The need for Continuous Testing in the DevOps era

```
const { Bodies, Body,    npos
Composites, Constraint, Engine, Mouse,
MouseConstraint, Render, Runner,
World } = Matterconst runner =
Runner.create()
const engine = Engine.create()
const world = engine.world
const render = Render.create({ eleme
docu    querySelector('.can-
vas'),engine,o   tions: { background:
'transparent', w   th: window.inner-
Width, height: wi   dow.innerHeight,
wireframes: false,    }) // create item
const createItem     ({ x: stringX, y:
stringY, length: s   ingLength, texture =
'']) => {
        oup = Body.nextGroup(true)
  const string = Compos-
tes.stack(stringX, stringY, 18, 1, 2, 2, (x,
y) =>
  Bodies.rectangle(x, y, stringLength / 2,
2, {
collisionFilter: { group },render: {fillStyle:
window.COLOR    kGrey,strokeStyle:
```

QMETRY

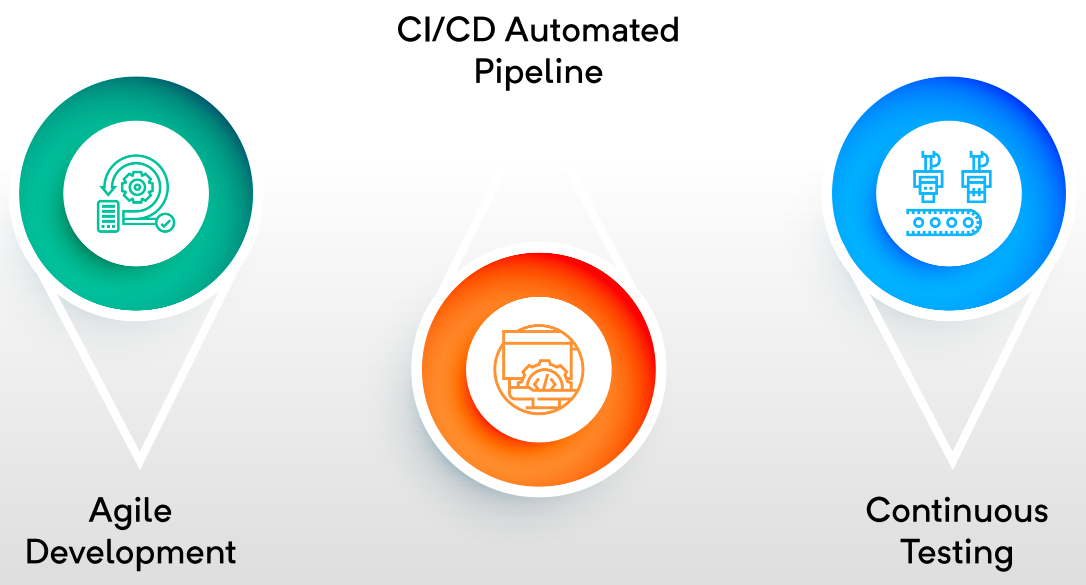
(408) 727-1101 info@qmetry.com www.qmetry.com

The need for Continuous Testing

No matter the industry, all organizations are battling fierce competition. And only way to stand out in this competitive world, is by providing impeccable customer experience. It becomes imperative for an organization then, be it service focused or product centric, to provide seamless & flawless user experience. With the advances in technology, customers are spoilt for choice. Customers expect speed and lack patience to wait for releases that take months or a year. It is a thing of the past when business needed to wait for 6 to 24 months to provide customers with the upgrades or bug fixes. Customers now expect and seek out the new, new features and frequent updates with continuous quality experience.

This ideology of “High Quality with Speed” has made organizations realize that there needs to be a radical change in software ideation, development, deployment and releasing processes. That is, all the pieces in this process need to move quicker, be available faster without compromising the ideal of the perfect high-quality product or service.

In order for an organization to achieve this, DevOps practice breaks this wall between development, testing and operations. The DevOps approach enables collaboration in agile teams and encourages them to come of out of these silos. Testing becomes a shared responsibility and covers all stages of development. Testing is no longer a task that starts and ends at a specific point. Under the DevOps practice, testing starts ideally right at the requirement stage and is normally initiated at the development stage.



