Byteman

Advanced Java Debugging & Testing

A Nate Hansberry Presentation

Byteman Discussion Overview

Intro - What is Byteman?

Use cases & Demos

Conclusion & Discussion

What is Byteman?

Intro

What is Byteman?

Byteman is a bytecode manipulation tool which makes it simple to change the operation of Java applications either at load time or while the application is running. It works without the need to rewrite or recompile the original program. In fact, Byteman can even be used to modify Java code which forms part of the Java virtual machine, classes such as String, Thread etc.

https://downloads.jboss.org/byteman/latest/byteman-programmers-guide.html

How is Byteman helpful?

How is Byteman helpful?

Easily create custom logging for products already released

How is Byteman helpful?

Easily create custom logging for products already released

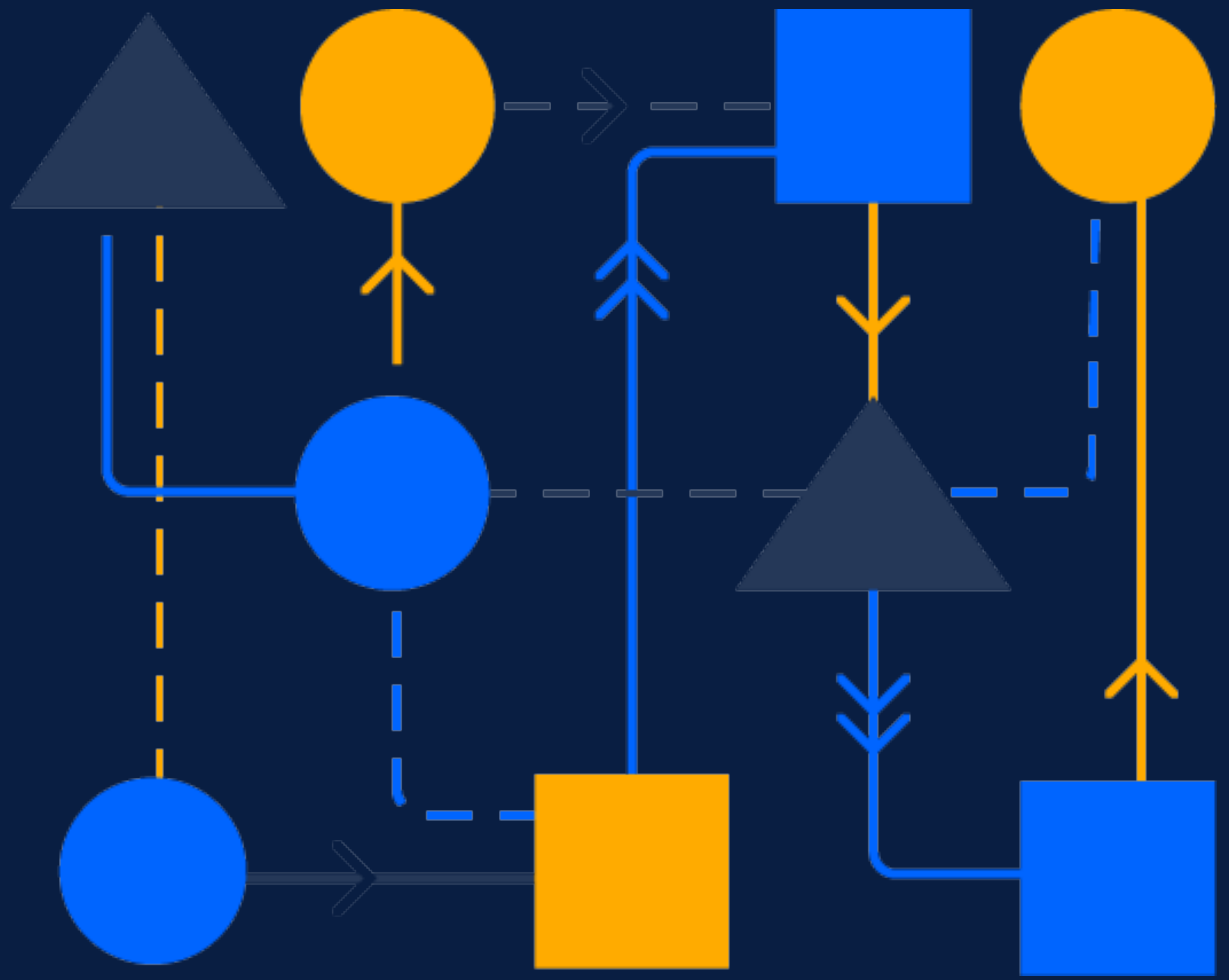
Implement temporary fixes until patch is released

