Increasing Efficiency in Code and Hardware by Implementing NeoLoad *Test as Code*

Evan Haberecht

Software Engineering Manager LexisNexis



- 17 Years in Software Testing (16 in Performance Analysis)
- Prior to current role, I worked for US Air Force & Fidelity Investments
- Joined LexisNexis in summer of 2017

What we'll talk about.

- What is Test as Code
- How does it differ from standard NeoLoad tests
- What we've implemented
- Benefits we've seen



The problem.







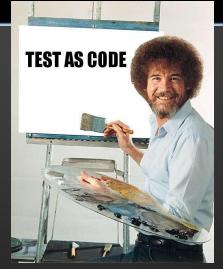
```
search.vaml ×
     name: YAML Example
     variables:
         name: serverName
         value: google
     sla profiles:

    name: exampleSLA

       - avg-request-resp-time warn >= 5000ms
     - name: myServer
       host: www.${serverName}.com
       scheme: http
     user paths:
     - name: hitServerUserPath
             name: hitServerTransaction
             sla profile: exampleSLA
                 name: hitServerRequest
                 sla profile: exampleSLA
                 url: /
                 server: myServer
                 method: GET
          - think time: 250ms
     - name: vamlPopulation
       user paths:
       - name: hitServerUserPath
         distribution: 100%
     - name: vamlScenario
       - name: vamlPopulation
         constant_load:
           users: 5
           duration: 1m30s
```

enario

The solution.





https://www.linkedin.com/in/evanhaberecht

- With Jenkins we've <u>automated</u> the process of starting and stopping pieces of our NeoLoad stack <u>when needed</u>
- This allows us to save money that was being spent on our testing instances
- Still have a few static machines up but we are working towards only having the Team Server and NeoLoad Web up full time for licensing and results.



name: hitServerUserPath

! searchUsersPath.yaml ×

user paths:

```
! searchVariables.yaml ×

1     hame: YAML Example
2     variables:
3     - constant:
4     name: serverName
5     value: google
```

```
! searchPopulation.yaml ×

1  populations:
2  - name: YAMLPopulation
3  user_paths:
4  - name: LoadServer
5  distribution: 100%
```

```
! searchServers.yaml x

1     servers:
2     - name: myServer
3     host: www.${serverName}.com
4     scheme: http
```

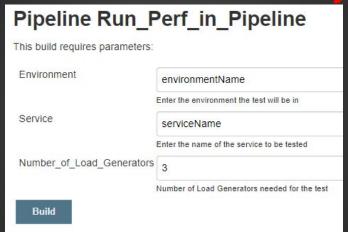
https://www.linkedin.com/in/evanhaberecht



 Storing a test's definition in multiple YAML files provides us the ability to switch between several different tests based upon variables.

These variables can be set manually set at run time:





 Or passed from other Jenkins jobs that want tests to be executed and wait for results

```
! searchVariables01.yaml ×

1    name: YAML Example
2    variables:
3    - constant:
4    name: serverName
5    value: bing
```

```
▲ CI.PERF

■ data

    autocomplete

                        stage('Run NeoLoad') {
   searchTerms.csv
                            steps {
                                script {
   ▶ csr
                                    try{
   ▶ HAMR
                                        sh "sleep ${env.Run_Sleep_Time}"
                                        sh "\"/usr/local/bin/NeoLoadCmd\" --noGUI -NTS http://${env.NLTeamServerURL} -NTSLogin \"${env.NLTeamServerUser}:$
  Jenkins
                                         {env.NLTeamServerPassword}\" -leaseLicense \"${env.NLLicense}:${env.NLUsers}:${env.NLTime}\" -project \"${WORKSPACE}/
   Jenkinsfile
                                        Projects/${env.service}.nlp\" \"${WORKSPACE}/YAML/populations/${env.Service}.yaml\" \"${WORKSPACE}/YAML/
                                        scenarios/${env.Service}.yaml\" \"${WORKSPACE}/YAML/servers/${env.Service}/${env.Environment}.yaml\" \"${WORKSPACE}/
  Projects
                                        YAML/SLA/${env.Service}.yaml\" \"${WORKSPACE}/YAML/user paths/${env.Service}.yaml\" \"${WORKSPACE}/YAML/variables/$
  python
                                         {env.Service}.yaml\" -launch ${env.Service} -nlweb -nlwebAPIURL \"http://${env.NLWebURL}\" -nlwebToken $
                                         {env.NLWebToken} --override-lg ${WORKSPACE}/${env.Second IP File}"

■ YAML

   populations
                                    catch(ex) {
                                        println("Run NeoLoad stage failed with this error: ${ex.getMessage()}\n")
   scenarios
                                        println("Stack trace:\n ${Arrays.toString(ex.getStackTrace())}\n")
   servers
   ▶ SLA
   user_paths
```

variables

scenarioDefinitions.yml

The mistakes people make.

- YAML formatting is serious business
- NeoLoad 6.9 and earlier requires a NeoLoad project (.nlp) file to be referenced with Test as Code

Our goal is to simplify things. Don't over-complicate your solutions

Links and resources.

- NeoLoad Resources updated with v6.10
 - NeoLoad's Test as Code Syntax Documentation
 - NeoLoad's Command Line Syntax
- Repository with examples from this presentation
 - https://github.com/habereet/NeoLoadPresentation

Q and A

https://www.linkedin.com/in/evanhaberecht Evan.Haberecht@LexisNexis.Com

