ANALYTICS SIMULATOR LAB

Introduction and Purpose

The Analytics Simulator lab will walk Sales Engineers though the setup and execution of the AppDynamics Analytics Simulator. This simulator has been designed to allow the user to create customer specific transactional analytic data. This transactional data can then be used to create compelling dashboards and reports for customer engagements such as demonstrations and POV wrap-ups without the need to fully implement Analytics at a customer site.

All required assets including VM's, software, and documentation can be found within the Ravello Systems Blueprint – Analytics Lab Simulator-bp. If you do not have access to Ravello, please contact your manager before you begin for account approval.

Lab Credentials

This section will outline all credentials used within the Analytics Simulator Lab.

Ravello Systems:

Access: www.ravellosystems.com

User ID: Assigned individually, if you do not have a Ravello account please contact your manager

Password: Assigned individually, if you do not have a Ravello account please contact your manager

Controller VM:

Access: Terminal – ssh controller@[yourIP]

User: controller

Password: appdynamics123

AppDynamics Controller:

Access: http://[your IP]:8090/controller

User ID: admin

Password: appdynamics123

All Other Systems:

Password: appdynamics123

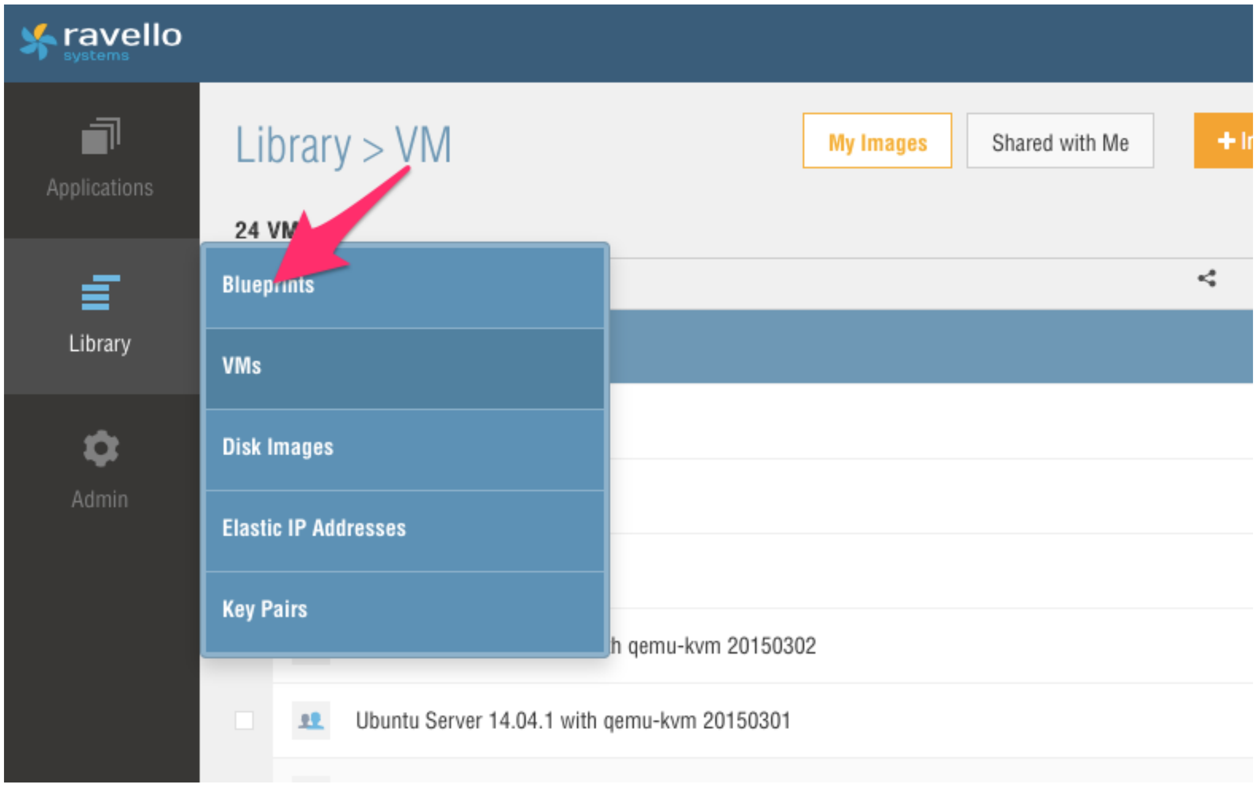
Please note that all credentials are CaSe SeNsItIvE

Prepare the Ravello Environment

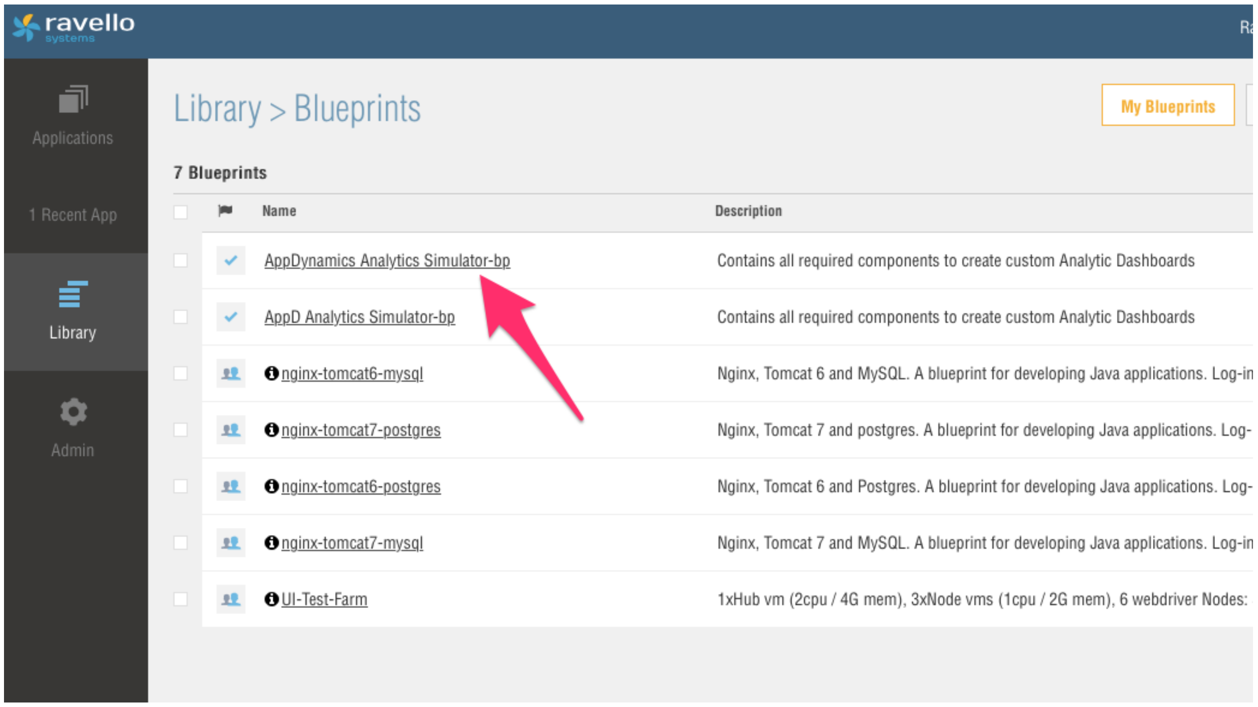
Ravello is a Cloud Application Hypervisor provider. It allows for the encapsulation of multi-VM applications that can be packaged and run from anywhere. Users can select a pre-defined blueprint and deploy their own application or build their own application environment from scratch using the available VM library. This lab will use a pre-existing blueprint.

Accessing Revello:

1) Log into Ravello Systems at www.ravellosystems.com
2) User your individual user ID and Password for access
    a. If you do not have access, please contact your manager for approval
3) Navigate to the Blueprints



4) The Blueprints screen appears. Select the “AppDynamics Analytics Simulator-bp” from the available Blueprints.

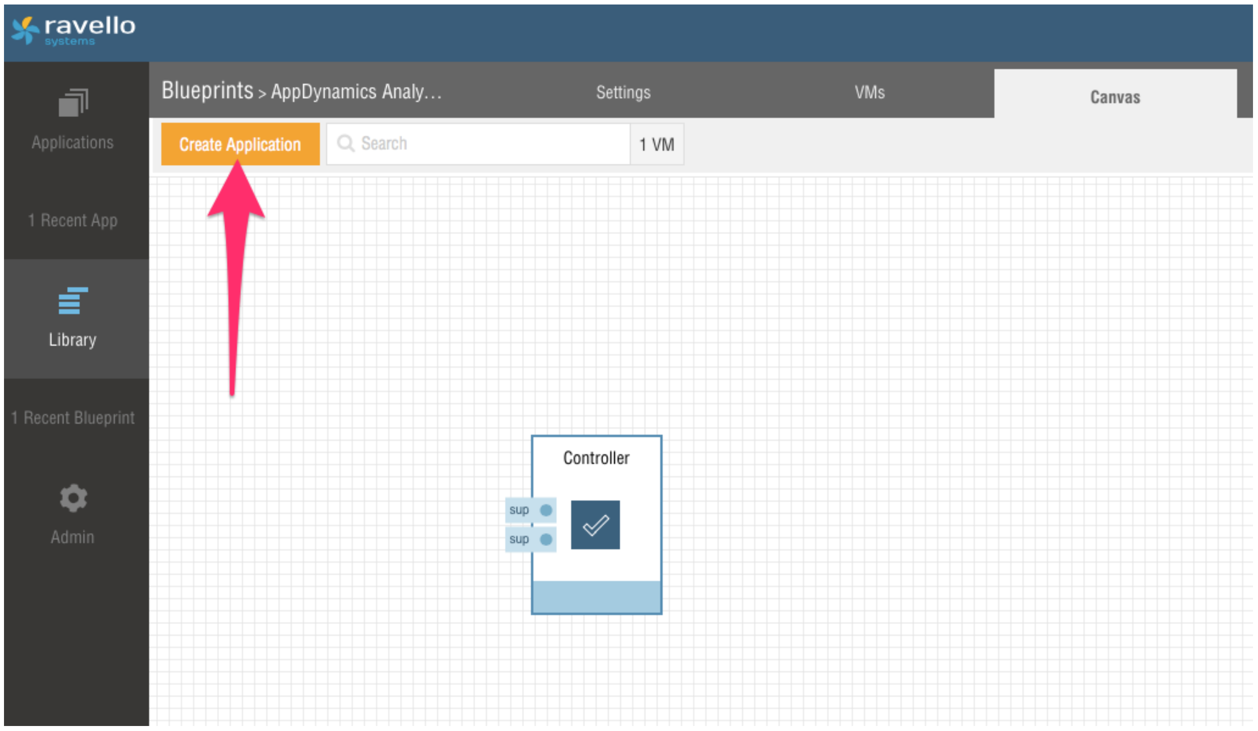


Library > Blueprints

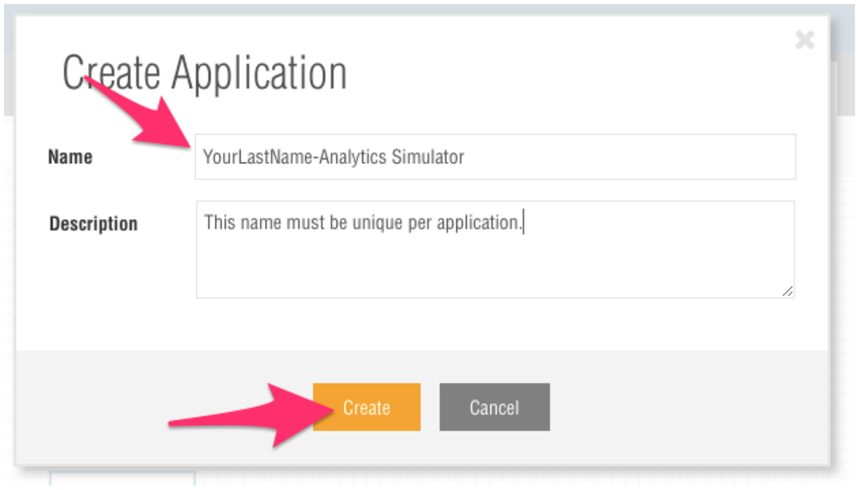
7 Blueprints

Name	Description
AppDynamics Analytics Simulator-bp	Contains all required components to create custom Analytic Dashboards
AppD Analytics Simulator-bp	Contains all required components to create custom Analytic Dashboards
nginx-tomcat6-mysql	Nginx, Tomcat 6 and MySQL. A blueprint for developing Java applications. Log-in
nginx-tomcat7-postgres	Nginx, Tomcat 7 and postgres. A blueprint for developing Java applications. Log-
nginx-tomcat6-postgres	Nginx, Tomcat 6 and Postgres. A blueprint for developing Java applications. Log-
nginx-tomcat7-mysql	Nginx, Tomcat 7 and MySQL. A blueprint for developing Java applications. Log-in
UI-Test-Farm	1xHub vm (2cpu / 4G mem), 3xNode vms (1cpu / 2G mem), 6 webdriver Nodes:

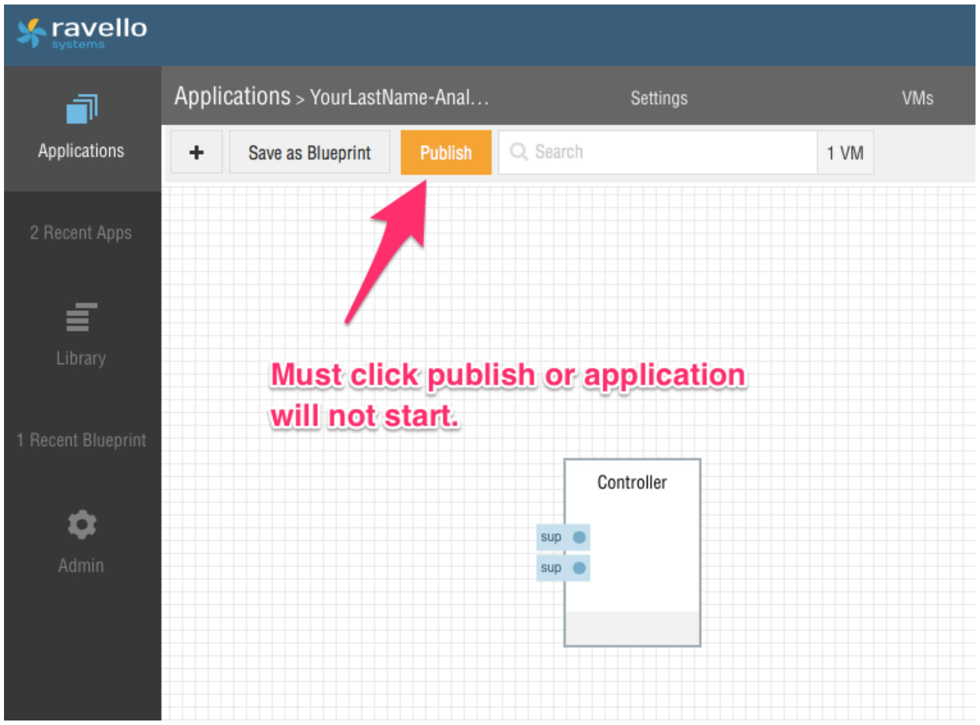
5) The *AppDynamics Analytics Simulator-bp* blueprint appears and contains one VM labeled Controller. Click the Create Application button in the top left corner. ***Do not change any application settings.***