Agile Methodology for customer centric software development.
Early customer feedbacks enable faster changes. This ensures that the software is release ready sooner than late.

Set of Continuous Activities - Continuous Integration, Continuous Delivery & Continuous Deployment.
“Continuous Everything” allows automation and streamlining of process that increases the efficiency providing greater time and cost benefits in software delivery.

Continuous Testing for high quality with speed
And as these organizations increasingly adopt DevOps & continuous delivery, there is a need for end-to-end continuous testing.

Continuous Testing

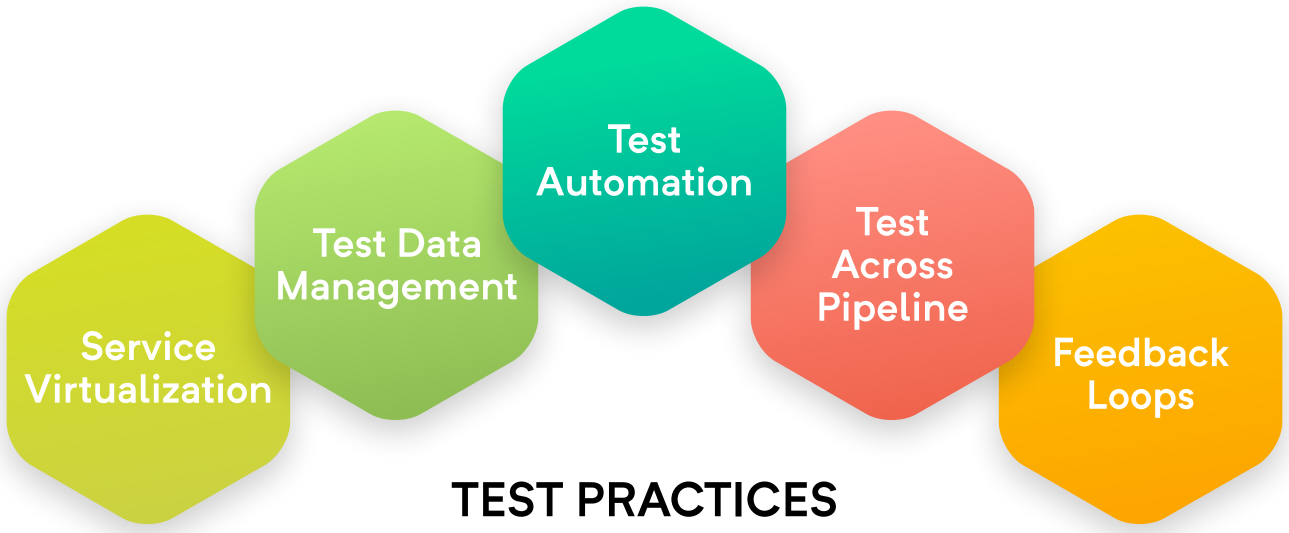
Continuous Testing helps in achieving more comprehensive and detail-oriented testing carried out through the development and release lifecycle. This allows all the team members to take the ownership of quality right from the beginning of the software development.

Continuous Testing addresses the automated testing for each task in the delivery pipeline. It ensures that you have a quality product at all the times. It does so by building a continuous feedback loop that helps the developers to identify issues quickly and fix them in the early stages of implementation. This reduces the business risk of product failure.

In practice, continuous testing is the process of testing early, testing often, testing comprehensively and using automation to achieve the release goals.

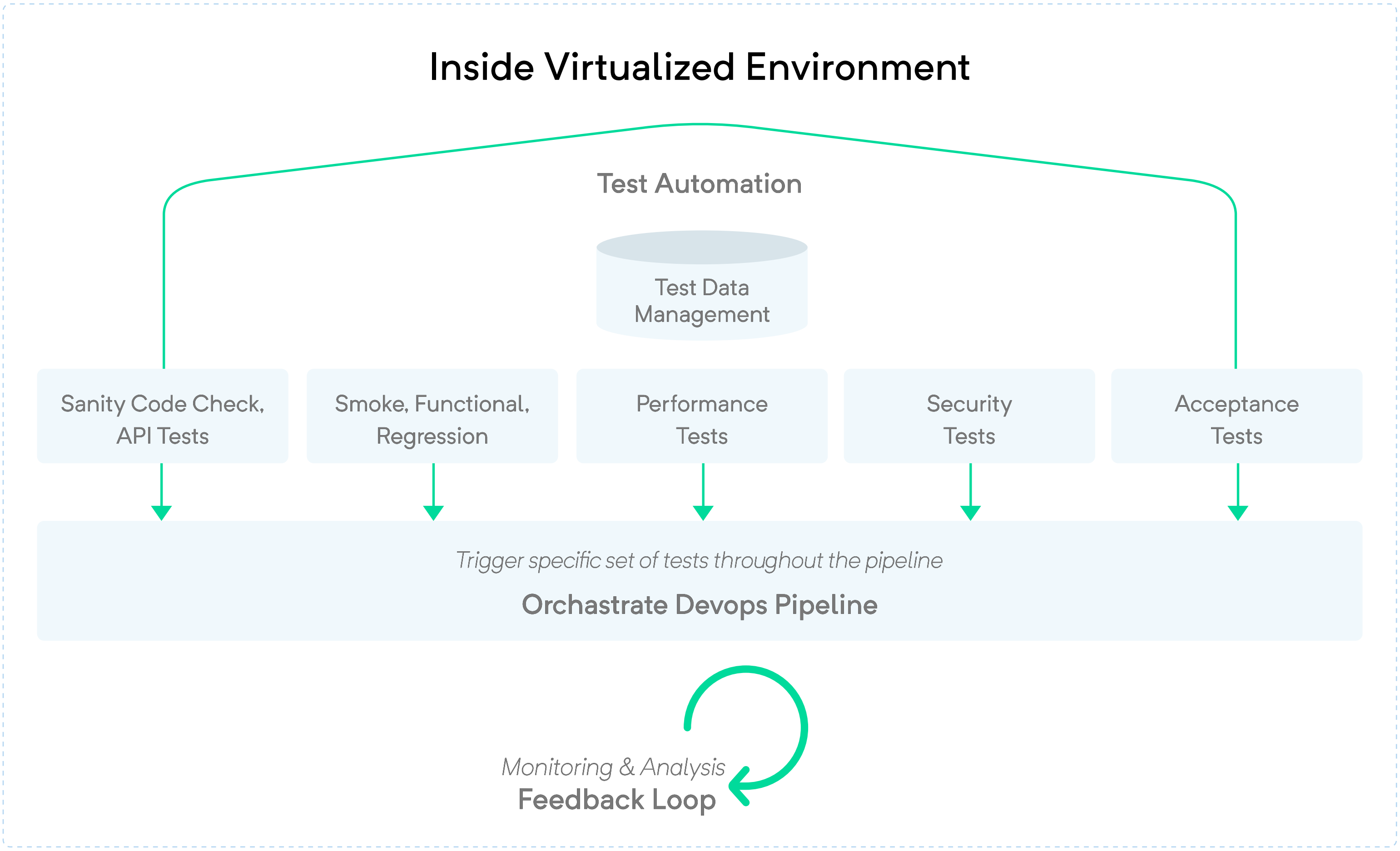
How can an organization adopt Continuous Testing?

To implement continuous testing in your organization, you not only need to move testing upstream, but you also need to adopt the right set of test practices.



- The main aim of continuous testing is to achieve **high quality with great speed**.
- To do this, test **more frequently along with adequate test coverage**.
- But this can only be achieved when **testing starts early** in the development lifecycle.
- However, the readiness of testing environment at such an early stage is a huge challenge. So, **service virtualization for the test environment** is necessary.
- At the same time, testing frequently can be achieved through extensive **Test Automation and Test Data Management**.
- Integrate **regression testing right from the beginning** to maintain the balance between rapid development and frequent testing.
- **Testing maturity and depth must be scaled up as you grow** through the CI/CD pipeline to ensure product readiness.
- **Continuous Feedback loops** are required for faster bug fixing and analytics must be leveraged to mitigate business risks. Test Management Tools such as QMetry helps agile teams to leverage analytics and get real-time feedback through various reports and visual analytics.

The following figure describes what a continuous testing environment looks like when all the test practices come together —


Inside Virtualized Environment
Test Automation
Test Data Management
Sanity Code Check, API Tests
Smoke, Functional, Regression
Performance Tests
Security Tests
Acceptance Tests
Trigger specific set of tests throughout the pipeline
Orchastrate Devops Pipeline
Monitoring & Analysis
Feedback Loop

Service Virtualization

Since Continuous Testing requires testing more frequently, you need to hit multiple environments with a greater frequency. This indicates a potential bottleneck if the environments are not available when you need them. Teams spend about 30% to 50%, and often more of their time just waiting for test environments with the latest build to become available for testing. This is inefficient and you cannot realize the full benefits of the DevOps practice.

By virtualizing the environments that help you test the code frequently, you can achieve testing continuity without worrying about the impact on areas that haven't been changed. Another benefit of test service virtualization is that it helps QA teams to test the components without waiting for all the pieces to be ready and available. This is done by simulating the missing dependencies to eliminate the roadblocks in the path of continuous testing. Virtualizing those environments allows you to test your code without having to worry with areas you are not changing (i.e. other systems and environments). You can access the environments and eliminate this constraint from your dev cycle by ensuring 24/7 availability.

Test Data Management

It is equally important to have access to use cases based realistic test data for the effectiveness of a continuous testing strategy. With the help of quality test data and sound test data management you can increase coverage and drive results with further accuracy.

For this you need to:

- Create test data synthetically for all the scenarios that are necessary for testing.
- Use service virtualization to capture requests and response traffic and reuse the data for consequent scenarios.

The ability to reuse this data and share it across multiple teams, projects, versions and releases is crucial to increase the speed of test construction, management and maintenance.

Count for Testing while Orchestrating CI/CD Pipeline

The CI/CD pipeline is the backbone of the modern DevOps practice. It is what helps you bridge the gap between development and operation teams by automating the build, the testing, and deployment of apps. It needs to integrate with your automation suite.

A DevOps practice cannot achieve its continuous testing ideals without the stability and speed of an automated CI/CD pipeline. Here's how you can set up an effective pipeline:

- Make testing the central and integral activity throughout the development cycle rather than a hygiene activity that occurs post-development.
- Build and automate tests concurrently and prepare to execute immediately after new functions or features are built.
- Work together with the team to analyse and determine the tests that should be run at various stages in the delivery pipeline.
- Configure test suites to run faster and efficiently to avoid bottlenecks at different stages in the software delivery pipeline.
- One key factor to prevent constant changes from raising false positives is the environment stabilization. This will help in smooth and confident deployment of the product across different platforms. You can use virtualization for the stabilization.

Test Automation

Test Automation is required throughout the CI/CD pipeline for continuous testing to succeed. Since Continuous Testing requires testing early and often, we need automated testing to get the desired cadence of feedback on the Application Under Test (AUT). These are some of the best practices to implement test automation:

- Automation of regression tests, non-functional performance and security tests where possible.
- Implementing the “Shift Left” principle by scaling up automated unit tests, APIs and Integration tests, and only a couple of automated tests through the UI.
- Leveraging a Continuous Integration (CI) server to run automated tests.
- Building smoke regression packs that run fast and as often as the application is updated.
- Automation of new functionality and stories in parallel to development rather than later.

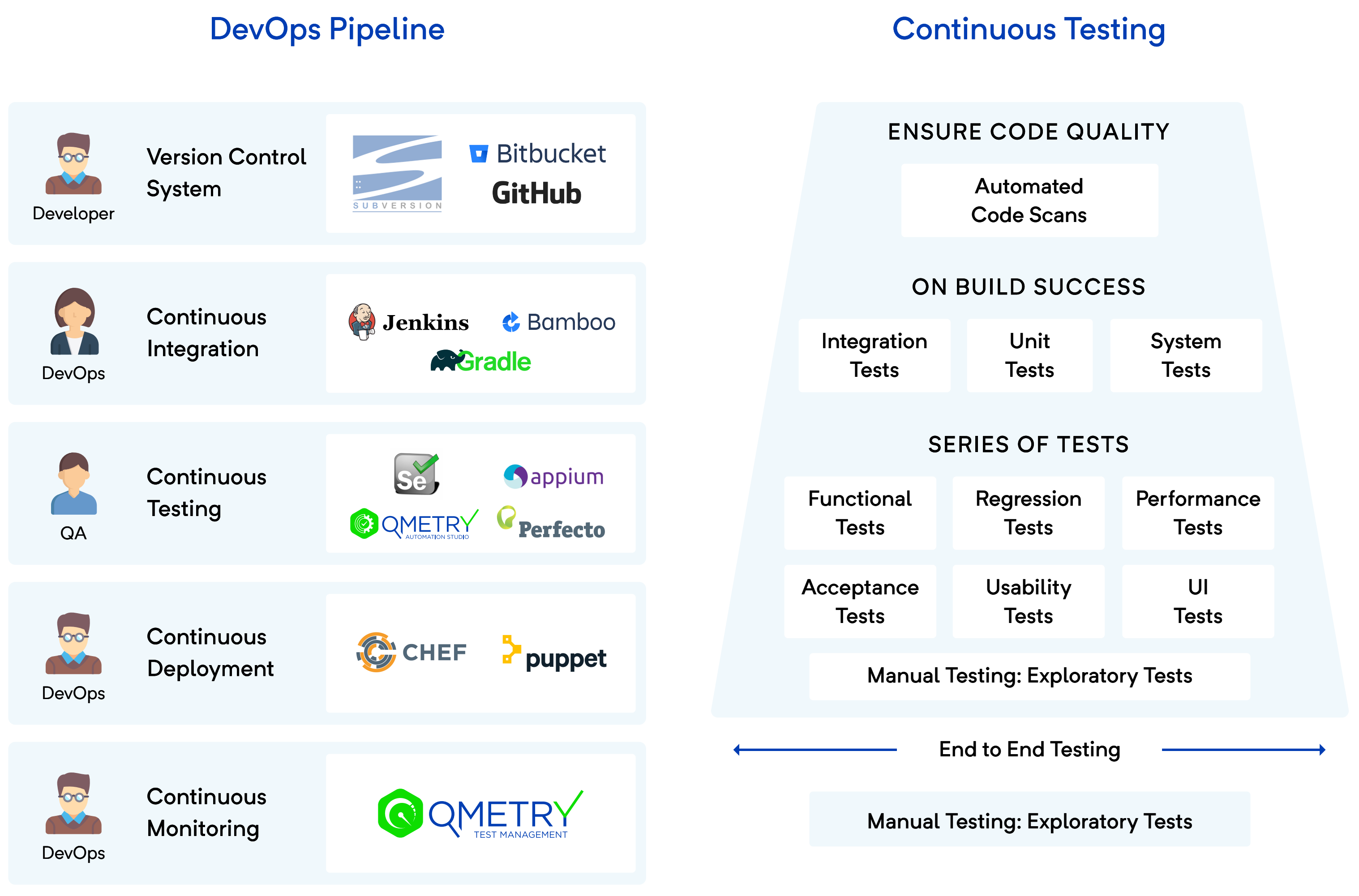
Feedback Loop

Businesses can achieve the full potential of continuous delivery and continuous testing by having a large suite of test cases that execute continuously or as a function of code check-ins. This will not only automate the go/no-go decisions but also helps you to determine and understand the current risk-levels associated with any release or application.

Using a powerful results or analytics dashboard, the business can visualize the heat maps and problem areas and re-prioritize resources when needed. Continuous feedback enables C-suites to have instant access to the results which can then be useful in understanding whether their expectations are met or not. Another advantage of continuous feedback is that it makes developers aware of the bugs/defects frequently. Hence these issues are fixed quickly when they are likely to be less costly and less progressive in the pipeline.

This type of agility is what embodies the DevOps strategy and allows them to maximise their potential. Automated testing is the key factor to get quick feedback on the application status.

What Continuous Testing looks like in DevOps Process?



We have already established the importance of testing, testing each line of code and testing often at various stages. This is impossible to achieve manually if you execute all of these tests every time a line of code is updated. This is where continuous testing comes in to play.

- Automatic Code Scan
- Testing on Successful Build
- Increasing Testing Maturity & Complexity
- Acceptance Tests at Continuous Deployment Stag
- Monitoring: Requirements Tracability & Advanced Analysise

Automatic Code Scans

The timely release of high-quality software depends on the ability to continuously integrate, test and ensure the required specifications are met. The purpose of continuous testing is to execute various types of tests continuously on the code base in different environments as defined in the CD pipeline. The earlier you implement Continuous Testing , the more value it provides to your business. Implementing your test strategy at the outset will help you identify a large number of process defects as and when they are introduced while coding. Continuous Testing helps you with the insight into fixing these bugs before they are part of the integrated system.

Whenever the code is changed, automated test scans run to ensure that your code doesn't break with any change or new introductions. Feedback is recorded frequently so that developers have the timely insight to fix bugs before the buggy code progresses into the development and delivery pipeline. And it is because the early stages of application development that have the worst defects, you can take advantage of Continuous Testing by introducing it as early as possible.

For example, as soon as developers check in the code in the source code repository such as Bitbucket, GitHub, SVN, etc., automated testing is triggered with static analysis and unit testing. Using the results from these code scans, you can pass/fail the builds based on the threshold you have set. So, if you have set a threshold wherein 85% code coverage is achieved and no more than five level 3 or higher defects are found, then the build is a success.

Conversely, if the coverage drops below 85% with high-risk defects then the integration is rejected, and dev team is notified. The advantage here is that development teams can work independently with the help of any dev or test tools they prefer. While businesses can set up policy gates to ensure that teams continue to operate within the risk parameters set by the key stakeholders.

Testing on Successful Build

Continuous Integration (CI) is a development practice that integrates tests into source code changes as part of the CI/CD process. CI requires an automated build and unit testing capability. What this means is that, if the developer breaks the build or if the new code fails the series of automated tests, that feedback is provided instantly. This allows the defects to be remedied before the broken code impacts other developers or test teams. A successful build and unit test will be delivered immediately to the deployment pipeline so that the repair or error can be fixed before the broken code affects other developers or the test teams. This process will be repeated every time a code change is committed by a developer. And with each commit, it triggers the automated process to test the software so that towards the end, the organization has a build that is deployment ready. The aim of this is to reduce the defects in the code released for deployment. It is now the developer's responsibility to fix the errors and the build. Continuous Integration reduces the amount of time taken to resolve the issues by identifying them as soon as code changes are delivered. It is evident that Continuous Testing is integral part of CI process. Every time code changes occur & are checked into the shared repository; the automated test suite is run. These suites are developed using automation testing tools such as Selenium, Appium, QMetry Automation Studio & UFT. The entire process can be achieved by using CI tools like Jenkins.

Continuous Testing best practices for Testing

For the deployment pipeline to be effective, there are many best practices followed. For instance, unit tests occur on the CI server that tests each unit of the system separately. Integration tests are implemented in integration environments where they test the various components integrated together. System tests take place in the testing environment where the larger system with various components and interfaces are tested through system-level scenarios. The depth of testing increases as the simulation of environment becomes more like production.

The benefits

- Finding low level errors early in the cycle
- Integrated code base to ensure the build doesn't break
- Shorter feedback loops to notify developers when builds break

Increasing Testing Maturity & Complexity

Once unit testing is completed on the new build, the deployment pipeline automatically deploys the code to the next environment. Here it will take some level of integration testing and once this deployment pipeline is initiated, automated testing becomes increasingly important.In the extended testing process, developers would initiate the automated test suite to run after the completion of a successful build or on a schedule at the end of the day as part of the nightly build process. This automated testing suite also checks out the integration. hat is various components that developers have worked on, work as a full application.

The automated test suite comprises thousands of test cases covered for the entire code base. Ideally, the code base should achieve 100% coverage. Once automated testing is done successfully in the next environment, it may progress to additional environments. This process can be repeated with automated application deployments and different types of tests against each environment. Ultimately, the objective is to have an automated, repeatable process to eliminate human errors, effort and time taken for these tasks. Continuous testing progressively becomes more challenging and complex as it progresses towards the production environment. Similarly, the tests grow in number and complexity as the code matures and the environment becomes more complex.

Further, the tests need to be updated each time different phases and automated scripts are updated, with the code becoming more mature and progressing to more complex and advanced environments, where the configurations and infrastructure advance until it reaches production. So, the time taken to run the tests increases as the testing progresses towards release. For example, unit tests might take lesser time to run but some integration or system tests or load tests may take much longer to run — ranging from hours to days.

The entire process is automated with continuous testing carried out at various stages with the help of policy gates and manual intervention, before the code is pushed to production. These quality gates at each stage of testing increase the confidence in the code and overall build.

Acceptance Tests at Continuous Deployment Stage

Continuous testing pipeline includes unit testing with preliminary automated security verifications. It then progresses to an integration level of testing and then on to system level testing and performance test scenarios. Finally, it progresses to Acceptance Testing that includes the automated site acceptance test cases and then on to the User Acceptance Testing that could be manual execution and include end-users to carry out the tests. Deployment is simply a technical decision to release a feature within the application, while the release is a result of a business decision to update a version, release new features into production and make it available to the end user. This is the final sign-off for the product or feature where the manual gates are invoked and finally deployed on the production site.

The unit-tested code produced by development teams will subsequently be deployed — hopefully using automated processes — into an integration test environment, where it will be tested using any number of automated testing tools and then deployed and validated in the UAT environment before final release into production.

Unit tests are typically automated as part of the Continuous Integration, but organizations still struggle to automate integration and acceptance testing. You can't automate manual testing. At this stage, manual testing in form of exploratory testing should be carried out to save time and efforts that otherwise would have been spent on manual activities.

As continuous testing progresses, the complexity of test and test environment increases and gets to a simulated environment that is closest to production. This is how a typical Continuous Testing pipeline can be designed by team on the basis of the product and various levels and types of testing that is required.

Monitoring: Requirements Tracability & Advanced Analysis

Reporting and analytics are needed to ensure that businesses can track the quality of their requirements and check that there are effective tests aligned to each requirement. The dashboards also help the stakeholders in making sure that all the crucial requirements are met, and rework is not required. Traceability also helps in assessing which requirements are at risk, working as expected or require further validation.

Additionally, you can meet compliance mandates and gain real-time visibility into the quality, and avoid any surprises late in the cycle.

Areas such as static code analysis, change impact analysis and scope assessment/prioritization benefit immensely from test automation. This helps in preventing defects and accomplishing more within each iteration. Real time monitoring of the test data remarkably improve the effectiveness of a continuous testing strategy.

With real time availability of reports, developers get the insights of the bugs immediately and thus resolving it faster without delaying further. At the same time, testers can check their test coverage against the requirements with the help of reports such as requirement coverage and traceability matrix.

Continuous Testing: Best Practices

Consumer experience is the new touchstone for the fast-evolving digital ecosystem. Users expect a seamless and smooth journey when interacting with you at various digital touchpoints. To match their evolving expectations, organizations are continually transforming themselves and improving both the quality and the pace of their development. This approach is highlighted by a shift to Agile Development and DevOps methods. Dev teams are increasingly playing a more strategic role and they require the tools and technologies to quickly respond to customer needs and feedback. The following best practices of testing are necessary for Continuous Testing to be optimal.

Focus on Mobile & Web Apps

Enterprises that are targeting digital excellence need to find the right balance between mobile and web apps. Don't assume that web apps should take precedence over mobile apps. Observe your business model and customer buying patterns closely so that you can dedicate your dev resources and investment smartly between the two.

Continuous Testing needs shared Ownership of Quality

Continuous Testing like most DevOps and Agile constructs lays more emphasis on people before processes. You need to develop the right mindset before you tackle the strategy. And for continuous testing to be effective, you need a cultural shift towards quality and efficiency across teams. The commitment to teamwork encourages collaboration between dev and test teams. Developers and test engineers need to work together and think of quality as a shared responsibility.

This needs changes across the organization, processes and tools. For instance, for developers and test engineers to work closely, require approaches like test-driven development (TDD) or behaviour-driven development (BDD). It also requires tools that promote collaboration and scalable testing.

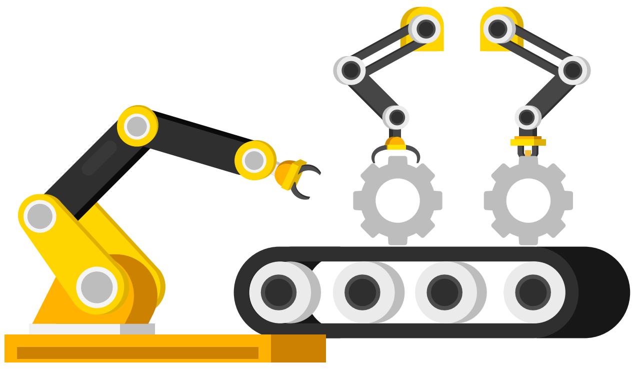
Shifting Left

Shifting left is one of the most important tactical steps for a continuous testing strategy. And shift-left requires both functional and non-functional tests as early as possible in the lifecycle. The sooner the feedback is passed on to the developer, the faster the dev teams can fix the issues and, the more efficient this process will be. Shifting left reduces the chances of avoidable risks related to delayed releases that happen when defects are detected late in the cycle. Or worse still, risk providing a poor customer experience because some bugs were not detected at all.

It's in the Cloud

The proliferation of mobile devices and growing list of browsers, operating systems and platforms means that testing cycles and requirements have grown more complex. If you want to keep the test infrastructure up to date, then it is a challenging task even for the largest teams with the best tools. Bearing in mind that you also need to scale up your environment to run parallel tests.

Allowing cloud-based testing platforms to do the heavy-lifting and the time consuming/repetitive tasks involved in scaling and maintaining the test infrastructure allows developers and test engineers to enhance the overall quality of your release.



Drive Automation

The increasing adoption of automation to drive efficiency in many areas of business includes software testing. Manual testing is still important and serves a definite purpose, but it is no longer enough to meet the needs of modern development practices. Test Automation can run tests in parallel to give the desired scalability and agility needed for modern Agile testing.

Using test automation team can reduce sprint times from a matter of weeks to minutes, accelerating the development and release process remarkably and also improving the quality. It also enables manual QA resources to focus on other core quality tasks like usability and exploratory tests.

Leverage Analytics

If you can't measure it, you can't improve it. Using analytics in your continuous testing practice helps you to view test results filtered by various parameters. You gain visibility into pass/fail and error trends and can observe how tests behave over time. This enables developer teams to identify and take actions easily.

Analytics is an important element to understand how tests are performing and to quickly find the root cause of the quality issues and bottlenecks.

Conclusion

It is technology that enables the continuous testing process, making it repeatable, reliable, and traceable. Perhaps you already have an optimized test management and automation solution, but what additional capabilities can you benefit from?

The ultimate strategic goal for any company is to reduce the business risks associated with releasing applications and reduce the time to market. Continuous Testing's primary role is in overcoming these risks to business — such that the new release of a reasonable quality and doesn't frustrate or alienate Continuous Testing answers the most important business question — Will our customers continue to be happy with the product when the new release comes out? Is our product release ready?

How can you enable Continuous Testing with the help of QMetry?

- Test early, often and comprehensively to empower your Agile QA with Continuous Testing
- Achieve Continuous testing by integrating with the CI/CD pipeline along with capturing and monitoring the test executions
- Continuously monitor with insightful feedback loops with 150+ reports

Find out more on how you can achieve Continuous Testing with QMetry Test Management

Know More

Get Continuous Testing Consultation to understand how you can empower your Agile QA teams with Continuous Testing

Continuous Testing Consultation

QMETRY

(408) 727-1101 info@qmetry.com www.qmetry.com