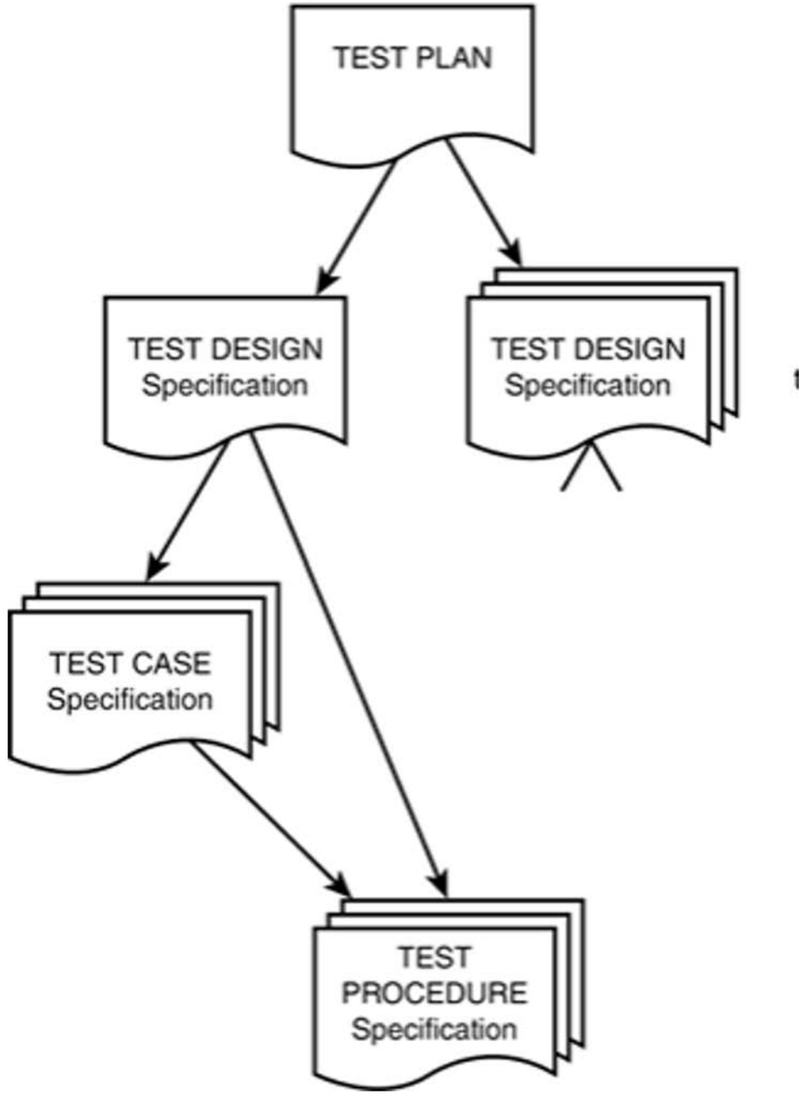
- Practical Test Pyramid - https://martinfowler.com/articles/practical-test-pyramid.html
- Robot Framework https://robotframework.org/#learning
- Axelrod's Complete Guide to Test Automation (https://www.apress.com/gp/book/9781484238318)

Different solutions work for different individuals

Multiple areas such as Udemy offer tests for different

- Code (Java, Python, C+, C#, Go, Ruby etc.)
- Tests (Development, Integration, Unit, functional, regression, runtime)
- Combined frameworks (Junit, Selenium, Robot Framework, Cucumber, Avocado

Find what works for you



Design the test case

- What do you want to test?
- What is the expected answer?
- What happens if test fails?
- Checking previous tests
- I am not a Robot expert
  - Going to use framework for examples
  - Free courses here: https://robocorp.com/docs/courses
  - Offers a community of practice for help in building and automating tests

1. Designing a test case
2. Analyze requirements
3. Set up a test environment
4. Analyze software/hardware needs
5. List how systems should respond
6. List testing methods
7. Design test cases
8. Run tests, study, save results

Test Case Type	Description	Test Step	Expected Result	Status
Functionality	Area should accommodate up to 20 characters	Input up to 20 characters	All 20 characters in the request should be appropriate	Pass or Fail
Security	Verify password rules are working	Create a new password in accordance with rules	The user's password will be accepted if it adheres to the rules	Pass or Fail
Usability	Ensure all links are working properly	Have users click on various links on the page	Links will take users to another web page according to the on-page URL	Pass or Fail

Code the test case

- Robot Framework offers an extensive array of libraries
  - https://github.com/robotframework/PythonLibCore Sample for building python libraries
  - https://franz-see.github.io/Robotframework-Database-Library/ - Queries databases for accuracy after change
  - https://github.com/MarketSquare/robotframework-difflibrary Compares files together - useful for variables and documentation
- Robot offers a quickstart to code some initial tests
  - https://github.com/robotframework/QuickStartGuide/blob/master/QuickStart.rst

Sample for User Login

```
*** Test Cases ***
User can change password
    Given a user has a valid account
    When she changes her password
    Then she can log in with the new password
    And she cannot use the old password anymore
```

Tabular Case

```
*** Test Cases ***
Invalid password
    [Template]    Creating user with invalid password should fail
    abCD5             ${PWD INVALID LENGTH}
    abCD567890123     ${PWD INVALID LENGTH}
    123DEFG           ${PWD INVALID CONTENT}
    abcd56789         ${PWD INVALID CONTENT}
    AbCdEfGh          ${PWD INVALID CONTENT}
    abCD56+           ${PWD INVALID CONTENT}
```

```
*** Test Cases ***
User can create an account and log in
    Create Valid User    fred    P4ssw0rd
    Attempt to Login with Credentials    fred    P4ssw0rd
    Status Should Be    Logged In

User cannot log in with bad password
    Create Valid User    betty    P4ssw0rd
    Attempt to Login with Credentials    betty    wrong
    Status Should Be    Access Denied
```

Complete the test case

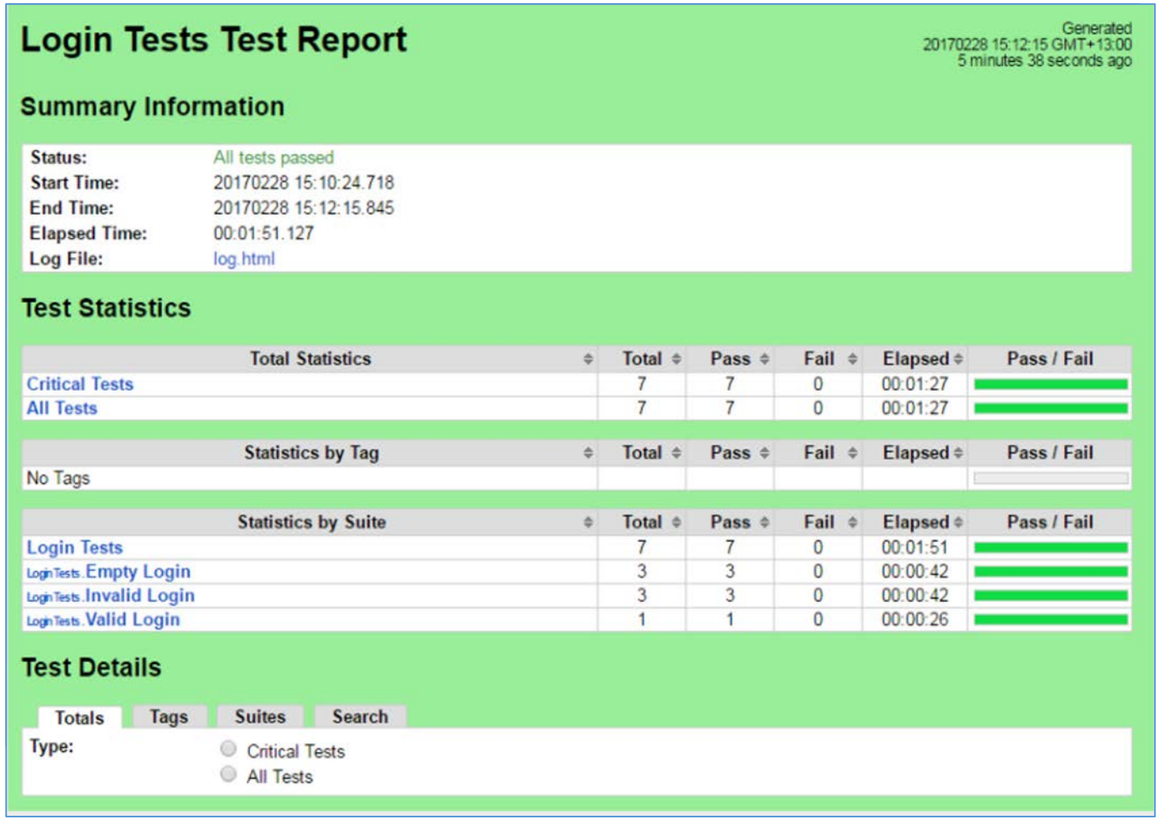
- The test case should run against a library to validate success
- Code checks against the database file
- Clear results

```
*** Keywords ***
Clear login database
    Remove file    ${DATABASE FILE}

Create valid user
    [Arguments]    ${username}    ${password}
    Create user    ${username}    ${password}
    Status should be    SUCCESS

Creating user with invalid password should fail
    [Arguments]    ${password}    ${error}
    Create user    example    ${password}
    Status should be    Creating user failed: ${error}

Login
    [Arguments]    ${username}    ${password}
    Attempt to login with credentials    ${username}    ${password}
    Status should be    Logged In
```



Login Tests Test Report

Generated
20170228 15:12:15 GMT+13:00
5 minutes 38 seconds ago

Summary Information

Status:	All tests passed
Start Time:	20170228 15:10:24.718
End Time:	20170228 15:12:15.845
Elapsed Time:	00:01:51.127
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	7	7	0	00:01:27	
All Tests	7	7	0	00:01:27	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
Login Tests	7	7	0	00:01:51	
LoginTests.Empty Login	3	3	0	00:00:42	
LoginTests.Invalid Login	3	3	0	00:00:42	
LoginTests.Valid Login	1	1	0	00:00:26	

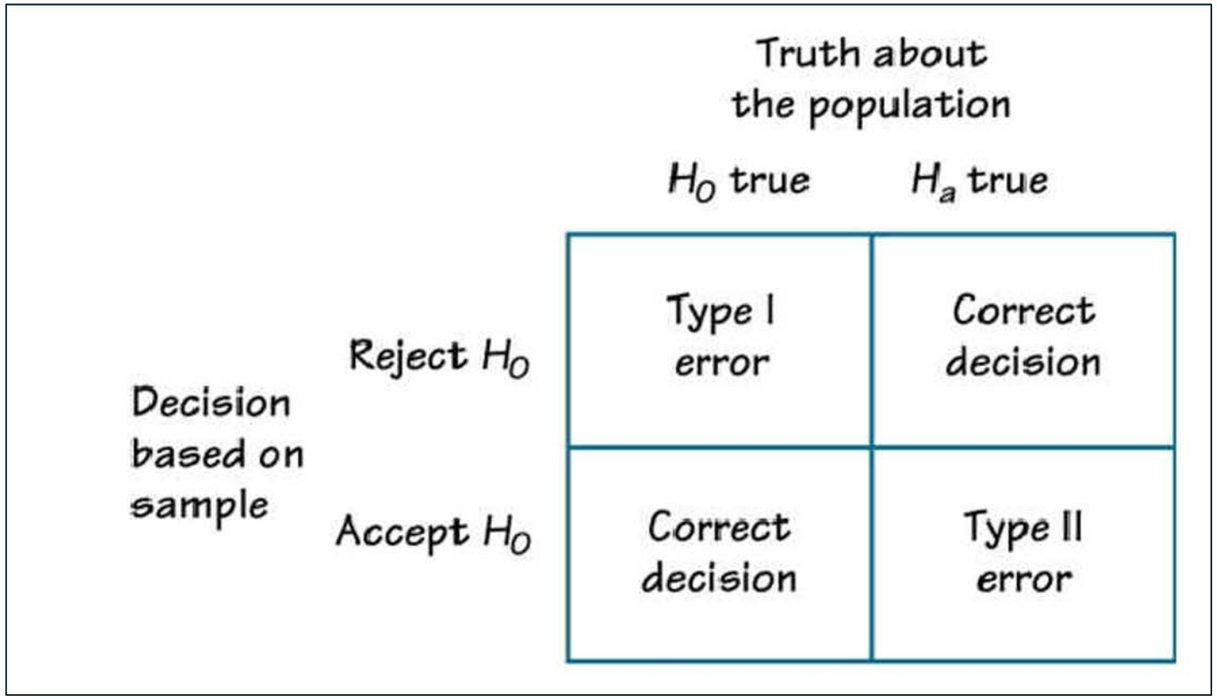
Test Details

Totals | Tags | Suites | Search

Type: Critical Tests / All Tests

Common errors and solutions

- Check with integration with tools
- Look for more fidelity in results
- Don’t debug immediately - gather results, and fix the whole
- Investigate root causes
- Resolve the problem and test again
  - Quick resolutions might not fix all the issues
- Capture the data, and get help



Challenging errors

- Some failures only happen on one machine
  - Test for multiple environments
  - Ensure environment (especially with VM) is correctly duplicated
- Are you running from IDE?
  - Or from the command line?
- Narrow the code to the specific problem
- Investigate influencing tests
- Flickering Tests
  - Some tests occur at unusual intervals
  - Wait for events to complete
  - Ensure area is not caused by test

Visual Studio Code

Next Steps

- Test automation contributes to flow and feedback in the DevOps model
  - Up to you to contribute learning
- Take online tutorials
  - Follow-on for this webinar offered by AWS
- Join a community of practice
  - Sign up for SKIL-Up Days with DevOps Institute
  - CT Foundation Course
- Practice, Practice, Practice

aws marketplace

Dilip Rajan

Sr. Partner Solution Architect at AWS

dilip-rajan-65643919

Agenda

1. Software Quality and Agility
2. Amazon CodeGuru
3. Code Quality with CodeGuru Reviewer
4. Application performance with CodeGuru Profiler
5. End to end developer workflow with Partners

Software agility and quality

aws | aws marketplace

Amazon CodeGuru

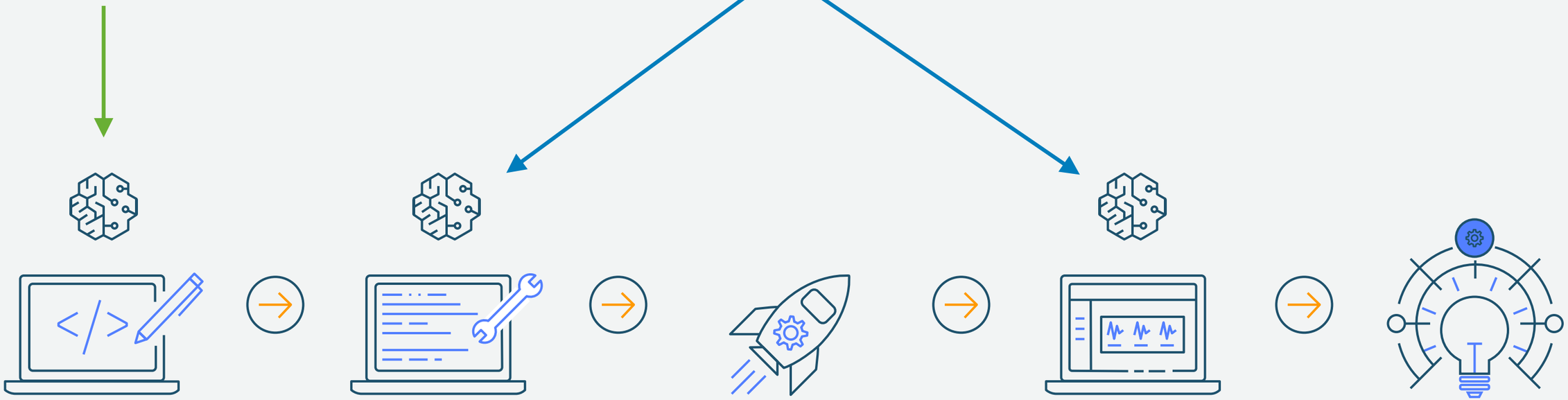
aws | aws marketplace

Amazon CodeGuru

Using ML to Code Review and Optimize High-Performing Applications

CodeGuru Reviewer

CodeGuru Profiler



CODING:

Built-in code reviews with actionable recommendations

BUILD & TEST:

Detect and optimize the expensive lines of code pre-production

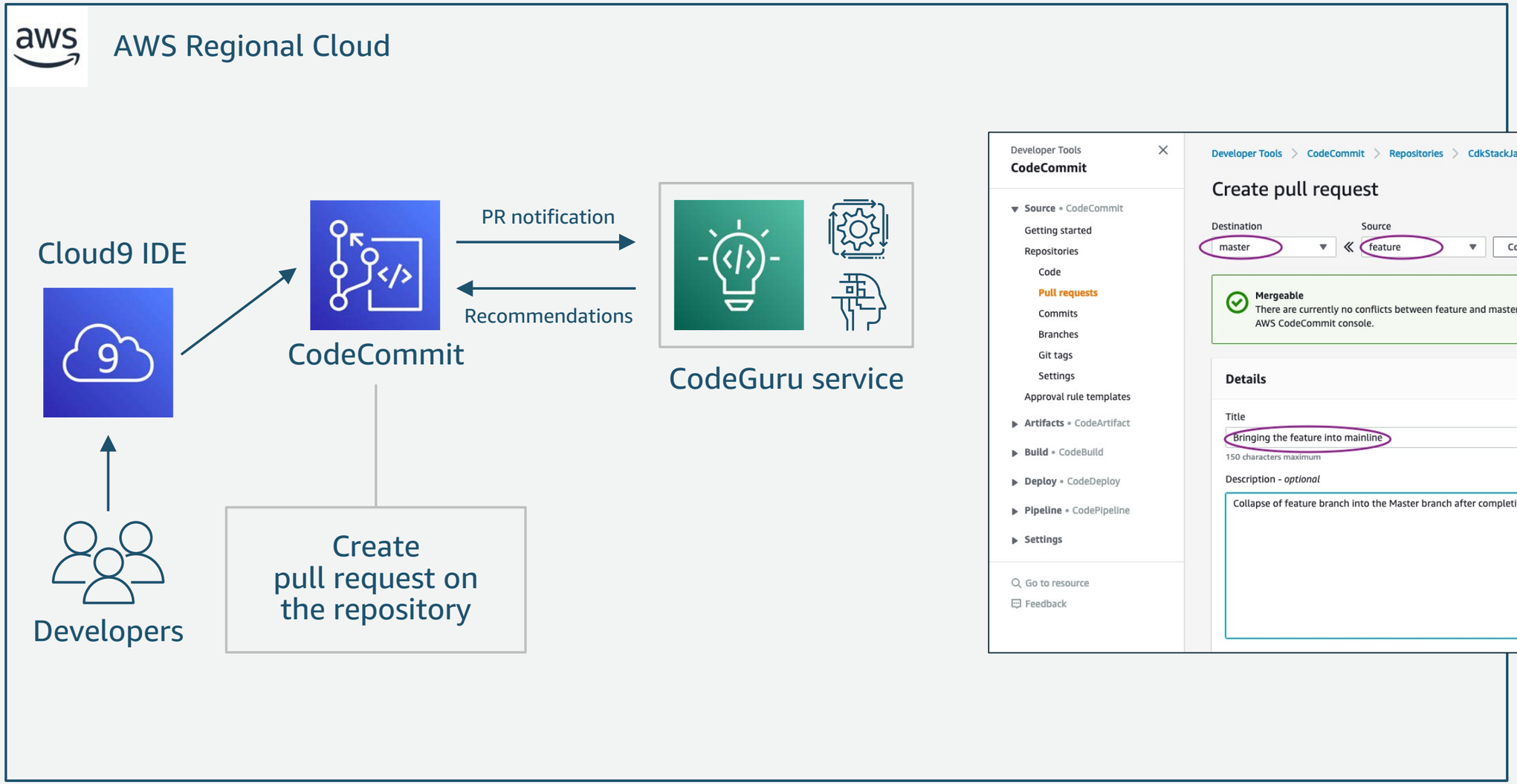
DEPLOY

MEASURE:

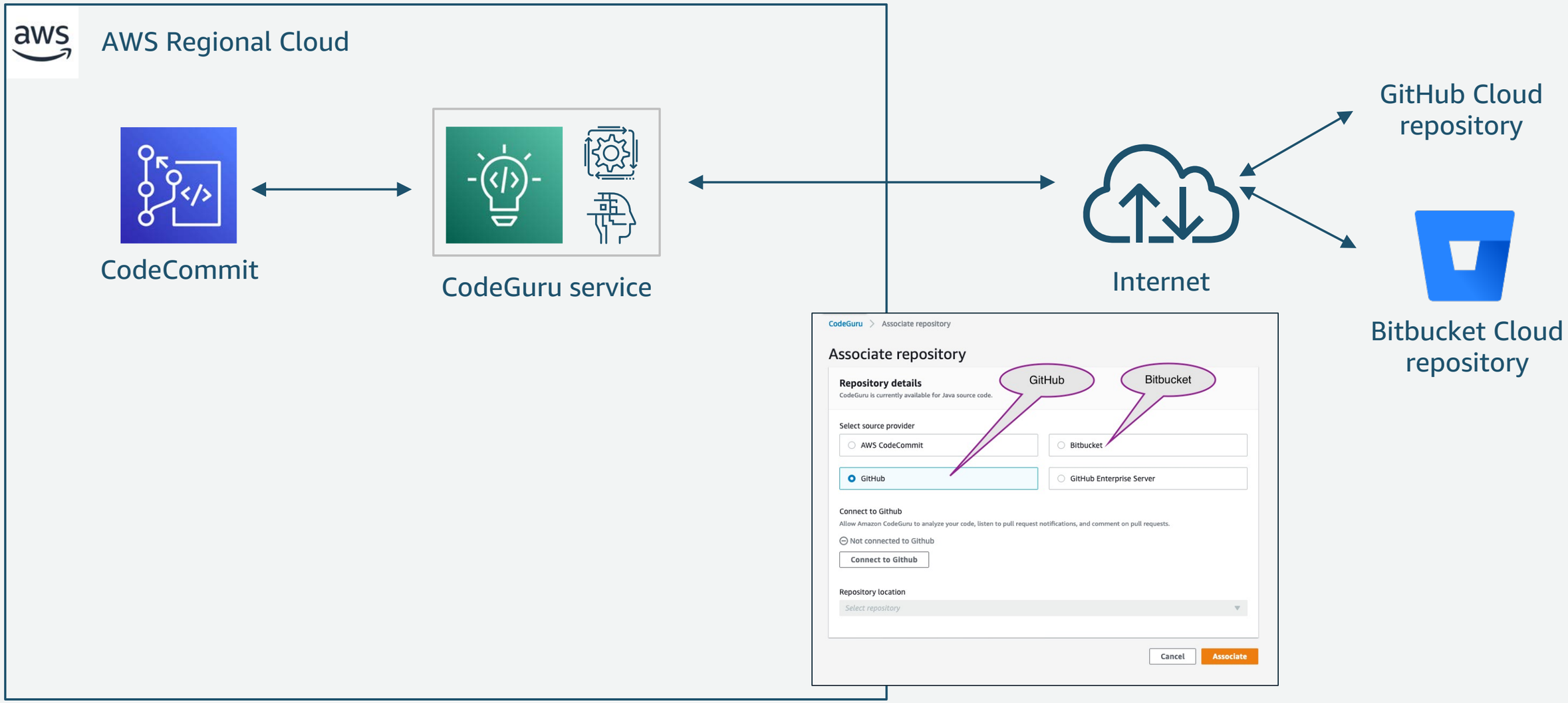
Easily identify performance and cost improvements in production environment

aws | aws marketplace

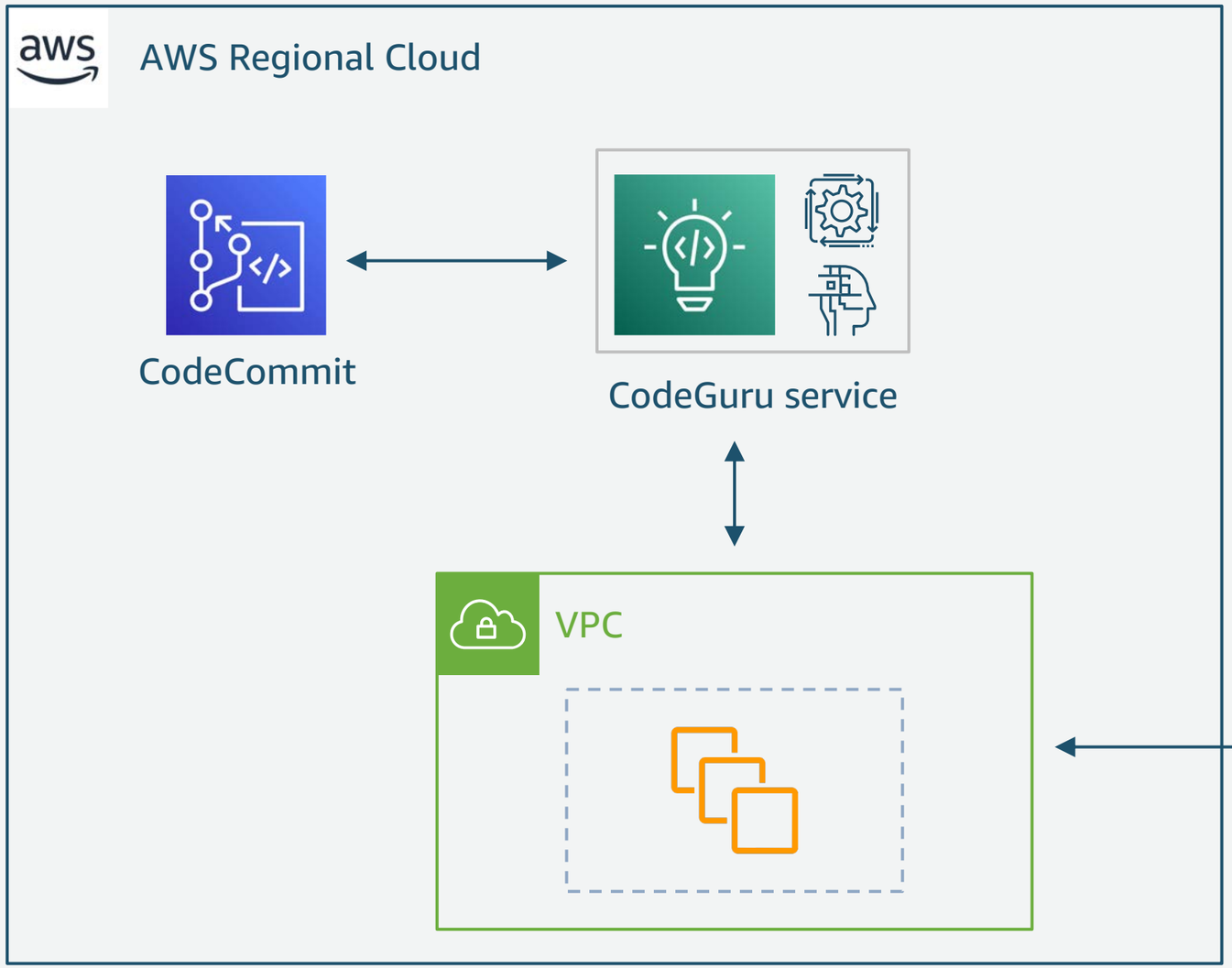
1. Triggering CodeGuru code review – Pull request

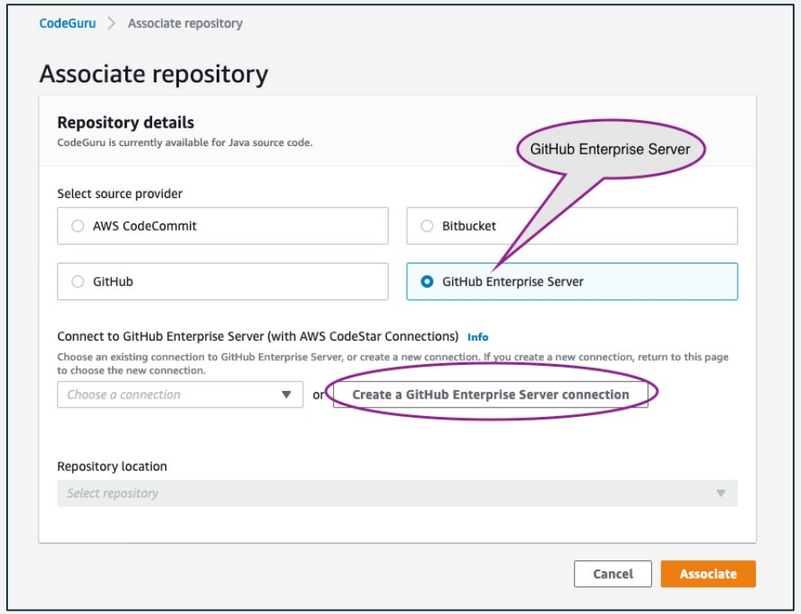


CodeGuru repository support

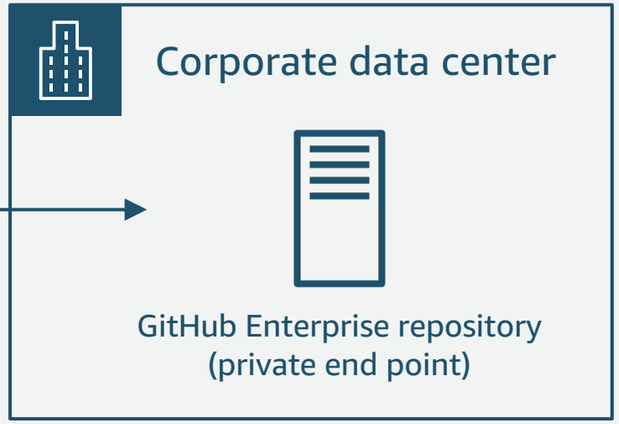


CodeGuru repository support





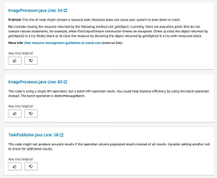
Direct Connect connectivity



aws | aws marketplace

2. Triggering CodeGuru code review - dashboard

AWS Regional Cloud



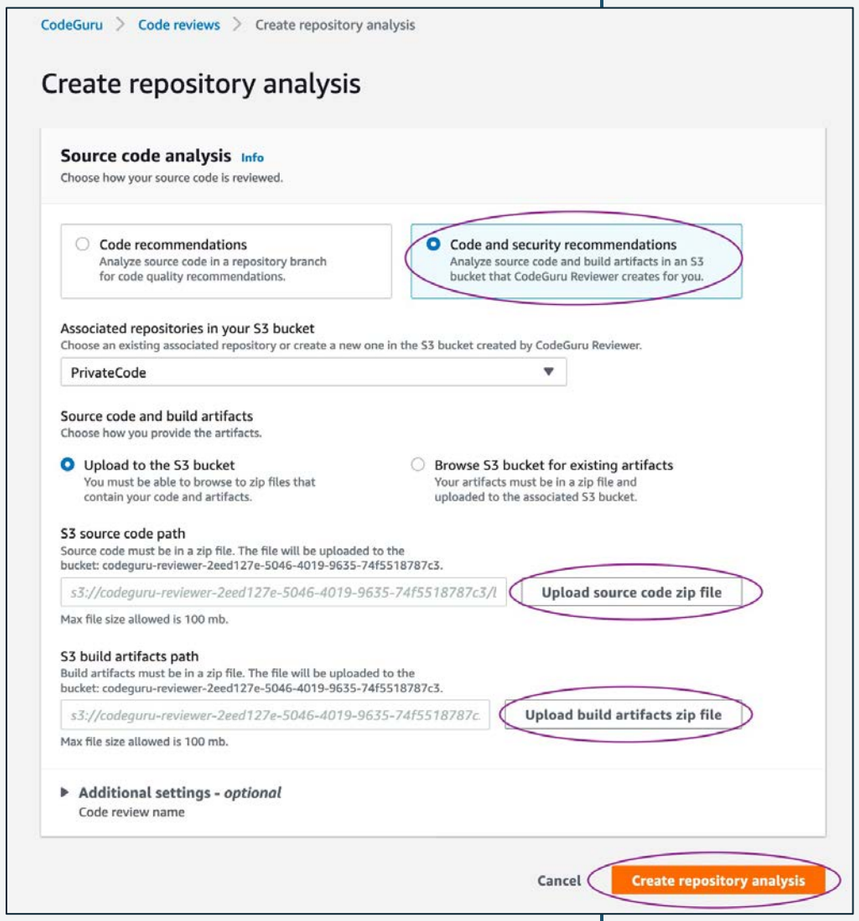
Recommendations

Scan

Associated S3 Bucket
with Code + Build artifact

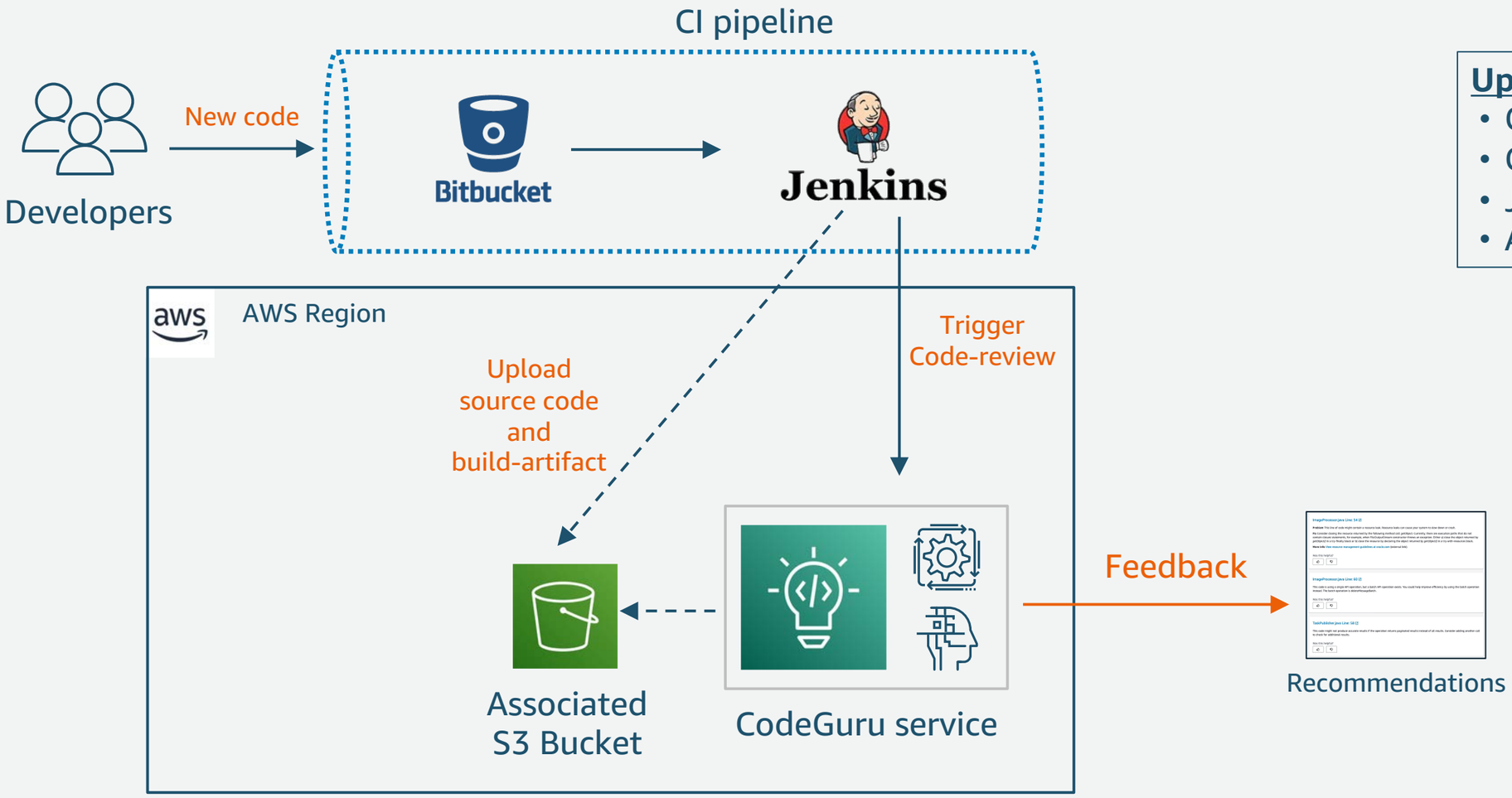
CodeGuru service

Trigger code-review scan



aws | aws marketplace

3. Coming soon: CodeGuru using Jenkins CI pipeline



Upcoming CI pipeline Support:

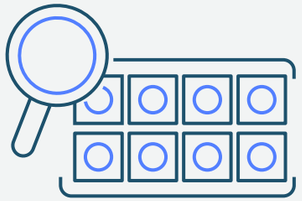
- GitHub Actions
- GitLab
- Jenkins
- AWS CodePipeline

aws | aws marketplace

Code areas addressed by CodeGuru Reviewer

AWS best practices

Correct use of AWS APIs

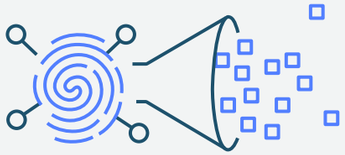


Concurrency

Correct implementation of concurrency constructs

Resource leaks

Correct resource handling



Sensitive information

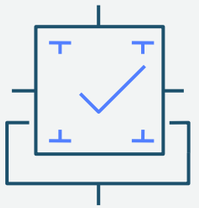
Unintended disclosure of personally identifiable information

Code efficiency

Hard-to-find defects

Code duplication

Refactor duplicate/ similar lines of code



Input validation

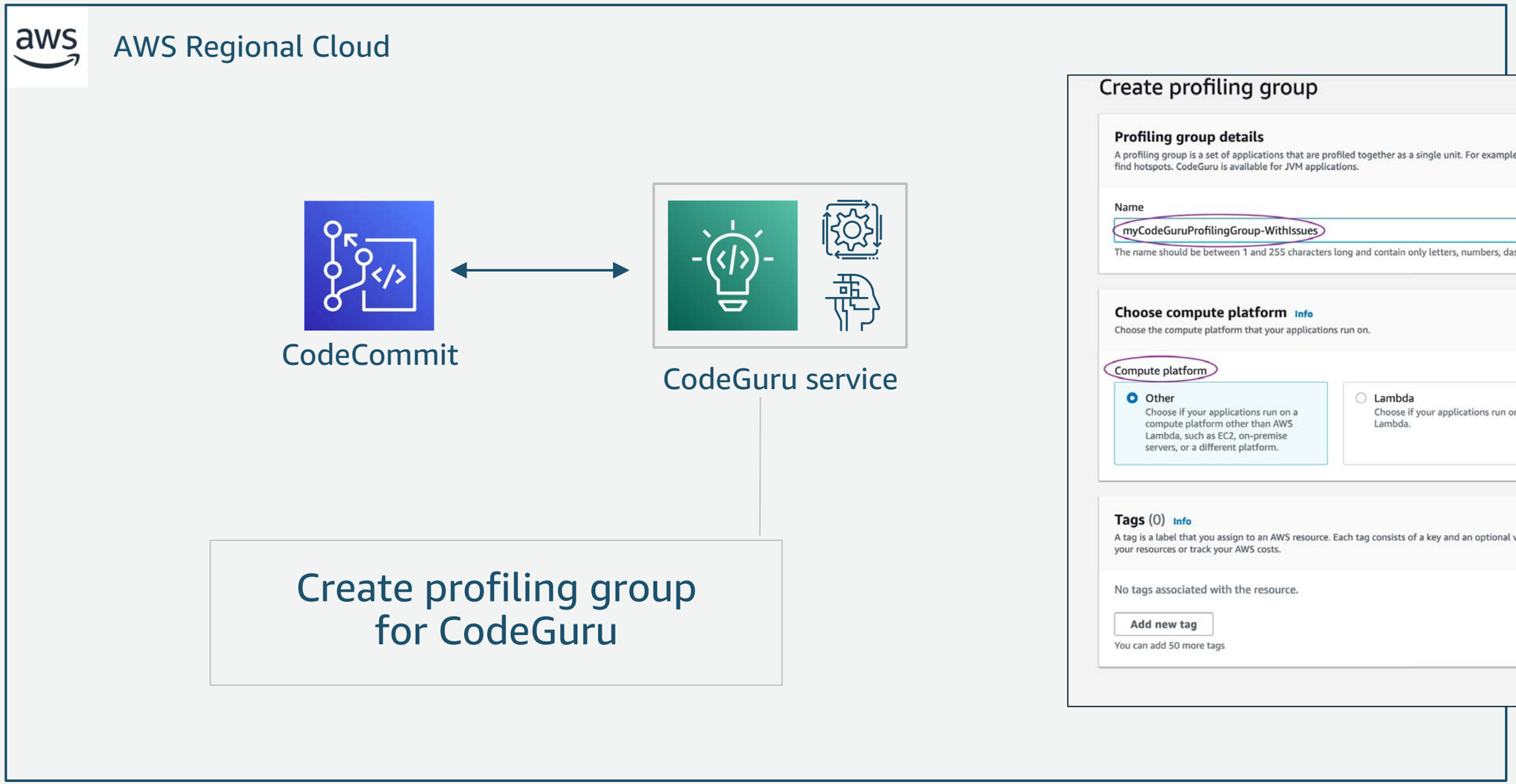
Ensure sufficient validation is done before processing the input data

aws | aws marketplace

Categories of security code issues addressed by Reviewer

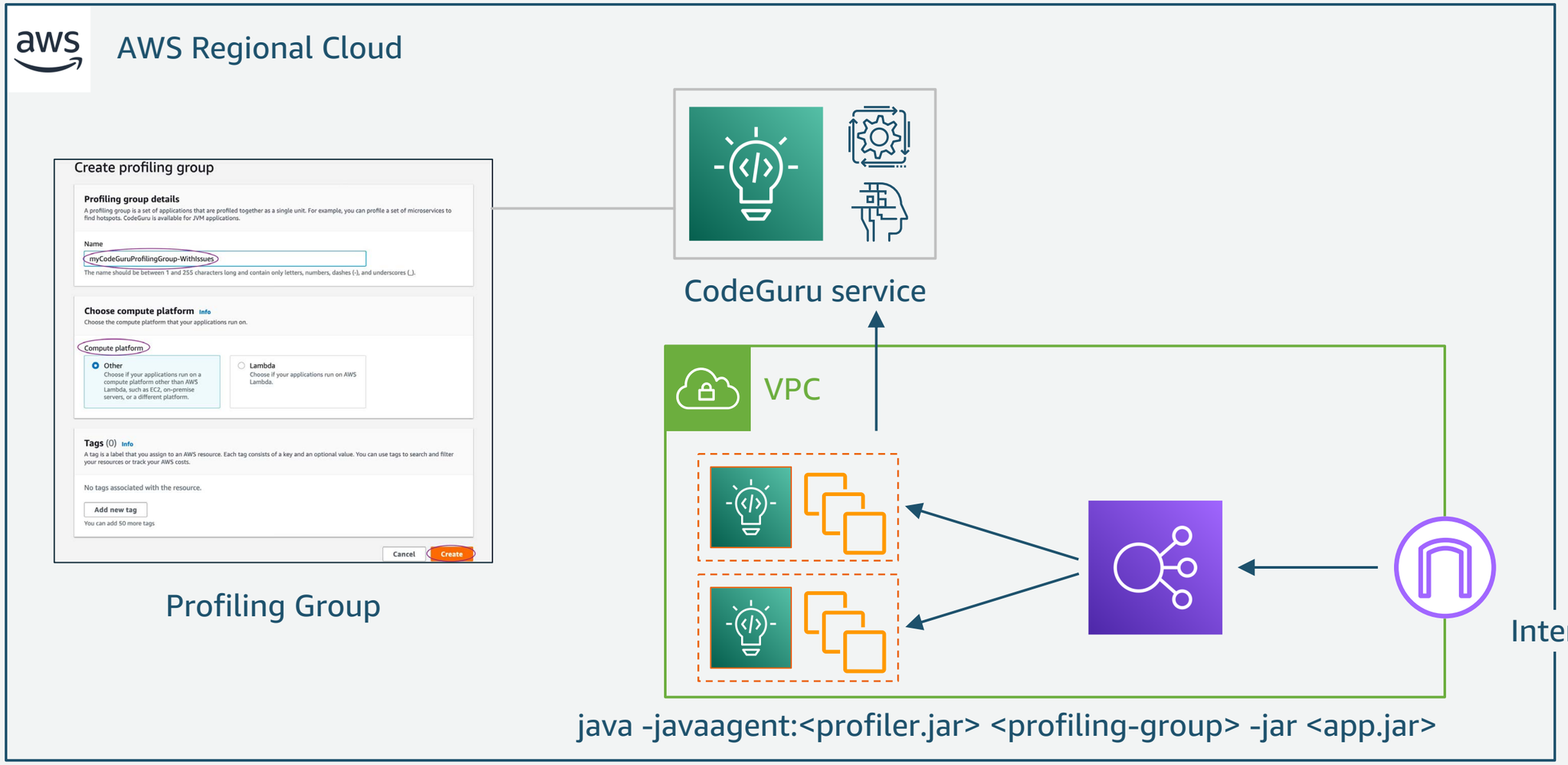
- AWS API Security Best Practices: *Following security best-practices when using AWS APIs.*
- Java Crypto Library Best Practices: *Best practices for common Java cryptography libraries.*
- Secure Web Applications: *Not sanitizing user-supplied can result in several security issues (e.g., cross-site scripting, SQL injection, LDAP injection, path traversal injection, etc.).*
- Sensitive Information Leak: *Leakage of sensitive information (e.g., logging of credit card number) leads to compliance issues.*
- AWS Security Best Practices: *Developed in collaboration with AWS security, bringing internal security expertise to our customers.*

CodeGuru onboarding – Profiler

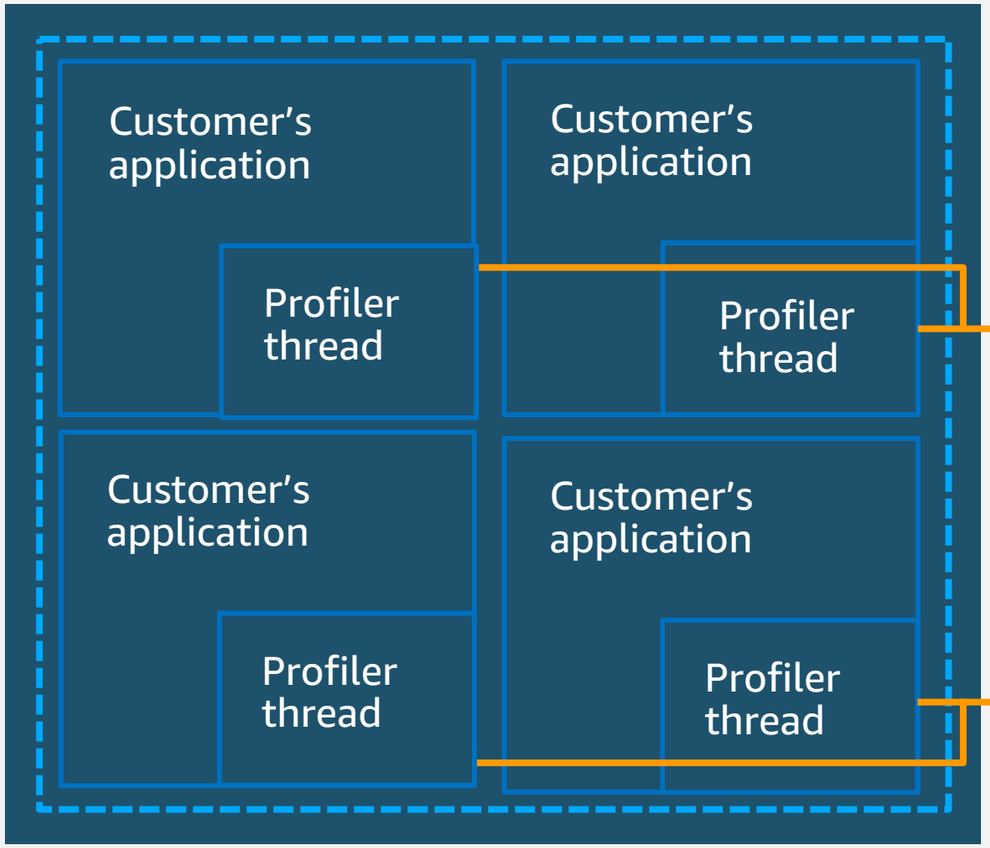


CodeGuru Profiler workflow

EC2 – Compute Instance

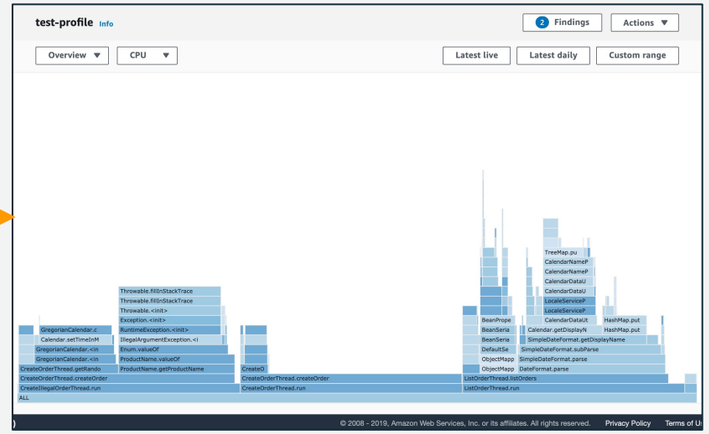


CodeGuru Profiler – how it works



Customer Application

CodeGuru Profiler backend



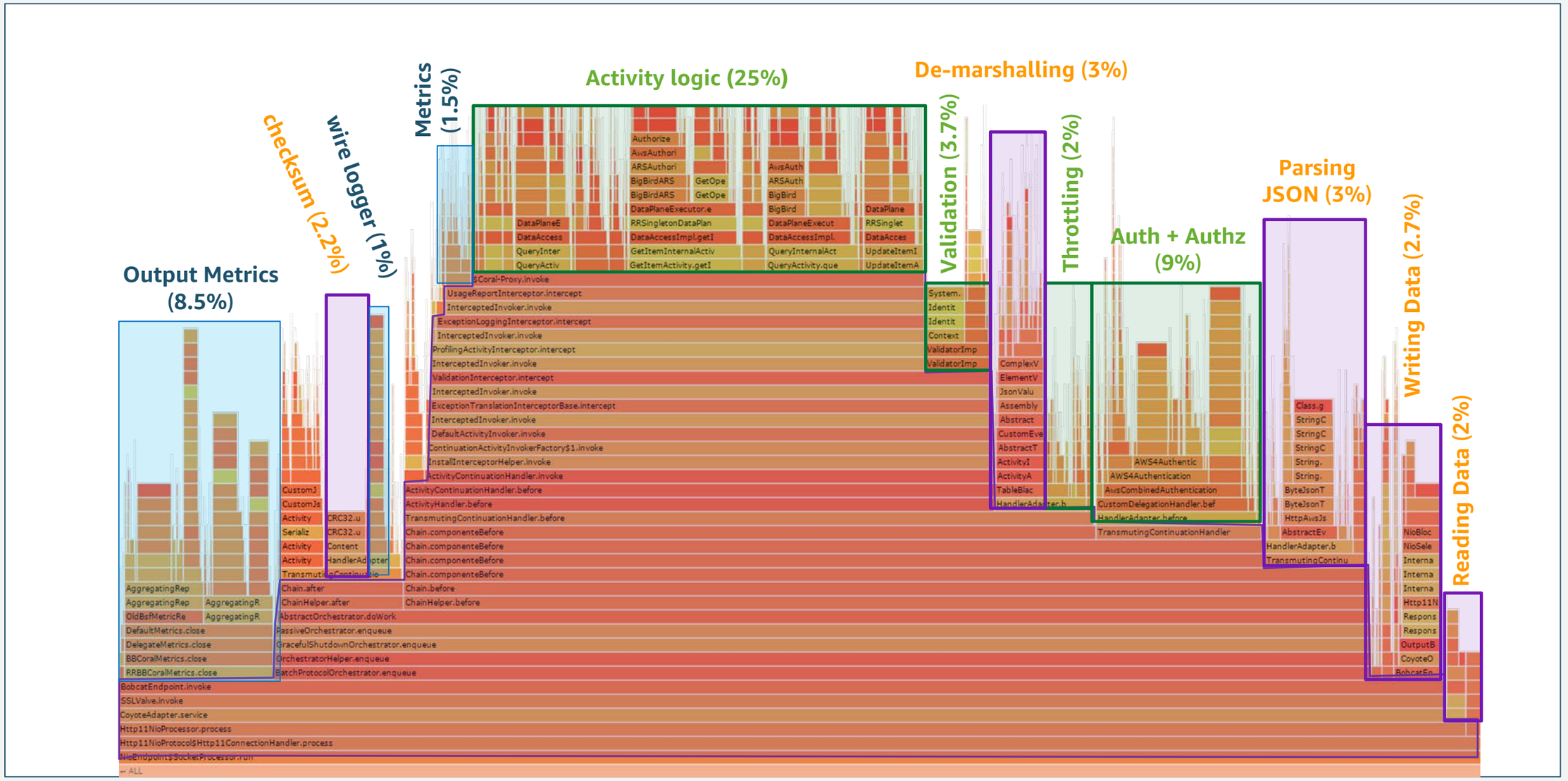
Console – Visualizations

Actionable recommendations

Console – Recommendation reports

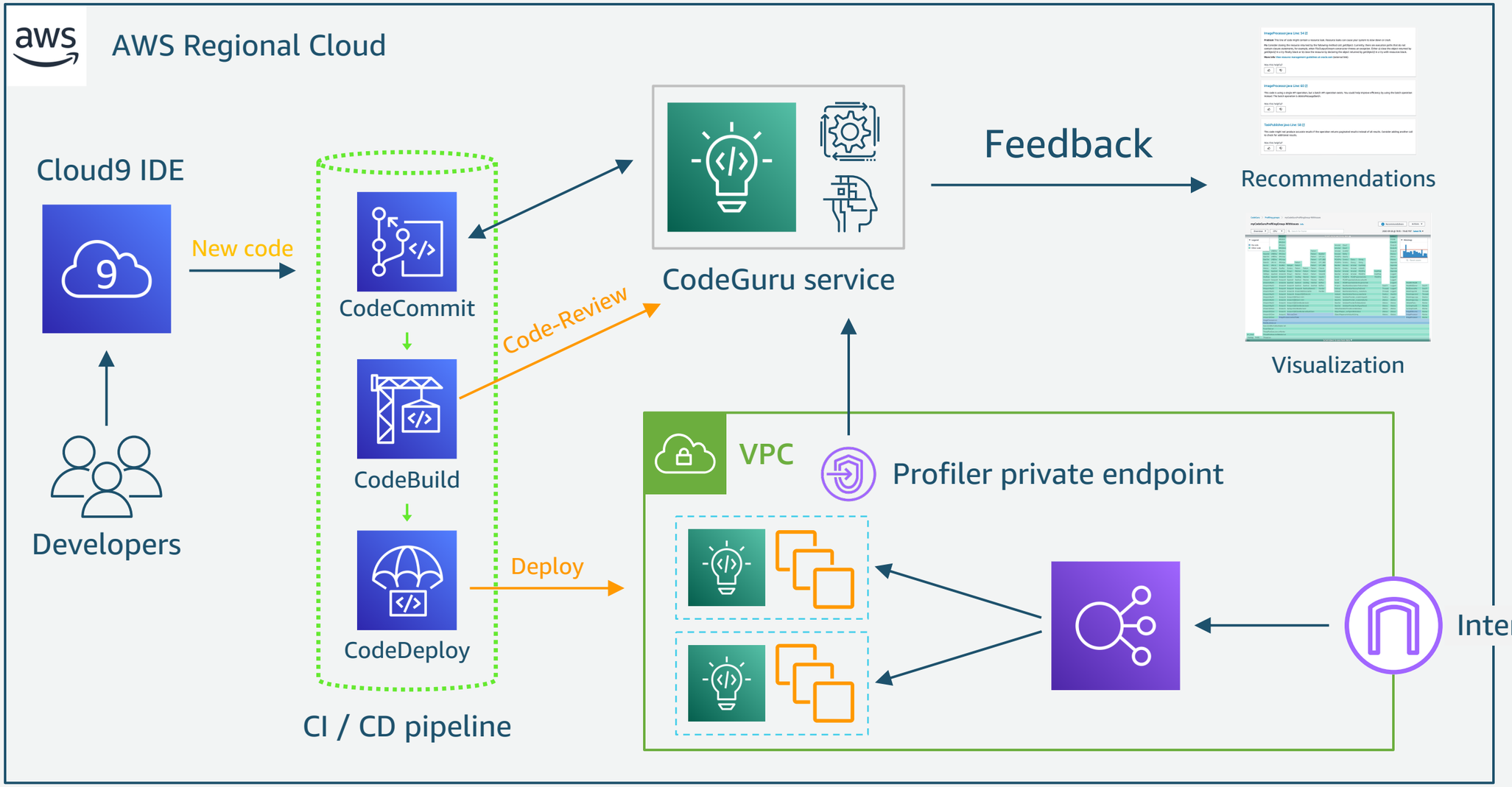
aws | aws marketplace

See where your app is spending its time – is it doing the right thing?

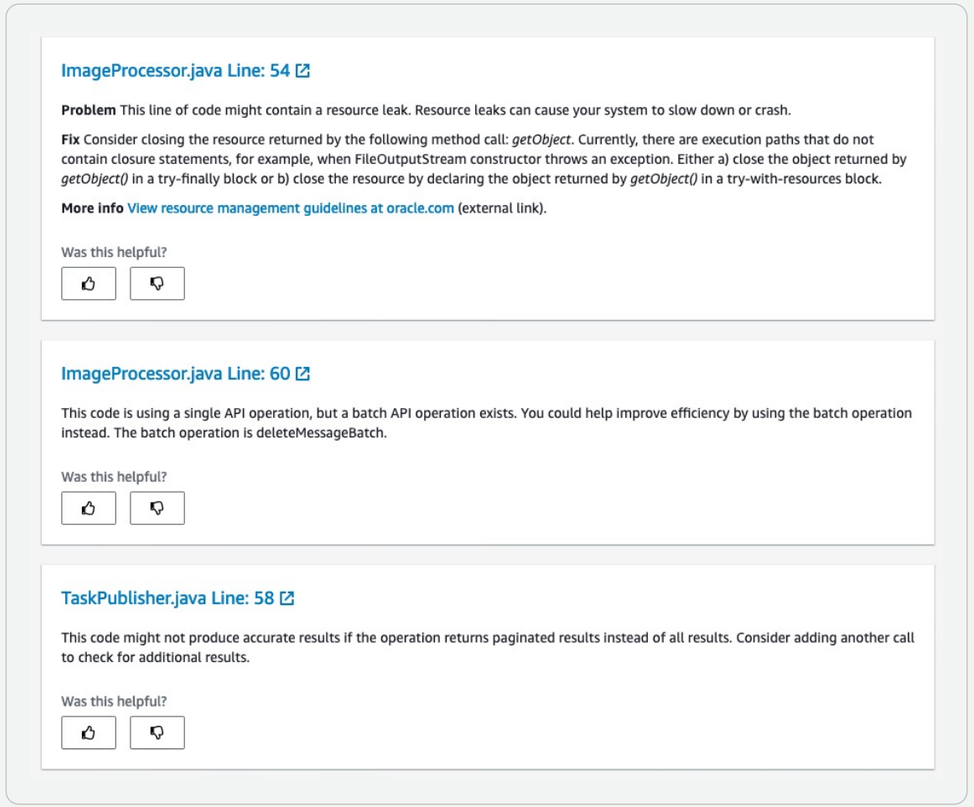


aws | aws marketplace

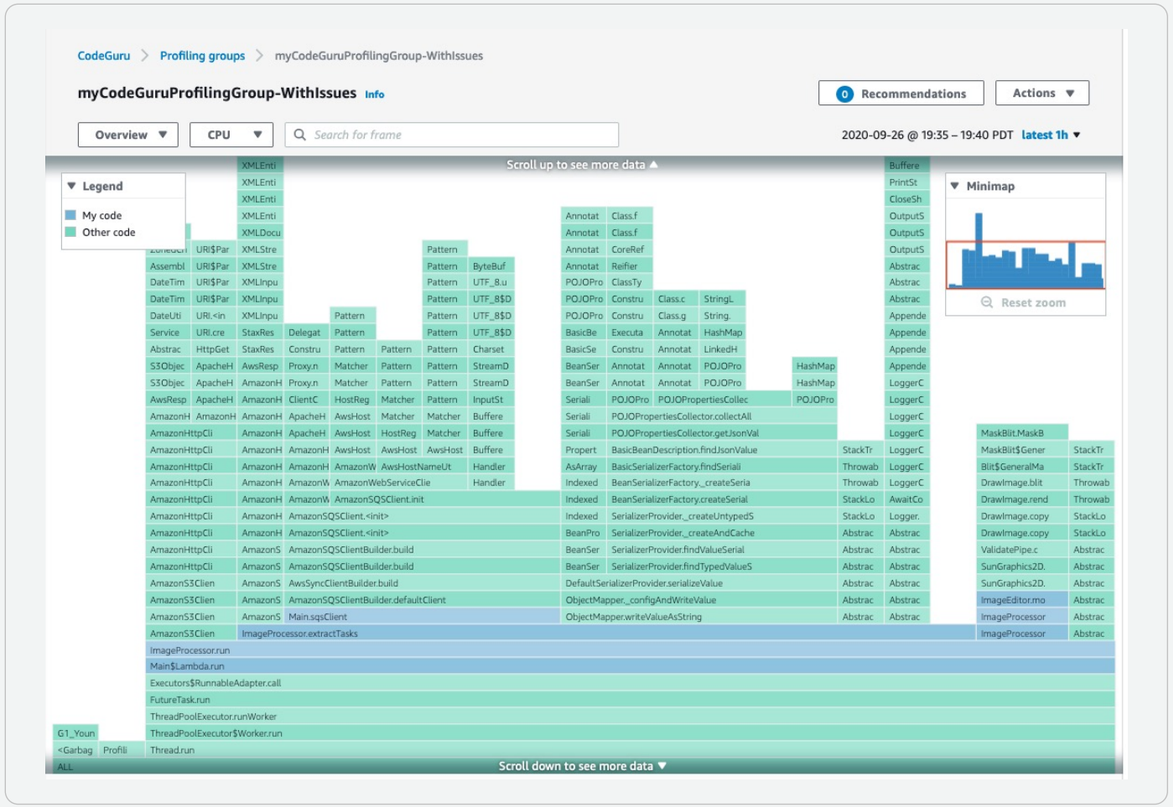
End-to-end CodeGuru workflow



CodeGuru output



Recommendations

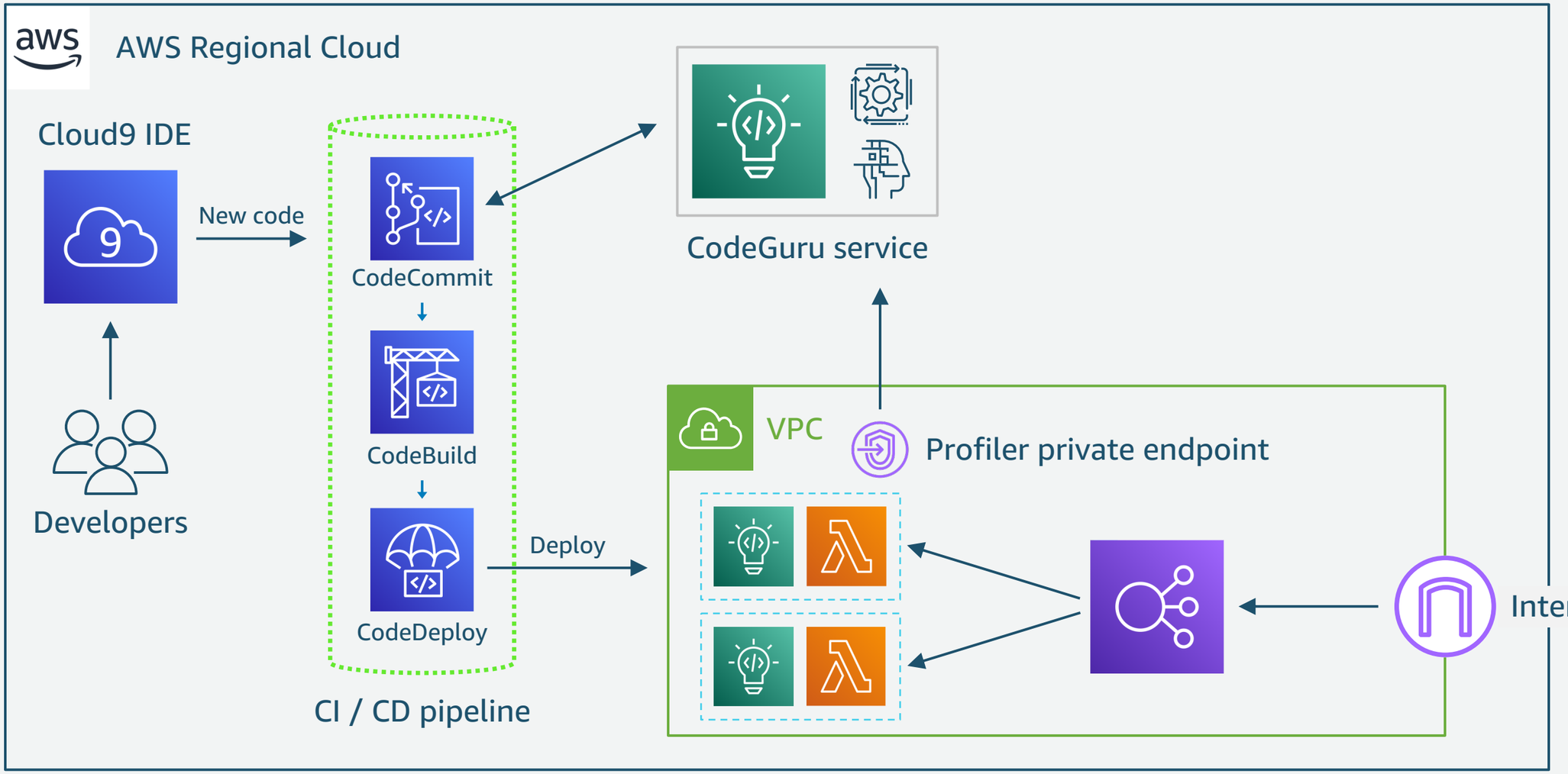


Application performance Visualization

aws | aws marketplace

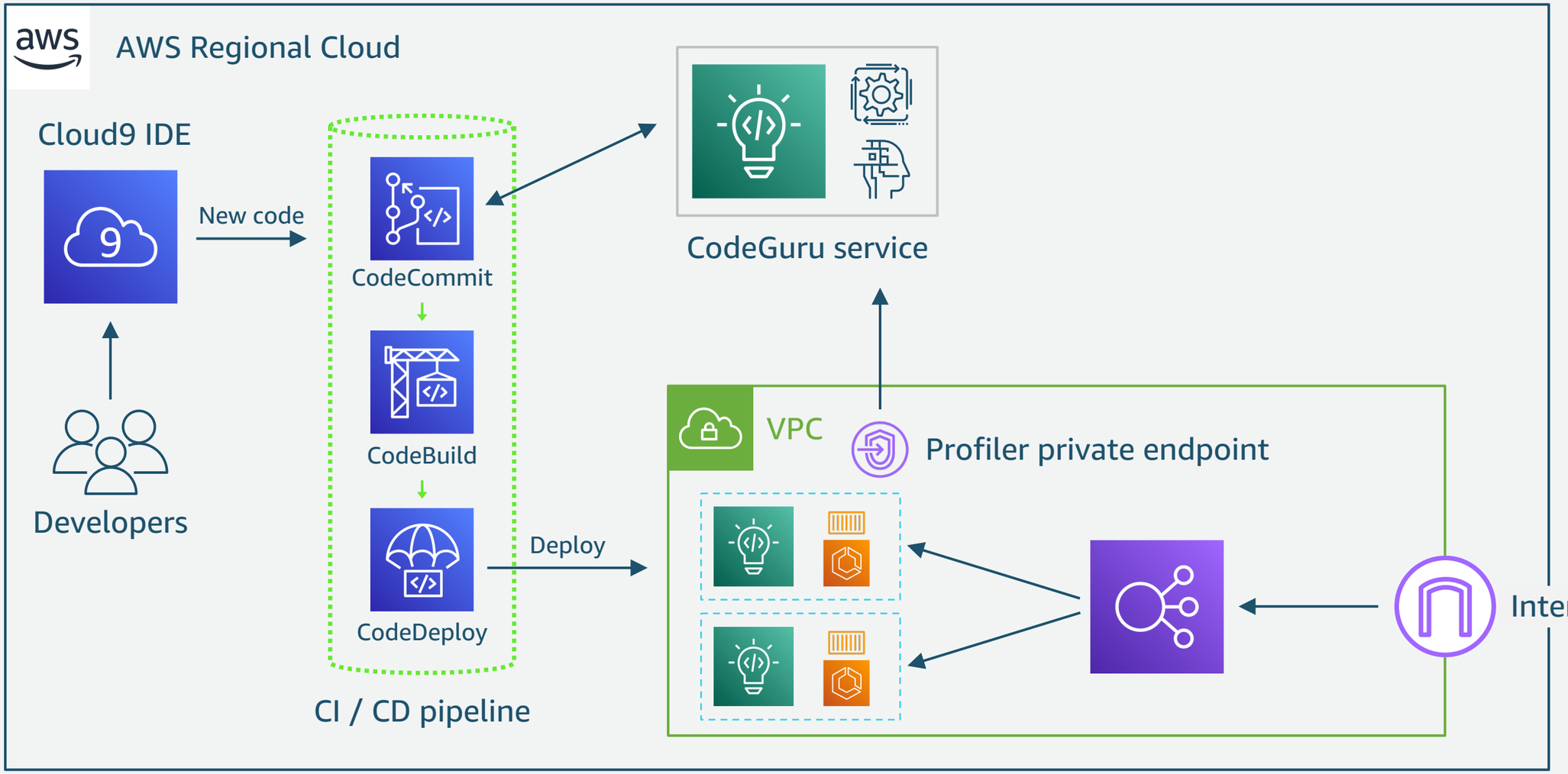
CodeGuru Profiler – Lambda

Lambda



CodeGuru Profiler – Containers (ECS/EKS)

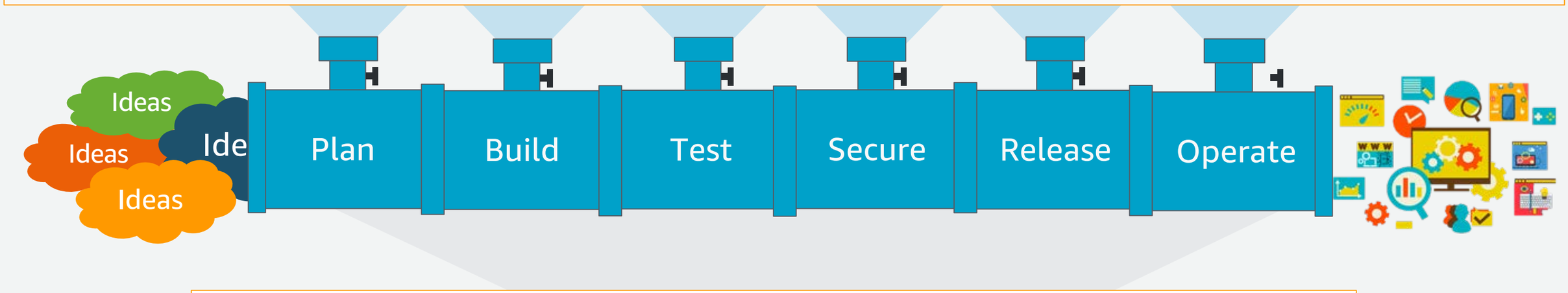
Containers



Expanding testing capabilities with AWS Marketplace

Continuous Testing Practices

Continuous integration	Dynamic App Sec Tests	Real & emulated mobile tests	Ephemeral test environments	Chaos testing
Automate unit & functional tests	Static App Sec Tests	End-to-end UX tests	Cloud load & performance tests	Feature flagging



Sample AWS Marketplace solution providers

CloudBees · Tricentis · MICRO FOCUS · Gatling · LaunchDarkly · Gremlin

circleci · GENYMOTION · GitLab · Akamai · JFrog · LoadRunner

1,600+ vendors • **10,000+** products

aws | aws marketplace

aws marketplace

10,000+ listings • **1,600+** ISVs • **24** regions • **310,000+** customers • **2M+** subscriptions

AWS Marketplace DevOps Workshop Series participating partner hands-on labs

GitLab · JFrog · circleci · harness · armory · CloudBees.

puppet · TREND MICRO · CORESTACK · HashiCorp Terraform · TRICENTIS · MICRO FOCUS

LaunchDarkly · GENYMOTION · PagerDuty · Gremlin · New Relic. · Quali

dynatrace · DATADOG · snyk · sysdig · sumo logic · *And more coming soon!*

aws | aws marketplace

Next steps

- Bookmark the AWS Marketplace DevOps Workshop Series landing page, check back for new content or subscribe to email updates
- Start a hands-on lab
- Move on to Module 6: Observability and Monitoring
- Visit the AWS Marketplace website to experiment with DevOps tooling

aws | aws marketplace