

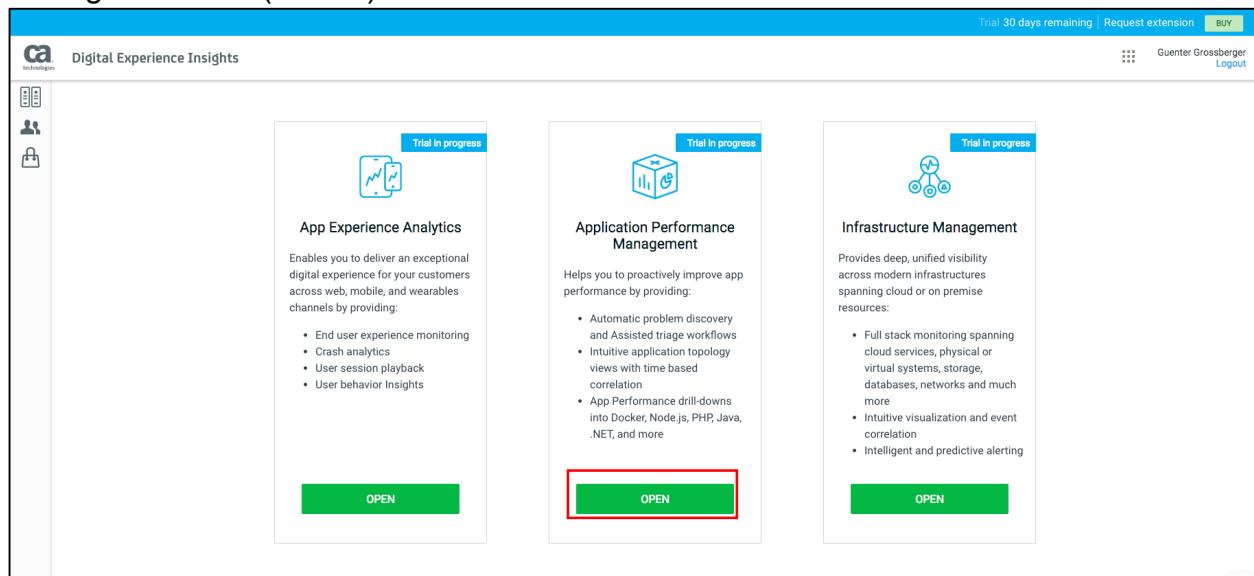
Monitor Modern Applications in the Cloud

Lab Instructions

1. Setup CA Digital Experience Insights (DXI) trial account

If you already have a CA Digital Experience Insights (DXI) trial account log in with your existing credentials (use your email address as tenant ID if asked) and skip this step.

1. Open <https://cloud.ca.com> in your browser
2. Click “Click here to register”
3. Click “1. Sign up”
4. Fill out the signup form and click “Get started”. Use your company email address. Public email providers like gmail, yahoo, ... are not supported.
5. On the DXI start page click the “Open” button in the Application Performance Management box (middle):

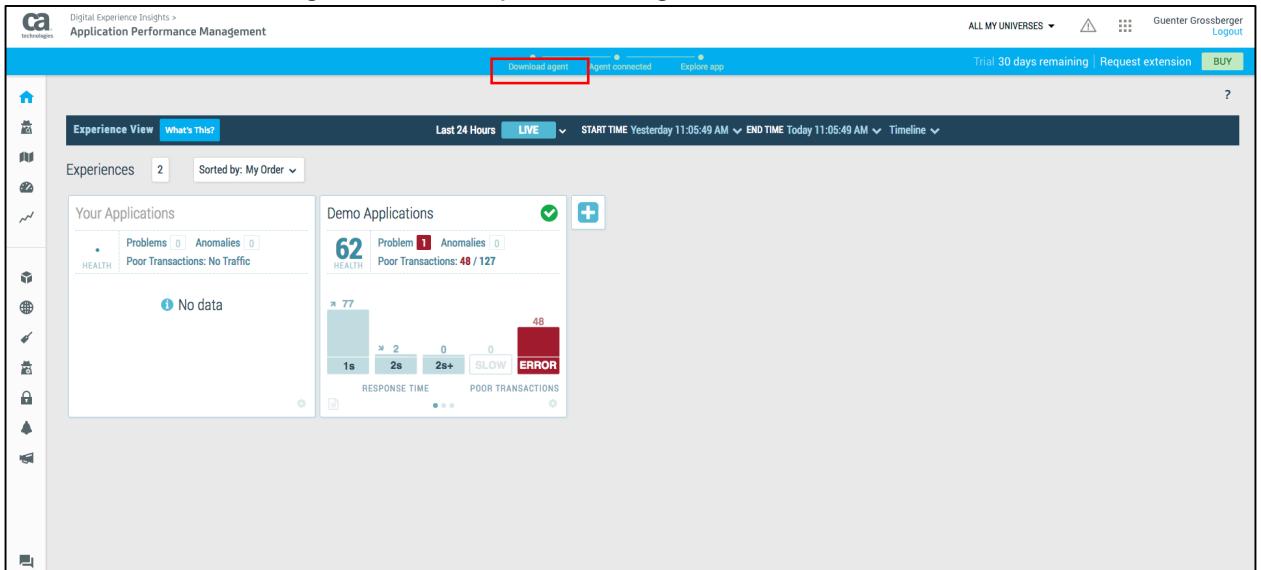


6. You see the CA APM home page (Experience View):

The screenshot shows the CA APM Experience View interface. At the top, there's a navigation bar with the CA Technologies logo, 'Digital Experience Insights > Application Performance Management', 'ALL MY UNIVERSES', 'Guenter Grossberger Logout', and a 'Trial 30 days remaining | Request extension | BUY' button. Below the navigation is a toolbar with icons for 'Download agent', 'Agent connected', 'Explore app', and a question mark. The main area has tabs for 'Experience View' (which is selected) and 'What's This?'. It shows two sections: 'Your Applications' and 'Demo Applications'. The 'Your Applications' section has a 'HEALTH' status indicator, 'Problems 0', 'Anomalies 0', and a note 'Poor Transactions: No Traffic'. The 'Demo Applications' section shows a green checkmark icon, a '62' in a blue box labeled 'HEALTH', 'Problem 1' (red), 'Anomalies 0', and 'Poor Transactions: 48 / 127'. Below these are two charts: 'RESPONSE TIME' showing bars for 1s (~77), 2s (~2), 2s+ (~0), SLOW (~0), and ERROR (~48); and 'POOR TRANSACTIONS' showing a single red bar for 'ERROR' (~48). On the left, a sidebar lists various monitoring categories with corresponding icons.