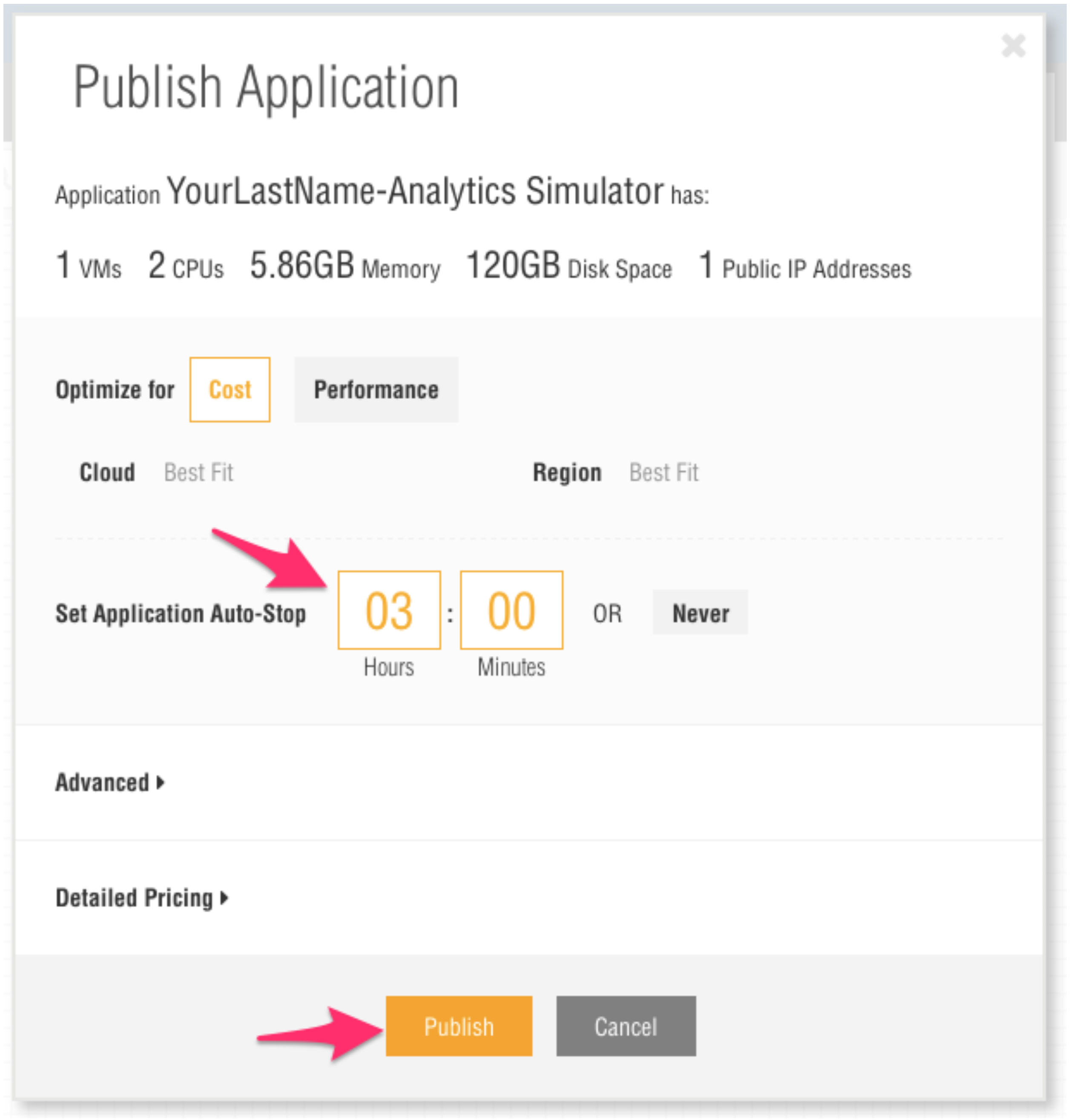
6) The Create Application dialog appears. Give the application a name with the format *LastName-AnalyticsSimulator*. ***The application name must be unique across the environment***. Click Create when finished.


Create Application
Name
YourLastName-Analytics Simulator
Description
This name must be unique per application.
Create
Cancel

7) Your unique application will be created. Click publish to finish application setup.



8) The Publish Application dialog appears. The most important component on this page is the Auto-Stop. Ravello only charges for running applications. Auto-Stop will shutdown all running VM's within the application at the end of the allotted time. Once running, the time can be extended as needed, however we need to make sure that Auto-Stop is used so that we are not needlessly charged for applications left running. ***Only publish an application for the time you will be using it.*** For this lab set Auto-Stop to 3 hours and click Publish.



Publish Application

Application YourLastName-Analytics Simulator has:

1 VMs 2 CPUs 5.86GB Memory 120GB Disk Space 1 Public IP Addresses

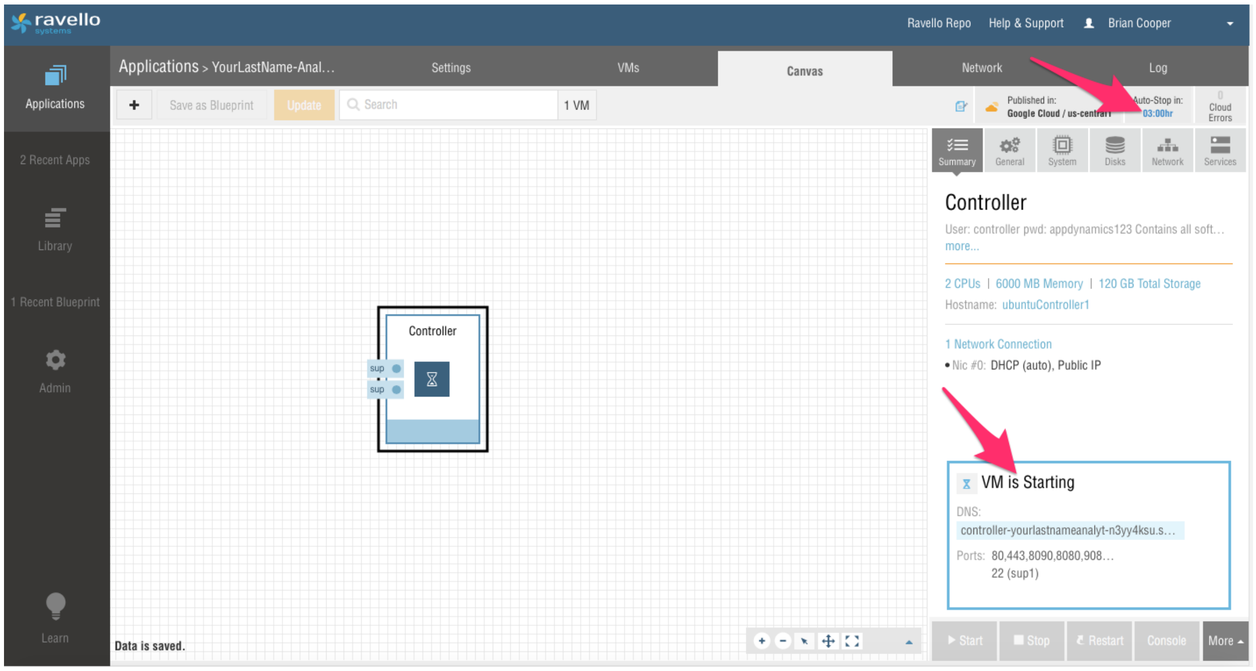
Optimize for [✓ Cost] Performance

Cloud Best Fit

Region Best Fit

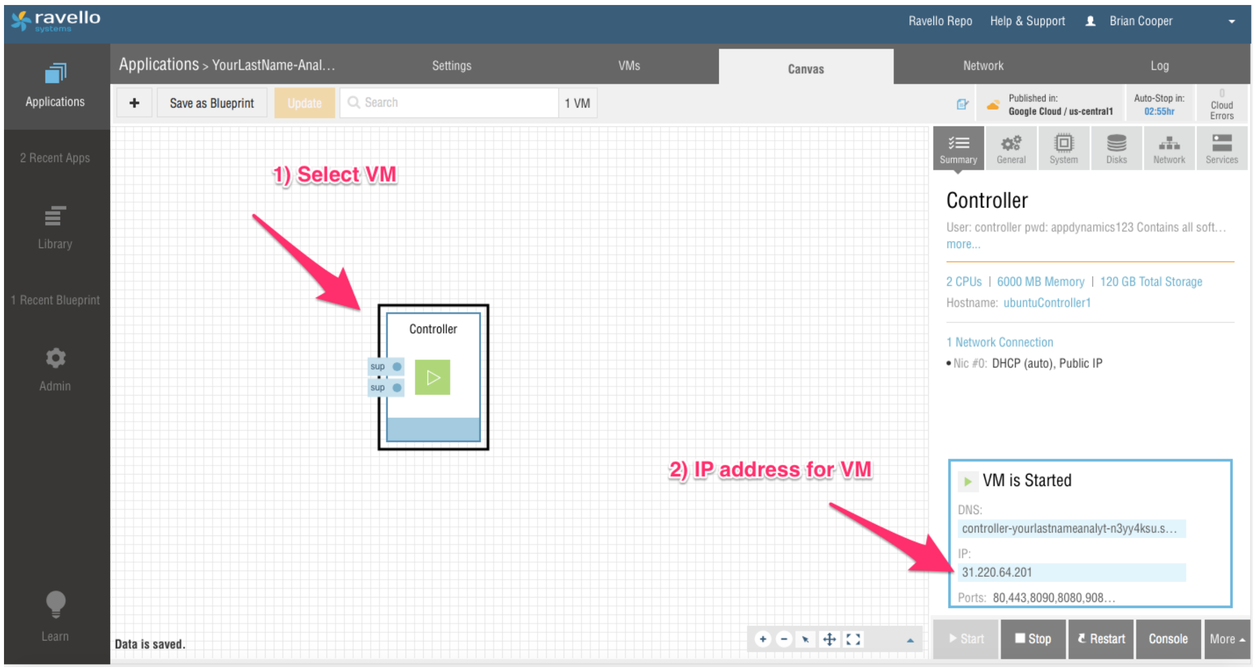
Set Application Auto-Stop 03 : 00 (Hours, Minutes) OR Never

9) Once published, your application will auto-start. You will also notice a countdown to power-off begins in the top right corner. Clicking this time will allow you to extend the Auto-Stop.



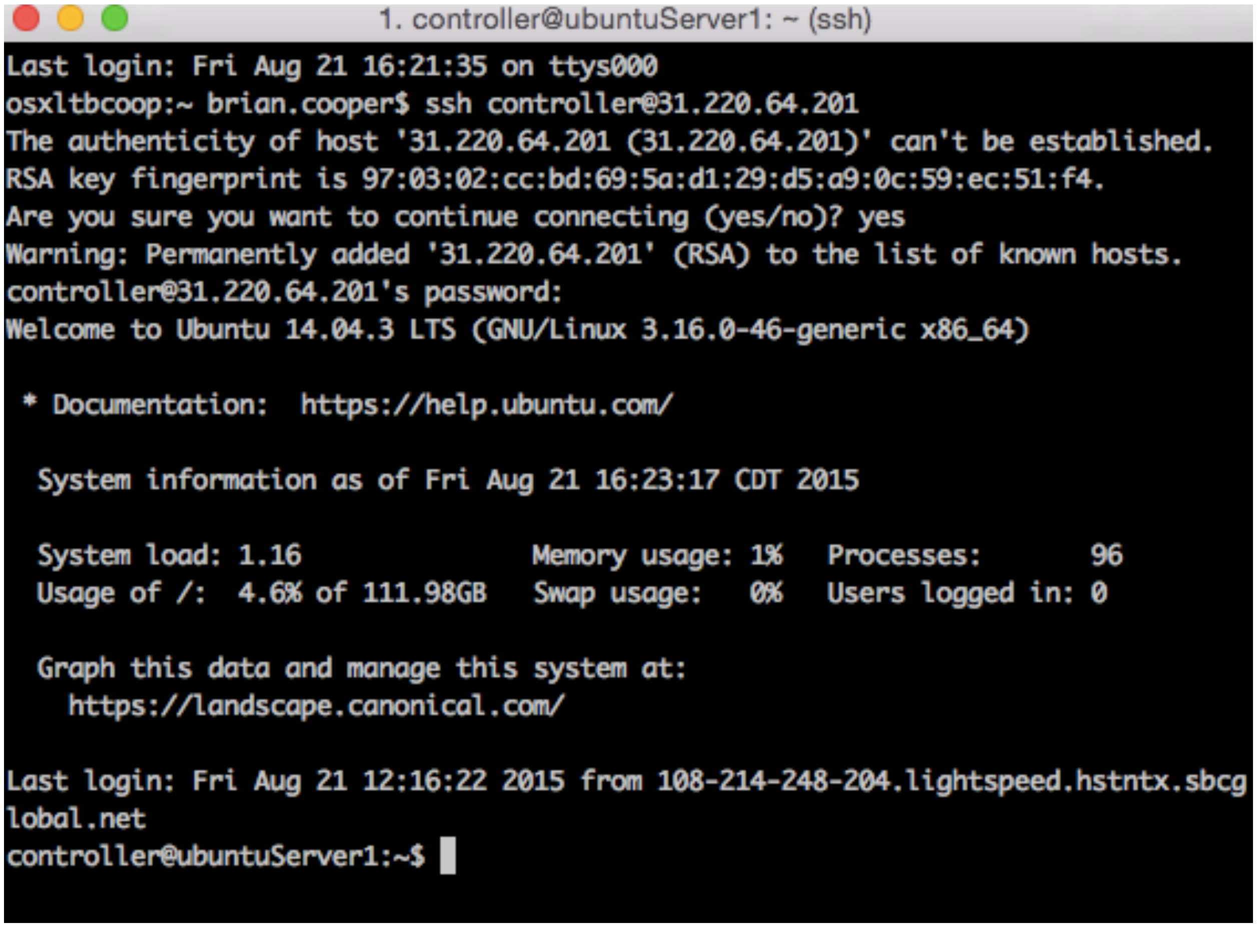
Controller
User: controller pwd: appdynamics123 Contains all soft… more…
2 CPUs | 6000 MB Memory | 120 GB Total Storage
Hostname: ubuntuController1
1 Network Connection
• Nic #0: DHCP (auto), Public IP
VM is Starting
DNS: controller-yourlastnameanalyt-n3yy4ksu.s…
Ports: 80,443,8090,8080,908…
22 (sup1)

10) Your application with take 5-7 minutes to provision and start for the first time. Be patient. Once all VM's have started, IP addresses will be available for each VM by selecting it in the Canvas view and reviewing the Summary screen in the far right pane.



Controller
User: controller pwd: appdynamics123 Contains all soft… more…
2 CPUs | 6000 MB Memory | 120 GB Total Storage
Hostname: ubuntuController1
1 Network Connection
• Nic #0: DHCP (auto), Public IP
VM is Started
DNS: controller-yourlastnameanalyt-n3yy4ksu.s…
IP: 31.220.64.201
Ports: 80,443,8090,8080,908…

11) Once your application is running, open a terminal and SSH to your Controller VM using the credentials supplied earlier in this document and its new IP address. ***It is not recommended to use the Ravello Console.***

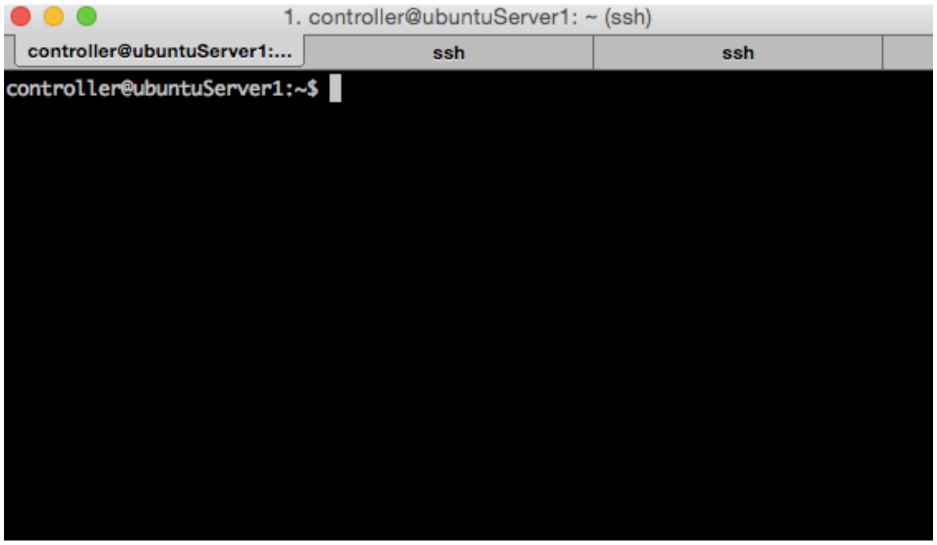


```
Last login: Fri Aug 21 16:21:35 on ttys000
osxltbcoop:~ brian.cooper$ ssh controller@31.220.64.201
The authenticity of host '31.220.64.201 (31.220.64.201)' can't be established.
RSA key fingerprint is 97:03:02:cc:bd:69:5a:d1:29:d5:a9:0c:59:ec:51:f4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '31.220.64.201' (RSA) to the list of known hosts.
controller@31.220.64.201's password:
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.16.0-46-generic x86_64)

 * Documentation:  https://help.ubuntu.com/

  System information as of Fri Aug 21 16:23:17 CDT 2015

  System load: 1.16               Memory usage: 1%   Processes:       96
  Usage of /:  4.6% of 111.98GB   Swap usage:   0%   Users logged in: 0

  Graph this data and manage this system at:
    https://landscape.canonical.com/

Last login: Fri Aug 21 12:16:22 2015 from 108-214-248-204.lightspeed.hstntx.sbcg
lobal.net
controller@ubuntuServer1:~$
```

Prepare the Analytics Environment

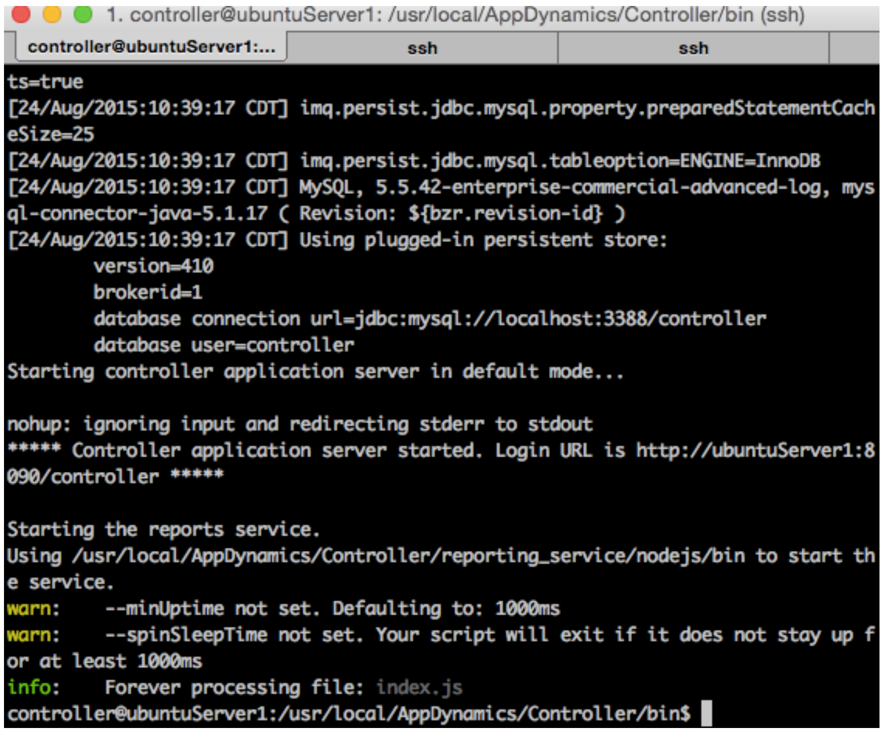
In this section you will prepare the environment to run the AppDynamics Analytics Simulator. This lab runs Analytics against a single Controller and its embedded Events Service. It also uses a single Machine Agent and embedded Analytics Agent. This section will walk through starting the existing Controller and Events Service and verifies they are both functioning as expected.

1) Open a terminal and SSH to your existing Controller environment created in the above section. Open an additional tab and repeat until there are three SSH connections open to your Controller (Will be used during subsequent lab sections).


1. controller@ubuntuServer1: ~ (ssh)
controller@ubuntuServer1:...
ssh
ssh

```
controller@ubuntuServer1:~$
```

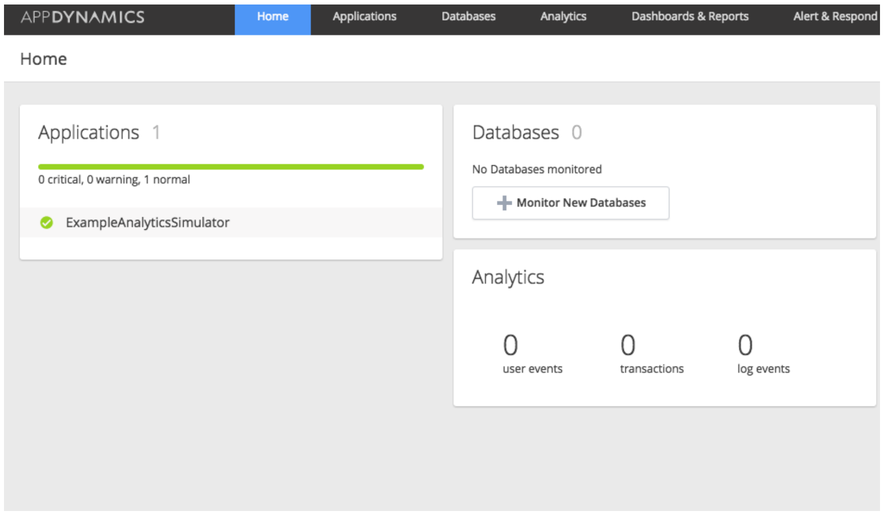
2) Start the AppDynamics Controller by navigating to /usr/local/AppDynamics/Controller/bin
   a. Execute the following to start controller: *sudo sh startController.sh*


1. controller@ubuntuServer1: /usr/local/AppDynamics/Controller/bin (ssh)
controller@ubuntuServer1:...
ssh
ssh

```
ts=true
[24/Aug/2015:10:39:17 CDT] imq.persist.jdbc.mysql.property.preparedStatementCach
eSize=25
[24/Aug/2015:10:39:17 CDT] imq.persist.jdbc.mysql.tableoption=ENGINE=InnoDB
[24/Aug/2015:10:39:17 CDT] MySQL, 5.5.42-enterprise-commercial-advanced-log, mys
ql-connector-java-5.1.17 ( Revision: ${bzr.revision-id} )
[24/Aug/2015:10:39:17 CDT] Using plugged-in persistent store:
        version=410
        brokerid=1
        database connection url=jdbc:mysql://localhost:3388/controller
        database user=controller
Starting controller application server in default mode...

nohup: ignoring input and redirecting stderr to stdout
***** Controller application server started. Login URL is http://ubuntuServer1:8
090/controller *****

Starting the reports service.
Using /usr/local/AppDynamics/Controller/reporting_service/nodejs/bin to start th
e service.
warn:    --minUptime not set. Defaulting to: 1000ms
warn:    --spinSleepTime not set. Your script will exit if it does not stay up f
or at least 1000ms
info:    Forever processing file: index.js
controller@ubuntuServer1:/usr/local/AppDynamics/Controller/bin$
```

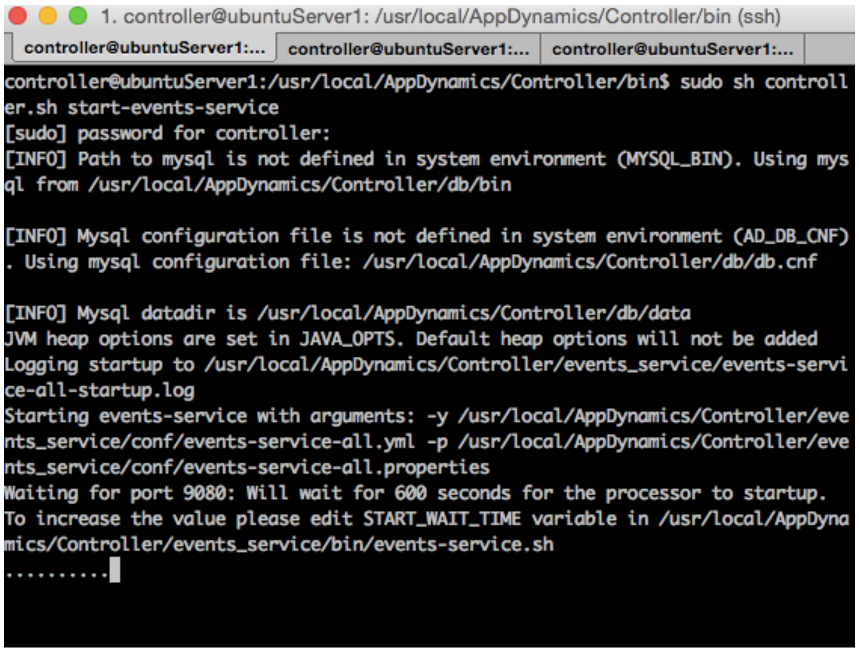
3) Verify the Controller has started by navigating to: http://[yourIP]:8090/controller

   a. Log into the Controller with credentials supplied above to verify it started properly. A single application titled “Example Analytics Simulator” is available.



4) Return to the terminal session and start the Events Service, ensure you have remained in the /usr/local/AppDynamics/Controller/bin directory.

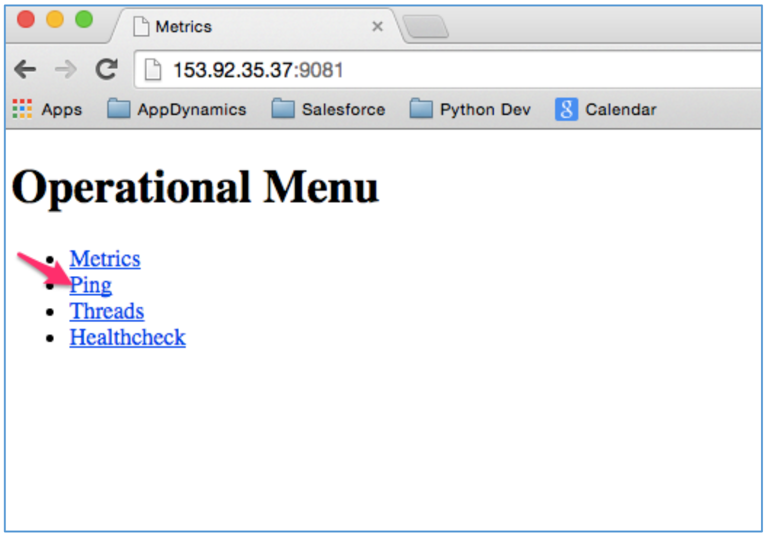
   a. Execute the following to start the Events Service: *sudo sh controller.sh start-events-service*


1. controller@ubuntuServer1: /usr/local/AppDynamics/Controller/bin (ssh)
controller@ubuntuServer1:...
controller@ubuntuServer1:...
controller@ubuntuServer1:...

```
controller@ubuntuServer1:/usr/local/AppDynamics/Controller/bin$ sudo sh controll
er.sh start-events-service
[sudo] password for controller:
[INFO] Path to mysql is not defined in system environment (MYSQL_BIN). Using mys
ql from /usr/local/AppDynamics/Controller/db/bin

[INFO] Mysql configuration file is not defined in system environment (AD_DB_CNF)
. Using mysql configuration file: /usr/local/AppDynamics/Controller/db/db.cnf

[INFO] Mysql datadir is /usr/local/AppDynamics/Controller/db/data
JVM heap options are set in JAVA_OPTS. Default heap options will not be added
Logging startup to /usr/local/AppDynamics/Controller/events_service/events-servi
ce-all-startup.log
Starting events-service with arguments: -y /usr/local/AppDynamics/Controller/eve
nts_service/conf/events-service-all.yml -p /usr/local/AppDynamics/Controller/eve
nts_service/conf/events-service-all.properties
Waiting for port 9080: Will wait for 600 seconds for the processor to startup.
To increase the value please edit START_WAIT_TIME variable in /usr/local/AppDyna
mics/Controller/events_service/bin/events-service.sh
..........
```

5) Verify Events Service has started correctly.

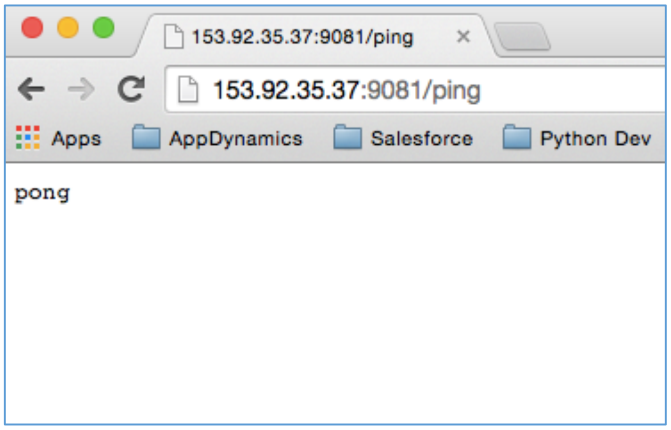
   a. Open a browser and navigate to http://[yourIP]:9081 and choose Ping from available options.



Operational Menu

- Metrics
- Ping
- Threads
- Healthcheck

   b. Receive Pong to verify Events Service is functioning properly



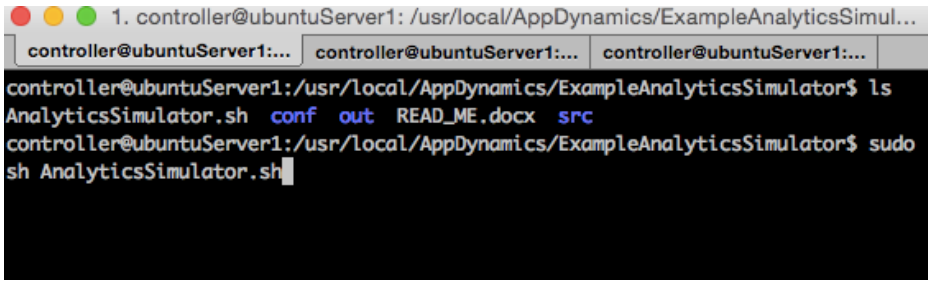
```
pong
```

Run Example Analytics Simulator

In this section you will execute the example Analytics Simulator that has been installed by default. The example utilizes a pre-existing Analytics Load file and will generate generic (user1, user2, search, checkout) transactional data. Once completed you will have a fully functional analytics environment including transactions, data, and a simple dashboard.

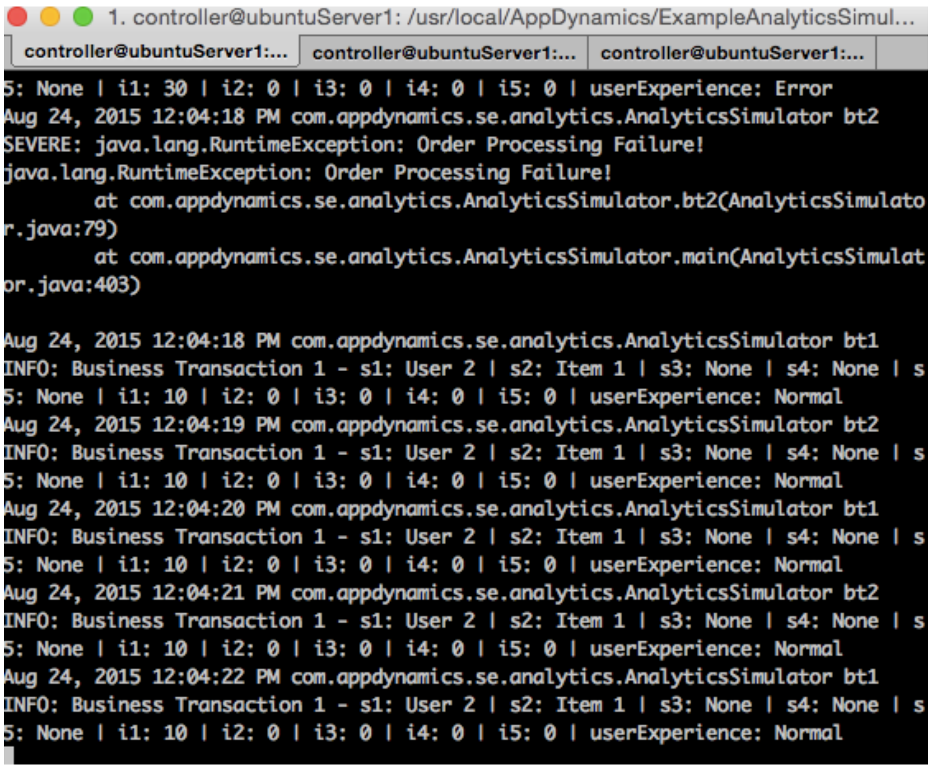
1) Start the Example Analytics Simulator by navigating to the following directory from within your SSH session: */usr/local/AppDynamics/ExampleAnalyticsSimulator/*

2) Execute the following to start the simulator and begin load generation: *sudo sh AnalyticsSimulator.sh*


1. controller@ubuntuServer1: /usr/local/AppDynamics/ExampleAnalyticsSimul...
controller@ubuntuServer1:...
controller@ubuntuServer1:...
controller@ubuntuServer1:...

```
controller@ubuntuServer1:/usr/local/AppDynamics/ExampleAnalyticsSimulator$ ls
AnalyticsSimulator.sh  conf  out  READ_ME.docx  src
controller@ubuntuServer1:/usr/local/AppDynamics/ExampleAnalyticsSimulator$ sudo
sh AnalyticsSimulator.sh
```

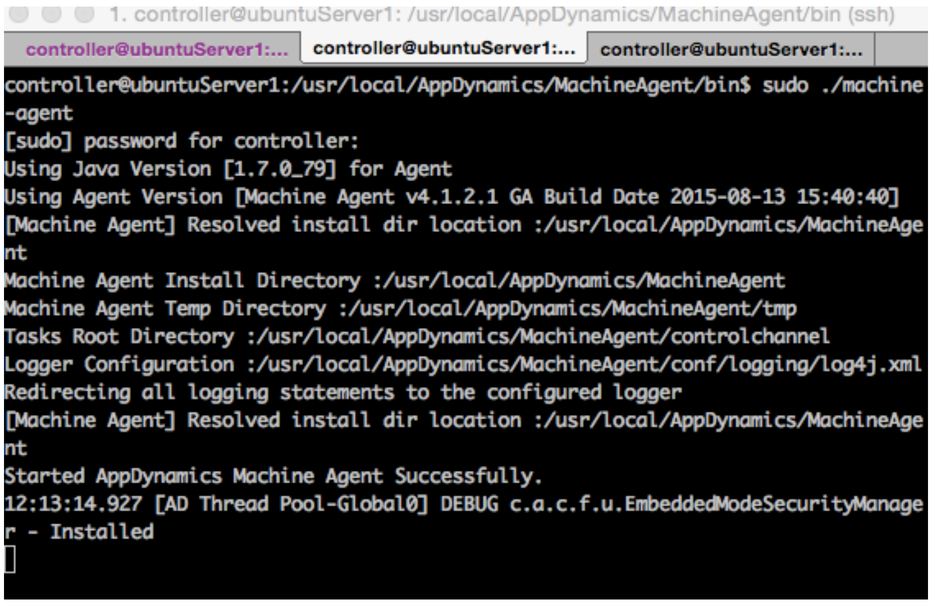
3) Once executed transactional load should begin to populate on the terminal screen. You should notice normal, very slow, and errored business transactions. ***Please note you must be in the ExampleAnalyticsSimulator directory in order for the simulator to successfully launch.***


1. controller@ubuntuServer1: /usr/local/AppDynamics/ExampleAnalyticsSimul...
controller@ubuntuServer1:...
controller@ubuntuServer1:...
controller@ubuntuServer1:...

```
5: None | i1: 30 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Error
Aug 24, 2015 12:04:18 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
SEVERE: java.lang.RuntimeException: Order Processing Failure!
java.lang.RuntimeException: Order Processing Failure!
        at com.appdynamics.se.analytics.AnalyticsSimulator.bt2(AnalyticsSimulato
r.java:79)
        at com.appdynamics.se.analytics.AnalyticsSimulator.main(AnalyticsSimulat
or.java:403)

Aug 24, 2015 12:04:18 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 2 | s2: Item 1 | s3: None | s4: None | s
5: None | i1: 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 12:04:19 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
INFO: Business Transaction 1 - s1: User 2 | s2: Item 1 | s3: None | s4: None | s
5: None | i1: 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 12:04:20 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 2 | s2: Item 1 | s3: None | s4: None | s
5: None | i1: 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 12:04:21 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
INFO: Business Transaction 1 - s1: User 2 | s2: Item 1 | s3: None | s4: None | s
5: None | i1: 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 12:04:22 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 2 | s2: Item 1 | s3: None | s4: None | s
5: None | i1: 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
```

4) Using the second terminal tab SSH to your Controller machine if you have not already and navigate to the installed Machine Agent's bin directory: */usr/local/AppDynamics/MachineAgent/bin*

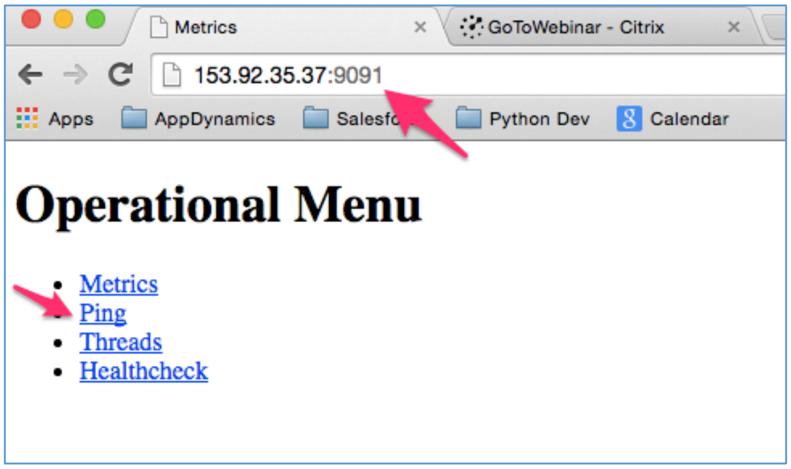
5) Execute the following to start the Machine Agent: *sudo ./machine-agent*



1. controller@ubuntuServer1: /usr/local/AppDynamics/MachineAgent/bin (ssh)
controller@ubuntuServer1:...
controller@ubuntuServer1:...
controller@ubuntuServer1:...

```
controller@ubuntuServer1:/usr/local/AppDynamics/MachineAgent/bin$ sudo ./machine
-agent
[sudo] password for controller:
Using Java Version [1.7.0_79] for Agent
Using Agent Version [Machine Agent v4.1.2.1 GA Build Date 2015-08-13 15:40:40]
[Machine Agent] Resolved install dir location :/usr/local/AppDynamics/MachineAge
nt
Machine Agent Install Directory :/usr/local/AppDynamics/MachineAgent
Machine Agent Temp Directory :/usr/local/AppDynamics/MachineAgent/tmp
Tasks Root Directory :/usr/local/AppDynamics/MachineAgent/controlchannel
Logger Configuration :/usr/local/AppDynamics/MachineAgent/conf/logging/log4j.xml
Redirecting all logging statements to the configured logger
[Machine Agent] Resolved install dir location :/usr/local/AppDynamics/MachineAge
nt
Started AppDynamics Machine Agent Successfully.
12:13:14.927 [AD Thread Pool-Global0] DEBUG c.a.c.f.u.EmbeddedModeSecurityManage
r - Installed
```

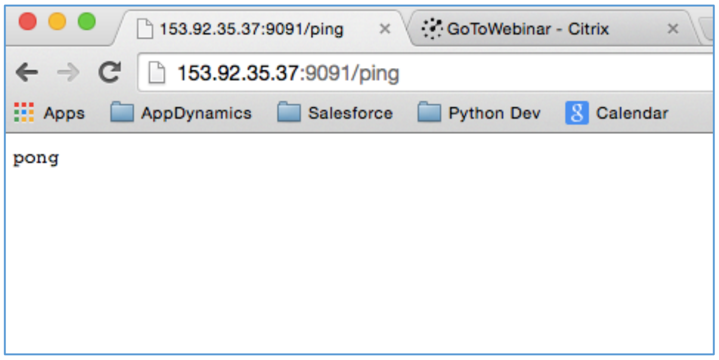
6) Verify the Machine Agent has started successfully by opening a browser and navigating to: http://[yourIP]:9091 and choose Ping from the available options.



Operational Menu

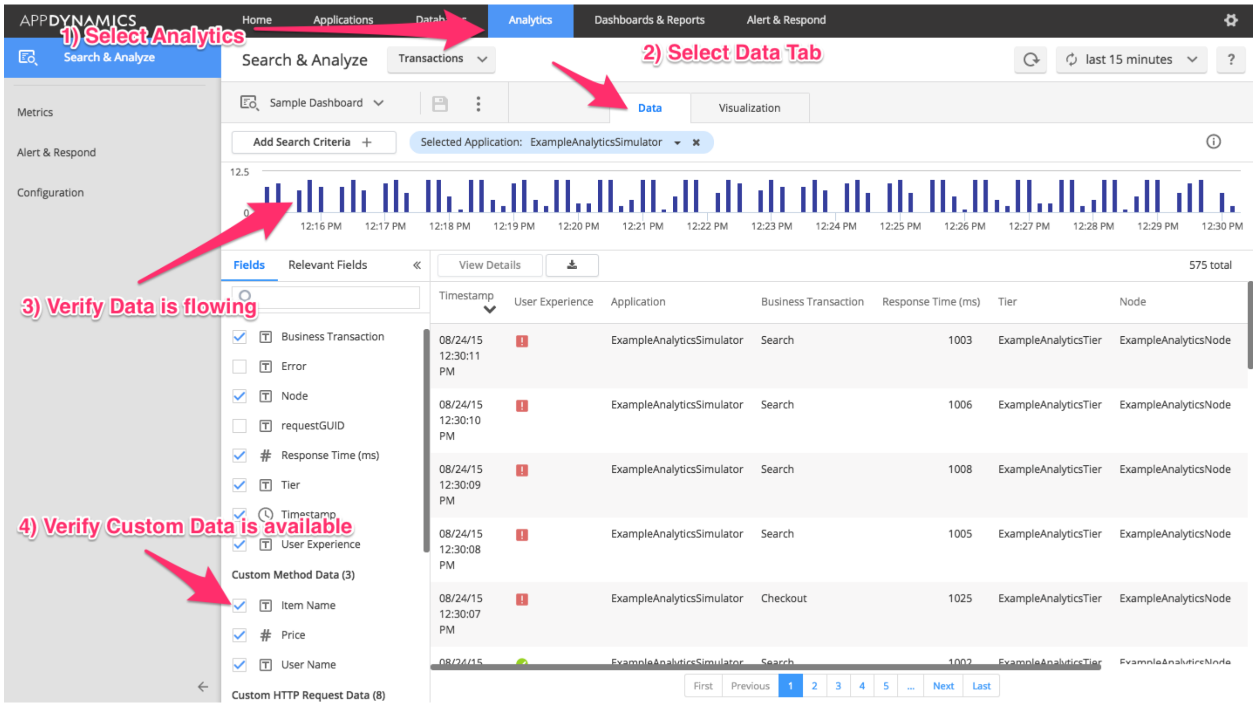
- Metrics
- Ping
- Threads
- Healthcheck

7) Receive Pong to verify Machine Agent is running successfully.



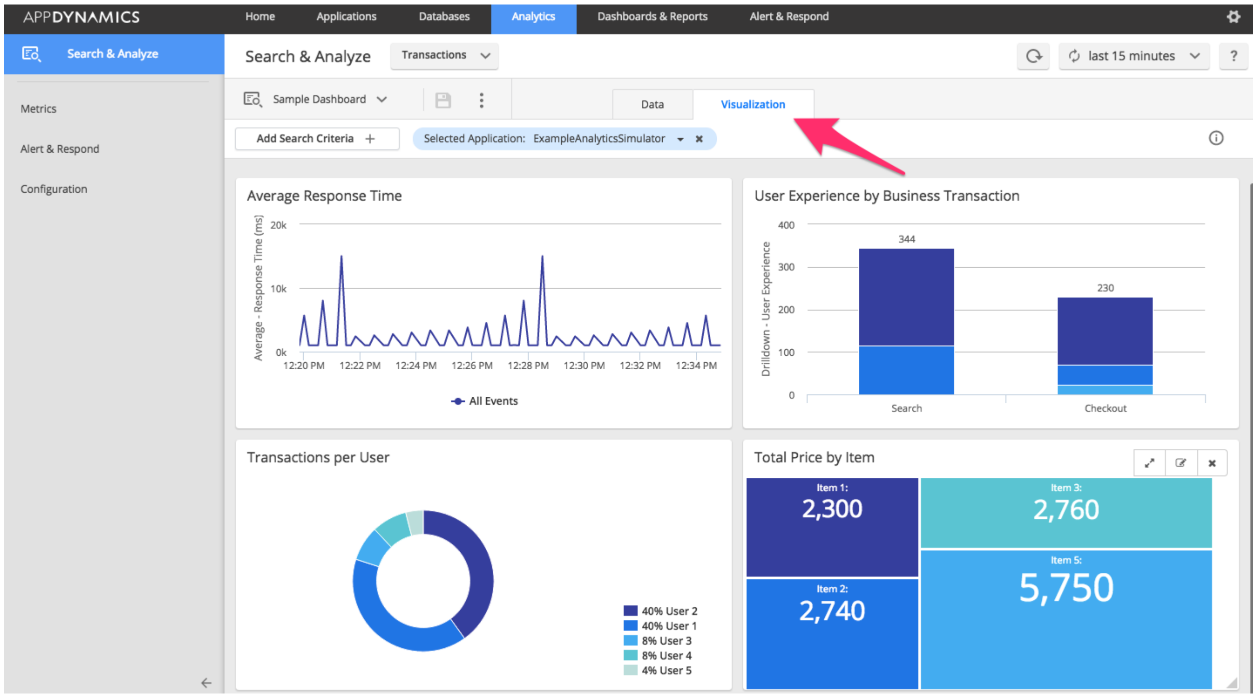
```
pong
```

8) Log into the Controller UI at: http://[yourIP]:8090/controller and click on the Analytics Tab at the top. Select the Data Tab and verify that transaction data is flowing and custom method data is available.



Timestamp	User Experience	Application	Business Transaction	Response Time (ms)	Tier	Node
08/24/15 12:30:11 PM	🟥	ExampleAnalyticsSimulator	Search	1003	ExampleAnalyticsTier	ExampleAnalyticsNode
08/24/15 12:30:10 PM	🟥	ExampleAnalyticsSimulator	Search	1006	ExampleAnalyticsTier	ExampleAnalyticsNode
08/24/15 12:30:09 PM	🟥	ExampleAnalyticsSimulator	Search	1008	ExampleAnalyticsTier	ExampleAnalyticsNode
08/24/15 12:30:08 PM	🟥	ExampleAnalyticsSimulator	Search	1005	ExampleAnalyticsTier	ExampleAnalyticsNode
08/24/15 12:30:07 PM	🟥	ExampleAnalyticsSimulator	Checkout	1025	ExampleAnalyticsTier	ExampleAnalyticsNode
08/24/15	●	ExampleAnalyticsSimulator	Search	1002	ExampleAnalyticsTier	ExampleAnalyticsNode

9) Click on the Visualization tab to view the ExampleAnalyticsSimulator custom dashboard.



10) Stop the Machine Agent by opening the terminal window where it is running and pressing *control-c*

11) Stop the ExampleAnalyticsSimulator by opening the terminal window where it is running and pressing *control-c*

Prepare Custom Analytics Load

In this section you will customize the Analytics Load document in order to create custom data for your assigned scenario. There are no right or wrong answers to this section, rather you must use the data to mock up a compelling story that you would deliver to the customer. Oftentimes this can be done as part of the POV wrap-up or as a standalone Analytics demo that uses your customers' terms, applications, and data.

Currently the Analytics Simulator is hardcoded with 10 different available Business Transactions, each having 5 String and 5 Integer values for use with Transaction Analytics. Business Transaction names and method parameters are generic (IE:bt1 - bt10, s1 - s5, i1 - i5) so they can be labeled via the configuration in the AppDynamics Controller UI to something that makes sense for your scenario.

1) Open a new SSH tab to your Controller if necessary and navigate to the following directory: */usr/local/AppDynamics/CustomAnalyticsSimulator/conf*
2) Open the load.txt file with your favorite editor. You may find it easier to copy this file back to your desktop and edit there, then replace when finished.

```
1|User 1|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
2|User 1|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
1|User 1|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
2|User 1|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
1|User 1|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
2|User 1|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
1|User 1|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
2|User 1|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
1|User 1|Item 3|None|None|None|30|0|0|0|0|Normal|No Error
2|User 1|Item 3|None|None|None|30|0|0|0|0|Error|Order
Processing Failure!
1|User 2|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
2|User 2|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
1|User 2|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
2|User 2|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
1|User 2|Item 1|None|None|None|10|0|0|0|0|Normal|No Error
2|User 2|Item 1|None|None|None|10|0|0|0|0|Very Slow|No
Error
1|User 2|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
2|User 2|Item 2|None|None|None|20|0|0|0|0|Normal|No Error
1|User 2|Item 3|None|None|None|30|0|0|0|0|Normal|No Error
2|User 2|Item 3|None|None|None|30|0|0|0|0|Error|Order
Processing Failure!
1|User 3|Item 5|None|None|None|50|0|0|0|0|Error|Cache
Timeout!
1|User 3|Item 5|None|None|None|50|0|0|0|0|Error|Cache
Timeout!
1|User 4|Item 5|None|None|None|50|0|0|0|0|Error|Cache
Timeout!
1|User 4|Item 5|None|None|None|50|0|0|0|0|Error|Cache
```

```
Timeout!
1|User 5|Item 5|None|None|None|50|0|0|0|0|Error|Cache
Timeout!
```

3) Edit the load.txt file to create your own Analytics scenario. Keep the following in mind as you do so:
   - Use the sample *load.txt* file as a template for generating your own load
   - Research websites, 10K's and Google for relevant information and ideas for your scenario
   - You can configure up to 10 Business Transactions with 5 String columns and 5 Integer columns each. If you do not need all of the columns, then just use None / 0 for placeholder values.
   - Normal / Error transactions will have ~1s response times and Very Slow transactions will have ~15 response times
   - Make sure you have many more Normal transactions than Very Slow otherwise the agent might not correctly classify the transactions
   - The Analytics Simulator will run through each transaction in this file and then reload the file repeatedly in a loop until you turn off the program, so you only need to ***configure the patterns*** that you want to show up in the data not 100% of the load
   - BT's can either:
     - Use the same columns and data (IE: in the ExampleAnalyticsSimulator load file Search and Checkout both use User Name in the second column)
     - Use different data in the same or different columns (IE: Search could alternatively use column 2 for Product Category)
     - You just need to remember which columns are used for which BTs when you configure Transaction Analytics in the Controller UI in the next section.
   - The columns in the load.txt file represent the following:
     1. ***Business Transactions:*** A number 1 – 10 that represents a Business Transaction that will be configured in the UI. (1=Search, 2=Checkout in the ExampleAnalyticsSimulator example)
     2. ***String Parameter 1:*** Any String value that represents a parameter (User Name in the ExampleAnalyticsSimulator example).
     3. ***String Parameter 2:*** Any String value that represents a parameter (Item Name in the ExampleAnalyticsSimulator example).
     4. ***String Parameter 3:*** Any String value that represents a parameter (None in the ExampleAnalyticsSimulator load – the parameter is not used so we put None as a placeholder).
     5. ***String Parameter 4:*** Any String value that represents a parameter (None in the ExampleAnalyticsSimulator load – the parameter is not used so we put None as a placeholder).
     6. ***String Parameter 5:*** Any String value that represents a parameter (None in the ExampleAnalyticsSimulator load – the parameter is not used so we put None as a placeholder).
     7. ***Integer Parameter 1:*** Any Integer value that represents a parameter (Price in the ExampleAnalyticsSimulator load).
     8. ***Integer Parameter 2:*** Any Integer value that represents a parameter (0 in the ExampleAnalyticsSimulator load – the parameter is not used so we put 0 as a placeholder).
     9. ***Integer Parameter 3:*** Any Integer value that represents a parameter (0 in the ExampleAnalyticsSimulator load – the parameter is not used so we put 0 as a placeholder).

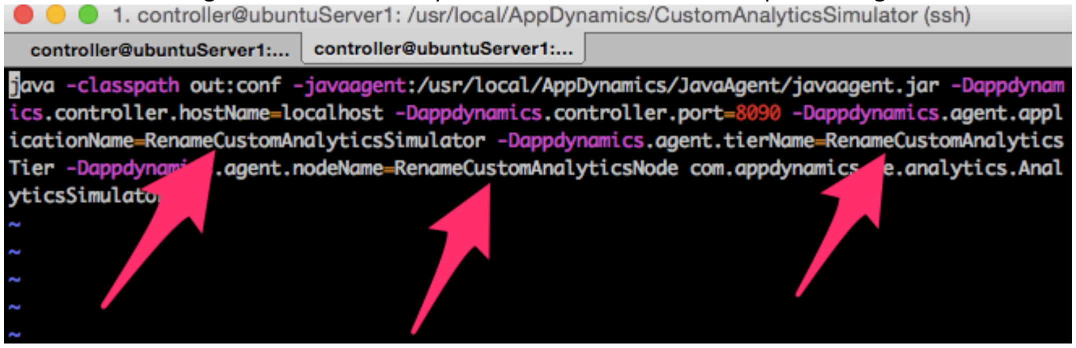
10. ***Integer Parameter 4****:* Any Integer value that represents a parameter (0 in the ExampleAnalyticsSimulator load – the parameter is not used so we put 0 as a placeholder).
11. ***Integer Parameter 5****:* Any Integer value that represents a parameter (0 in the ExampleAnalyticsSimulator load – the parameter is not used so we put 0 as a placeholder).
12. ***User Experience Type****:* Normal, Very Slow or Error
13. ***Error Text****:* Any String that will be displayed as a real error in the search details (this value will only show up for lines with *User Experience = Error*)

4) Once your load.txt file has been edited, save and /or transfer the file back to the */usr/local/AppDynamics/CustomAnalyticsSimulator/conf* directory.

CONFIGURE AND RUN CUSTOM ANALYTICS SIMULATOR

In this section you will configure and run the Custom Analytics Simulator with the data created in the previous section. The goal is to use your custom load.txt file and application name configurations to create a set up compelling stories and use cases. You can then use this data to develop custom dashboards and present to customers during POV wrap-ups or stand alone Analytics presentations.

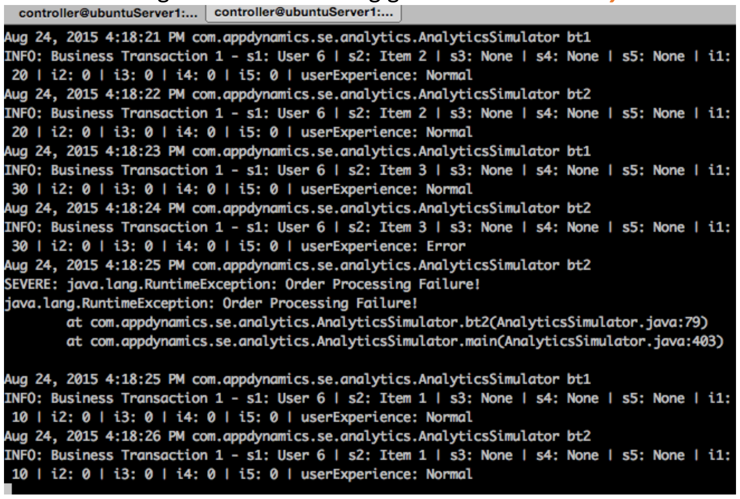
1) Navigate to /usr/local/AppDynamics/CustomAnalyticsSimulator
2) Open AnalyticsSimulator.sh using your favorite editor and update the Application, Node, and Tier values. Give them meaningful names based on your scenario. Use sudo when performing this action.



1. controller@ubuntuServer1: /usr/local/AppDynamics/CustomAnalyticsSimulator (ssh)
controller@ubuntuServer1:...
controller@ubuntuServer1:...

```
java -classpath out:conf -javaagent:/usr/local/AppDynamics/JavaAgent/javaagent.jar -Dappdynam
ics.controller.hostName=localhost -Dappdynamics.controller.port=8090 -Dappdynamics.agent.appl
icationName=RenameCustomAnalyticsSimulator -Dappdynamics.agent.tierName=RenameCustomAnalytics
Tier -Dappdynam    .agent.nodeName=RenameCustomAnalyticsNode com.appdynamics  e.analytics.Anal
yticsSimulato
~
~
~
~
~
```

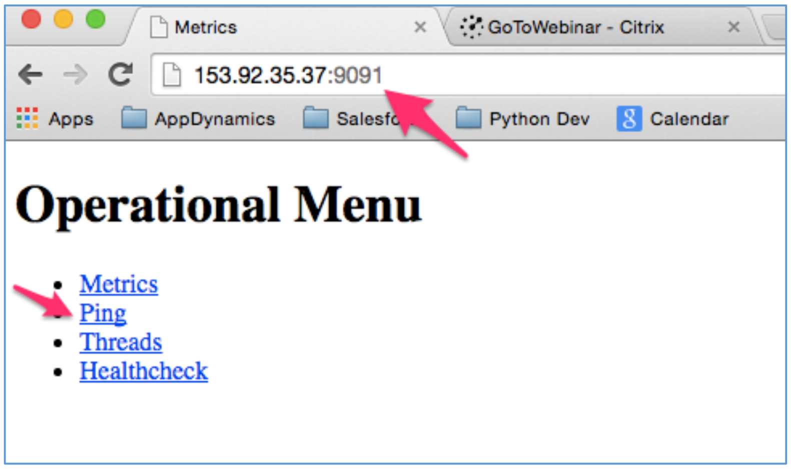
3) Execute AnalyticsSimulator using: sudo sh AnalyticsSimulator.sh
   a. You should begin to see load being generated based on ***your custom load.txt*** file



```
Aug 24, 2015 4:18:21 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 6 | s2: Item 2 | s3: None | s4: None | s5: None | i1:
 20 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 4:18:22 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
INFO: Business Transaction 1 - s1: User 6 | s2: Item 2 | s3: None | s4: None | s5: None | i1:
 20 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 4:18:23 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 6 | s2: Item 3 | s3: None | s4: None | s5: None | i1:
 30 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 4:18:24 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
INFO: Business Transaction 1 - s1: User 6 | s2: Item 3 | s3: None | s4: None | s5: None | i1:
 30 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Error
Aug 24, 2015 4:18:25 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
SEVERE: java.lang.RuntimeException: Order Processing Failure!
java.lang.RuntimeException: Order Processing Failure!
        at com.appdynamics.se.analytics.AnalyticsSimulator.bt2(AnalyticsSimulator.java:79)
        at com.appdynamics.se.analytics.AnalyticsSimulator.main(AnalyticsSimulator.java:403)

Aug 24, 2015 4:18:25 PM com.appdynamics.se.analytics.AnalyticsSimulator bt1
INFO: Business Transaction 1 - s1: User 6 | s2: Item 1 | s3: None | s4: None | s5: None | i1:
 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
Aug 24, 2015 4:18:26 PM com.appdynamics.se.analytics.AnalyticsSimulator bt2
INFO: Business Transaction 1 - s1: User 6 | s2: Item 1 | s3: None | s4: None | s5: None | i1:
 10 | i2: 0 | i3: 0 | i4: 0 | i5: 0 | userExperience: Normal
```

4) Configure the Machine Agent with new Application, Node, and Tier values.
   a. Navigate to /usr/local/AppDynamics/MachineAgent/conf and open controller-info.xml for editing using sudo
   b. Make certain Application, Node, and Tier names match what was entered for the Java Agent in step 2 above
5) Start Machine Agent
   a. Navigate to /usr/local/AppDynamics/MachineAgent/bin
   b. Execute *sudo ./machine-agent* to start the Machine Agent

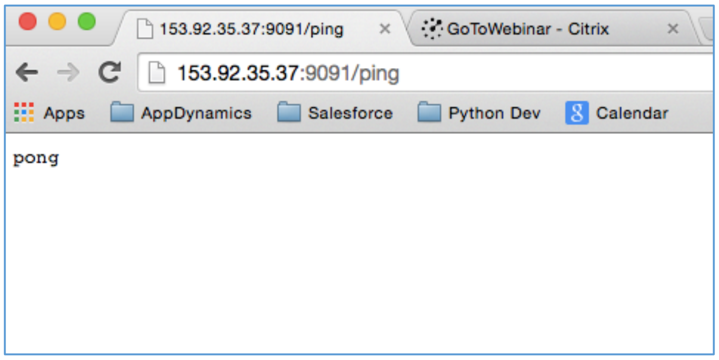
6) Verify the Machine Agent has started successfully by opening a browser and navigating to: http://[yourIP]:9091 and choose Ping from the available options.



Operational Menu

- Metrics
- Ping
- Threads
- Healthcheck

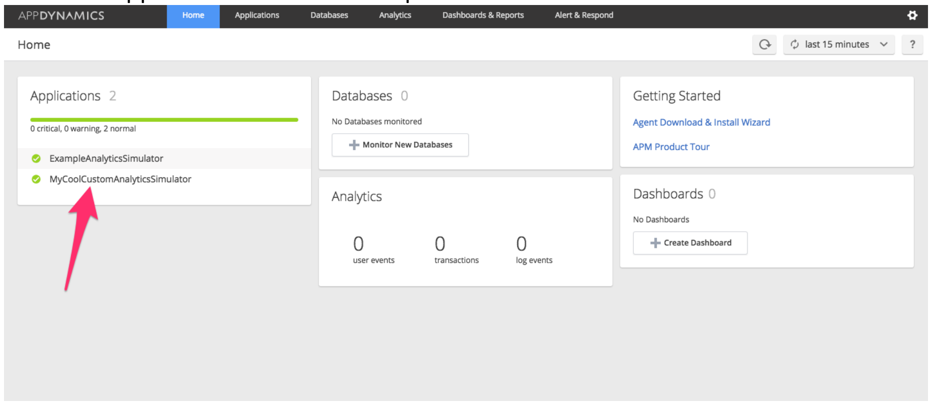
7) Receive Pong to verify Machine Agent is running successfully.



```
pong
```

8) Open a browser and log into the Controller UI using http://[yourIP]:8090/controller

   a. Your new application should now be present



Applications 2

0 critical, 0 warning, 2 normal

- ExampleAnalyticsSimulator
- MyCoolCustomAnalyticsSimulator

Databases 0

No Databases monitored

Getting Started

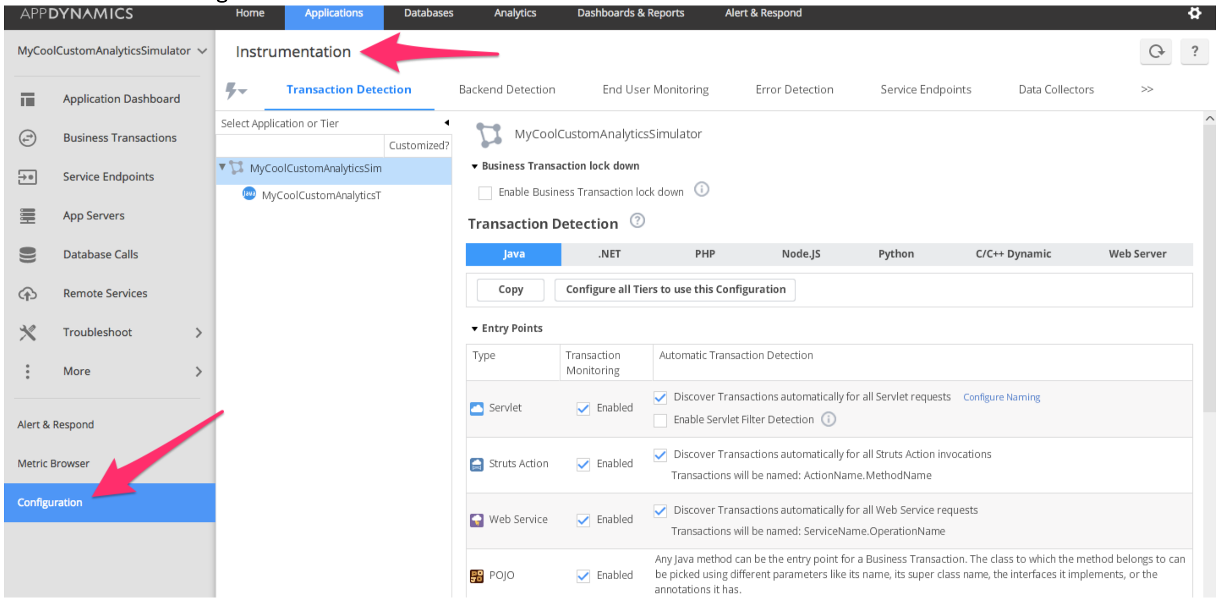
Analytics

0	0	0
user events	transactions	log events

Dashboards 0

No Dashboards

9) Select your new application and choose Configuration from the bottom left of the screen, then choose Instrumentation to get to the Transaction Detection screen.



MyCoolCustomAnalyticsSimulator

Business Transaction lock down

☐ Enable Business Transaction lock down

Transaction Detection

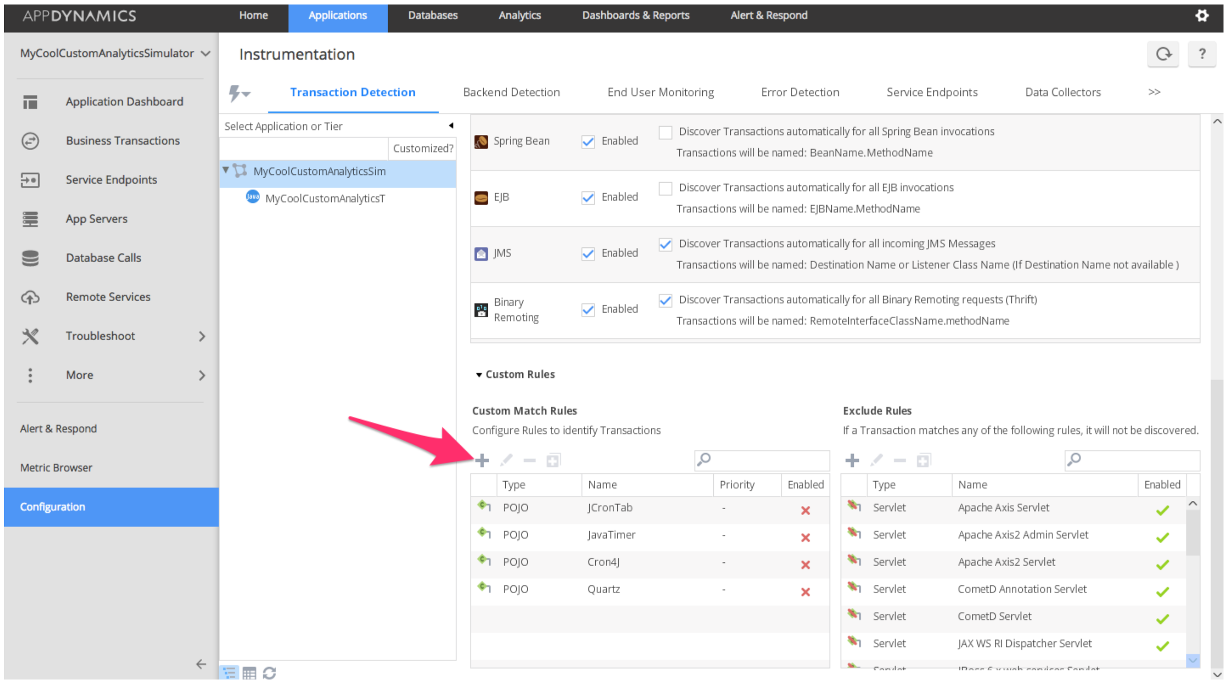
Java | .NET | PHP | Node.JS | Python | C/C++ Dynamic | Web Server

Copy | Configure all Tiers to use this Configuration

Entry Points

Type	Transaction Monitoring	Automatic Transaction Detection
Servlet	✓ Enabled	✓ Discover Transactions automatically for all Servlet requests Configure Naming ☐ Enable Servlet Filter Detection
Struts Action	✓ Enabled	✓ Discover Transactions automatically for all Struts Action invocations Transactions will be named: ActionName.MethodName
Web Service	✓ Enabled	✓ Discover Transactions automatically for all Web Service requests Transactions will be named: ServiceName.OperationName
POJO	✓ Enabled	Any Java method can be the entry point for a Business Transaction. The class to which the method belongs to can be picked using different parameters like its name, its super class name, the interfaces it implements, or the annotations it has.

10) Scroll to the bottom of the screen and select the + icon to add a new Custom Match Rule



Spring Bean	✓ Enabled	☐ Discover Transactions automatically for all Spring Bean invocations Transactions will be named: BeanName.MethodName
EJB	✓ Enabled	☐ Discover Transactions automatically for all EJB invocations Transactions will be named: EJBName.MethodName
JMS	✓ Enabled	✓ Discover Transactions automatically for all incoming JMS Messages Transactions will be named: Destination Name or Listener Class Name (If Destination Name not available)
Binary Remoting	✓ Enabled	✓ Discover Transactions automatically for all Binary Remoting requests (Thrift) Transactions will be named: RemoteInterfaceClassName.methodName

Custom Rules

Custom Match Rules

Configure Rules to identify Transactions

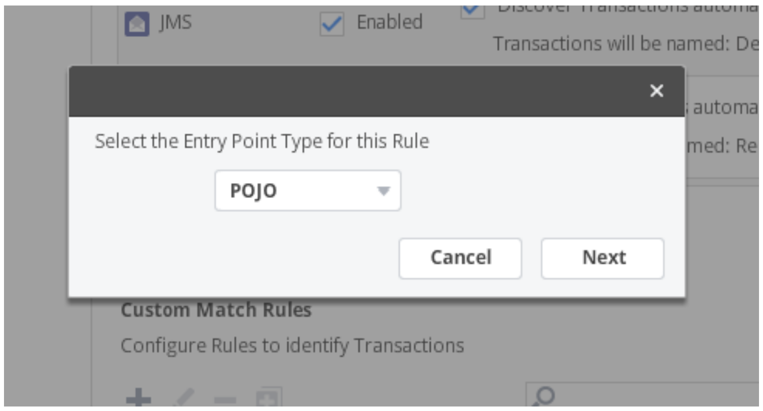
Type	Name	Priority	Enabled
POJO	JCronTab	-	✗
POJO	JavaTimer	-	✗
POJO	Cron4J	-	✗
POJO	Quartz	-	✗

Exclude Rules

If a Transaction matches any of the following rules, it will not be discovered.

Type	Name	Enabled
Servlet	Apache Axis Servlet	✓
Servlet	Apache Axis2 Admin Servlet	✓
Servlet	Apache Axis2 Servlet	✓
Servlet	CometD Annotation Servlet	✓
Servlet	CometD Servlet	✓
Servlet	JAX WS RI Dispatcher Servlet	✓

11) Select POJO as the Entry Point Type and click Next

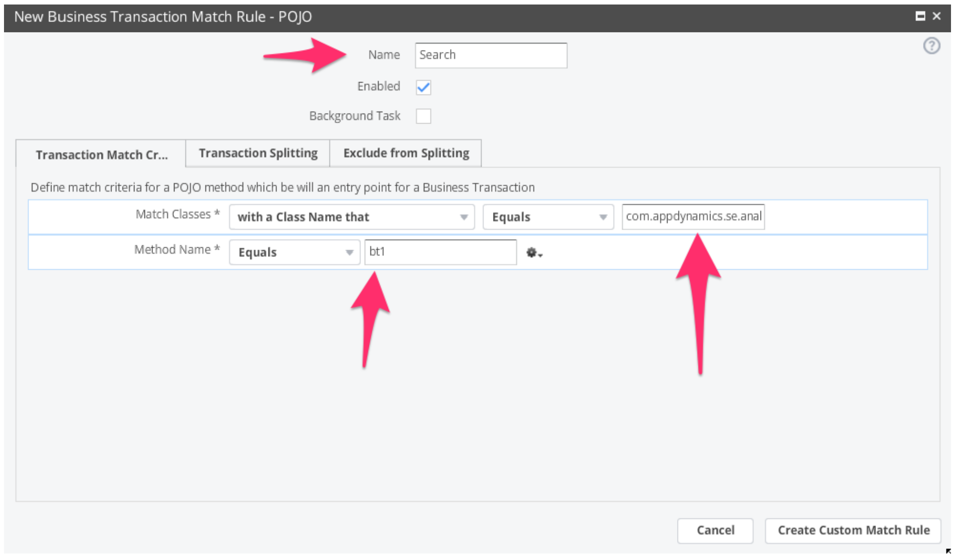

JMS
Enabled
Transactions will be named: De
Custom Match Rules
Configure Rules to identify Transactions
Cancel
Next

Select the Entry Point Type for this Rule

POJO

12) The new business transaction match rule dialog appears.

a. Give the New Business Transaction and meaningful name
b. Set Match Classes to a Class Name that **Equals** *com.appdynamics.se.analytics.AnalyticsSimulator*
c. Set Method Name to **Equals** bt1
d. Repeat steps 10-12 for each additional transaction replacing bt1 with bt# up to 10



Name: Search

Enabled: ✓

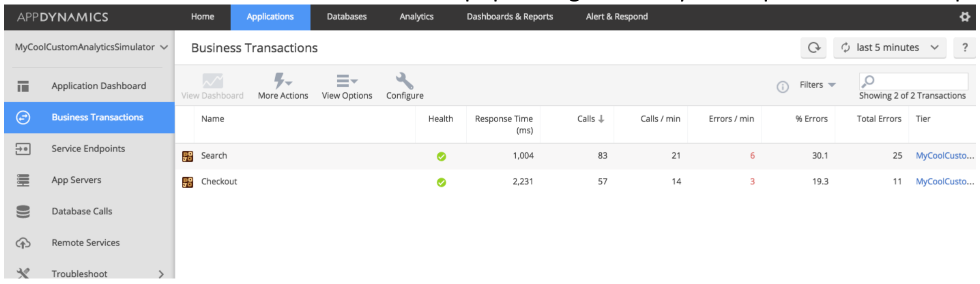
Background Task: ☐

Define match criteria for a POJO method which be will an entry point for a Business Transaction

Match Classes *: with a Class Name that | Equals | com.appdynamics.se.anal

Method Name *: Equals | bt1

13) Return to the Application Flow Map and review the transaction data and newly created flow map. Your new Business Transactions should now be populating. This may take up to 5 minutes to populate.



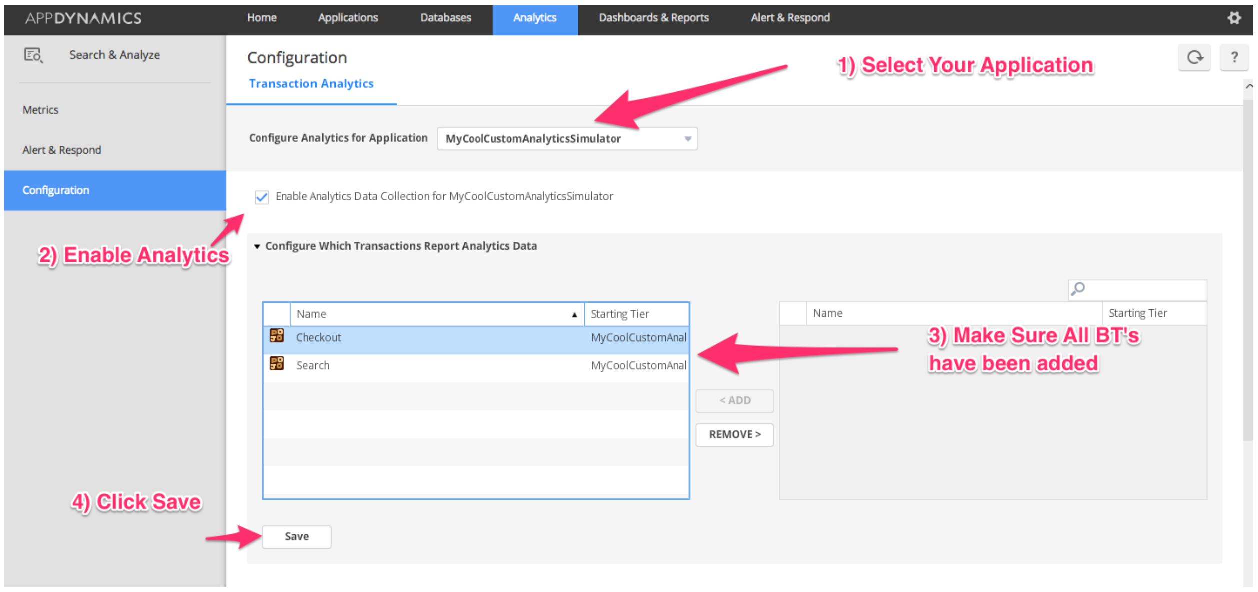
Business Transactions

Showing 2 of 2 Transactions

Name	Health	Response Time (ms)	Calls	Calls / min	Errors / min	% Errors	Total Errors	Tier
Search	✓	1,004	83	21	6	30.1	25	MyCoolCusto...
Checkout	✓	2,231	57	14	3	19.3	11	MyCoolCusto...

14) Navigate to the Analytics Tab and select Configuration from the left side menu.

15) Select your application from the dropdown menu available on the Transaction Analytics Pane and check Enable Analytics Data Collection. Ensure all of your custom Business Transactions have been added and click save.



Configure Analytics for Application: MyCoolCustomAnalyticsSimulator

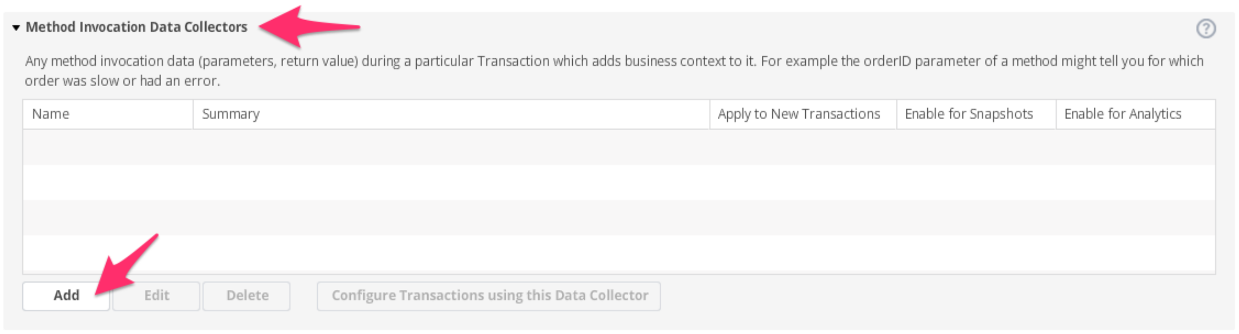
☑ Enable Analytics Data Collection for MyCoolCustomAnalyticsSimulator

Configure Which Transactions Report Analytics Data

Name	Starting Tier
Checkout	MyCoolCustomAnal
Search	MyCoolCustomAnal

Name	Starting Tier

16) Scroll down and click Add to create Method Invocation Data Collectors for each of your Business Transactions.



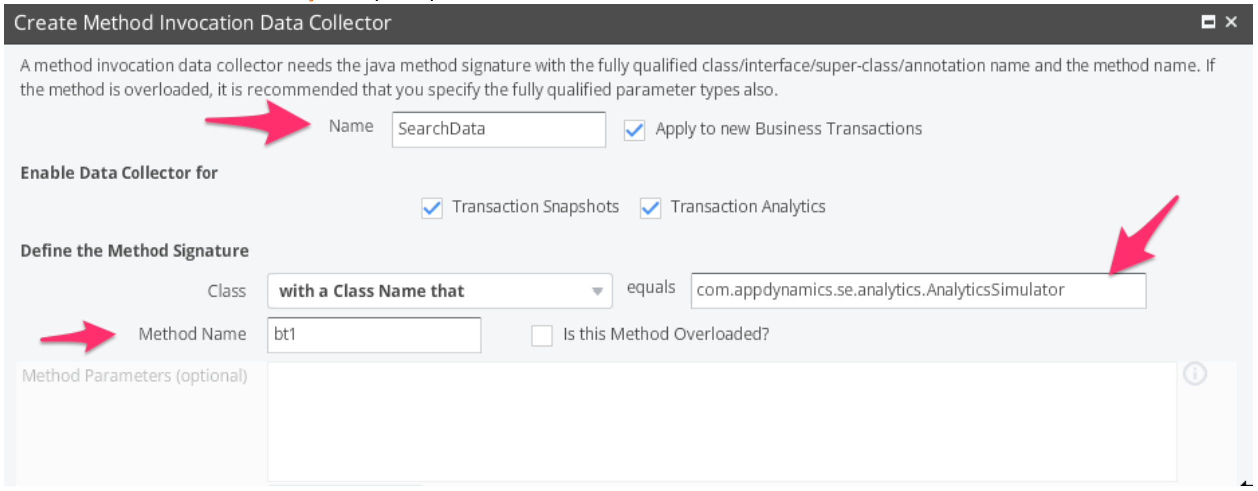
Method Invocation Data Collectors

Any method invocation data (parameters, return value) during a particular Transaction which adds business context to it. For example the orderID parameter of a method might tell you for which order was slow or had an error.

Name	Summary	Apply to New Transactions	Enable for Snapshots	Enable for Analytics

17) For each Method Invocation Data Collector fill in the following fields:

  a. ***Name*** (BT Name + "Data")
  b. ***Class Name*** (*com.appdynamics.se.analytics.AnalyticsSimulator)*
  c. ***Method Name*** (same method as configured for the Business Transaction ex. bt1, bt2, etc.)
  d. ***Apply to new Business Transactions*** (True)
  e. ***Transaction Snapshots*** (True)
  f. ***Transaction Analytics*** (True)



Create Method Invocation Data Collector

A method invocation data collector needs the java method signature with the fully qualified class/interface/super-class/annotation name and the method name. If the method is overloaded, it is recommended that you specify the fully qualified parameter types also.

Name: SearchData ✓ Apply to new Business Transactions

Enable Data Collector for

✓ Transaction Snapshots ✓ Transaction Analytics

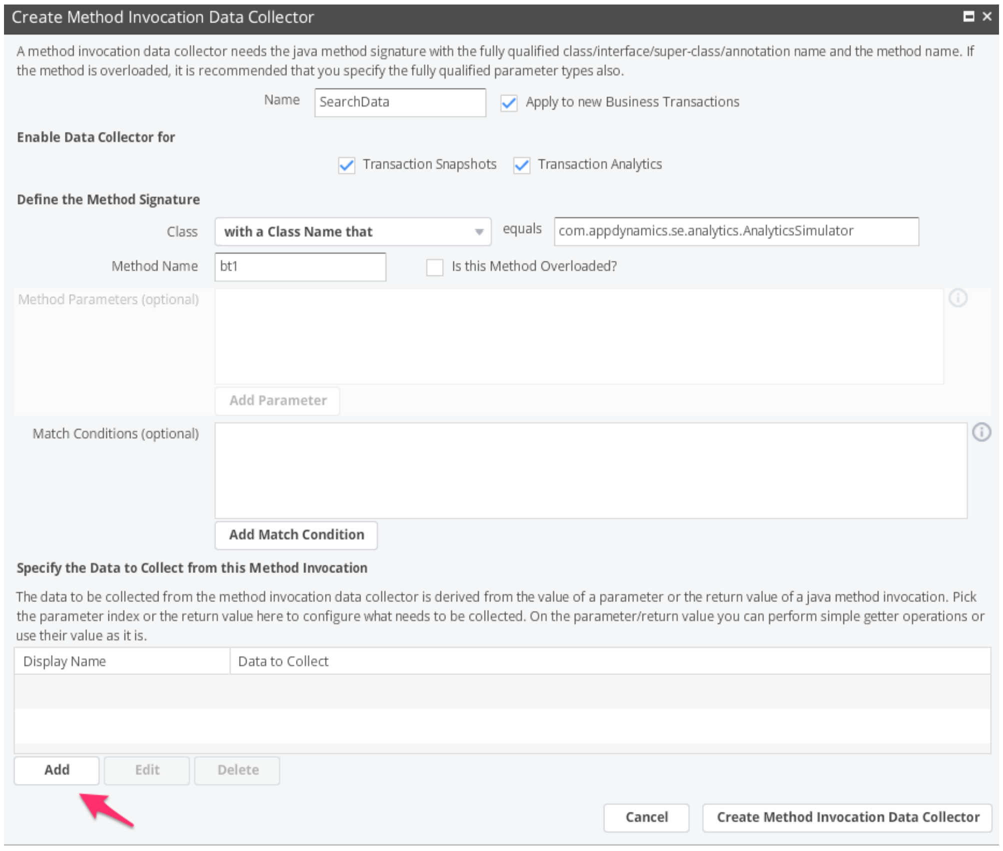
Define the Method Signature

Class: with a Class Name that — equals: com.appdynamics.se.analytics.AnalyticsSimulator

Method Name: bt1 ☐ Is this Method Overloaded?

Method Parameters (optional)

18) Scroll down to specify the data to be collected from this method. Click Add.



Create Method Invocation Data Collector

A method invocation data collector needs the java method signature with the fully qualified class/interface/super-class/annotation name and the method name. If the method is overloaded, it is recommended that you specify the fully qualified parameter types also.

Name: SearchData ✓ Apply to new Business Transactions

Enable Data Collector for

✓ Transaction Snapshots ✓ Transaction Analytics

Define the Method Signature

Class: with a Class Name that — equals: com.appdynamics.se.analytics.AnalyticsSimulator

Method Name: bt1 ☐ Is this Method Overloaded?

Method Parameters (optional)

Add Parameter

Match Conditions (optional)

Add Match Condition

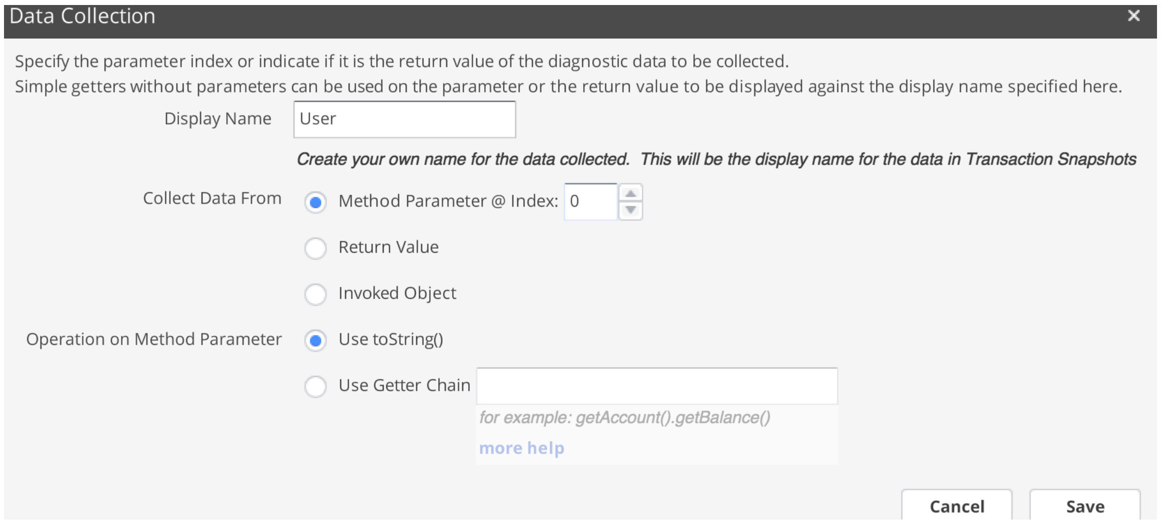
Specify the Data to Collect from this Method Invocation

The data to be collected from the method invocation data collector is derived from the value of a parameter or the return value of a java method invocation. Pick the parameter index or the return value here to configure what needs to be collected. On the parameter/return value you can perform simple getter operations or use their value as it is.

Display Name	Data to Collect

19) The Data Collection dialog appears. This is where you will specify which columns of your load.txt file will be consumed by Analytics. The Analytics Simulator Application was written with simple getter methods for each String and each Integer value available in the load.txt file created earlier in the lab.

20) Each String parameter in load.txt is represented here by an index of 0-4 and is retrieved using a simple toString() method call. Give the Data Collection a Display Name and then choose the proper index to retrieve the string value you want to use. For instance, if you put a user name in the first available string position in load.txt you would use index 0 and your Data Collection would look like the below screen capture.



Data Collection

Specify the parameter index or indicate if it is the return value of the diagnostic data to be collected.
Simple getters without parameters can be used on the parameter or the return value to be displayed against the display name specified here.

Display Name: User

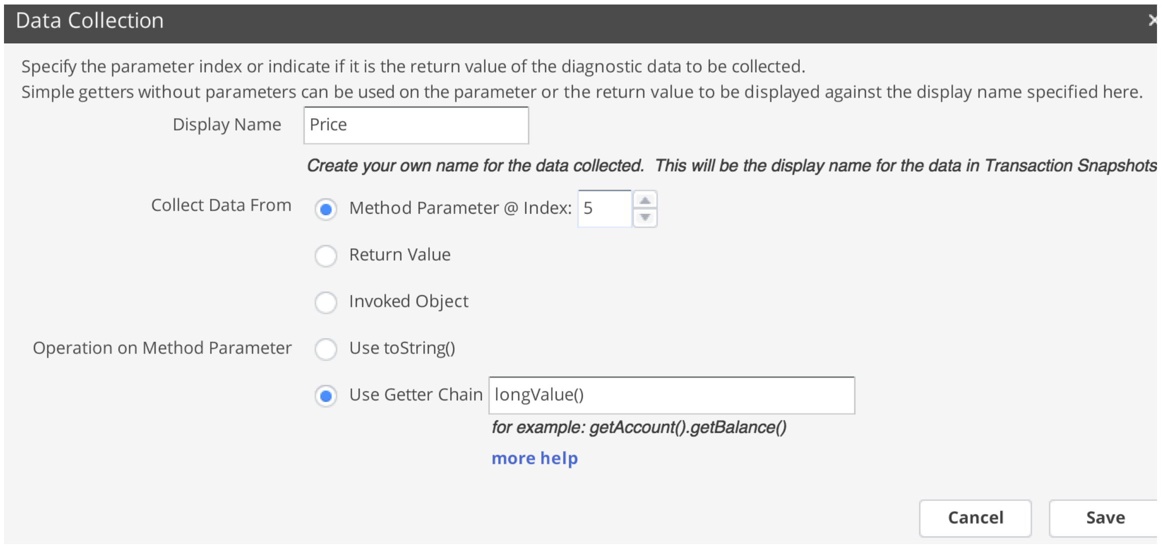
Create your own name for the data collected. This will be the display name for the data in Transaction Snapshots

Collect Data From: [✓ Method Parameter @ Index: 0] Return Value Invoked Object

Operation on Method Parameter: [✓ Use toString()] Use Getter Chain

for example: getAccount().getBalance()

21) Each Integer value in load.txt is represented here by an index of 5-9 and is retrieved using the Getter Chain longValue(). Give this Data Collection a Display Name and then choose the proper index to retrieve the integer value you want to use. For instance, if you put price in the first available integer position in load.txt you would use index 5 and your Data Collection would look like the below screen capture.



Data Collection

Specify the parameter index or indicate if it is the return value of the diagnostic data to be collected.
Simple getters without parameters can be used on the parameter or the return value to be displayed against the display name specified here.

Display Name: Price

Create your own name for the data collected. This will be the display name for the data in Transaction Snapshots

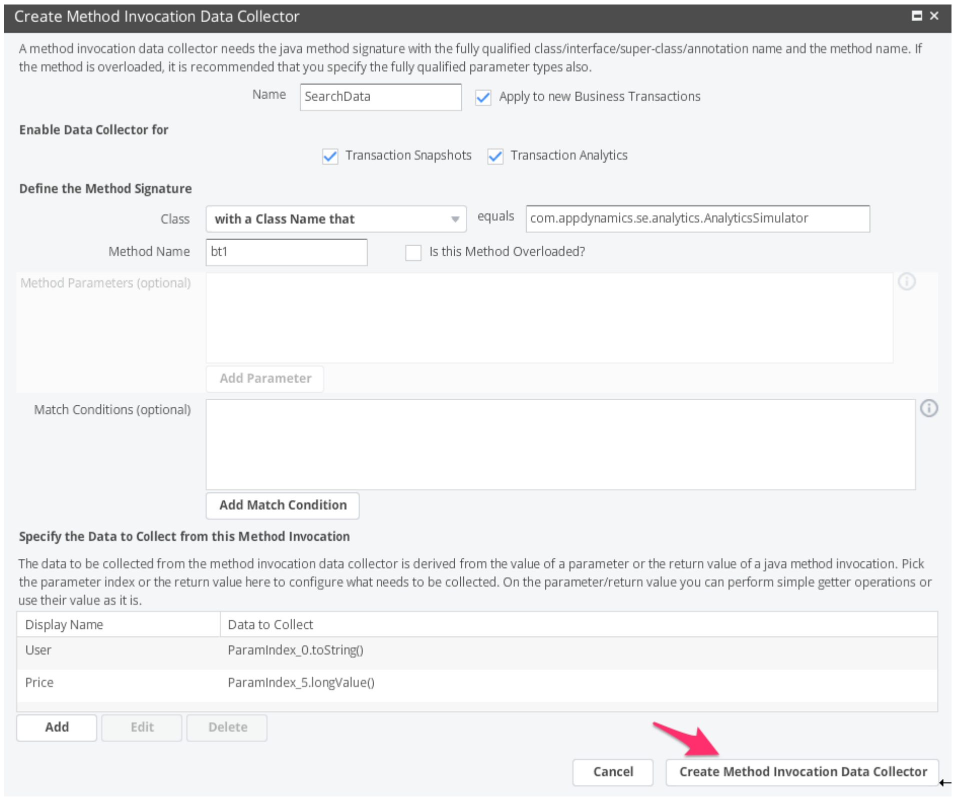
Collect Data From: [✓ Method Parameter @ Index: 5] Return Value Invoked Object

Operation on Method Parameter: Use toString() [✓ Use Getter Chain: longValue()]

for example: getAccount().getBalance()

22) Repeat steps 18 - 21 for each value you wish to capture for this Business Transaction.

23) When you have finished adding the appropriate data collectors for your Business Transaction, click Create Method Invocation Data Collector.



Create Method Invocation Data Collector

A method invocation data collector needs the java method signature with the fully qualified class/interface/super-class/annotation name and the method name. If the method is overloaded, it is recommended that you specify the fully qualified parameter types also.

Name: SearchData ✓ Apply to new Business Transactions

Enable Data Collector for

✓ Transaction Snapshots ✓ Transaction Analytics

Define the Method Signature

Class: with a Class Name that equals com.appdynamics.se.analytics.AnalyticsSimulator

Method Name: bt1 ☐ Is this Method Overloaded?

Method Parameters (optional)

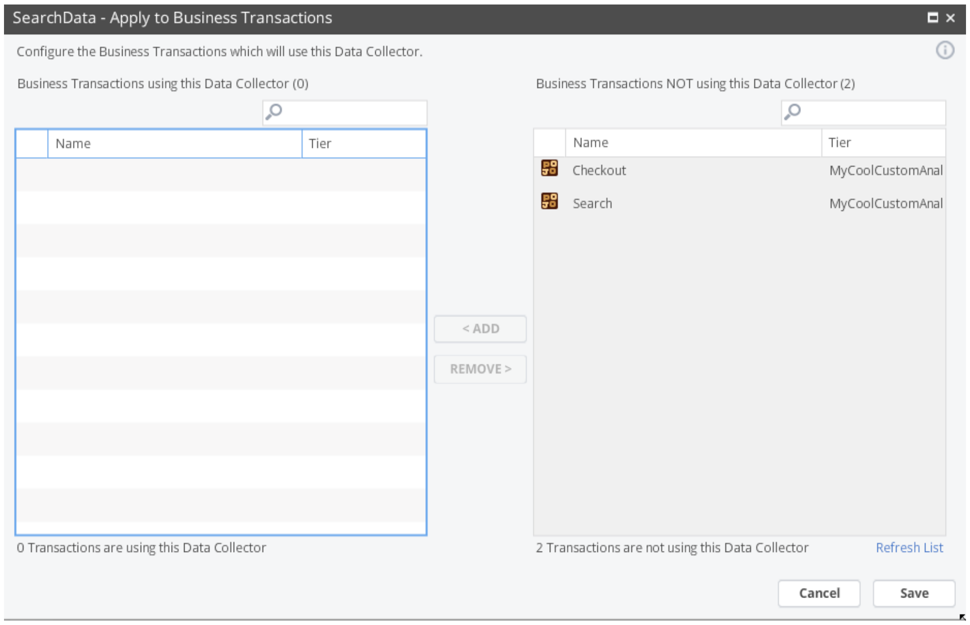
Match Conditions (optional)

Specify the Data to Collect from this Method Invocation

The data to be collected from the method invocation data collector is derived from the value of a parameter or the return value of a java method invocation. Pick the parameter index or the return value here to configure what needs to be collected. On the parameter/return value you can perform simple getter operations or use their value as it is.

Display Name	Data to Collect
User	ParamIndex_0.toString()
Price	ParamIndex_5.longValue()

24) Choose which Business Transactions your new Data Collector applies to and click Save.



SearchData - Apply to Business Transactions

Configure the Business Transactions which will use this Data Collector.

Business Transactions using this Data Collector (0)

Name	Tier

0 Transactions are using this Data Collector

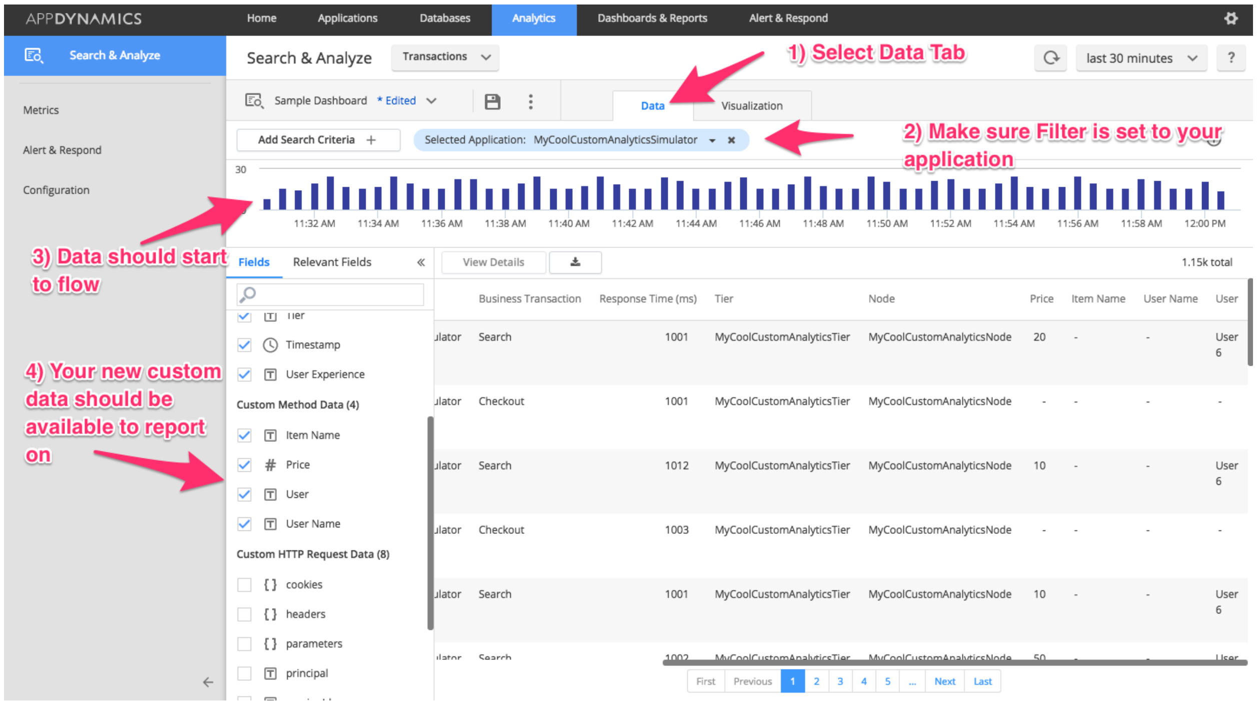
Business Transactions NOT using this Data Collector (2)

Name	Tier
Checkout	MyCoolCustomAnal
Search	MyCoolCustomAnal

2 Transactions are not using this Data Collector

25) Repeat steps 16 – 24 for each Business Transaction you wish to capture data for.

26) Return to the main Analytics screen and select the Data tab. You should now be able to see data flowing from your application. Your new custom method data points will be listed in available fields and the values for each can be seen in the transaction details.



	Business Transaction	Response Time (ms)	Tier	Node	Price	Item Name	User Name	User
ulator	Search	1001	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	20	-	-	User 6
ulator	Checkout	1001	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	-	-	-	-
ulator	Search	1012	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	10	-	-	User 6
ulator	Checkout	1003	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	-	-	-	-
ulator	Search	1001	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	10	-	-	User 6
ulator	Search	1002	MyCoolCustomAnalyticsTier	MyCoolCustomAnalyticsNode	50	-	-	User

27) You are now ready to begin building your own custom dashboards. Using your scenario and the data patterns you created, build a short customer demo and present it to the team. Your presentation should be no longer than 10 minutes and should tell a compelling story.