How is Byteman used?

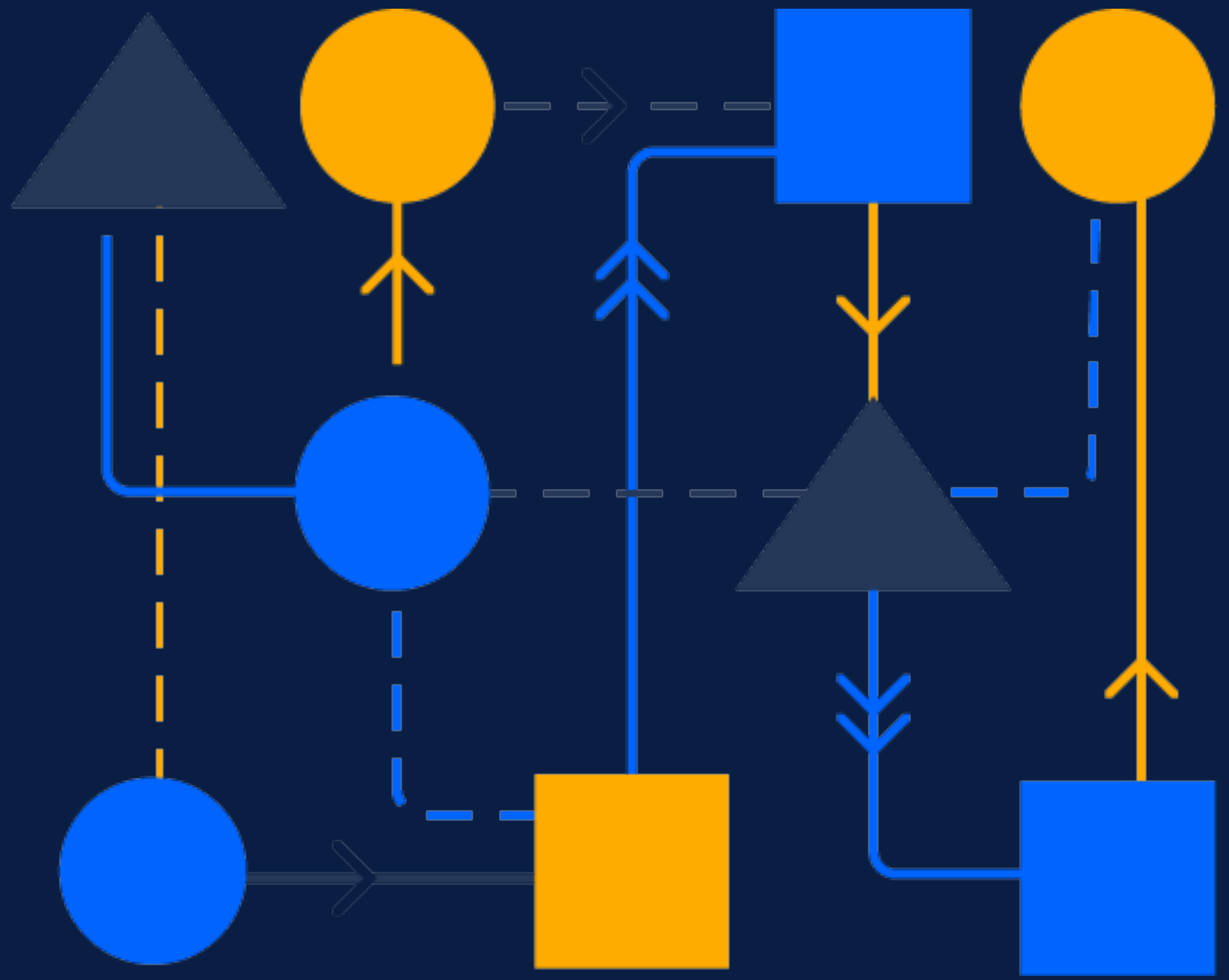
Byteman uses a simple and easy-to-use Event Condition Action (ECA) rule language, based on Java.