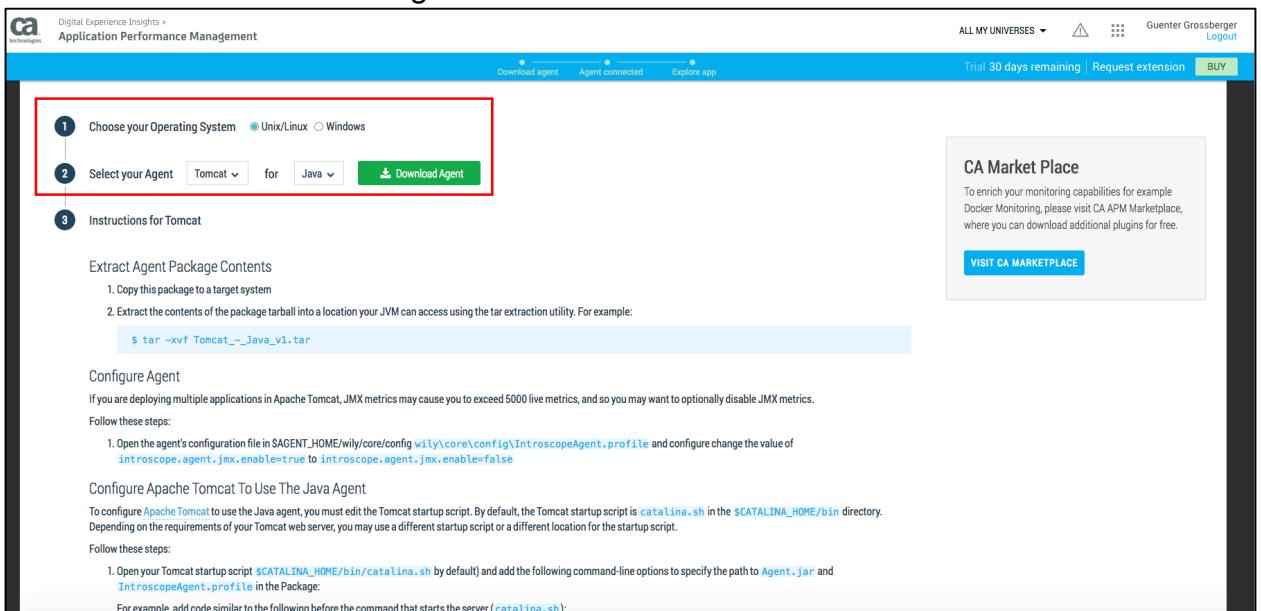
2. Download and install application agents

1. Click on “Download agent” in the top blue navigation bar:



The screenshot shows the CA Digital Experience Insights Application Performance Management interface. At the top, there's a blue navigation bar with the CA logo, "Digital Experience Insights > Application Performance Management", and user information like "ALL MY UNIVERSES", "Guenther Grossberger", and "Logout". In the center of the page, there's a dashboard titled "Experience View" with tabs for "What's This?", "LIVE", and "Timeline". Below the dashboard, there are sections for "Your Applications" and "Demo Applications". A prominent red box highlights the "Download agent" button in the top right corner of the main content area.

2. Download Java agent: choose operating system “Unix/Linux” and “Tomcat” for “Java”. Click on “Download agent” and save to local disk.



The screenshot shows the "CA Market Place" download agent configuration page. It has three numbered steps: 1. "Choose your Operating System" (radio buttons for Unix/Linux and Windows), 2. "Select your Agent" (dropdowns for Tomcat and Java, and a "Download Agent" button), and 3. "Instructions for Tomcat". The "Tomcat" and "Java" dropdowns are selected. To the right, there's a sidebar titled "CA Market Place" with a "VISIT CA MARKETPLACE" button. The main content area also includes instructions for extracting agent package contents and configuring the agent for Tomcat.

Extract Agent Package Contents

1. Copy this package to a target system
2. Extract the contents of the package tarball into a location your JVM can access using the tar extraction utility. For example:
`$ tar -xvf Tomcat_--_Java_v1.tar`

Configure Agent

If you are deploying multiple applications in Apache Tomcat, JMX metrics may cause you to exceed 5000 live metrics, and so you may want to optionally disable JMX metrics.

Follow these steps:

1. Open the agent's configuration file in \$AGENT_HOME/wily/core/config wily\core\config\IntroscopeAgent.profile and configure change the value of `introscope.agent.jmx.enable=true` to `introscope.agent.jmx.enable=false`

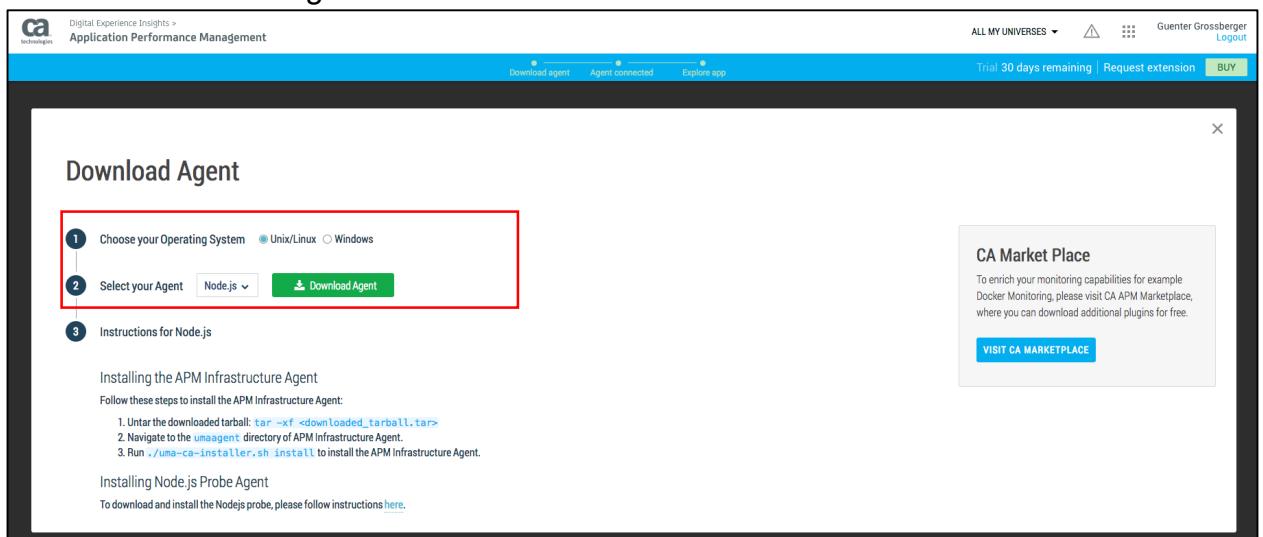
Configure Apache Tomcat To Use The Java Agent

To configure Apache Tomcat to use the Java agent, you must edit the Tomcat startup script. By default, the Tomcat startup script is `catalina.sh` in the `$CATALINA_HOME/bin` directory. Depending on the requirements of your Tomcat web server, you may use a different startup script or a different location for the startup script.

