Embedding Performance Engineering into the CI/CD Pipeline

Presented by:
Anjeneya Dubey
McGraw-Hill Education

Brought to you by:

350 Corporate Way, Suite 400, Orange Park, FL 32073
888-268-8770 - 904-278-0524 - info@techwell.com - http://www.stareast.techwell.com/
Anjeneya Dubey is the director of performance engineering for McGraw-Hill Education, a learning science company that delivers personalized learning experiences. His responsibilities include ensuring that every product built is high performing, highly scalable, highly available, highly reliable, and fault tolerant. In his past five years with McGraw-Hill, Anjeneya has built automated performance engineering frameworks that detect performance and scalability issues early on in a fast-paced agile environment. Previously he was a technology consultant, focused on providing enterprise quality and performance engineering solutions. Anjeneya has worked with large institutions to set-up enterprise performance and quality engineering solutions.
Embedding Performance Engineering Into Continuous Integration & Continuous Delivery Pipeline
By – Anjeneya Dubey

Little Context about McGraw-Hill Education and Me

McGraw-Hill Education
Improving outcomes and delivering ed tech at scale across higher education and K-12

We partner with
14,000+ authors and educators in various fields of study:

connect
- 3.8 billion interactions
- Serving 8,577 institutions

ALEKS
- 4.0 billion questions
- Serving 446 institutions

LEARNSMART®
- 9.3 billion questions
- Recently growing at an average of 100 million interactions per month

connectedED
- 8.6 million teachers
- Serving 22 percent of U.S. students

Anjeneya Dubey
Director of Performance Engineering
Anjeneya.dubey@mhedu.com

- Software engineering
- Performance Engineering
- Capacity Engineering
- Infrastructure Planning and implementation
- AWS Cloud Architecture & Operations
- Site Reliability Engineering
Agenda

- Continuous Integration and Continuous Delivery
- What does it mean to include performance engineering into the CI CD Pipeline
- Challenges
- What did we do to include performance engineering in the pipeline
  - Process changes
  - Performance test types
  - Test Environment management
  - Test Data management
  - Tools and Technologies we use
  - Pass/fail Decision Making
- Self Service Performance Engineering
- Using AI in production
- Do’s and Don’ts
- Summary

Continuous Integration/Continuous Delivery

- Automated build process and build verification tests for each environment in Continuous integration
- Extend Continuous integration by rapidly deploying capabilities to users to gain competitive advantage
- Reduce test cycle time & time to market
- Highly automated testing & release/roll-back
- Quicker automated decision making & feedback loop
Embedding Performance into the Pipeline

Your pipeline as code
• Dev -> Test-> Prod
• Dev -> Test-> Performance -> Prod

What does it mean?
Adding Performance environment into the pipeline means that now the performance tests are blocking your code promotion

Challenges - Cultural

• Performance is an after thought
• Is not part of the agile teams
• Is not part of the quality teams
• Do not get included in the agile ceremonies
• Create awareness on performance tasks
• Empower dev to test
Challenges - Technical

• Automating the performance testing and analysis
• Reducing Time to prepare and execute test
• Quickly reacting to performance metrics
• Automatic Pass/Fail
• Scaling the load test tool for variety of tests
• Keep the testing env/data consistent
• What to Shift left what to shift right
• Cost of running performance test on every build

How do we do it @MHE?
Process changes

• Make non functional/Performance requirement as part of the functional requirement
• API contracts
• Include performance as part of definition of done for sprints
• Clear definition of performance ready product
• Discuss Performance results as part of the sprint demos with all stake holders

Performance Requirements Workflow

Stories with acceptance criteria that includes clear performance requirements
• API X must handle load of xx transactions per sec with 95%ile response time as 100 ms
• All Stories must be evaluated if they require performance criteria
• Performance tests should be created to validate the criteria within the sprints
• Poor Performance = Functional Bug
Typical PE Process

Test Environment

- Use production like performance env
- Spin up only when you run test to save cost
- Refresh DBs for test data management
Cloud makes it easier

- Can expand and contract - Autoscaling
- Infrastructure as code – Terraform, Puppet
  - Creating & destroying envs at ease
  - Create parallel envs for parallel executions

Spin up parallel envs for parallel executions

- Production capacity instance
- Protocol Level full load test
- UI performance test using functional test scripts

- Scaled down env
- Stress/capacity test

- Testing outside the pipeline
- Troubleshooting
- Benchmarking/baselining tests
Performance Test types in CI CD

- User Experience - Browser side performance
- Load tests
- Capacity/Stress Tests

Single user performance

Good UX = Customer Happy

How do we measure that?

- Collect single user browser side response times
- Leverage functional test scripts (selenium)
- Create scenarios that you want to measure through our self service automation framework
- All Methods in the scripts have the snippet that collects the response times
- Executed from various geo locations
- Usable time vs last byte
- Collecting HAR & Creating videos of the tests for offline analysis
- Upload the data to S3
- MHE Performance Platform takes over from there
Load tests

• Full load tests
• Scaled Down tests
• Stress test to find capacity

Feature Flags

What to do when you find performance issues?
• Block the release
• Turn Off the feature that creates the performance issue
Test Data management

- Make our tests self contained
- Create & destroy data as part of the test as much as possible
- For the ones you can't create during the test you create as part of the environment build out
  - Spin up parallel Aurora RDS with pre-seeded test data to speed up env build out

Tools & Technologies we use
Performance Engineering Platform

- Singular platform to manage performance lifecycle for all of our products
- Powers CI CD for Performance engineering
- Central repo for all metrics
- Dynamic thresholds
- Pass fail decision making
- Powers Self Service Performance Engineering

PE Platform Overview

- Collector Service
- Aggregator Service
- Reporting & Alerting Service
- Central Repository

Developers and other tools like New Relic, Amazon, JMeter, Datadog, sumologic, CircleCI, Jenkins, and JIRA are connected to these services.
PE Platform – Performance test types

Trending – Performance graph for each build
**Containerize JMeter**

- We use JMeter heavily for all the CI CD testing
- Distributed load testing – we need 1 master & N number of slaves to generate huge load
- Scaling the JMeter for thousands of users was a challenge
- Dockerize JMeter gives the scale needed
- Speeds up the provisioning
- Part of the infra as code – which means when the code gets deployed automatically JMeter farm gets provisioned where the test gets executed

**Automated Pass/Fail**

Based on 3 basic rules

- Simple & Easy
- Implementable
- Dependent on throughput, response times and system KPIs
Thresholds for pass/fail

- Static Business response times SLAs
- Dynamic user experience/API level Response times thresholds
- Dynamic System Resource utilization thresholds
- Based on historical trend for each API and alerts if it deviated from last n tests
- Allows separate threshold for each API
- Doesn’t allow slippage even within the contract

Self-Service Performance Engineering

- You don’t need to be performance engineer to run test
- Automate the entire performance cycle
  - Script Creation through a UI
  - Execute test as part of CI CD or Execute it on demand through voice enabled Alexa or a chatBot
  - Analysis through APM and MHE built Performance Platform
  - Automated Notification through Hipchat/Email/Pager Duty
  - Automated Defect creation with details in jira
Self-Service Performance Engineering

Test Creation
- CI CD
- Alexa
- Chat Bots

Execution
- APM
- MHE PE Platform

Analysis
- Hipchat
- Email
- Pagerduty

Defect
- Jira

Notifications
- Automated defect creation
- Summary of the test result
- APM dashboard links with drill down

• Automated real time hipchat notifications
• With Jira link and details
Shift Right - Anomaly detection

- Twitter Anomaly Detection
- Twitter’s Breakout Detection
- Pearson Correlation Algorithm
- K-Means Clustering
- New Relic Radar

Do’s & Don’ts

**Do’s**
- Start with simple
- Perfect it later
- Remove false positives - Get it right from the beginning
- Know your applications KPIs
- Run parallel tests
- Run continuous tests

**Don’ts**
- Don’t run benchmark & endurance test in CI
- Don’t remove the failing tests to pass through CD
- Don’t keep increasing the thresholds to pass tests
- Don’t reinvent your PE framework rather see how you can leverage your existing tools and framework in CI CD
Summary

• Include performance engineering in your CI CD pipeline
• Automate & automate
• Make your tests repeatable
• Collect metrics along the way
• Avoid false positives
• Keep analysis & decision making simple
• Empower devs to test

Questions?