The Byteman rule skeleton is:

RULE - arbitrary name for humans
CLASS - what is the fqn for target class?
METHOD - what is the target method of target class?
AT - should we do something when we enter
BIND - do we need variables?
IF - what condition are we waiting for?
DO - how are we changing behaviour?
ENDRULE - end of rule so we can have multiple in one file



Use Cases



Use Case #1

Jira & Crucible Application link

Applink Issue

Customer

Jira Support

Crucible Support

Crucible DoS

Summary

Opens ticket with Jira Support

The issue describes a problem with the Jira UI. Specifically, when clicking on the review link in the dev panel, the view from Crucible does not load.

Applink Issue

Customer

Jira Support

Fisheye Support

Fisheye DoS

Summary

Opens ticket with Jira Support

The issue describes a problem with the Jira UI. Specifically, when clicking on the review link in the dev panel, the view from Crucible does not load.

Responds to the ticket

After checking the Jira logs they do not see any issues so they send the ticket to Fisheye Support

Applink Issue

Customer

Jira Support

Fisheye Support

Fisheye DoS

Summary

Opens ticket with Jira Support

The issue describes a problem with the Jira UI. Specifically, when clicking on the review link in the dev panel, the view from Crucible does not load.

Responds to the ticket

After checking the Jira logs they do not see any issues so they send the ticket to Fisheye Support

Responds to the ticket

After checking the Crucible logs they do not see any issues so they send the ticket to Crucible DoS

Applink Issue

Customer

Jira Support

Fisheye Support

Fisheye DoS

Summary

Opens ticket with Jira Support

The issue describes a problem with the Jira UI. Specifically, when clicking on the review link in the dev panel, the view from Crucible does not load.

Responds to the ticket

After checking the Jira logs they do not see any issues so they send the ticket to Fisheye Support

Responds to the ticket

After checking the Crucible logs they do not see any issues so they send the ticket to Crucible DoS

Responds to the escalation

And decides to use byteman

Applinks Order of Events

User clicks review link in Jira UI

Applinks Order of Events

Jira sends request to Crucible for review data
User clicks review link in Jira UI

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Crucible checks internally for review data

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Crucible checks internally for review data
Crucible forms a response and sends it to Jira

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Crucible checks internally for review data
Crucible forms a response and sends it to Jira
Jira uses response from Crucible and displays information

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Byteman
Crucible checks internally for review data
Crucible forms a response and sends it to Jira
Jira uses response from Crucible and displays information

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Byteman
Crucible checks internally for review data
Byteman
Crucible forms a response and sends it to Jira
Jira uses response from Crucible and displays information

Applinks Order of Events

User clicks review link in Jira UI
Jira sends request to Crucible for review data
Byteman
Crucible checks internally for review data
Byteman
Crucible forms a response and sends it to Jira
Byteman
Jira uses response from Crucible and displays information

Byteman Script

Rule 1

Rule 2

Rule 3

We know we got something from Jira if we search with data

```
# this should log the directory where the Lucene index in Crucible is located

# As a result of Jira asking for data to populate the dev panel

RULE org.apache.lucene.search.IndexSearcher#search entry

CLASS org.apache.lucene.search.IndexSearcher

METHOD search(org.apache.lucene.search.Query,
org.apache.lucene.search.Collector)

AT EXIT

BIND rosr : org.apache.lucene.index.ReadOnlySegmentReader = $0.subReaders[0];

IF $1.toString().contains("<UPDATE-JIRA-ISSUEKEY-HERE>")

DO org.slf4j.LoggerFactory.getLogger("Byteman").info("->
org.apache.lucene.search.IndexSearcher#search({},{})", $1,
rosr.singleNormStream)

ENDRULE
```

Byteman Script

Rule 1

Rule 2

Rule 3

Next we check to see how many hits we get in the index

```
# this will return the number of search results the jira summary service
uses to build the review response

RULE
com.atlassian.fecru.plugin.jira.summary.service.DefaultReviewDetailsService
#getReviewDetails exit

CLASS
com.atlassian.fecru.plugin.jira.summary.service.DefaultReviewDetailsService

METHOD getReviewDetails

AFTER WRITE $searchResults

IF TRUE

DO org.slf4j.LoggerFactory.getLogger("Byteman").info("-
>com.atlassian.fecru.plugin.jira.summary.service.DefaultReviewDetailsServic
e#getReviewDetails; number of searchResults: {}",
String.valueOf($searchResults.size()))

ENDRULE
```

Byteman Script

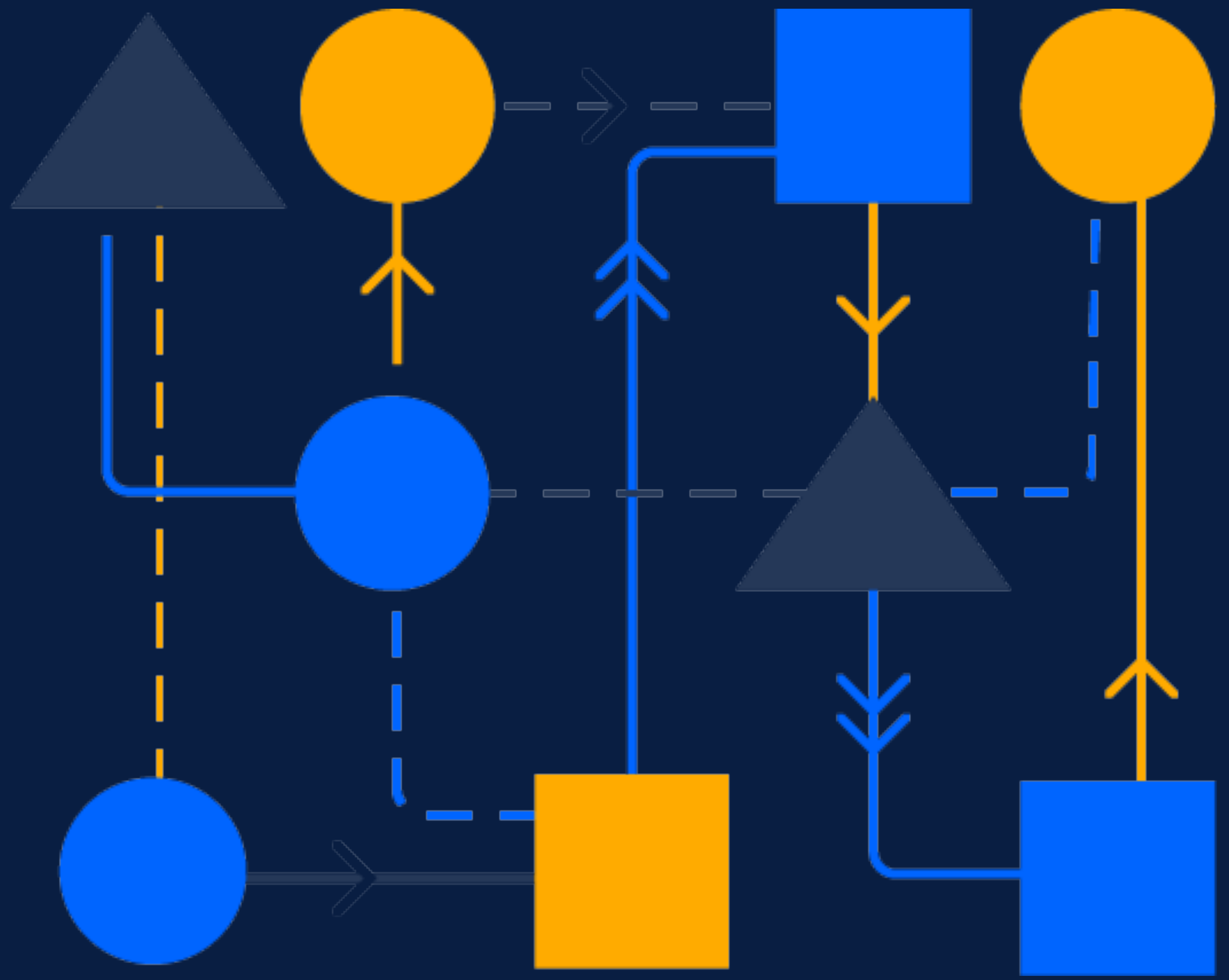
Rule 1

Rule 2

Rule 3

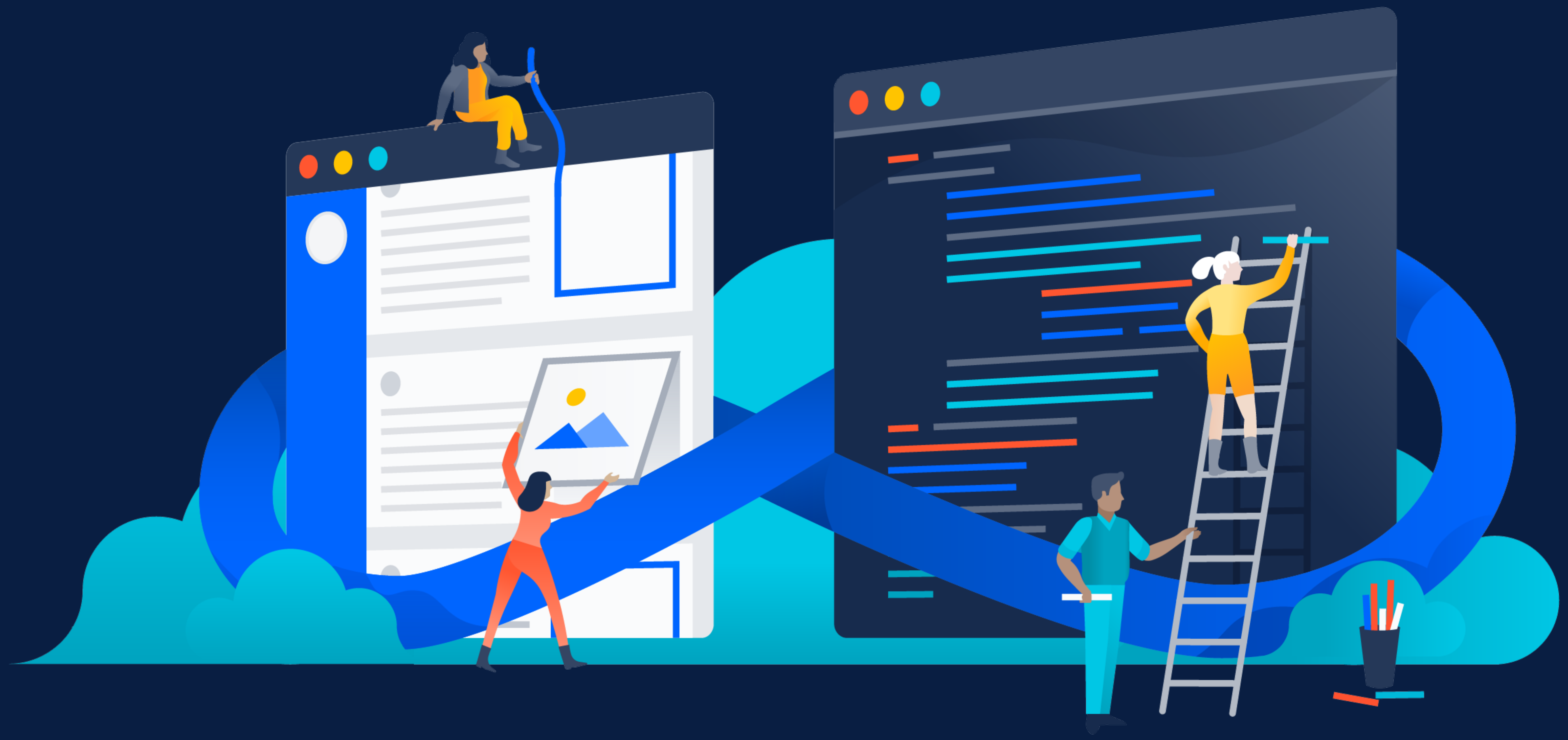
Last we check the response being sent back to the requester

```
# this will log the json payload being sent back from fecru (by way of
atlassian-rest-common plugin) to jira

RULE
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelpe
r$TypeOutInvoker#_dispatch exit

CLASS
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelpe
r$TypeOutInvoker

METHOD _dispatch

AT EXIT

IF TRUE

DO org.slf4j.LoggerFactory.getLogger("Byteman").info("->
com.atlassian.plugins.rest.common.interceptor.impl.DispatchProviderHelpe
r$TypeOutInvoker#_dispatch; response: {}", $2.getResponse().getEntity())

ENDRULE
```



Use Case #2

[Fisheye UI Performance Timings & temporary fixes]

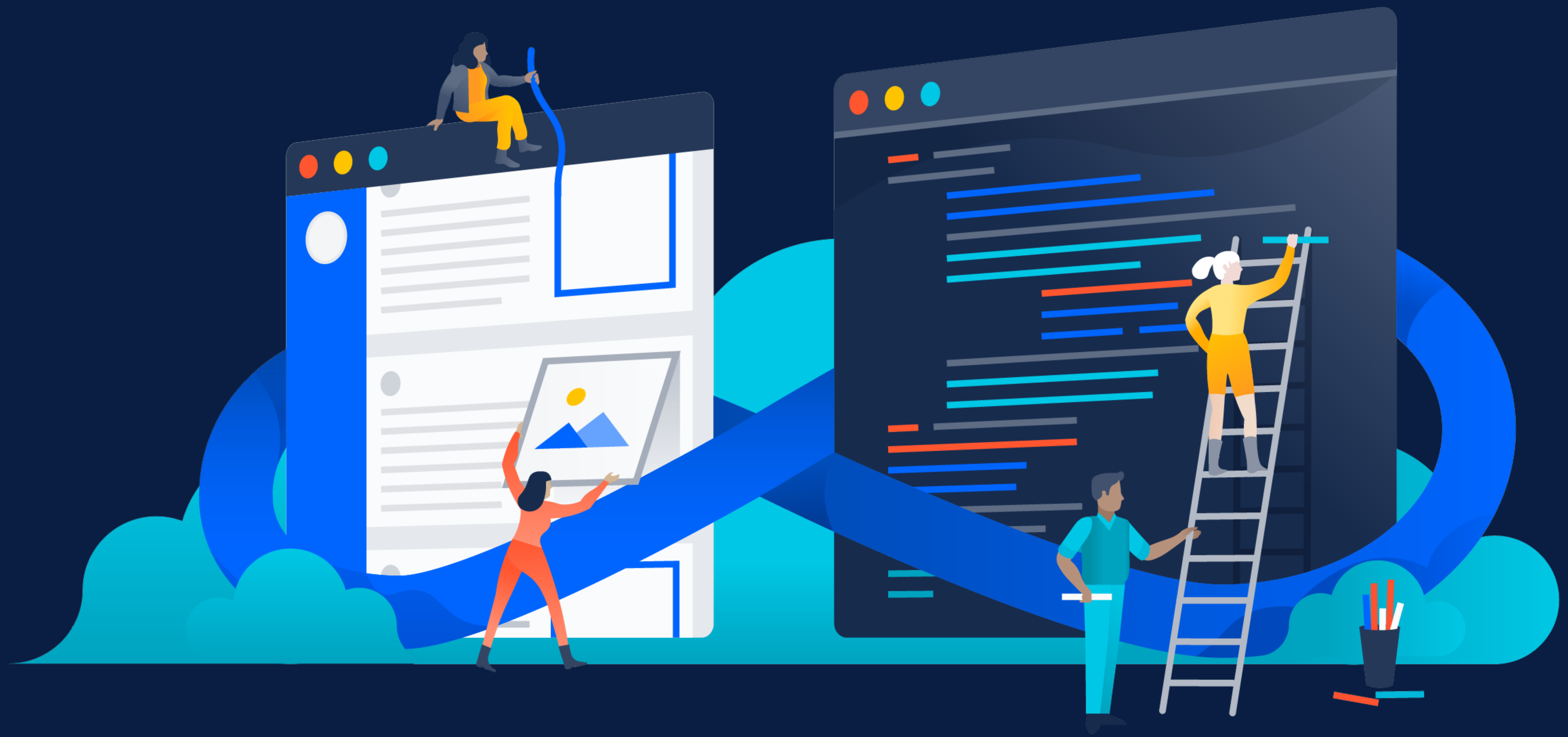


Demo Time

Custom Logging

Demo Time

Custom Logging and Temporary Patching



Advanced Topics
Testing

Byteman and JUnit

Maven

Test Class Option 1

Test Class Option 2

To include dependencies using Maven add the following elements to the main pom.xml in your project

```
    <dependency>
        <groupId>org.jboss.byteman</
groupId>
        <artifactId>byteman</artifactId>
        <scope>test</scope>
        <version>${byteman.version}</
version>
    </dependency>
    <dependency>
        <groupId>org.jboss.byteman</
groupId>
        <artifactId>byteman-submit</
artifactId>
        <scope>test</scope>
        <version>${byteman.version}</
version>
    </dependency>
    <dependency>
        <groupId>org.jboss.byteman</
groupId>
        <artifactId>byteman-install</
artifactId>
        <scope>test</scope>
        <version>${byteman.version}</
version>
    </dependency>
    <dependency>
        <groupId>org.jboss.byteman</
groupId>
        <artifactId>byteman-bmunit</
artifactId>
        <scope>test</scope>
        <version>${byteman.version}</
version>
    </dependency>
```

Byteman and JUnit

Maven

Test Class Option 1

Test Class Option 2

Add annotation to specify configuration and location

```
@RunWith(org.jboss.byteman.contrib.bmunit.
BMUnitRunner.class)
@BMUnitConfig(loadDirectory="target/test-
classes")
@BMScript(value="check.btm")
```