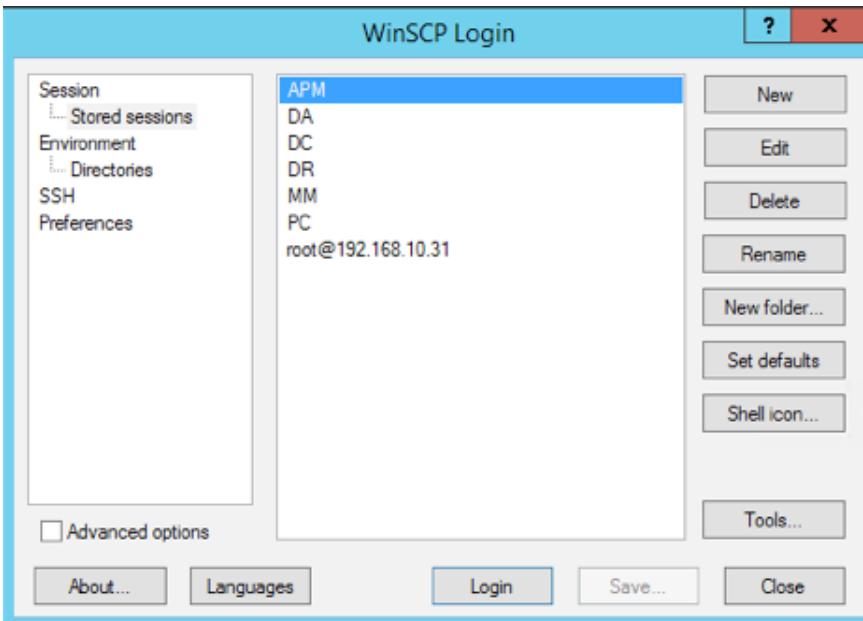
Follow these steps:

1. Open your Tomcat startup script `$CATALINA_HOME/bin/catalina.sh` by default and add the following command-line options to specify the path to `Agent.jar` and `IntroscopeAgent.profile` in the Package:
For example, add code similar to the following before the command that starts the server (`catalina.sh`):

3. Download Node.js agent: choose operating system “Unix/Linux” and “Node.js”. Click on “Download agent” and save to local disk.



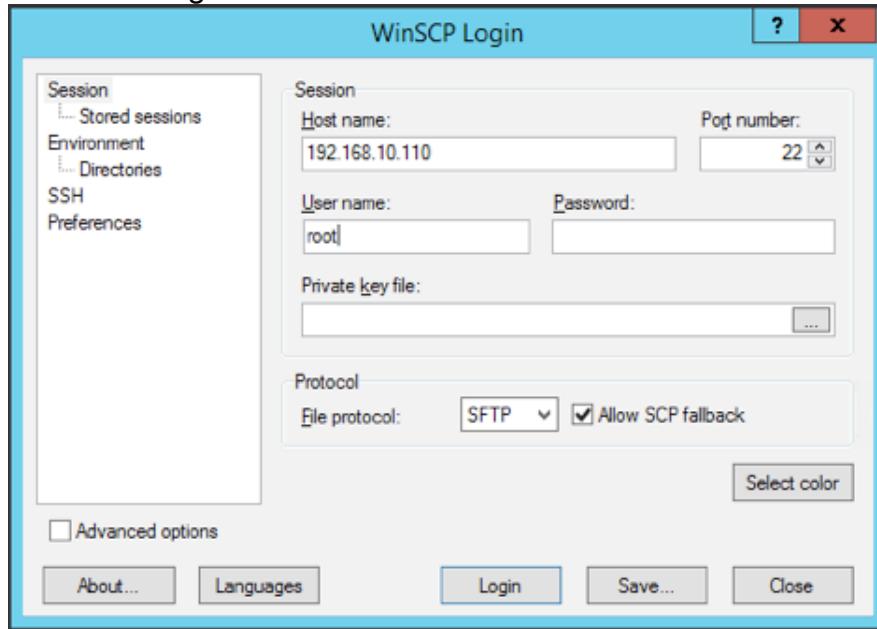
4. Copy agents to Linux VM: open the application “WinSCP” from the task bar and click “New”



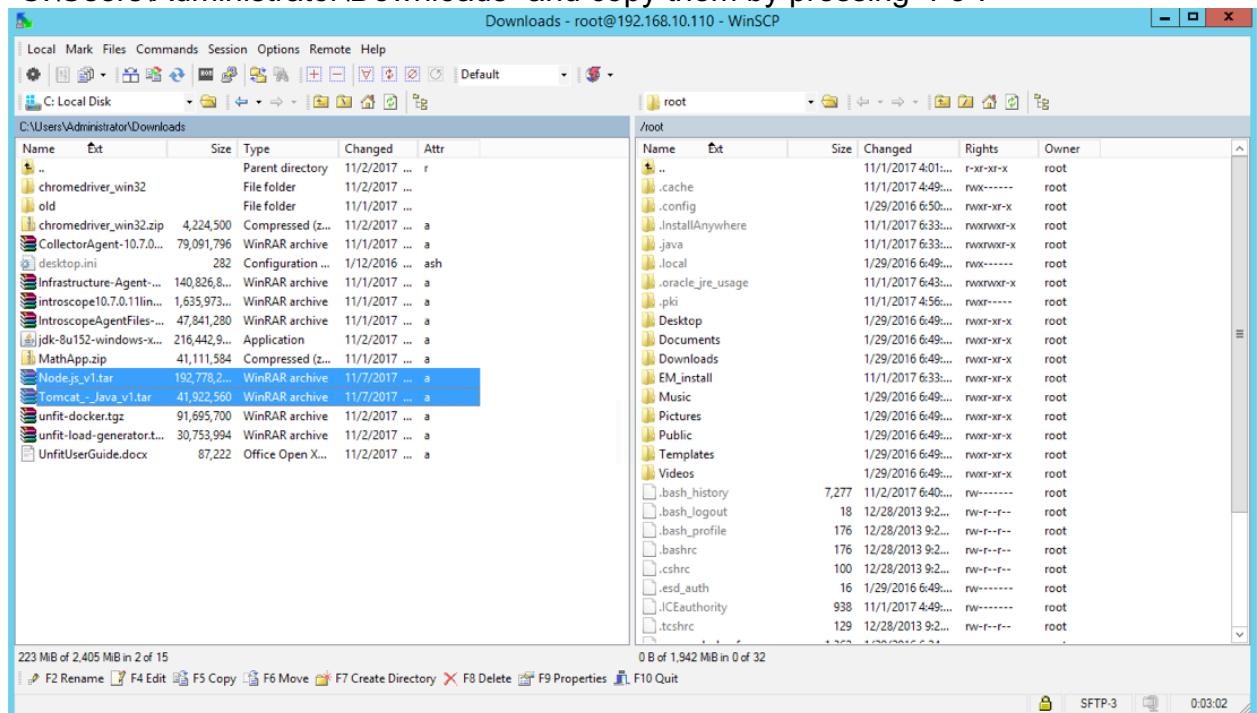
5. Enter:

- host name “192.168.10.110”
- user name “root”
- password “CAdemo123”

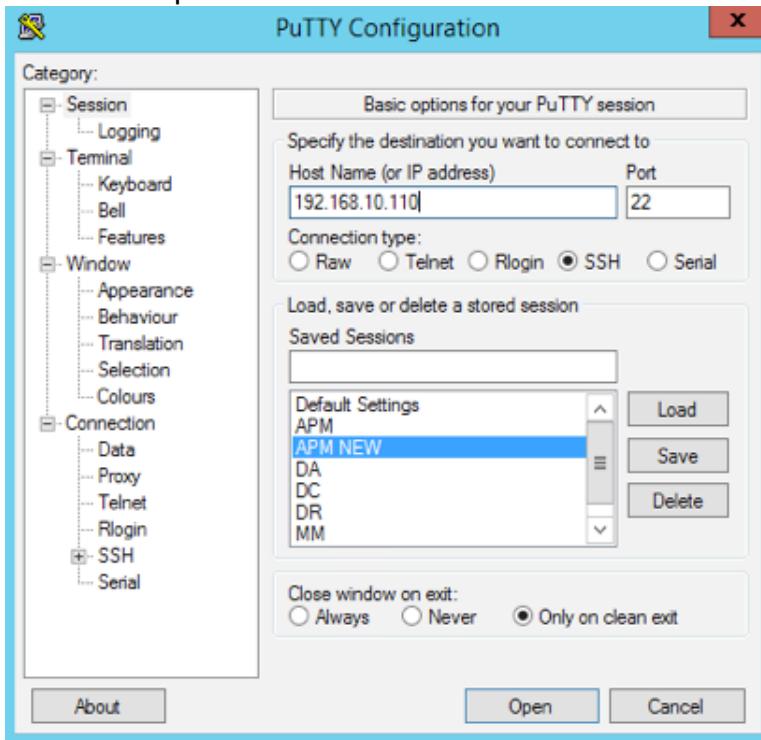
and click “Login”



6. Select the files “Node.js_v1.tar” and “Tomcat_-_Java_v1.tar” from the directory “C:\Users\Administrator\Downloads” and copy them by pressing “F5”.



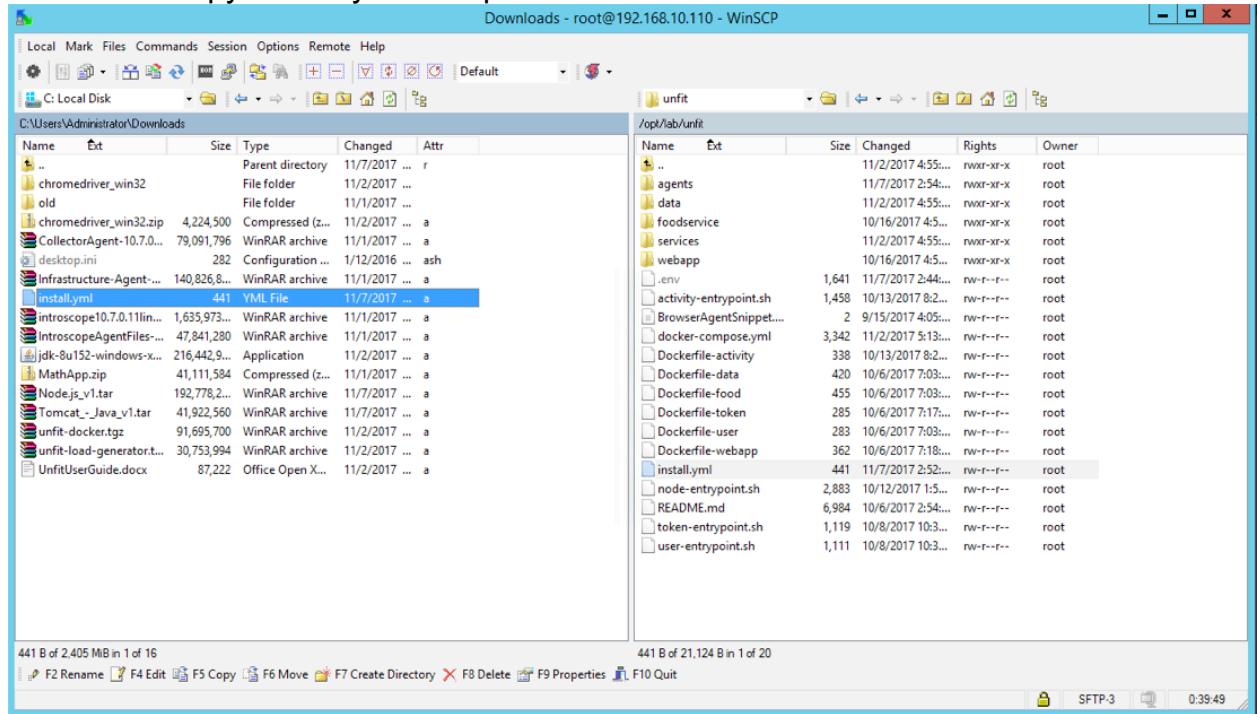
7. Open the application “Putty” from the task bar, enter host name “192.168.10.110” and click “Open”:



8. Login as user “root” with password “CAdemo123”
9. Type “cd /opt/lab/unfit”
10. Type “cp ~/Tomcat_-_Java_v1.tar agents”
11. Type “cp ~/Node.js_v1.tar agents”
12. Open the file “.env” in text editor: “vi .env”
13. Change the following agent properties and comment out the two “EM” properties:
- JAVA_AGENT=Tomcat_-_Java_v1.tar
 - NODE_AGENT=Node.js_v1.tar
 - #EM_HOST=192.168.10.110
 - #EM_PORT=5001
14. Make sure docker is running by typing “systemctl start docker”
15. Type “docker-compose build” to build the unfit application with the downloaded agents

3. Download and install Docker agent

1. In WinSCP copy “install.yml” to /opt/lab/unit”:



2. In CA DXI in your browser select “Settings/Agents”:

The screenshot shows the CA DXI interface. On the left, there is a navigation sidebar with links like 'Experience View', 'Agents View', 'Map', 'Dashboard', 'Metric View', 'SETTINGS', 'Perspectives', 'Universes', 'Attributes', and 'Agents'. The 'Agents' link is highlighted with a red box. The main content area displays a dashboard for 'Demo Applications'. It shows a summary card with '63' applications, '1 Problem', '1 Anomalies', and 'Poor Transactions: 48 / 128'. Below this are cards for 'data' (with 78 items), '1s' (1 item), '2s' (2 items), '2s+' (0 items), 'SLOW' (0 items), and 'ERROR' (48 items). The top of the page includes a header with the CA logo, 'Digital Experience Insights > Application Performance Management', user information for 'Guenter Grossberger', and a 'Logout' button.

3. Click on “Show Agent Connection Details”:

The screenshot shows the CA Application Performance Management interface. At the top, there are navigation links like 'Digital Experience Insights > Application Performance Management', 'ALL MY UNIVERSES', 'Logout', and a trial status 'Trial 30 days remaining | Request extension BUY'. Below the header is a toolbar with icons for 'Download Agent', 'Trace All Agents', and 'Show Agent Connection Details'. The main area is a table titled 'Agent Connection Details' with columns: Agent name, Host, Process URL, Applications, State, and Trace Agent. Two rows are visible: 'CA APM Demo Agent - Tomcat' and 'Logstash-APM-Plugin'. The 'Show Agent Connection Details' button is highlighted with a red box.

Leave that window open:

The screenshot shows a modal window titled 'Agent Connection Details'. It contains two fields: 'Manager URL' with the value 'https://464139.apm.cloud.ca.com:443' and 'Manager Token' with the value '85443878-56a1-427a-85c2-958f78adbbc2'. A green 'Close' button is at the bottom right.

4. Insert **your** values into install.yml with “vi install.yml”:

version: "3"

services:

dockermanager:

image : bhaab01/dockermanager:latest

environment:

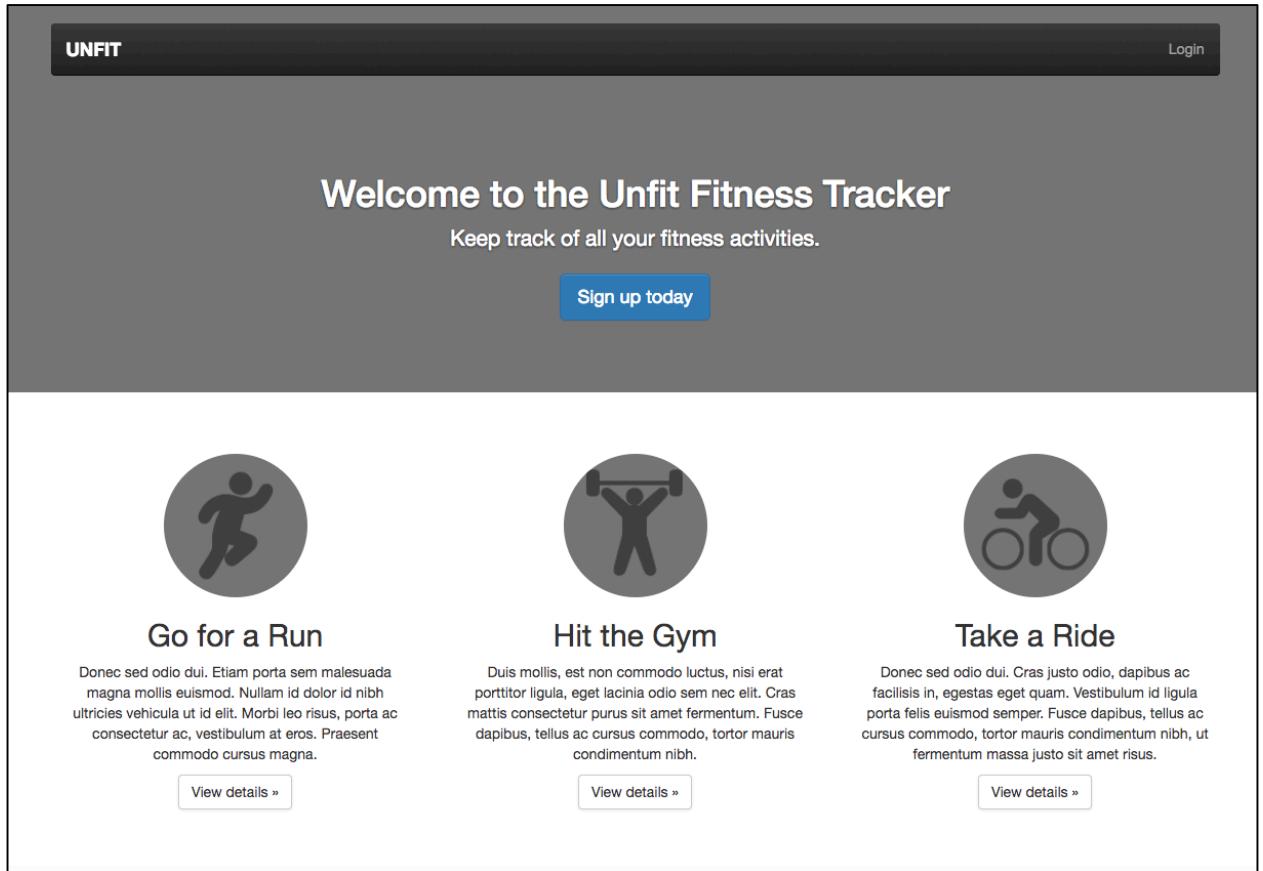
```
- agentManager_url_1=https://464139.apm.cloud.ca.com:443
- agentManager_credential=85443878-56a1-427a-85c2-958f78adbbc2
- containerflow=enabled
[...]
```

