Transferring Performance Testing to FOLIO Teams

Overview

One of the objectives of the Performance Task Force is to enable teams in the FOLIO community to do performance testing by themselves early and often during the release cycle, as it is a good practice, so that performance issues are caught and addressed before the release. This document outlines the challenges and strategy to hand over performance testing to the FOLIO teams. However, it does not prescribe how teams will integrate performance testing in their day-to-day activities. Such discussion will be held in due time after teams have experienced with performance testing for a couple of quarters.

Current Environment

- 1 carrier-io instance
- 1 HA large-scale FOLIO deployment
- UChicago dataset: ~27M records dataset
- Github repo: https://github.com/folio-org/perf-testing

Challenges

- Knowledge transfer is the key because carrier-io has a steep learning curve
  - Using and administering carrier-io
  - Creating JMeter test scripts
  - Interpreting test results and troubleshoot issues
- Environment: enabling team to share, reuse testing environment
  - Each carrier-io instance has its own InfluxDB that stores all test data: JMeter test data, JVM profiling data, custom metrics data, etc...
  - Currently we are developing a large scale FOLIO deployment in the community's AWS account for PTF to use. Ideally we would have multiple.
  - Creating and maintaining a Big database.
  - Upgrading carrier-io

Proposals

1. Teams need to learn how to work with carrier-io ASAP. The best way is to embed team members within PTF so that they can learn and be trained.
   a. Each team chooses an engineer who loves tackling performance problems. This person will start out working with PTF to create JMeter test scripts for their team. PTF will spend 20% of the time to work with these team members to train them to write carrier-io compatible JMeter scripts, deploying the test script and its artifacts to carrier-io, execute the Jenkins job, and interpreting test data with carrier-io.
   b. These team members will go back to their team with this knowledge and lead/teach their team doing performance testing tasks.
2. Because each carrier-io instance has its own InfluxDB that stores all test data, all 8 teams can't use one carrier-io. Each team should have its own carrier-io instance.
   a. The carrier-io instance should be spun up and scaled down to save cost.

3. Because teams will need to performance-test their work before releases, having one large-scale FOLIO environment is not enough. There should be 2 to 3 (+1 for PTF?) large scale FOLIO environments that could be spun up and torn down on demand to save costs, all should be the same and should be on the same software versions. The "+1" is a dedicated environment for PTF as long as it exists.
   a. The 2-3 large scale FOLIO environments are to be shared among the teams
   b. Initially PTF will be responsible for upgrading the environments at the beginning of every sprint with the latest snapshots (usually of the commits whose stories were approved at the end of the previous sprints) based on FOLIO-SNAPSHOT software versions. Later on teams should take over this responsibility.
      i. Upgrading the environments regularly allows teams to test with the modules of latest versions, while balancing the instability of the frequent commits with upgrading the chain of dependencies required by the modules.
      ii. This upgrade includes running any database migration script to update the database. These scripts are run automatically when the module is enabled.
   c. Teams spin up an env to run tests, then drop env after testing - to restore the state for the next team to use. The restored state is what was deployed at the beginning of each sprint.
      i. When teams spin up an environment, they will be able to customize the version of any module to be loaded if desired. This includes any released version or from master or from any branch.
      1. Ex: Team wants to performance test the mod-circulation-storage code on a branch. They will set the version of this mod-circulation-storage for it to be loaded on start up.
   d. There should be a Wiki page for teams to schedule a timeslot or timeslots to run their tests.
   e. Teams will have two hours after running a test to collect data and examine results, after which the environment would automatically get shut down. (Since the data will be stored in a persistent EBS volume, the test data won't be dropped after the environment is shut down.)
      i. If a team needs to run database migration script for its performance testing, in some cases it could take more than 2 hours. In this case the team running migration script will need to communicate with the teams that are after their timeslot to update them of the overtime use.
      ii. Note that once testing is finished and the environment is dropped, the data will be restored to the beginning of the sprint state, so all this migration will be gone.
4. **Environment Costs**
   a. The following assumptions are made to determine costs
      i. Environments will be used for 1/3 of the time, or 10 days in a month
      ii. Each day will be used for about 12 hours, or 1/2 day
      iii. Therefore it will be a total of five (5) 24-hours days in a month, which equals to 1/6 of the full month
      iv. The costs per environment is 1/6 of the normal use, taking advantage of the ability to spin up and tearing down the environment.