Byteman and JUnit

Maven

Test Class Option 1

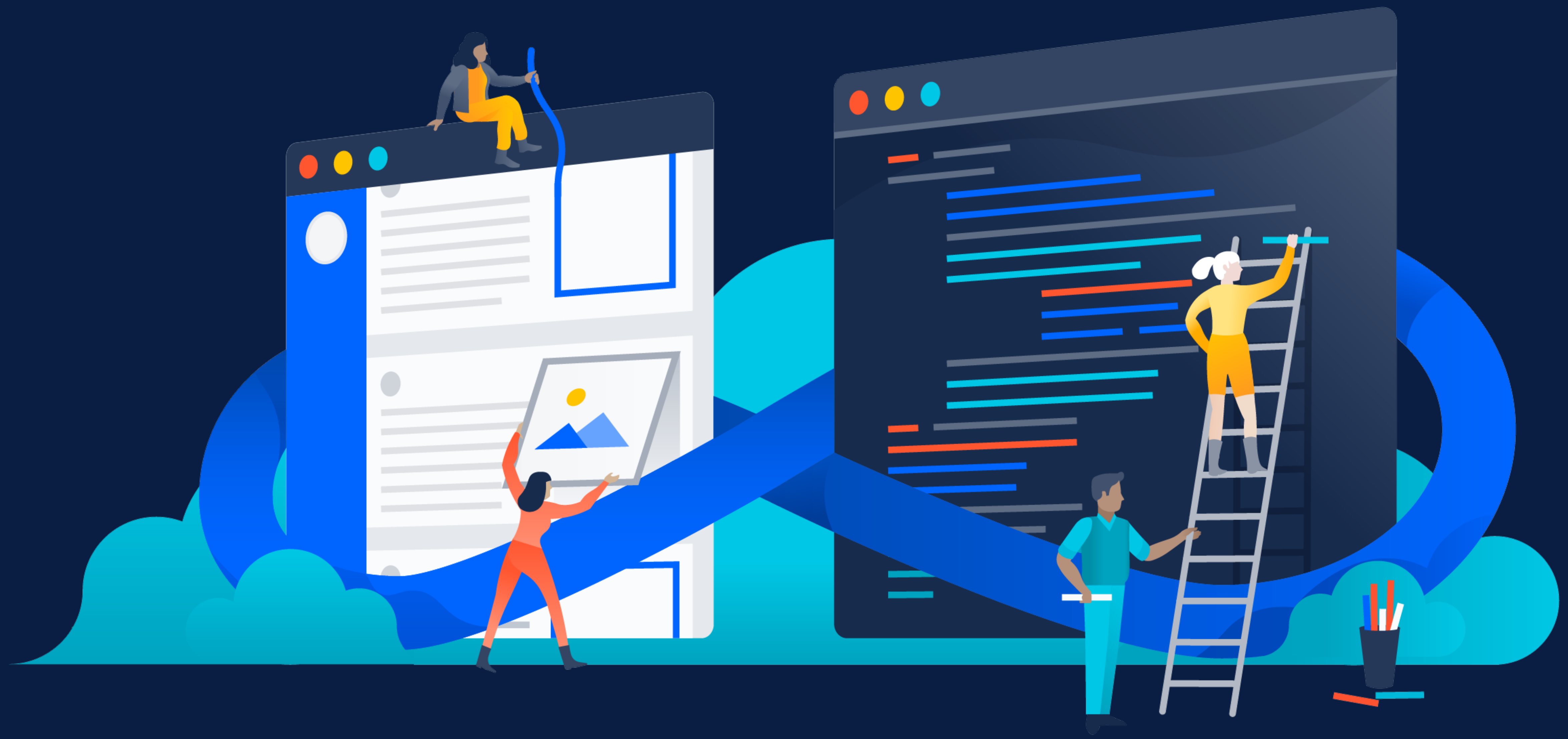
Test Class Option 2

Provide rule in annotation instead of separate script

```
@BMRule(name = "handle file not found",
            targetClass =
"FileOutputStream",
            targetMethod = "<init>(File)",
            action = "throw new
FileNotFoundException( \"Ha ha Byteman
fooled you again!\" )"
            )
```

Demo Time

Running JUnit and Byteman



Advanced Topics
Offline Testing

What happens when the code is changed?

Helpful Resource Links

Byteman Tutorial
https://developer.jboss.org/docs/DOC-17213

BMUnit Tutorial
https://developer.jboss.org/docs/DOC-52953

BM Rule Check
https://developer.jboss.org/docs/DOC-48911

Conclusions & Discussion

For pdf version of this presentation go to:

https://github.com/dark3rMatt3r/bytemanPresentation

Or scan this QR

Thank you :)

This has been A Nate Hansberry Presentation