5. Start Docker Monitor with “docker-compose -f install.yml up –d
6. Verify that Docker Monitor is running by typing “docker ps”. You should see output like the following with **two containers running** from the image “dockermanager”, the Docker Monitor and the ContainerFlow agent:

```
docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
af5bb549ee0e bhaab01/dockermanager:latest "bash run.sh" 2 minutes ago Up 2 minutes dockermanager
ea9375cbd3a0 bhaab01/dockermanager:latest "sh install.sh" 2 minutes ago Up 2 minutes unfit_dockermanager_1
```

4. Start unfit application

1. Start unfit application with “docker-compose up –d”
2. Open application in browser via shortcut or <http://192.168.10.110:8000/>



3. Open a windows command prompt from the taskbar
4. Type “cd \apps\unfit-load-generator”
5. Start load generation: type “baseLoad.bat”. Two Chrome windows will open and run load against the unfit application

Section 2: Use CA APM to quickly find the root cause of a performance issue. The Assisted Triage engine looks at application issues, the transactions they affect, and the underlying infrastructure. Utilize CA Docker and infrastructure monitoring for additional insights into the underlying architecture.

Problem Description

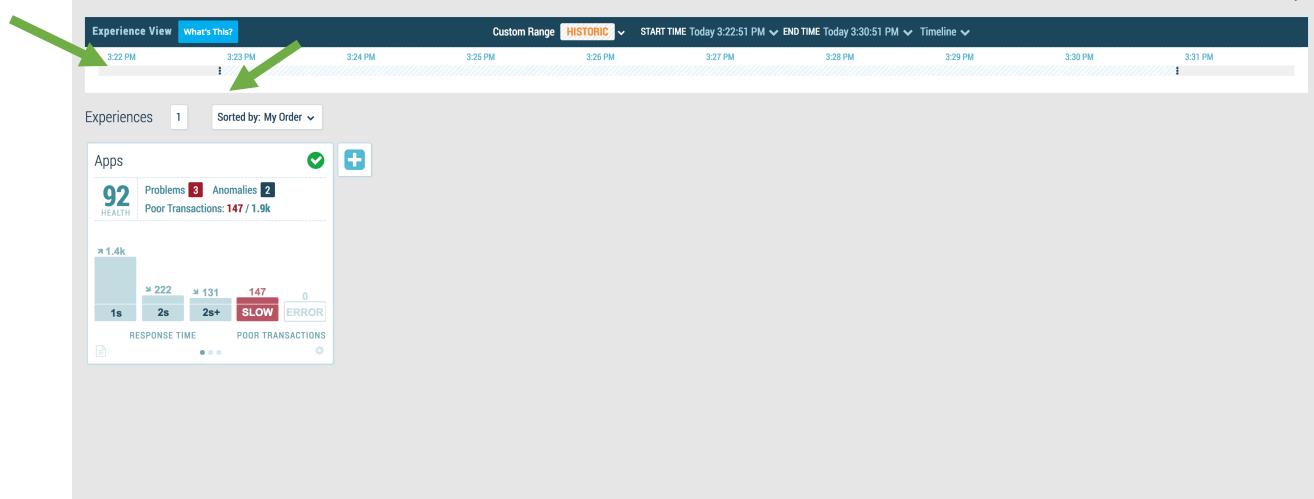
(Example Issue) There has been a steady ramp in traffic since 8 am. At the peak of the traffic, roughly 100,000 concurrent users, the NOC reported that there was a sudden increase in the response time of key transactions, and a decrease in the amount of place order transactions being completed.

1. Open another windows command prompt from the taskbar
2. Type “cd \apps\unfit-load-generator”
3. Start load generation: type “dbtcLoad.bat”. Two Chrome windows will open and run load against the unfit application. If there is no dbtcLoad.bat save the baseLoad.bat as dbtcLoad.bat and replace “baseLoad.xml” with “dbtcLoad.xml” and save again.

Wait a few minutes for the problem to develop.

Open Chrome browser and goto the APM Homepage. Click on Experience View. Note that your view may be different than the screenshot below

Here we see that there have been 147 slow transactions, creating 3 separate problems for the end users. Drill in to see which applications are affected by this issue.



Let's check out the high call ratio issue listed on the right hand 'Problems' panel. Click on the 'Potential high call ratio' issue on the right.

The screenshot shows the Experience View interface with several service cards. From left to right, they are:

- UserService**: Health 100, Problem 1, Anomalies 0, Poor Transactions: 0 / 89. Transaction volume chart.
- Default**: Health 99, Problems 2, Anomalies 0, Poor Transactions: 14 / 1.2k. Transaction volume chart.
- TokenService**: Health 100, Problem 1, Anomalies 0, Poor Transactions: 0 / 94. Transaction volume chart.
- ActivityService**: Health 74, Problems 3, Anomaly 1, Poor Transactions: 177 / 678. Transaction volume chart.
- foodservice**: Problems 0, Anomalies 0, Poor Transactions: No Traffic. Transaction volume chart.

PROBLEMS

- POSSIBLE CULPRIT: DispatcherServletService
Potential high call ratio from DispatcherServletService to localhost on 1527 (uniftdb instance) (Derby DB) in the order of 14
- POSSIBLE CULPRIT: napio03centos:1527
Backend napio03centos:1527 experienced Unstable Response Times, Alerts
- POSSIBLE CULPRIT: datengine:61616
Backend datengine:61616 experienced Unstable Response Times, Alerts

ANOMALIES

- POSSIBLE CULPRIT: Apps/DataEngine/URLs/.../api/
Frontend Apps/DataEngine/URLs/.../dataengine/api/ experienced Unstable Response Times, Alerts

Here, we see all the components involved in this problem. Notice that the affected end User Transaction is highlighted in red. Also note that *infrastructure components are included in the analysis of this issue*. Click on one of the infrastructure components under "Affected Infrastructure Components"

The screenshot shows the Experience View interface with a dependency graph and transaction traces. The graph highlights a component labeled 'DispatcherServletService' with a red oval. Below the graph is an 'Overview for GENERICFRONTEND Apps|ActivityService|URLs/api/v1/' section with metrics for Average Response Time (ms).