<table>
<thead>
<tr>
<th>1 Community FOLIO deployment (1-year Reserved)</th>
<th>Price/hour</th>
<th>Hours/month</th>
<th>Instances</th>
<th>Monthly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>EKS Cluster</td>
<td>0.1</td>
<td>120</td>
<td>1</td>
<td>12.00</td>
</tr>
<tr>
<td>Database (t3.xlarge)</td>
<td>0.104</td>
<td>120</td>
<td>1</td>
<td>12.48</td>
</tr>
<tr>
<td>EC2 (t3.xlarge)</td>
<td>0.104</td>
<td>120</td>
<td>6</td>
<td>74.88</td>
</tr>
<tr>
<td>Load Balancers (Classic)</td>
<td>0.025</td>
<td>120</td>
<td>4</td>
<td>12.00</td>
</tr>
<tr>
<td>EBS (general purpose, gp2)</td>
<td>0.1/GB</td>
<td>142 GB</td>
<td>8</td>
<td>113.6</td>
</tr>
<tr>
<td><strong>Total FOLIO</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>$125.53</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1 Carrier-io (1-year Reserved)</th>
<th>Price/hour</th>
<th>Hours/month</th>
<th>Instances</th>
<th>Monthly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>EC2 (m5.xlarge - Reserved)</td>
<td>0.121</td>
<td>120</td>
<td>1</td>
<td>14.52</td>
</tr>
<tr>
<td>Spot instance (t3.medium)</td>
<td>0.05</td>
<td>15</td>
<td>1</td>
<td>0.75</td>
</tr>
<tr>
<td>EBS (general purpose, gp2)</td>
<td>0.1/GB</td>
<td>200 GB</td>
<td>8</td>
<td>20.00</td>
</tr>
<tr>
<td><strong>Total Carrier-io</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>35.27</strong></td>
</tr>
<tr>
<td><strong>Monthly Grand Total</strong></td>
<td></td>
<td></td>
<td></td>
<td><strong>$160.80</strong></td>
</tr>
</tbody>
</table>

b. Each environment costs about $160/month, three environments = $482/month.

c. Using 3 years of reserved instances will bring down the costs to $112/month for one environment, 3 environments = $336/month (see attached spreadsheet CommunityPerfEnvironmentCosts.xlsx for more details)

**How to Get There**

- Create sandbox environments (carrier-io and FOLIO) for teams to play with during the transition.
  - [PERF-114](#) - Automate influxDB and Grafana installation **CLOSED**
  - [PERF-104](#) - Build a large-scale FOLIO in the community's AWS **CLOSED**

- [PERF-102](#) - Create Jenkins Job to automatically run tests in Github **CLOSED**
  - Automatically running tests checked into Github. Will need to add the step to spin up FOLIO when running tests. This helps to run tests and compare results against the previous test runs.

- Documentation:
  - [PERF-110](#) - Create a carrier.io documentation "How to work with..." **OPEN**
  - Upgrading carrier-io going forward
  - Create how-to documentations to administer carrier-io
  - Create a diagram or a set of diagrams showing pieces of carrier-io and of FOLIO to communicate the architecture and responsibilities
  - Performance Analysis documentation: what to look for, log analysis (missing indexes, database logs for slow queries), pgHero, pgAdmin, Performance Insight, metrics, trouble signs (such as slowness - runaway CPU/memory, 500 errors, database memory etc...), Giraffe analysis.