Magento Marketplace and EQP 2 Testing Framework

J Ravi Menon
Magento Marketplace Lead Architect
@jrmenon
ravmenon@adobe.com

Magento

imagine 2017

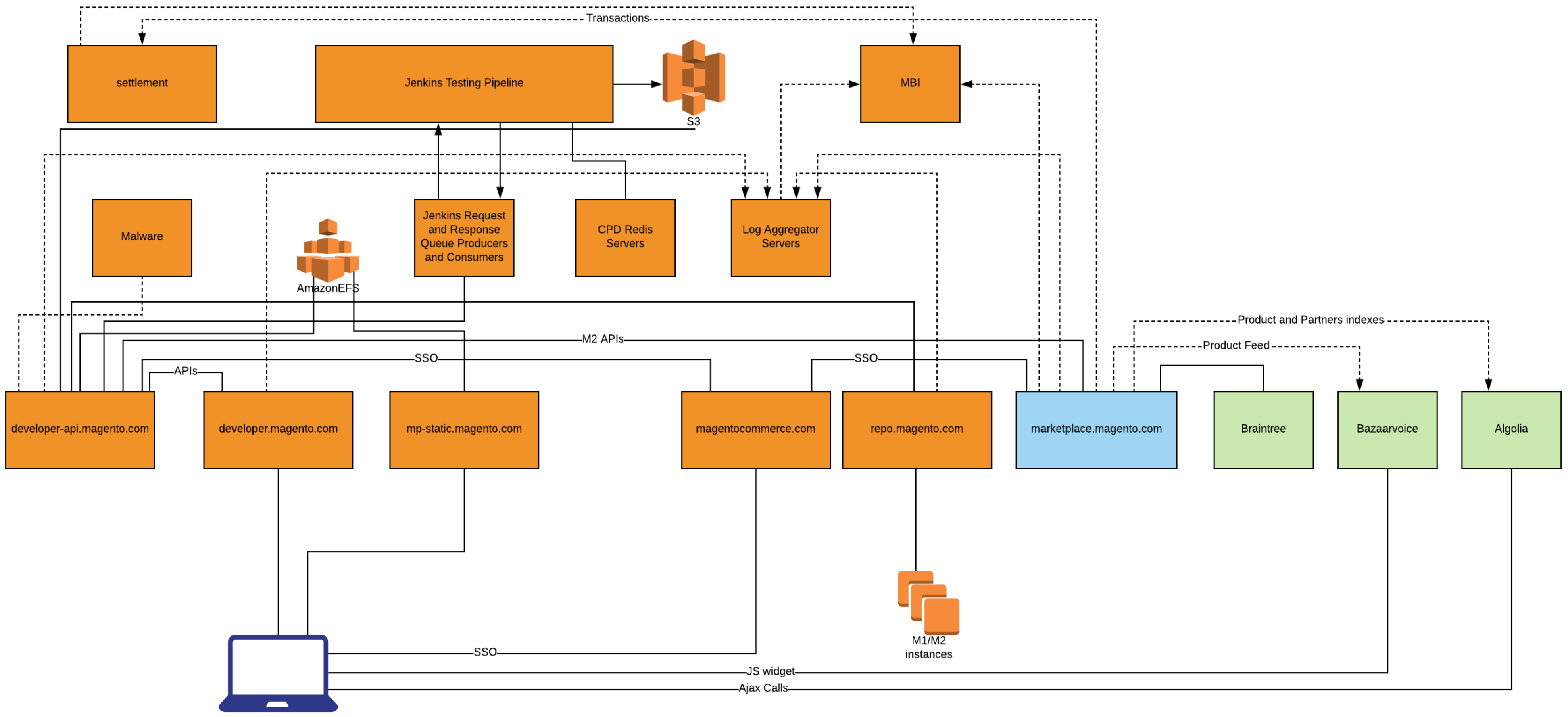
Introduction

Introduction

- Extension Quality Program (EQP) 2.0 went live on October 1st 2018
- Full revamp of the Developer Portal – https://developer.magento.com/
- Legacy M1 platform removed.
- API-First Design!
- A new extensible and scalable infrastructure for automated tests.
- Support for parallelization of technical (automated tests + manual QA) and marketing tracks of the EQP Pipeline.

Magento

Introduction – Arch Diagram



Introduction – Tech Stack

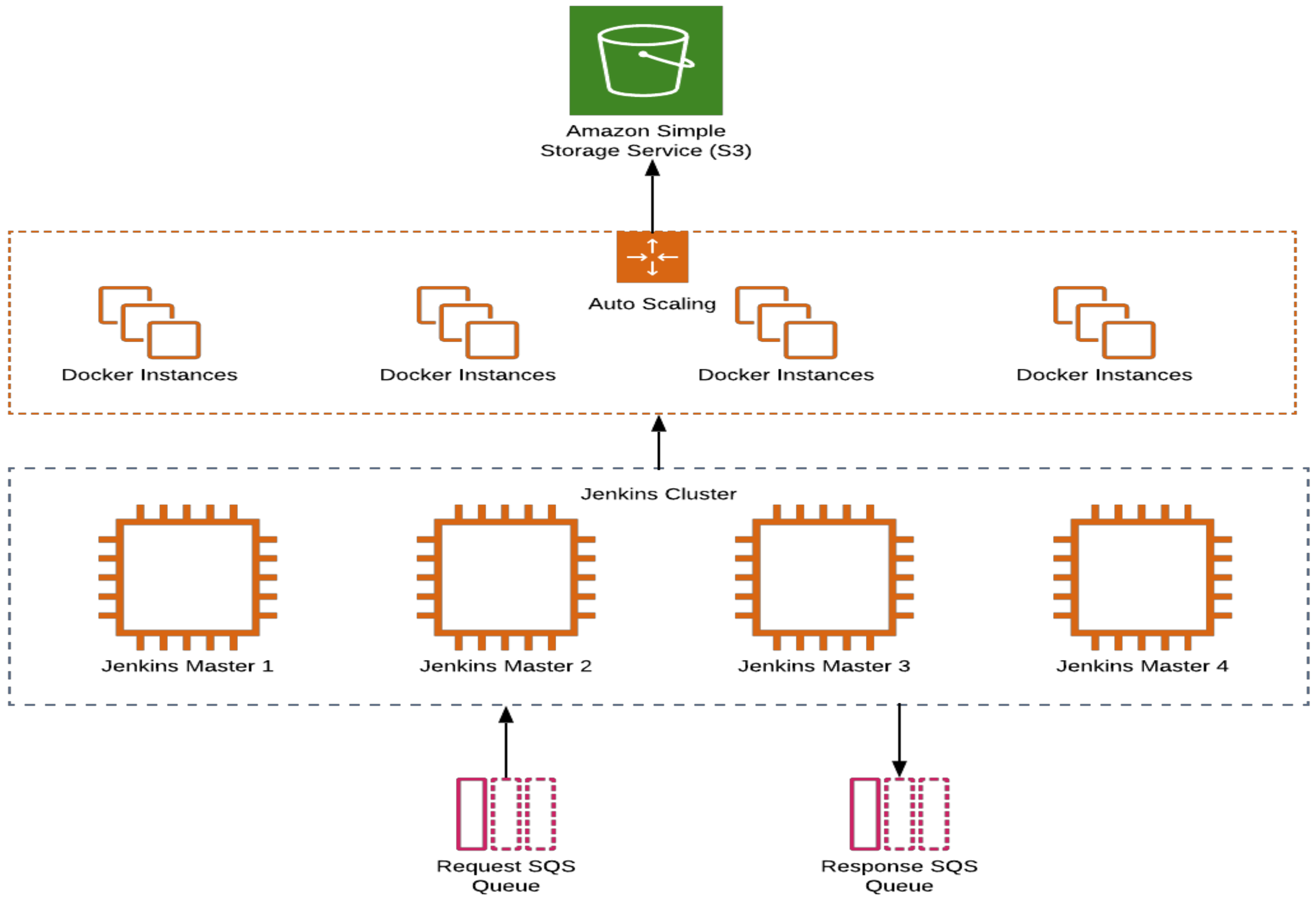
- Backend services on Symfony with PHP 7x and nginx + fpm workers.
- Datastore is MySQL accessible only to the API sub-system.
- RabbitMQ and Amazon SQS for async processing.
- Frontend is SPA using React + Redux.
- Testing infrastructure built on top of Jenkins and Docker.

Introduction – Tech Stack

- Details at https://live-eu.magento.com/sites/default/files/Breakout%20Session%20VIII_Winning%20Solutions%20Track.pdf

Jenkins Testing Infrastructure

Jenkins Testing Infrastructure



Jenkins Testing Infrastructure

- Designed to run automated tests in parallel.
- Test requests submitted on Amazon SQS.
- Multiple Jenkins Masters read requests from the queues (fault tolerance).
- Based on the test request, each Jenkins Master spawns the relevant number of instances, runs a Docker image, and invokes the test driver script with all the inputs supplied in the request.
- Each test tool runs the test, and outputs the result in a unified format – reports.json.

Jenkins Testing Infrastructure

- The test output is saved to S3 buckets.
- The test run completion notified to the response Amazon SQS.
- Overall test result can be in 3 states:
  - PASS
  - FAIL
  - REVIEW
- The test instance running Docker is then shutdown:
  - instance lifetime = lifetime of test, and designed to be stateless.

Jenkins Testing Infrastructure

- Launching of test instances designed to be elastically scalable with sufficient throttling logic to keep it within overall resource usage limits.

- Current set of tests:
  - Malware check (outside of main Jenkins pipeline for security reasons).
  - PHPCS (static analysis)
  - Install + Varnish test
  - Patch diff check:
    - If satisfied, manual Q/A step can be bypassed.
  - Copy-Paste Detector for plagiarism detection.

- Due to stateless nature of docker instances, it is easy plug-in new tests within the framework.

Magento

Jenkins Testing Infrastructure

- Example of number of Docker Images spawned for install + varnish test for an extension supporting 2.1, 2.2, and 2.3, both CE and EE:

	5.6	7.0	7.1	7.2
2.1 CE	✓	✓	✓	
2.2 CE		✓	✓	
2.3 CE			✓	✓
2.1 EE	✓	✓	✓	
2.2 EE		✓	✓	
2.3 EE			✓	✓

Demystifying Install + Varnish Test

Demystifying Install + Varnish Test

- EQP 2 Docker Images for CE editions to be publicly available. Instructions on EE builds to be available.

- Demo of the following setup:
  - M2 2.2.7 CE with PHP version 7.1.x.
  - Varnish setup.
    - Default varnish.vcl changed to see ‘X-EQP-Cache’ header in production mode (values can be HIT or MISS)
  - M2 root in docker instance at /var/www/html.
  - Running as root inside docker.
  - /root/.composer/auth.json setup with composer credentials.
  - Test products added via php bin/magento setup:performance:generate-fixtures

Demystifying Install + Varnish Test

- The following urls should be setup:
  - http://magento.local/
  - http://magento.local/category-1.html
  - http://magento.local/category-2.html
  - http://magento.local/simple-product-1.html
  - http://magento.local/simple-product-2.html
  - http://magento.local/simple-product-3.html

- Example docker commands on my local:
  - docker run -h magento.local -p 80:80 <EQP Docker Image>
  - docker exec –it <image name from docker ps> /bin/bash

Demystifying Install + Varnish Test

- For install test, the following commands need to be run at M2 root:
  - `composer require vendor/extension`
  - `php bin/magento setup:upgrade`
  - `php bin/magento deploy:mode:set production`
    - (Behind the scene this runs di:compile and static-content:deploy)
  - `php bin/magento indexer:reindex`
  - `php bin/magento cache:clean`

- Any errors detected in the aforesaid steps is deemed an install failure, so it needs to be fixed before proceeding to the varnish test.

Demystifying Install + Varnish Test

- Varnish test steps via curl:
  - Visit all the urls via curl for the first time – e.g.
    - `curl -vvv http://magento.local/category-1.html 2>&1 | less`
    - For each of the urls, the ‘X-EQP-Cache’ header should show ‘MISS’ (pages being loaded for the first time)
  - Visit all the same urls for the second or more times:
    - Now for each of the urls, every time the ‘X-EQP-Cache’ header should show ‘HIT’ (all the pages being served from the varnish cache).
  - Update product prices either via admin or via M2 APIs’ and visit all product pages via curl:
    - Now for each of the product urls, the ‘X-EQP-Cache’ header should show the ‘MISS’ (varnish caches invalidated due to product price changes).
  - Visit the product urls again:
    - Now for each of the product urls, the ‘X-EQP-Cache’ header should show ‘HIT’ (varnish caches updated and pages served from here).

Demystifying Install + Varnish Test

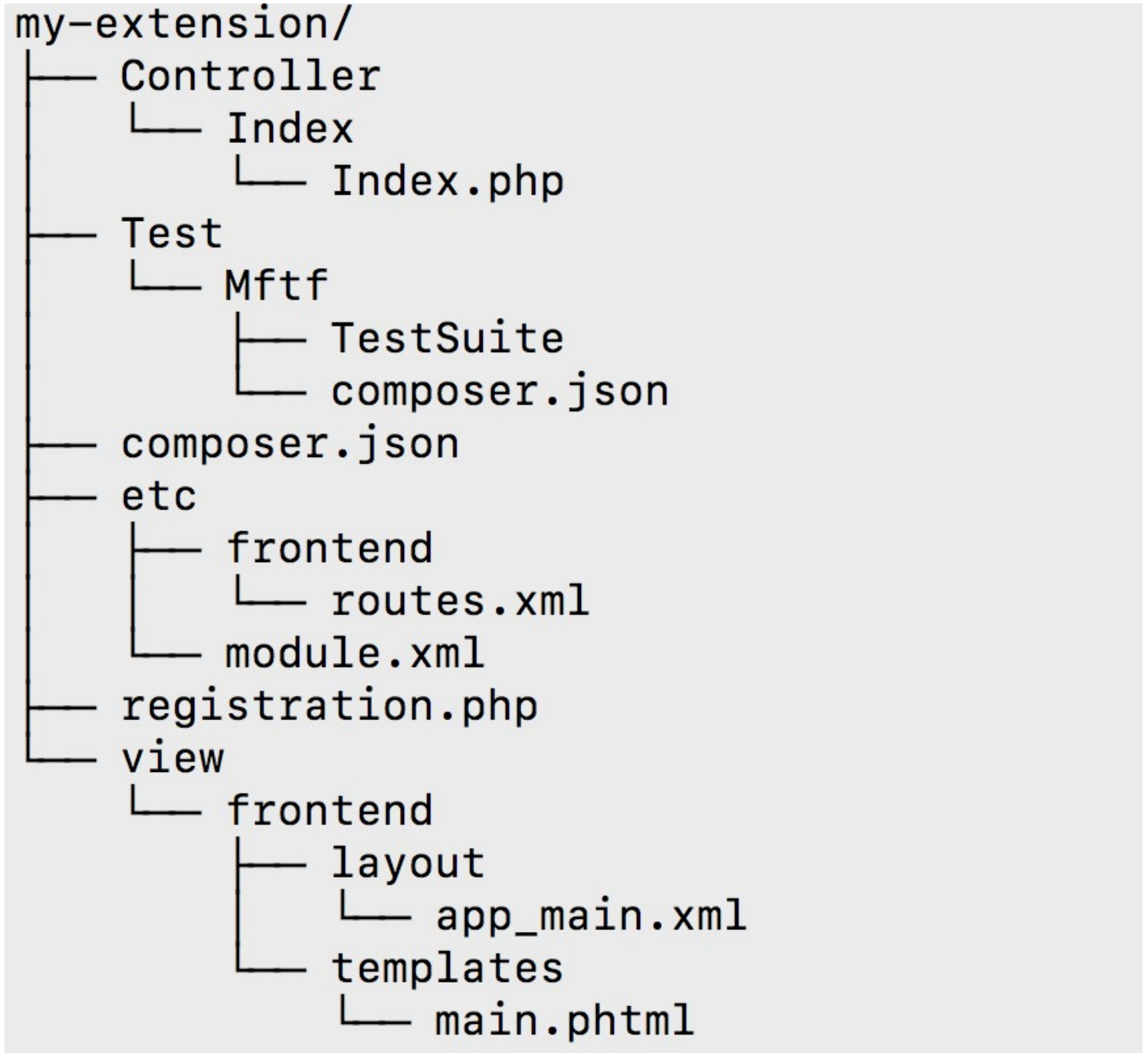
- The plan is to put the aforesaid steps into a wrapper script and have it available at https://github.com/magento/marketplace-tools.

MFTF Support for Extensions in Marketplace

Why MFTF?

- Encourages development of better quality extensions.
- Merchants/SI development teams can automate regressions as they customize the store.
- Helps in automation of the testing infrastructure to minimize time spent in manual QA stage.
- Can help in detecting interoperability issues between extensions.
- Path to building ‘fast-track’ option along with ‘quality score’ for rewarding vendors developing high-quality extensions.

MFTF Extension Layout



```
my-extension/
├── Controller
│   └── Index
│       └── Index.php
├── Test
│   └── Mftf
│       ├── TestSuite
│       └── composer.json
├── composer.json
├── etc
│   ├── frontend
│   │   └── routes.xml
│   └── module.xml
├── registration.php
└── view
    └── frontend
        ├── layout
        │   └── app_main.xml
        └── templates
            └── main.phtml
```

MFTF Extension Layout

- A single zip can be submitted, and the plan is to extract out the MFTF part to its own composer package.

- In the sample layout shown earlier, 2 composer packages will be setup in the repo: my-extension.zip (minus the MFTF portion) and my-extension-mftf.zip

- An extension buyer will automatically get entitlements to the MFTF package when purchasing the extension.

- This allows for merchant/SIs' to install MFTF packages in their DEV and QA environments for running regressions, but not have it installed in Production environments.

MFTF Extension Layout

- No need to have unnecessary test code in production:
  - Better from security perspective too.
- The MFTF package can be independently submitted too outside of the extension – i.e no change in extension code, but more tests added to increase coverage.
- Multi-phase rollout plan to be announced soon to give extension developers enough time to prepare for MFTF support.
- The Jenkins test infrastructure to support parallel run of MFTF tests to speed up execution.

What’s Next in EQP 2 Testing

Magento

What's next in EQP2 Testing

- PHPCS consolidation project rollout – session on contribution day being conducted by Olena Orobei (@LenaOrobei).

- MFTF support for Marketplace extensions.

- Weighted scoring plan for PHPCS to encourage developers to fix warnings – i.e. if score reaches a certain threshold due to too many warnings, an extension can be rejected.

- Security tests – static and dynamic:
  - Scanning tools for non-PHP code, javascript, css etc.
  - https://maxchadwick.xyz/blog/on-magento-module-vulnerabilities
  - https://store.fooman.co.nz/blog/mlau-2019-devexchange-extension-and-security-recap.html

- PHPStan: https://github.com/phpstan/phpstan - currently in investigation mode. Seems to offer better static analysis of PHP code via AST.

What’s next in EQP2 Testing

- Support for M2 ECE compatible docker image support for EE testing.
- Public release of EQP 2 Docker Images for CE, and instructions to build for EE with appropriate credentials:
  - This is to allow developers to run all EQP tests locally before submission to the Marketplace.
- Performance Tests.
- Magento component health index.
- Opening up the EQP 2 testing framework in collaboration with ExtDN.

What’s next in EQP2 Testing

- Opening up of EQP 2 REST APIs’ which will provide programmatic access to the testing infrastructure in sandbox and production environments.

References

- https://live-eu.magento.com/sites/default/files/Breakout%20Session%20VIII_Winning%20Solutions%20Track.pdf
- https://devdocs.magento.com/mftf/2.2/introduction.html
- https://github.com/magento/magento-coding-standard/wiki/Magento-Marketplace-Extensions-Verification
- https://devdocs.magento.com/guides/v2.3/cloud/docker/docker-config.html

Q/A