PROBLEMS

- POSSIBLE CULPRIT: DispatcherServletService
Potential high call ratio from DispatcherServletService to localhost on 1527 (uniftdb instance) (Derby DB) in the order of 14

AFFECTED INFRASTRUCTURE COMPONENTS [4]

- activityservice
- userservice
- datengine
- tokenservice

AFFECTED APPLICATION COMPONENTS [14]

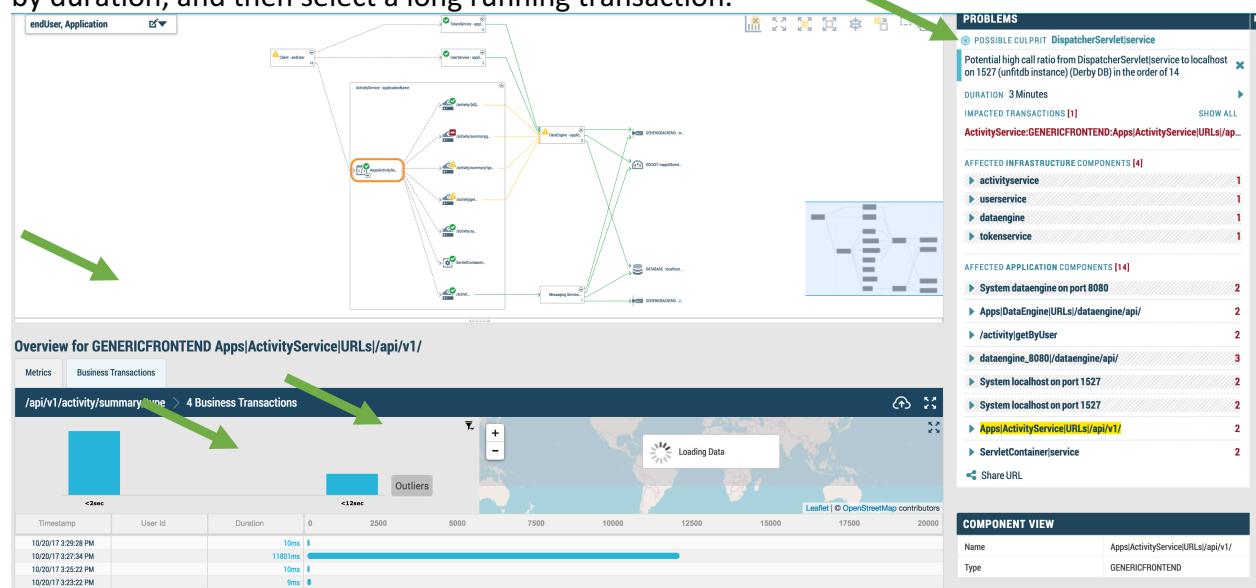
- System datengine on port 8080
- Apps/DataEngine/URLs/.../dataengine/api/
- /activity/getByUser
- dataengine_8080/.../dataengine/api/
- System localhost on port 1527
- System localhost on port 1527
- Apps/ActivityService/URLs/api/v1/
- ServletContainer/service

Note that we are monitoring host level metrics including memory, cpu, network and storage. The UIM integration provides even more infrastructure visibility. In this case, this issue has caused a spike in the available memory used.

AFFECTED INFRASTRUCTURE COMPONENTS [5]

- ▶ **dataengine** 1
- ▶ **tokensevice** 1
- ▶ **activityservice** 1
- ▼ **zibiscents.localdomain** 1
- ▼ **Alerts** 1
- SuperDomain:SaaS:Memory In Use Percent**
- ▶ **userservice** 1

Now, let's click on the "Impacted Transaction" to see what the impacted transactions look like. Once you have clicked the "Impacted transactions, click on the Business Transaction Tab, sort by duration, and then select a long running transaction.



Here, we see an individual transaction that had a high number of calls (2832) to a backend database. This churn is what has caused the issue in the dataengine component. Click on the Summary and tree views for different perspectives of call data.

| Level | Path | Cumulative Duration | Count |
|-------|---|---------------------|-------|
| 0 | FrontendApps[DataEngineURLs]/dataengine/api/ | 11796 ms | 1 |
| 1 | ServletDispatcherServlet | 11796 ms | 1 |
| 2 | org.springframework.web.servlet.FrameworkServlet::service | 11796 ms | 1 |
| 3 | javax.servlet.http.HttpServlet::service | 11796 ms | 1 |
| 4 | com.unifi.dataengine.controller.LoggingInterceptor::preHandle | 0 ms | 1 |
| 4 | Backenddspringframework.jdbc.DriverManagerDataSource::getConnectionFromDriverManager | 278 ms | 282 |
| 4 | org.apache.catalina.connector.ResponseFacade::getOutputStream | 0 ms | 2 |
| 4 | Backendlocalhost on 1527 (unifidb instance) (Derby DB) | 63 ms | 282 |
| 4 | org.apache.catalina.connector.Response::setContentLength | 0 ms | 1 |
| 4 | org.apache.derby.client.am.Connection::prepareStatement | 42 ms | 282 |
| 4 | Backenddspringframework.jdbcRowMapperResultSetExtractor::extractData | 49 ms | 282 |
| 4 | org.apache.derby.client.am.ResultSet::close | 0 ms | 282 |
| 4 | org.apache.derby.client.am.Connection::close | 24 ms | 282 |
| 4 | com.fasterxml.jackson.databind.ser.BasicSerializerFactory::buildContainerSerializer | 0 ms | 2 |
| 4 | org.apache.catalina.connector.ResponseFacade::setStatus | 0 ms | 1 |
| 4 | com.fasterxml.jackson.databind.type.TypeBindings::..._resolveBindings | 0 ms | 1 |
| 5 | Backendlocalhost on 1527 (unifidb instance) (Derby DB)[SQL]PreparedQuery[SELECT * FROM ACTIVITIES WHERE ID = ?] | 62 ms | 281 |
| 5 | org.apache.catalina.connector.Response::setStatus | 0 ms | 1 |
| 5 | java.net.Socket::close | 4 ms | 564 |
| 5 | Backendlocalhost on 1527 (unifidb instance) (Derby DB)[SQL]PreparedQuery[SELECT * FROM ACTIVITIES WHERE USERID = ?] | 1 ms | 1 |
| 5 | java.net.Socket::connect | 54 ms | 282 |
| 5 | org.apache.catalina.connector.Response::getOutputStream | 0 ms | 2 |
| 6 | org.apache.catalina.connector.Response::setStatus | 0 ms | 1 |

Docker Container Data

Now that we have quickly discovered the root cause of the issue, let's take a look at some of the other data collected about this problem. Click on 'View in Map' to see the full visualization.

The screenshot shows the CA APM Experience View interface. At the top, there is a timeline and a 'Relationship Flow' section. The 'Relationship Flow' section displays a network of nodes and connections between them, including 'Client - endUser', '1.0.1 - buildId', '1.0.2 - buildId', 'DataEngine - applicationName', 'Messaging Services (receive)', 'DATABASE - localhost on 1527 (unifidb instance) (Derby DB)', and 'SOCKET - nippo03centos:1527'. Below this, there is a 'PROBLEMS' section stating 'No problems detected', and an 'ANOMALIES' section listing several possible culprits and alerts related to the database and dispatcher service. A blue arrow points from the text above to the 'View in Map' button.

