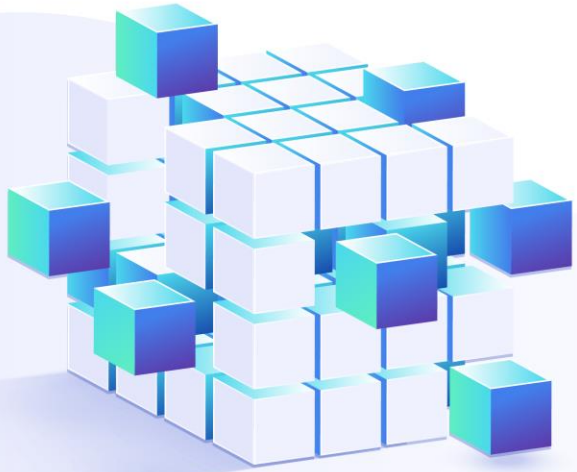
aim consulting

AN ADDISON GROUP COMPANY

Gated Test Automation

Improve the reliability of test automation and communicate the standing of product quality more clearly.

Contents



Introduction

Test automation grew out of the **recognition** and **acceptance** that manual testing is inefficient and cannot scale.

By automating testing as part of their Continuous Integration/Continuous Deployment pipeline, enterprise organizations took a useful step in the right direction.

However, **test automation is just the beginning.** Many organizations inadvertently introduce more fragility and inefficiency into their CI/CD with test automation, thus negating its benefits.

Testing and Test Automation

Let's define testing and test automation, no matter how obvious they are.

- **Testing** is the validation that software **meets its intended purpose without breaking other code** before being released to production.
- **Test automation** is the application of **tools and software to perform testing** that would **otherwise be done manually by humans**. Offloading the testing burden from humans to machines **increases efficiency**. It enables human testers to perform more complex testing that automated testing cannot efficiently solve.

Ideally, a test automation solution scales across the enterprise and **improves code quality and efficiency** while **reducing uncertainty**.

Unfortunately, organizations struggle to achieve this. **Why is this?**

Losing Improvement for Perfection

Many enterprise organizations unknowingly ignore the nature and purpose of their CI/CD pipeline because they are so focused on the goal of test automation: faster delivery with higher quality.

In this noble and correct pursuit, they **unconsciously believe that automated tests equal faster delivery and higher quality**.

This equation is only partially true.

Automated tests are necessary but will only meet the end goal when applied within the right strategy.

aim consulting
AN ADDISON GROUP COMPANY

The Wrong Strategy

Most teams have a strategy of "**test everything at once after the code is written**."

This creates an **unintentional divide** between developers and testers resulting in very impressive code and very impressive tests that don't work well together.

Failed tests **block promotion throughout the pipeline**, often demanding the team's time to troubleshoot the code and the tests.

Additionally, the team and the stakeholders **can't know the true state of the software.**

A Typical Conversation

The true purpose of software testing is to **communicate the true state of the software**.

Most teams don't understand this, even though it's ***critical in order to effectively inform the way in which they test***.

As a result, teams find themselves stuck in a process and conversations guessing at the true state of the software.

What This Tells Us

Our automated CI/CD is supposedly more efficient than manual testing. However, as we can see, the conflict between the code and the tests results in inefficient discussions within the team and **confusion about the state of the software**.

The team missed the true purpose of software testing.
It's not to test for everything "out of the gate" (*pun intended*). Again, the true purpose of software testing is to **communicate the state of the software**, internally and externally.

Proper Understanding and Alignment Lead to an Effective Framework

If the team and stakeholders understand and agree that testing is a way to effectively and efficiently communicate the state of the software, they can implement a CI/CD framework and process that meets the demand for **higher quality software delivered more quickly**.

So...

Working in Unison

Knowing and communicating the state of the software requires **tests and code to work together**, which requires Testers and Developers to work together.

This is where Gated Test Automation comes in.

In our previous example, the talented team with an initial strategy of **one** implicit gate in their process required perfection (test for everything "out of the gate").

What is Gated Test Automation?

Gates in a CI/CD pipeline define which tests must pass in which environment, forcing testers and developers to collaborate and communicate at the start of a Sprint through the end.

Code a little, test a little, code a little more, test a little more, and so on.

Consider Everything, but Address Iteratively

Coding and testing little-by-little requires addressing all factors in the CI/CD upfront.

With *well-defined gates*, which we'll get to shortly, teams are forced to think about often overlooked variables like runtime, installation, and anything else that must work in order to promote code through the environments to production.

The team that was previously blocked from figuring out if a failed test was an issue with the code or the test itself is now aligned on expectations for code and tests *before any fingers hit the keyboard.*

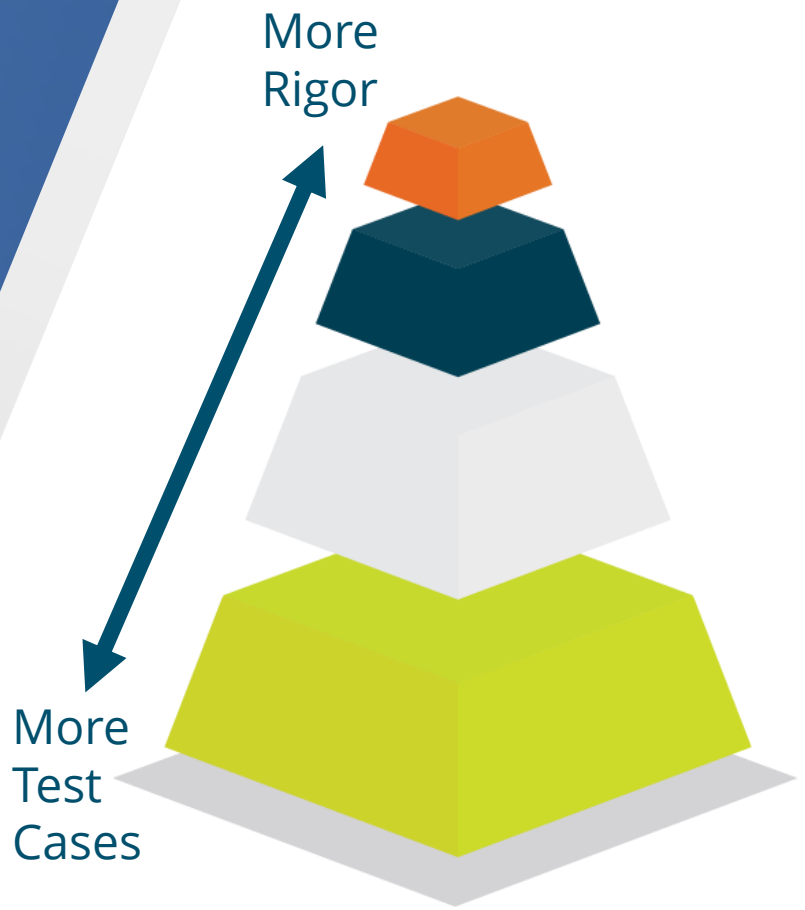
So, what does Gated Test Automation look like?

Now that we understand that test automation's purpose is to communicate the state of the software and that Gated Test Automation is the most efficient and effective way to do that, let's drill into the technical details.

The pyramid at the right demonstrates the process of Gated Test Automation, a far cry from ill-defined automation that tests for everything at once.

Differences Between Automated Tests

Test Volume and Rigor



HIGH LEVEL TESTING
Scenario, Performance, Boundary, Exploratory

ACCEPTANCE TESTING
Exercises the full requirements
Executes in full environment

SMOKE TESTING
Essential functions
First tests to execute at runtime

UNIT TESTING
Composed of several tests
Executes at compile-time

aim consulting
AN ADDISON GROUP COMPANY

Right Time for the Right Conversation

As you'll see in greater detail, Gated Test Automation enables the **right conversations and problem-solving** at the **right time**.

If we return to the conflict between the developer's elegant code and the tester's elegant tests, we can now see that they are set up for the right conversation at the right time.

There's no more 'feeling around in the dark' about the problem – shooting for perfection right away, without defined expectations.

Going Through Your CI/CD with Gated Test Automation

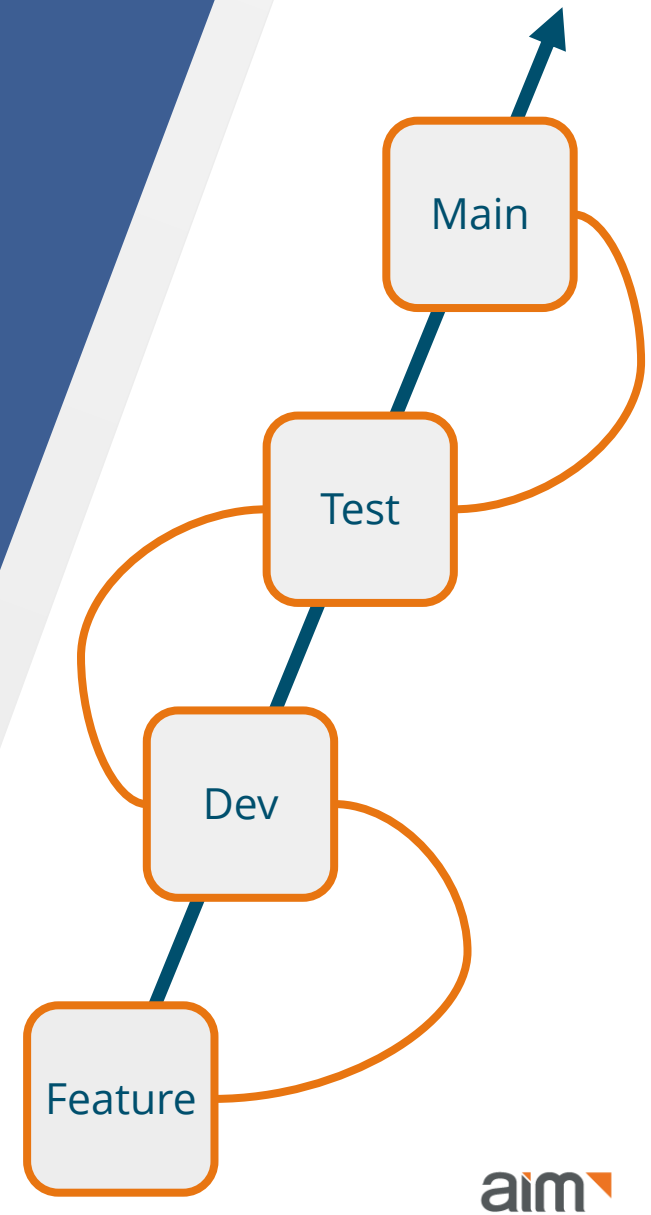
Now, let's look at the activities and gates from Source Control to Production.

Each branch of code:

- the **feature branch**,
- the **dev branch**
- and the **test branch**

have different levels of rigor expected and are subjected to different test automation suites.

Developers and testers will commit code changes to the feature branch and merge the code to upper branches when it is ready.



Enterprise Environments

Modern software projects use multiple environments, each with a different purpose to test a certain aspects of the deployed application.

Test Environment
Rigorous testing: Code will work in all specified cases on commit

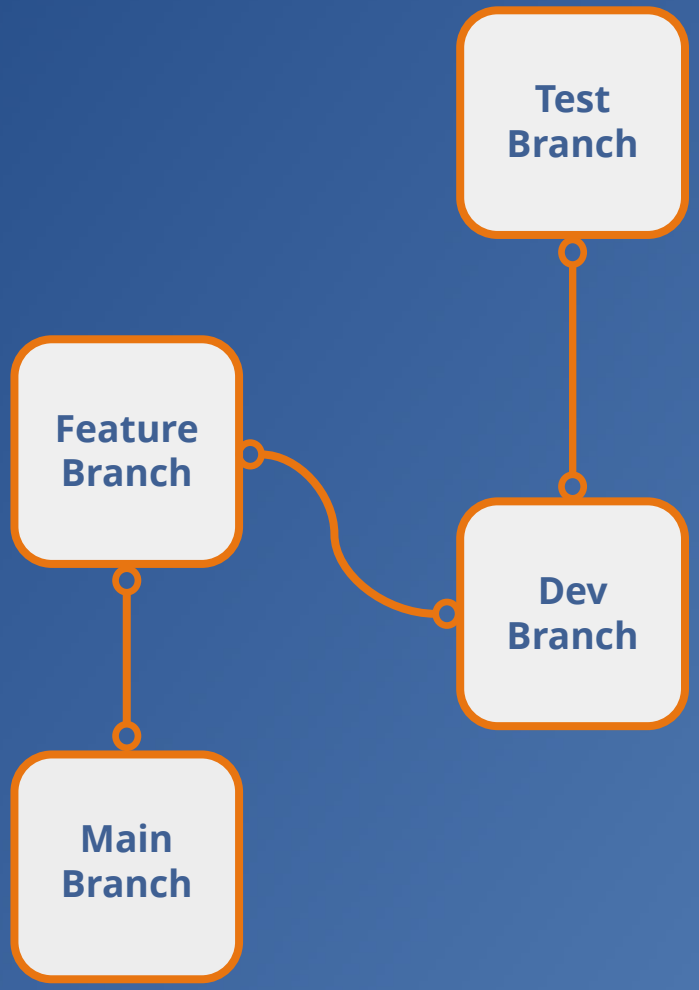
Dev Environment
Proving ground: Code will work in simple cases on commit

Branch Environment
Freeform sandbox: No code is really required to work on commit

Code Branches

The code running in each environment can be managed by keeping a branch for each environment.

- Core code development
- Branch is named by its Scrum board task ID



- PR from Dev to test shows that it has passed initial Quality Engineering gate
- PR approval from tester is required to enter

- First pull request (PR) is from feature to Dev code
- Cannot be committed directly to Dev
- PR approval from developers required to enter

Test Types

Then, it's possible to run various types of tests in new environments which each serve a different purpose

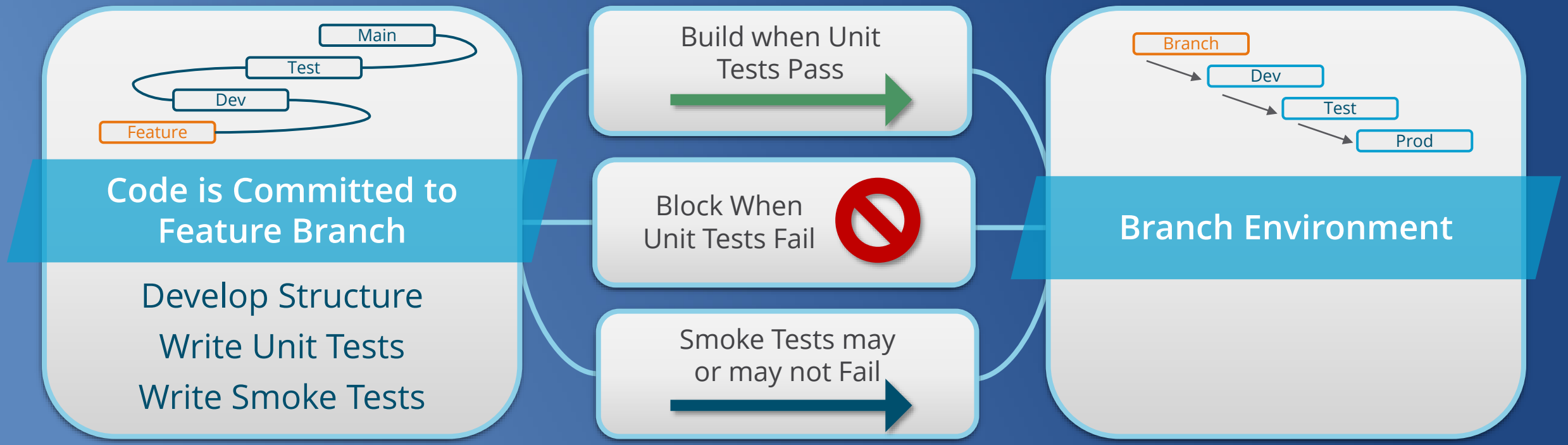
- **Branch environment** is the least rigorously managed
- Developers only need to have passed **unit tests** to enter

- **Smoke tests** are the agreed minimum function
- A single API call with non-failing result may be a **smoke test**

- **Acceptance tests** express conformance to specifications and requirements

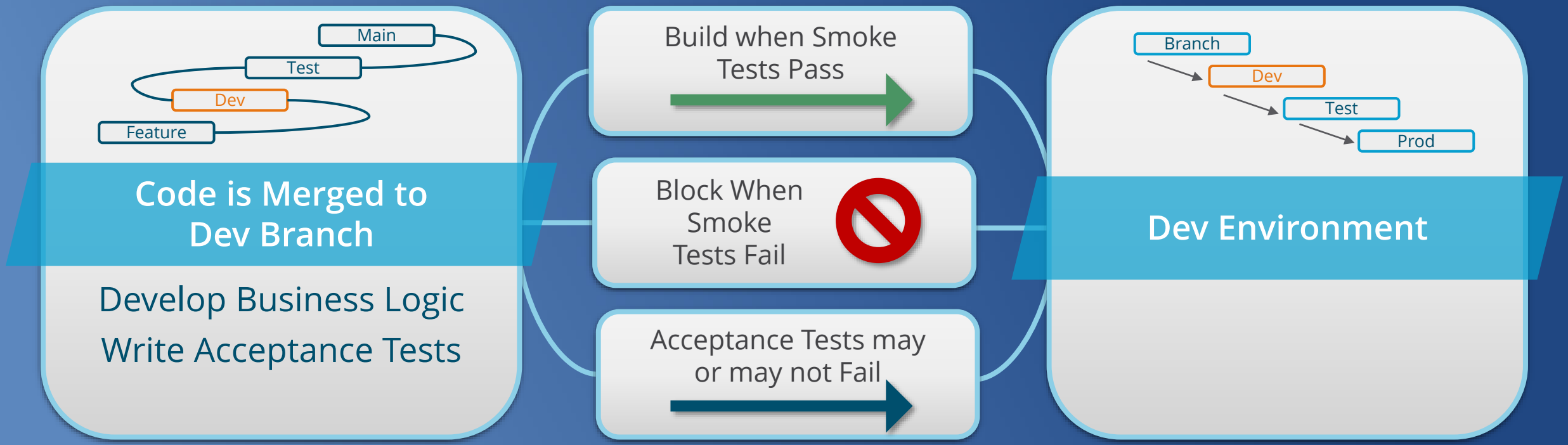
Let's see this in action...

Branch Phase: Build-to-Branch Environment



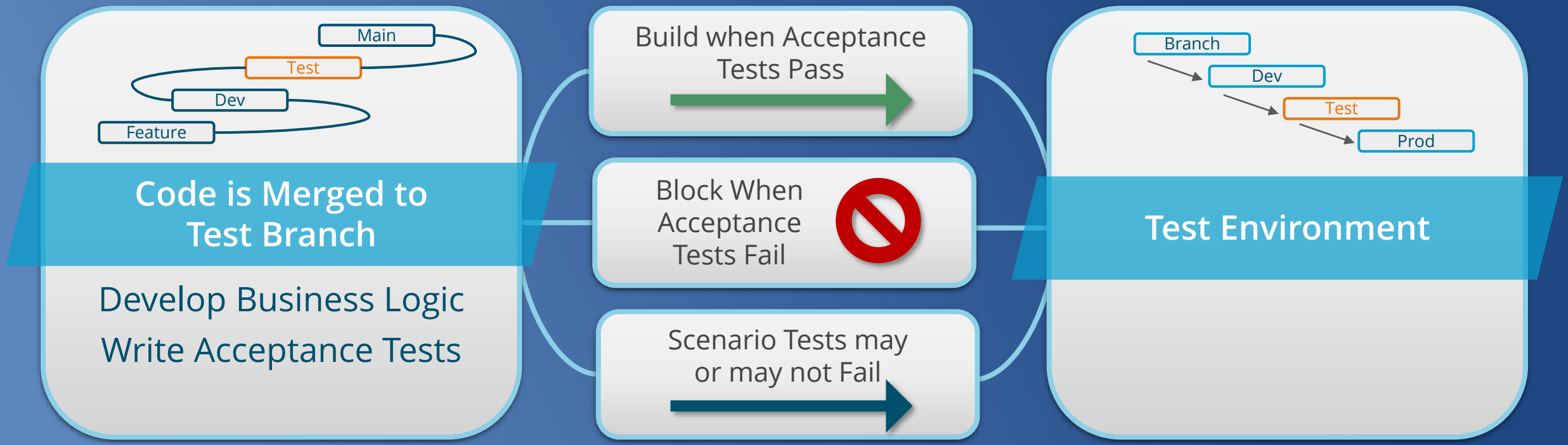
Unit tests are the requirement for the first check-ins and deployments.
Smoke tests will be developed but don't have to pass at this stage

Smoke Phase: Build-to-Dev Environment



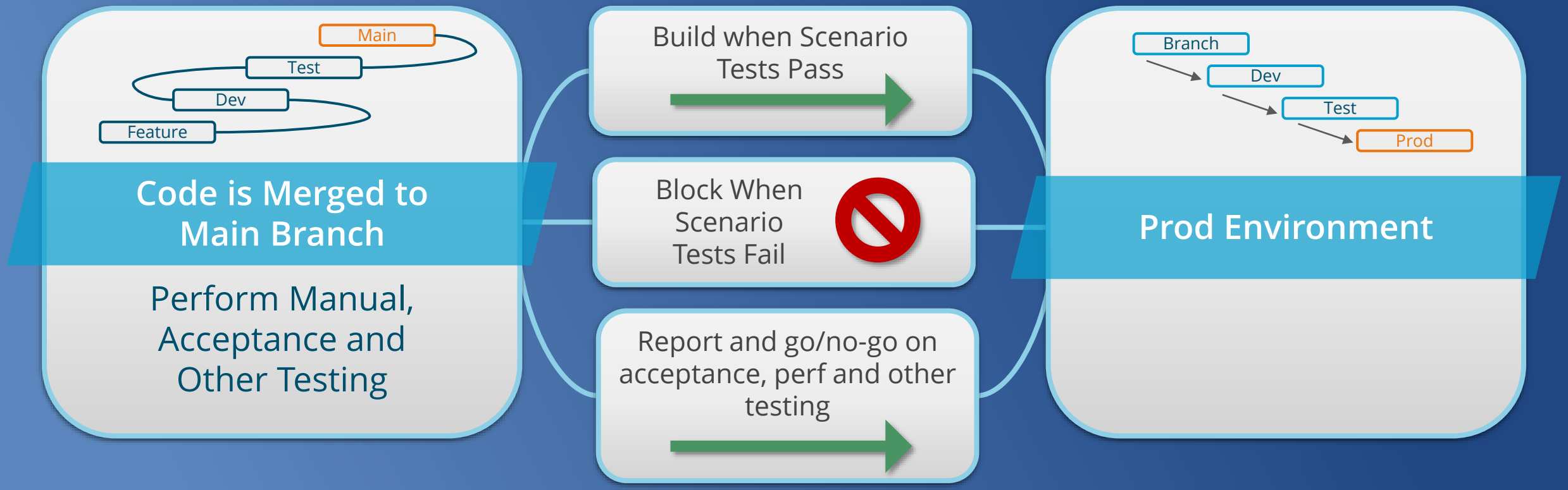
Smoke tests are the requirement for the Dev environment.
Acceptance tests are developed but are not required to pass.

Acceptance Phase: Build-to-Dev Environment



Smoke tests are the requirement for the Dev environment.
Acceptance tests are developed but are not required to pass.

Release Phase: Build-to-Prod Environment



The final decision to move into production takes all the data from automated testing, manual testing, user testing, and other activity and the team decides together when the software is ready for users.

aim consulting
AN ADDISON GROUP COMPANY

More On Communication

Often there is an expectation that all tests have to pass in all environments. This prevents us from having a full understanding of the software as it is being developed.

Teams who implement test automation without gates frequently rely on automation reports and dashboards which require time and effort. SDETs have to build and maintain reporting to communicate the state of the software.

Gated Test Automation reduces that need by providing a shorthand and informed agreed-upon assumptions for the team and stakeholders to quickly know the state of the software.

Immediately, the PM knows that Smoke Tests have passed, and the next gate is the passing of Acceptance Tests.

In Conclusion

Throughout the entire Sprint, **Gated Test Automation ensures every team member and stakeholder understands the state of the software** and that it's being delivered with better quality and efficiency.

About the Author

Patrick Taylor

National Technology Evangelist

Patrick Taylor is a National Technology Evangelist at AIM and holds over twenty years of experience in technology leadership.

He helps our clients solve their hardest technology problems by providing reference architectures, high-quality tools, and industry-leading techniques to software development. On a day to day, Patrick builds teams, designs cloud architectures, and drives velocity and quality.

His background spans all levels across software organizations, from individual contributor, to software director, to architect.

aim consulting

AN ADDISON GROUP COMPANY

About AIM Consulting

AIM Consulting, an Addison Group company, is an award-winning industry leader in technology consulting and solutions delivery. AIM's differentiation is our collaborative engagement model that provides cross-functional results. We work with clients, shoulder to shoulder, for one goal – their success. Founded in 2006, with offices in Seattle, Minneapolis, Denver, Houston, and Chicago, we are ranked among the fastest growing private companies and best companies to work for due to a long track record of success with our partners and consultants. Our long-term relationships with the best technology consulting talent allows us to deliver on expectations, execute on roadmaps, and drive modern technology initiatives.

Seattle Business Washington's 100 Best Companies to Work For

2017 Best Places to Work Minneapolis/St. Paul Business Journal

Inc. 5000

Minnesota Business 100 Best Companies to Work For

aimconsulting.com