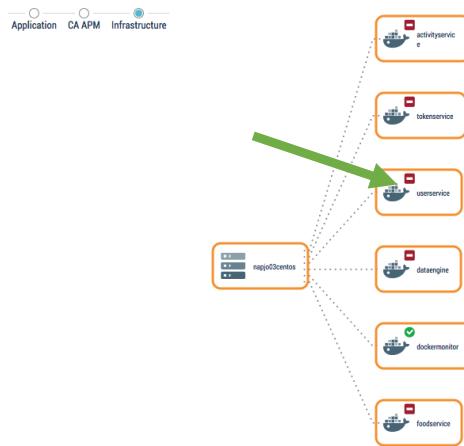
Note that there are three levels in this map:

Application: Application dependencies and relationships

CA APM: Visualization of the environment defined by the CA EM and Agents

Infrastructure: Network infrastructure of your environment

We see in this view that the problem has propagated to multiple containers, but since we are interested in the dataengine, let's start there. Click on the dataengine.

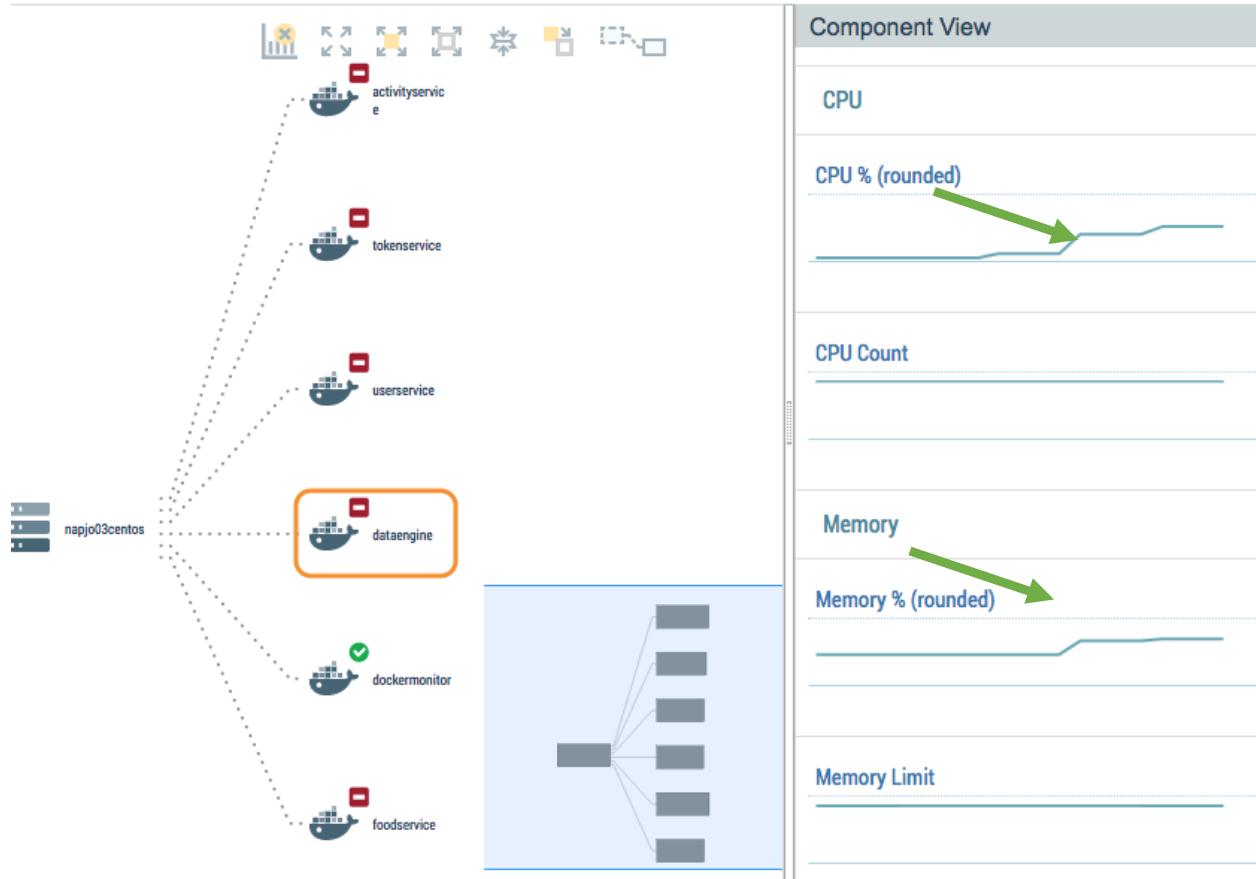


We notice that there are 4 tabs, each relating to a different level of monitoring within the application: Docker, Agent, Application and Host. Clicking on the Docker monitor, we see that two alerts – CPU and Memory have been triggered.

Screenshot of the Component View showing the Docker tab selected. The interface includes a navigation bar with tabs: Docker, Agent, Application, and Host. The Docker tab is active. Below the tabs, there is a table for 'Alerts Summary' and a list of 'Alerts (6)'. A green arrow points from the text 'Clicking on the Docker monitor, we see that two alerts – CPU and Memory have been triggered.' to the 'Alerts Summary' section. The 'Alerts Summary' section shows 1 Status (Danger) and Alert Intensity (3/5). The 'Alerts' list includes: 3 / 5 Network - Dropped Packets, 2 / 5 Memory Utilization, CPU Utilization (green checkmark), Network - Dropped Packets (green checkmark), Network - Errors (green checkmark), and Status Value (green checkmark). Below this is a section for 'Topological events (27)' with a table showing Node name, Change, and Event time.

| Node name | Change | Event time |
|--|--------------------|---------------------|
| Apps ActivityService URL- /activity/summary/getDail... | ◆ Connection added | 10/23/17 6:02:53 PM |
| /app.html#/profile/9_AJ... | ◆ Connection added | 10/23/17 6:02:53 PM |
| /app.html#/profile/9_AJ... | ◆ Connection added | 10/23/17 6:02:18 PM |

Scrolling down, we can see the metrics that triggered those alerts, and the attributes automatically collected by the Docker monitor. This information, along with the host level metrics are included in the analysis conducted by Assisted Triage.



Note that in addition to individual container health metrics, we also collect host level metrics such as CPU, Memory, Storage, and Network Metrics:

