



Java. Cloud. Leadership.

Real Java Enterprise Testing

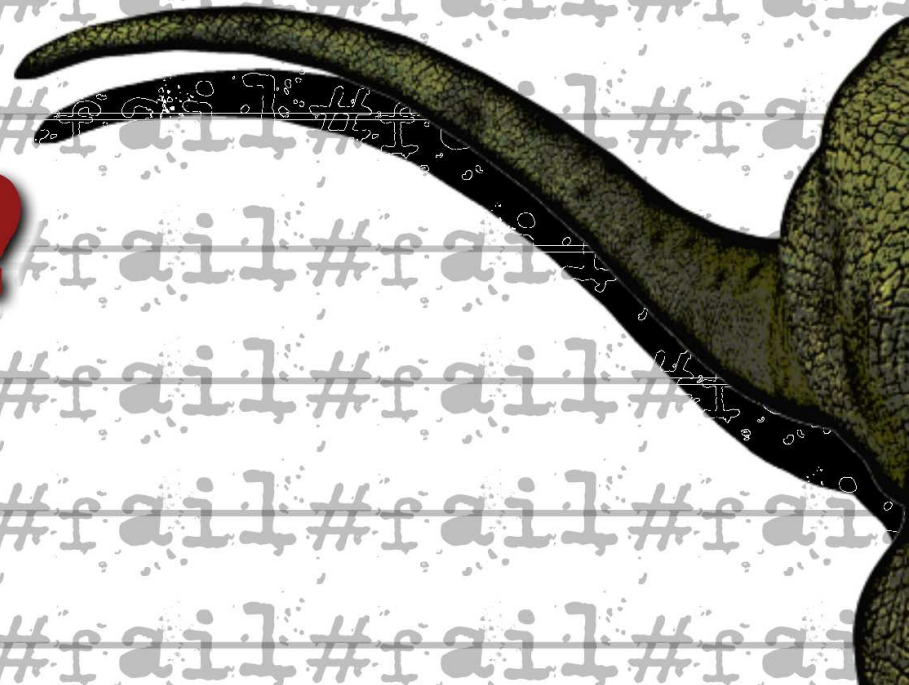
Aslak Knutsen
@aslakknutsen



**AHH, IT'S TEST
HELL ON EARTH!**



**CAN WE SAVE
DEVELOPER
PRODUCTIVITY?**





How do we prevent
software failures?

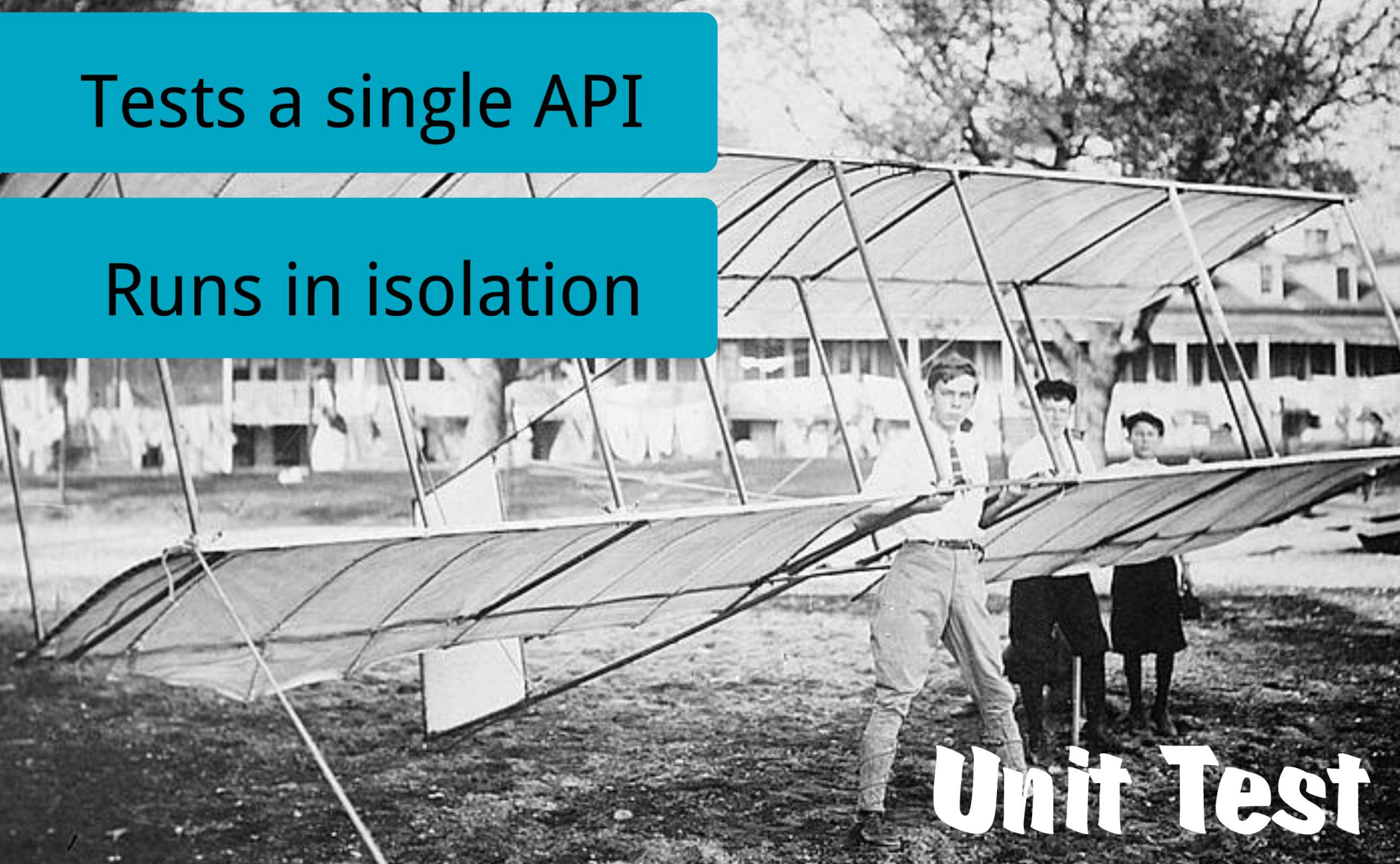


Unit Test **VERSUS** **Integration Test**

Tests a single API

Runs in isolation

Unit Test



Unit tests are so
fine-grained.

Fast, simple
and easy.
Just as
testing
should be.



Integration Test


A black and white photograph of Orville Wright sitting in the cockpit of the Wright Flyer. The aircraft is a biplane with a high-wing configuration, supported by a complex system of cables and struts. The background is a clear sky.

Tests many parts

Revs up the engine

Nice if they
could run in an
IDE.

Ha! In your
dreams.



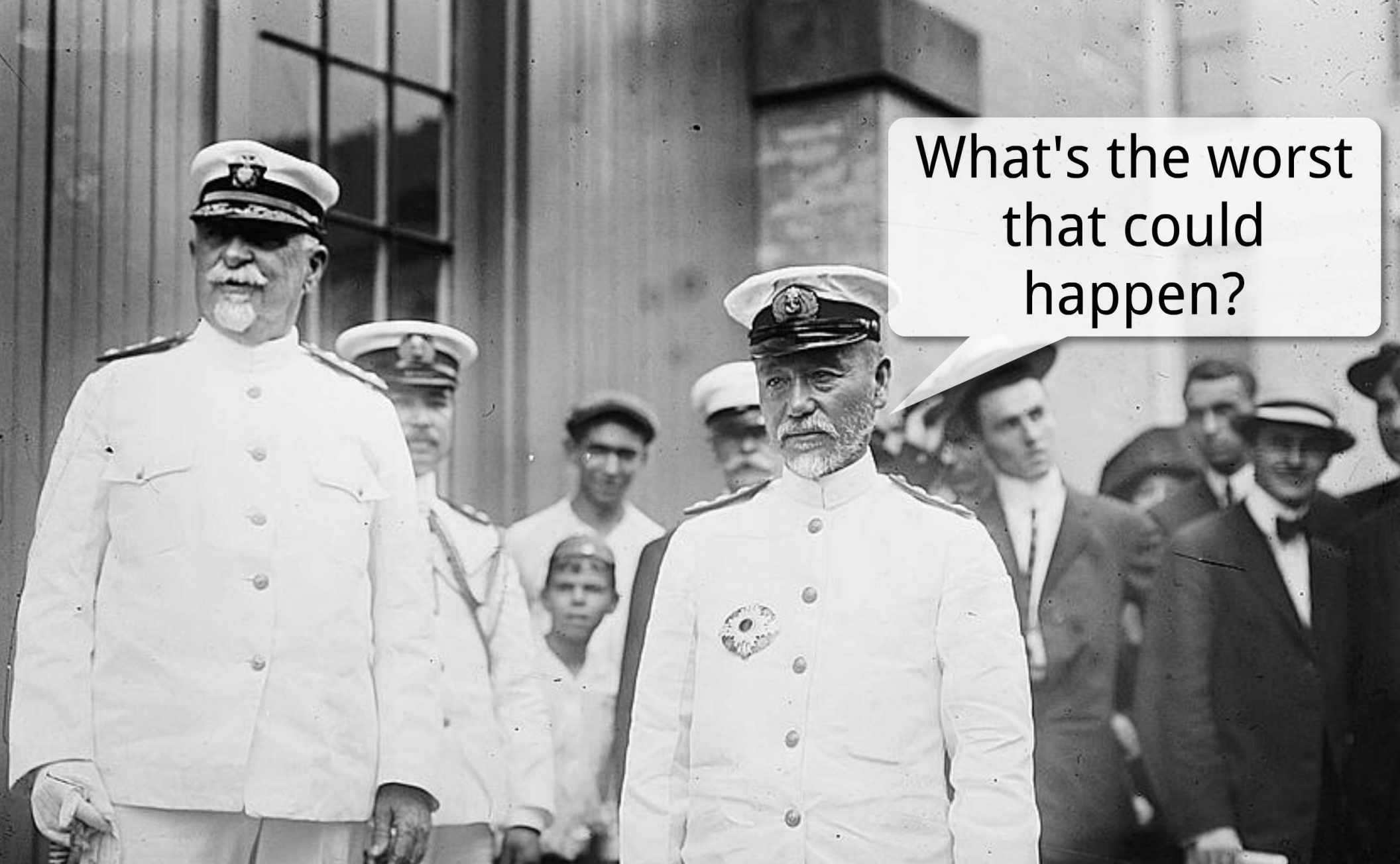
Is this test ever
going to finish
building?

Lunch is
doomed.

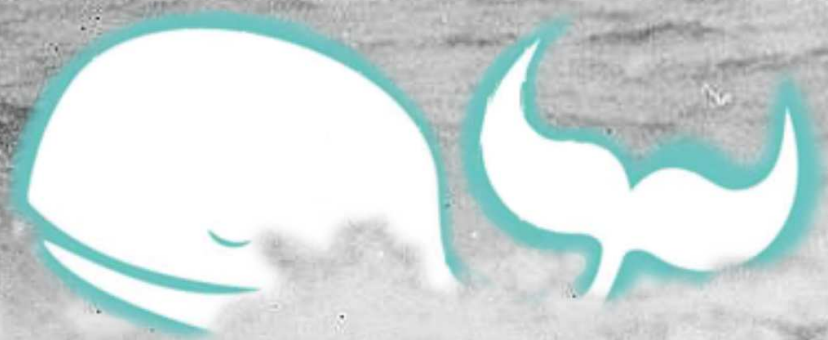



Let's just can it.

What's the worst
that could
happen?



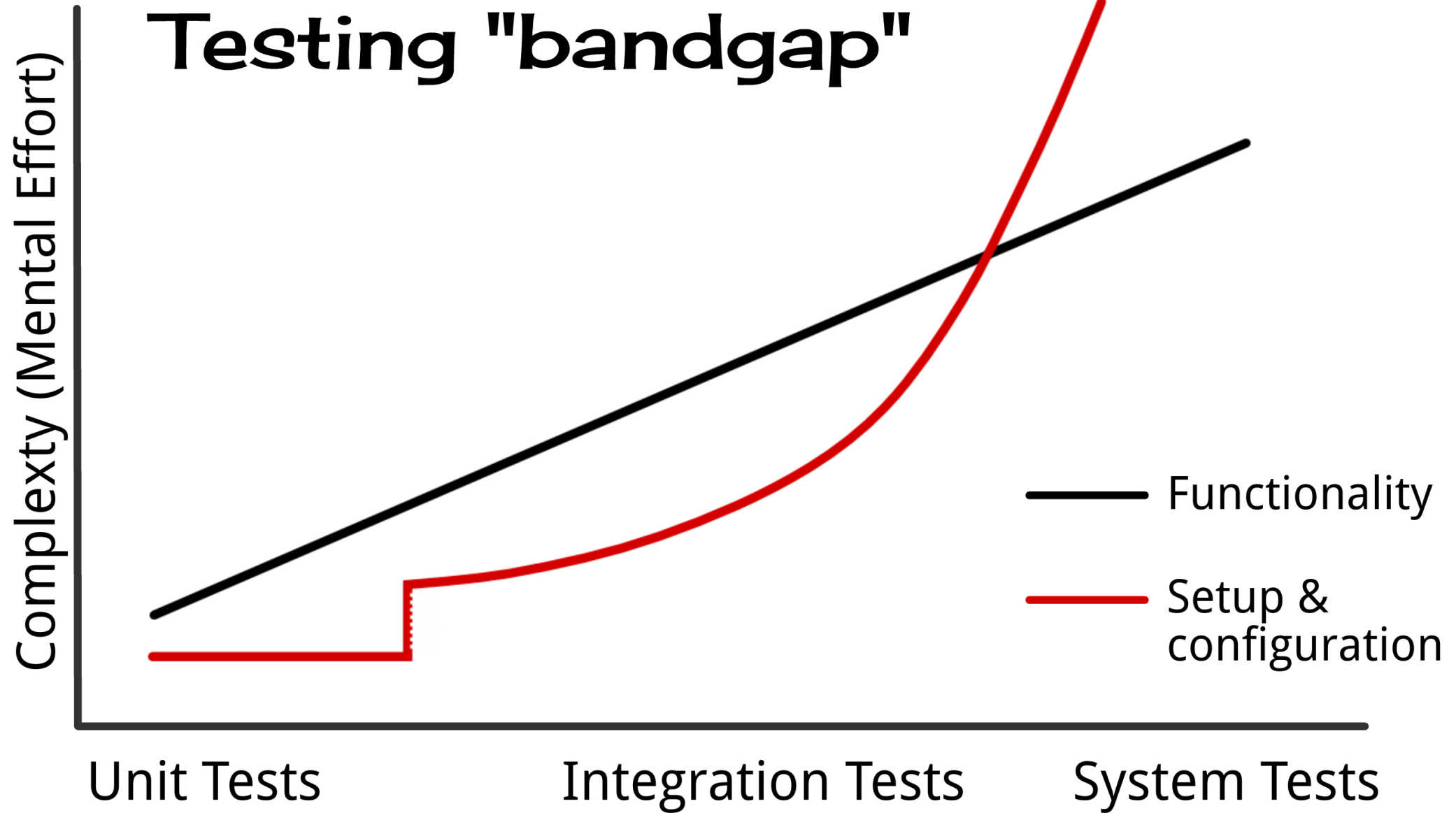
#fail



A black and white photograph of a man in a pinstriped suit, white shirt, and patterned tie, looking directly at the camera. He is framed by a dark, rounded television set. To his right, a speech bubble contains the text "Two words: intended environment".

Two words:
intended
environment

Testing "bandgap"



*graph theoretical



We've got to
bridge this gap



What would make
us more productive
when writing tests?

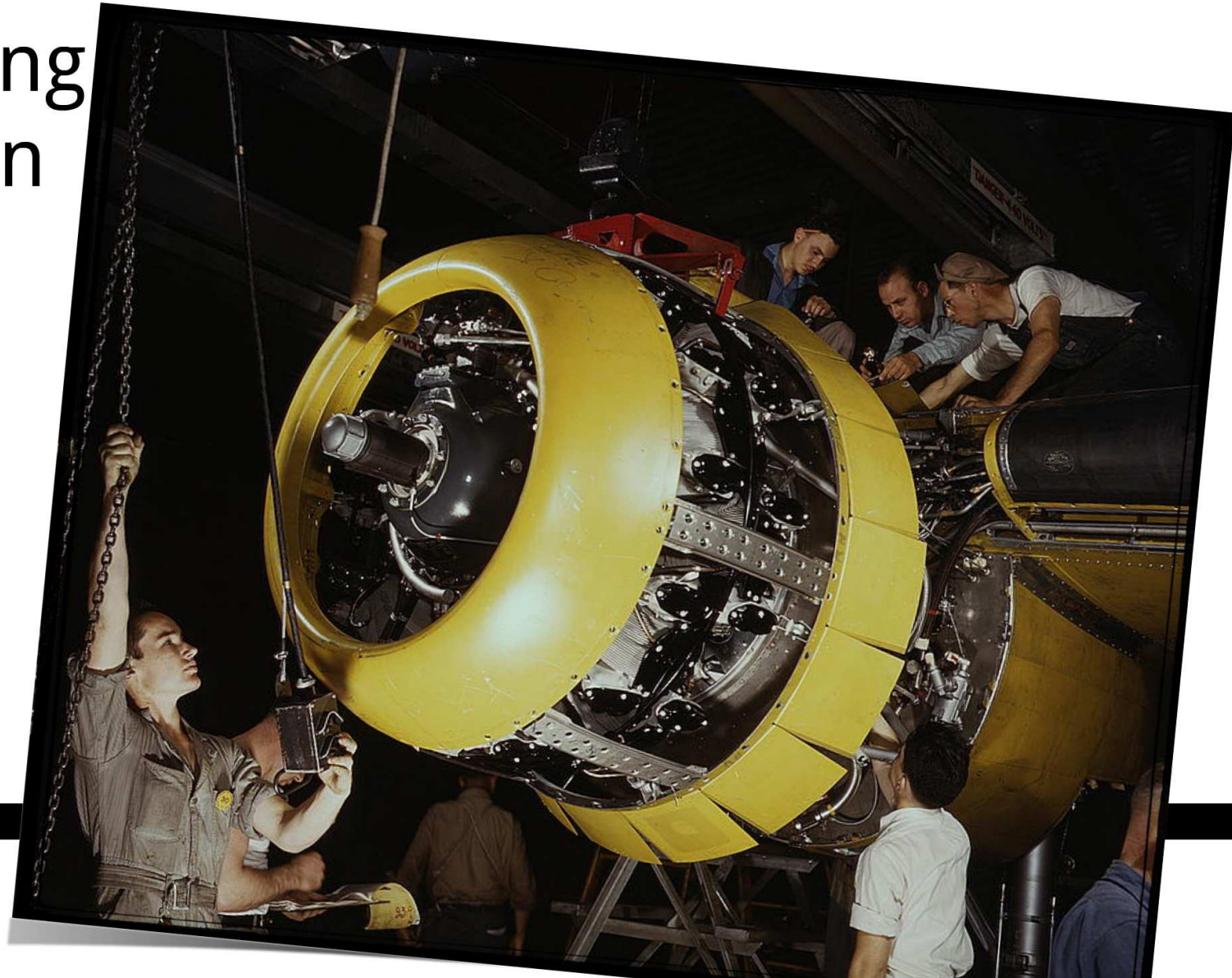
It's called a
component model
for testing.





Provides a
standard
programming
model

Manages the
runtime during
test execution





Gives you
powerful
mechanisms
for free

Keeps you **focused** on the test at hand





Can we speed up
the integration
testing cycle?



Code

V00000

Build



Test

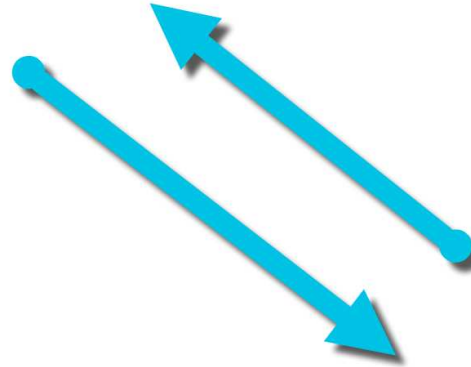


Skip the build.





Code



Test



~~VOODOO~~

~~Build~~



SHRINKWRAP



SHRINKWRAP

An API for creating archives
(e.g., JARs, WARs and EARs).
...in Java! ^

@alrubringer



Andrew Rubringer

Project Lead

```
JavaArchive jar = ShrinkWrap.create(JavaArchive.class, "beans.jar")  
    .addClasses(Greeter.class, GreeterBean.class)  
    .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
```

Simple and
fluent API

* What does ShrinkWrap
give us?

Skip the build!

```
System.out.println(jar.toString(true));
```

```
/META-INF
```

```
/META-INF/beans.xml
```

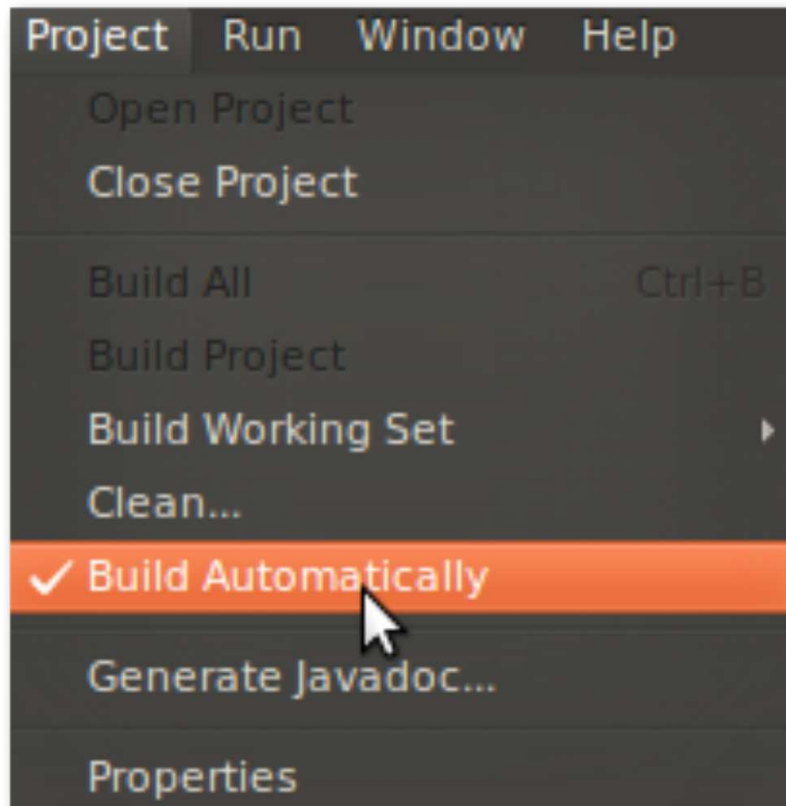
```
/org
```

```
/org/example
```

```
/org/example/Greeter.class
```

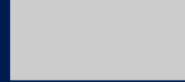
```
/org/example/GreeterBean.class
```


Incremental IDE compilation



Micro deployments

```
.addClasses(Greeter.class, GreeterBean.class)  
    .addClasses(Greeter.class, MockGreeter.class)
```



What could we use
to simplify
integration tests?



Hm. Test in
container?

Bring your test to the runtime...



...instead of managing
the runtime from your
test.



<http://arquillian.org>



arquillian

A container-oriented testing
framework



Aslak Knutsen

@aslakknutsen
Project Lead

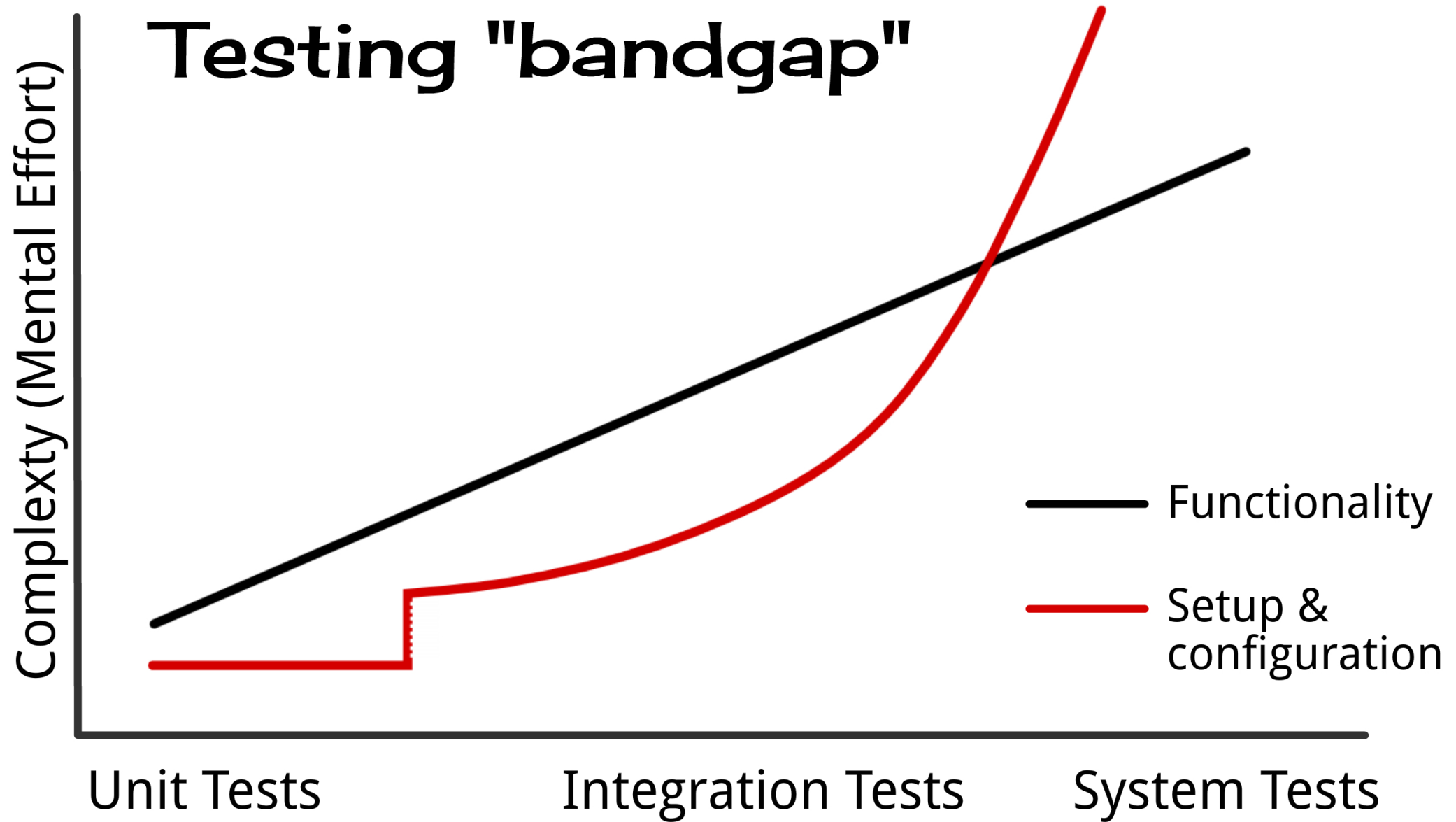




**Arquillian
enables**

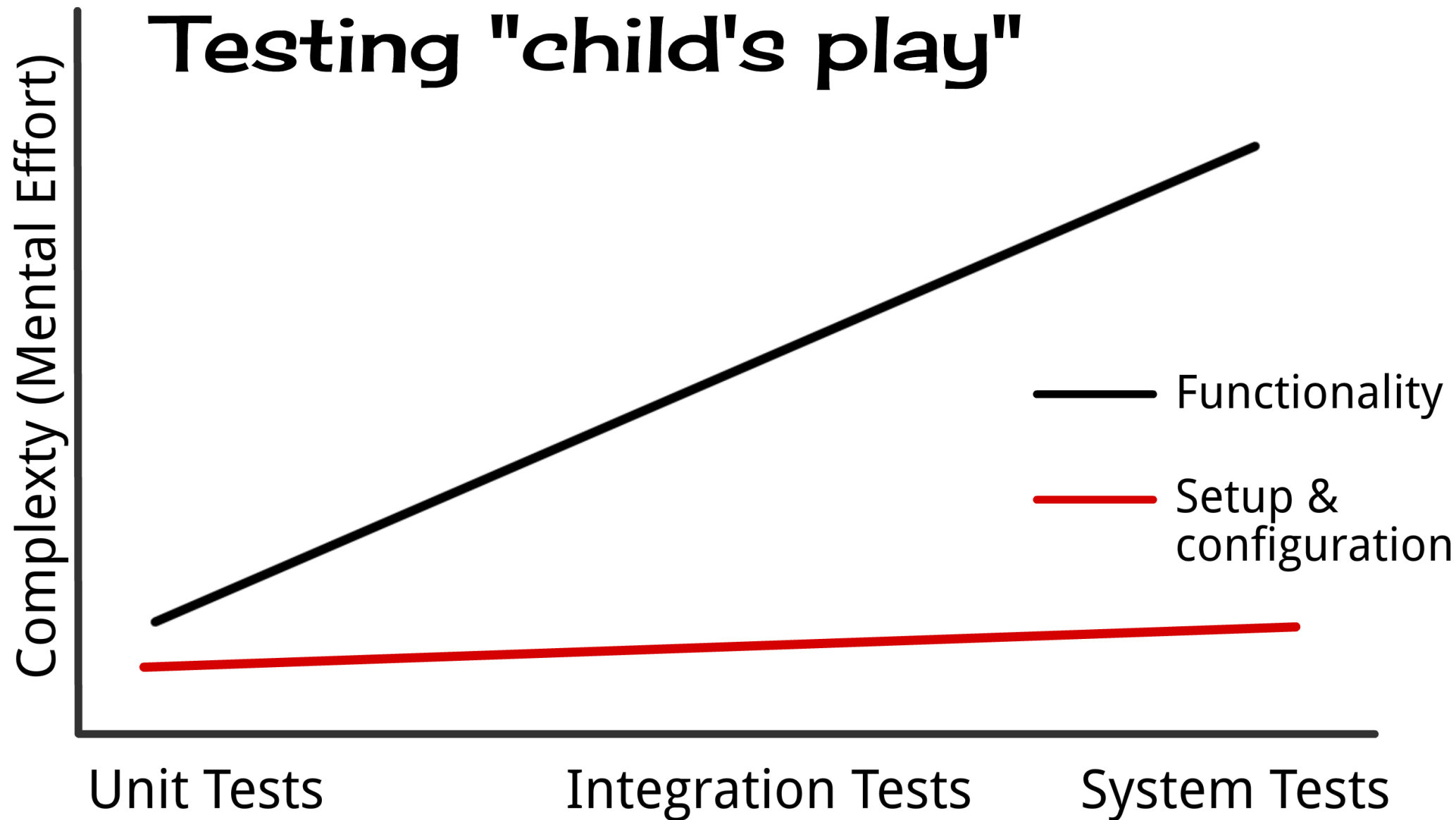
**portable integration tests for
enterprise applications**

Testing "bandgap"



*graph theoretical

Testing "child's play"



*graph theoretical


```
public class GreeterTestCase {
```

```
}
```

```
@RunWith(Arquillian.class)  
public class GreeterTestCase {
```

```
}
```

```
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }
}
```

```
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject Greeter greeter;
}
```

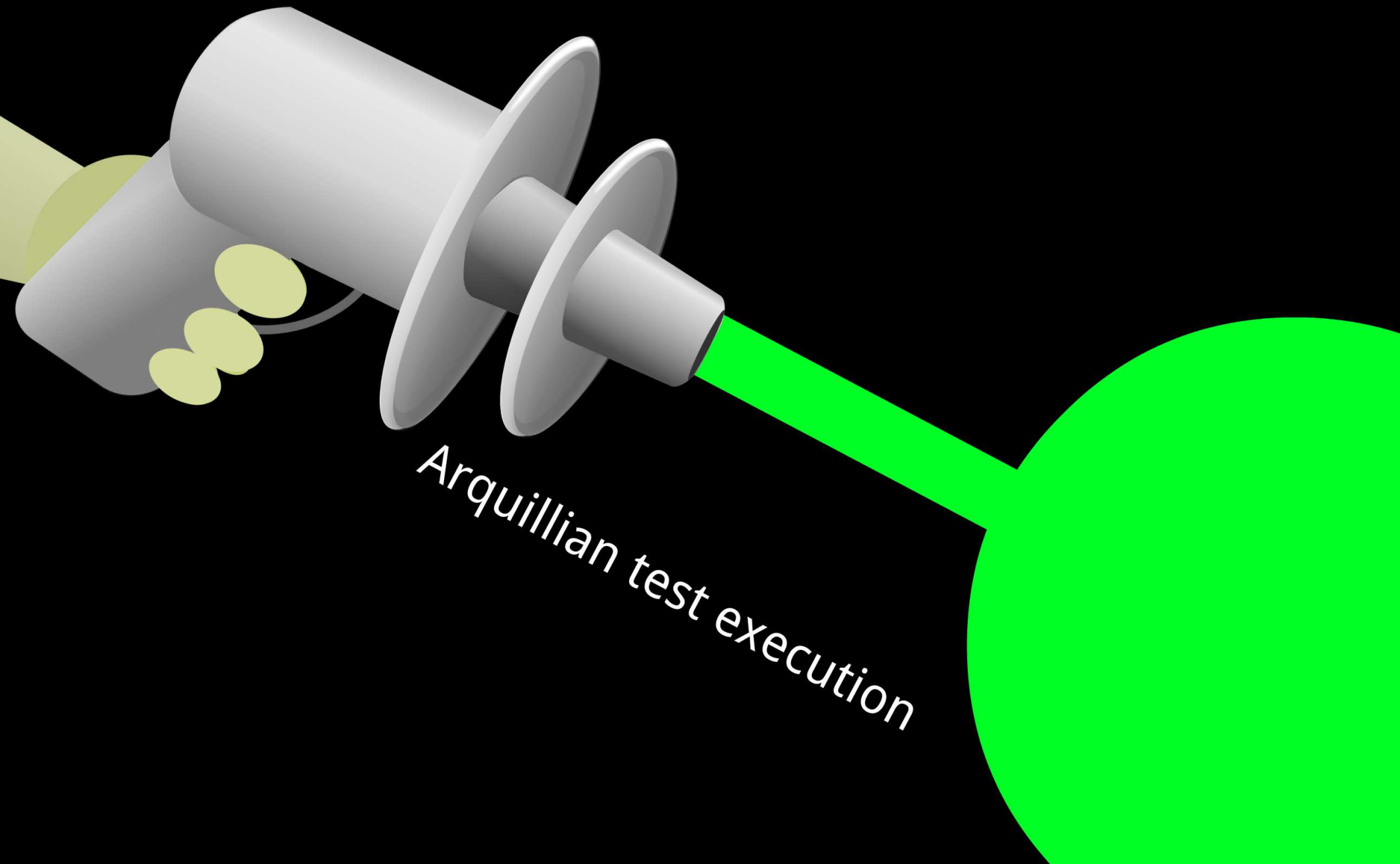


```
@RunWith(Arquillian.class)
public class GreeterTestCase {
    @Deployment
    public static JavaArchive createDeployment() {
        return ShrinkWrap.create(JavaArchive.class)
            .addClass(Greeter.class)
            .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
    }

    @Inject Greeter greeter;

    @Test
    public void shouldBeAbleToInvokeBean() throws Exception {
        Assert.assertEquals("Hello, Earthlings", greeter.greet("Earthlings"));
    }
}
```

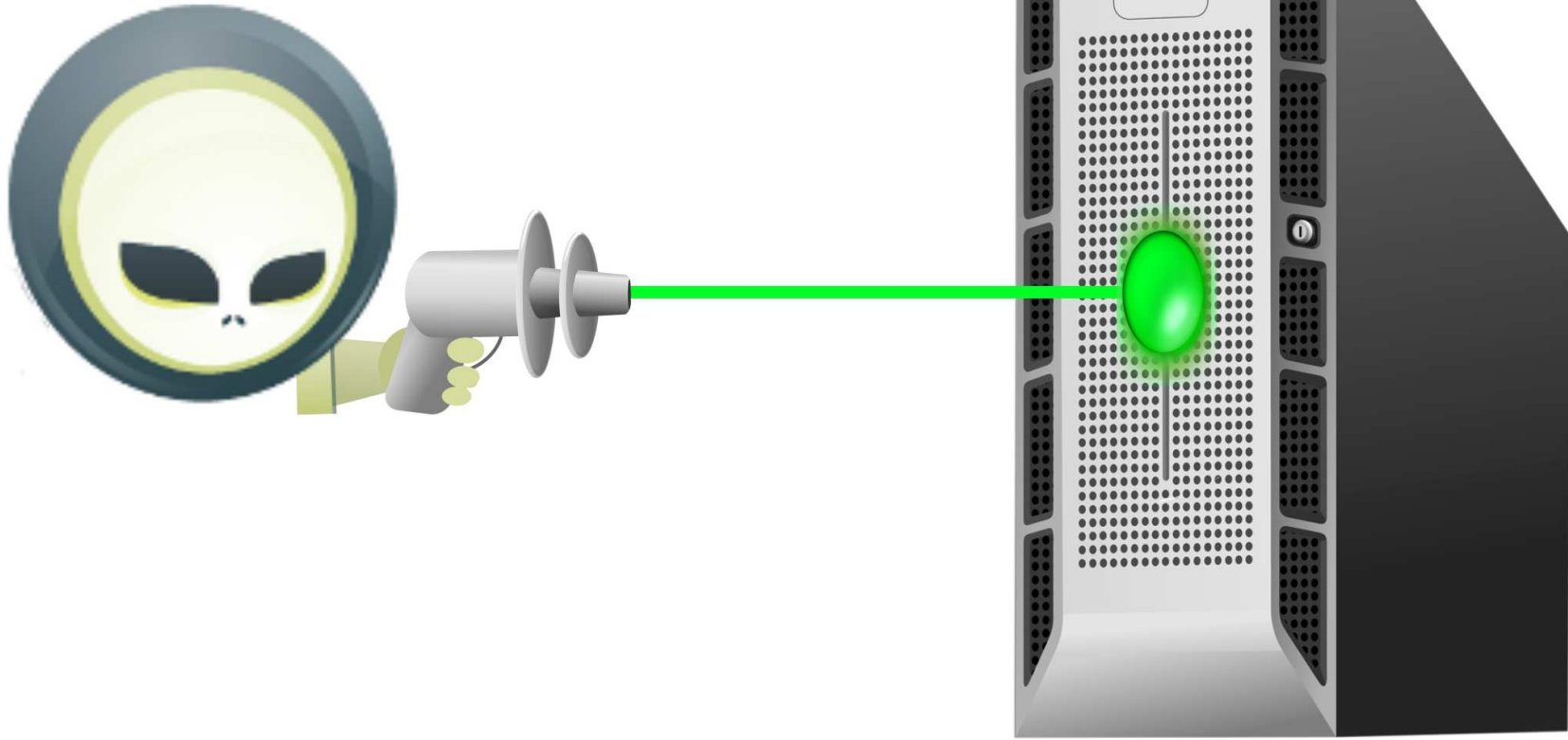




Arquillian test execution

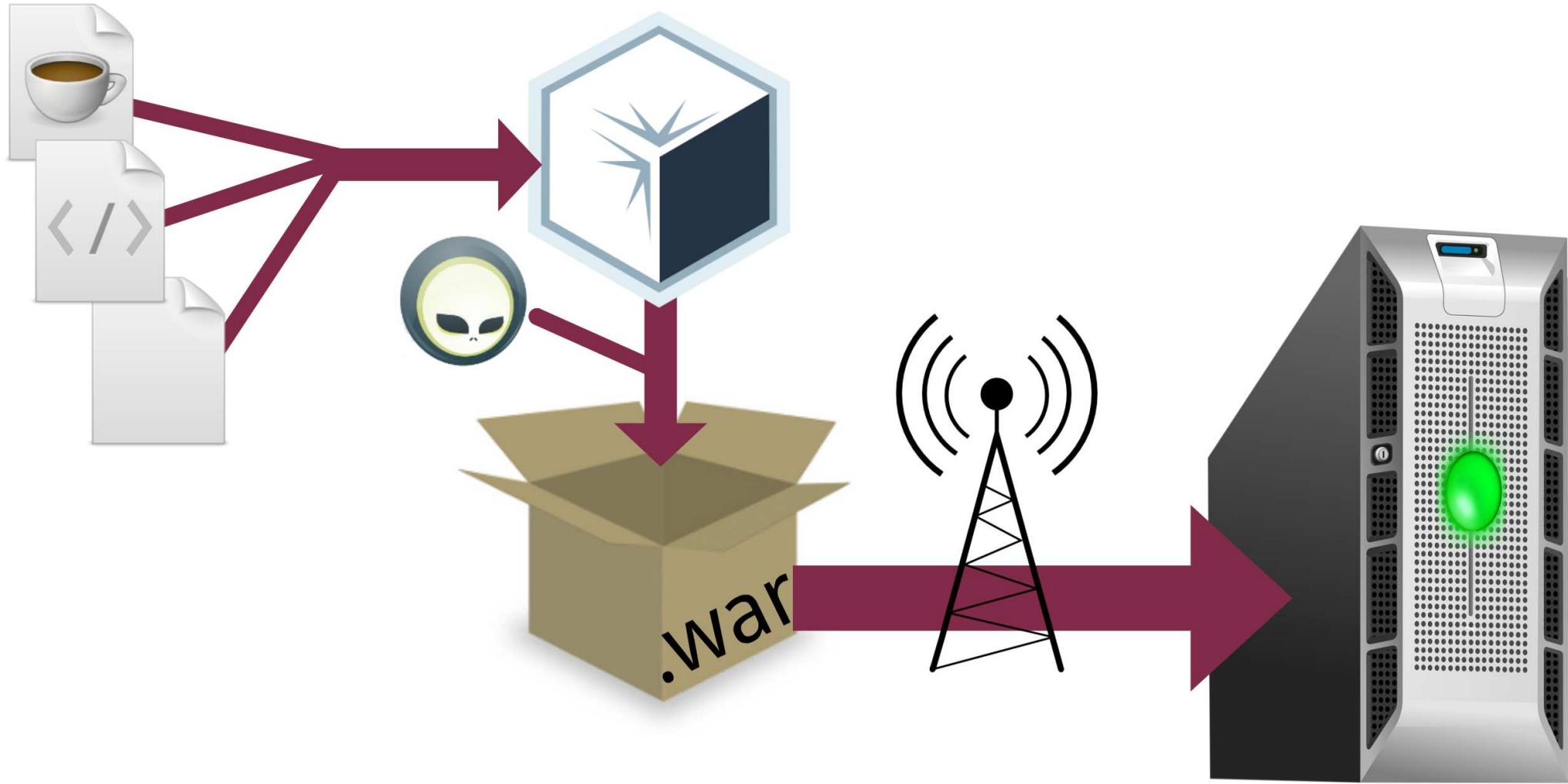


1 Select a container



2

Start or connect to a container

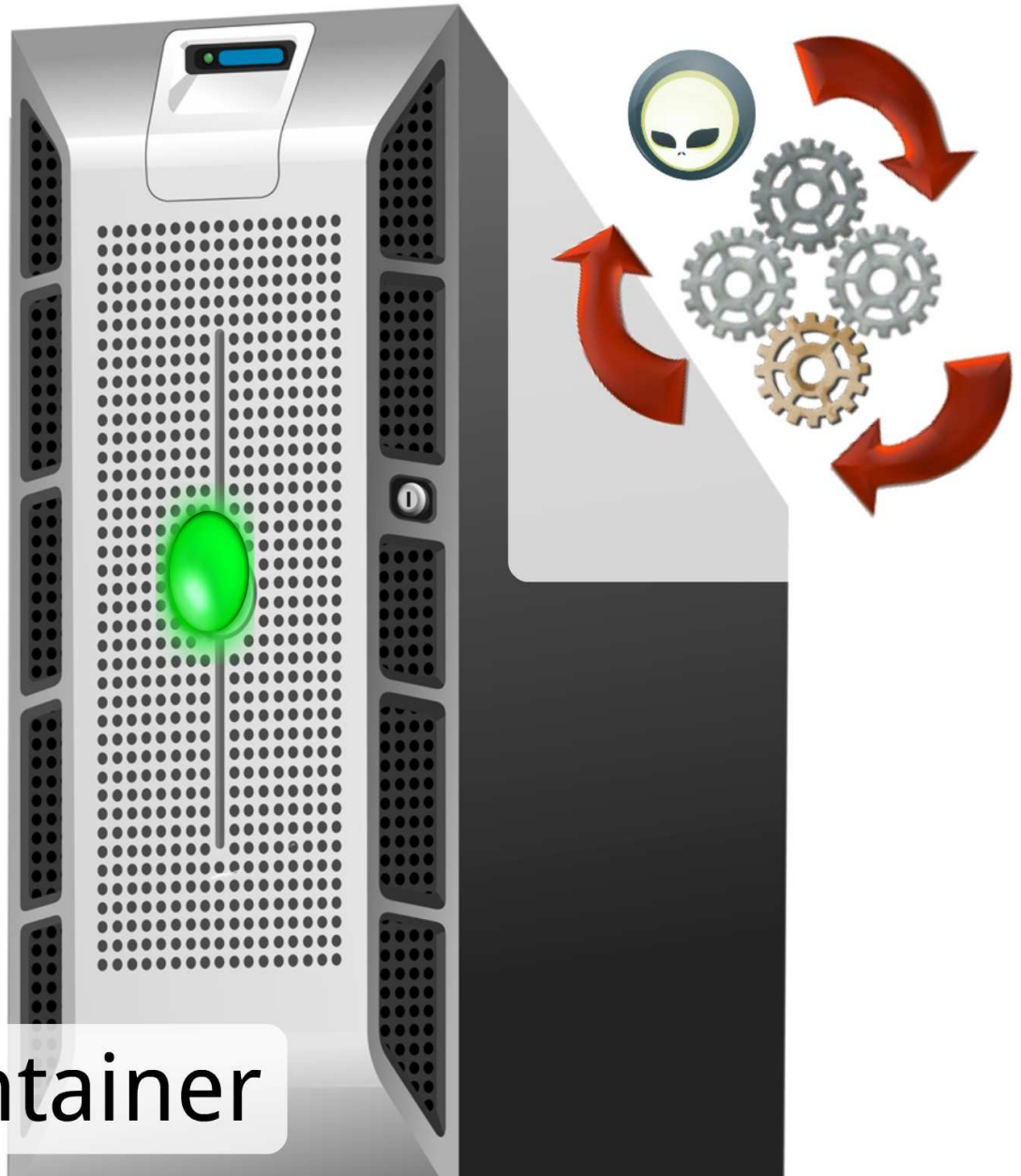


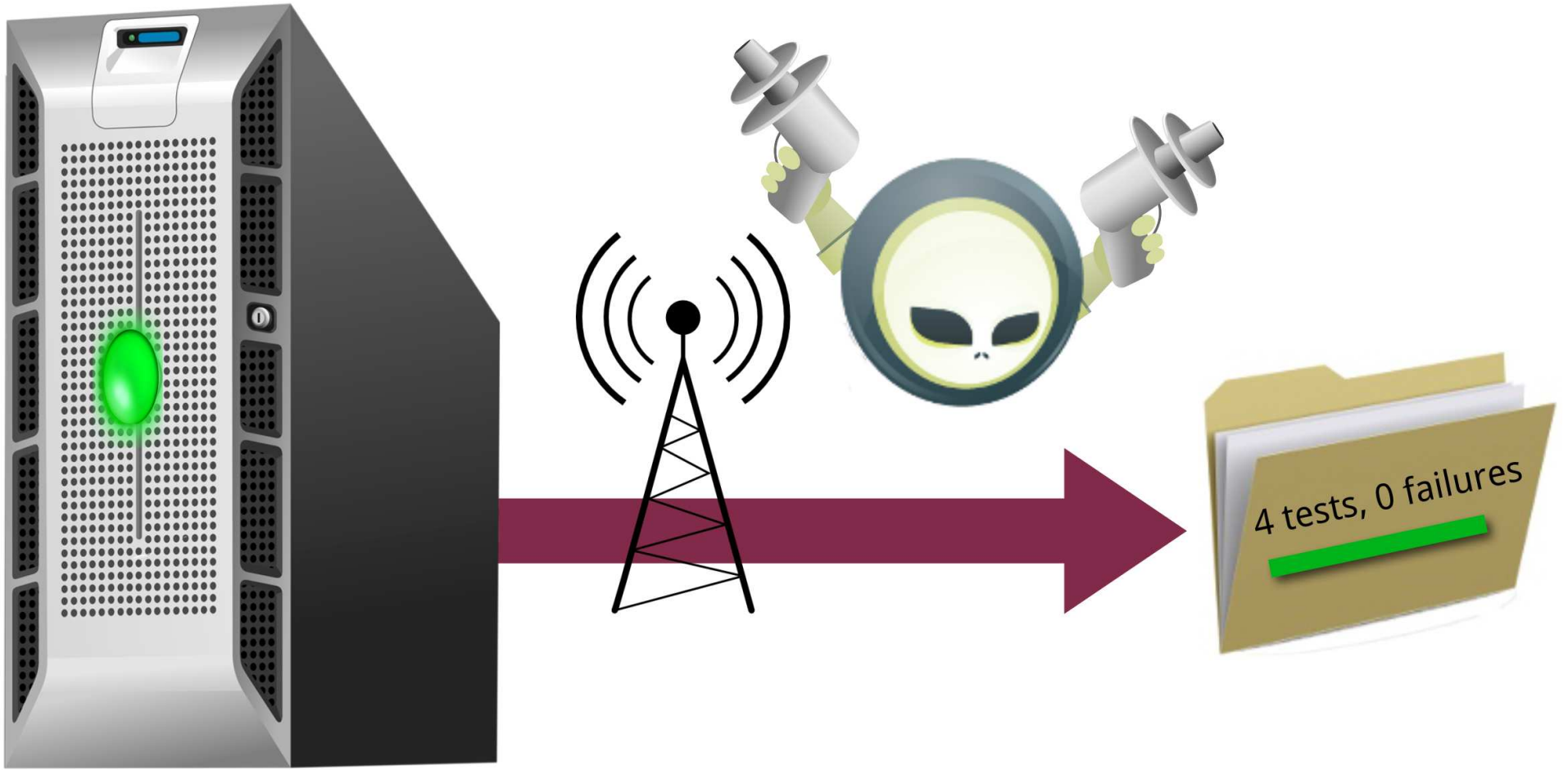
3

Package test archive, deploy to container

4

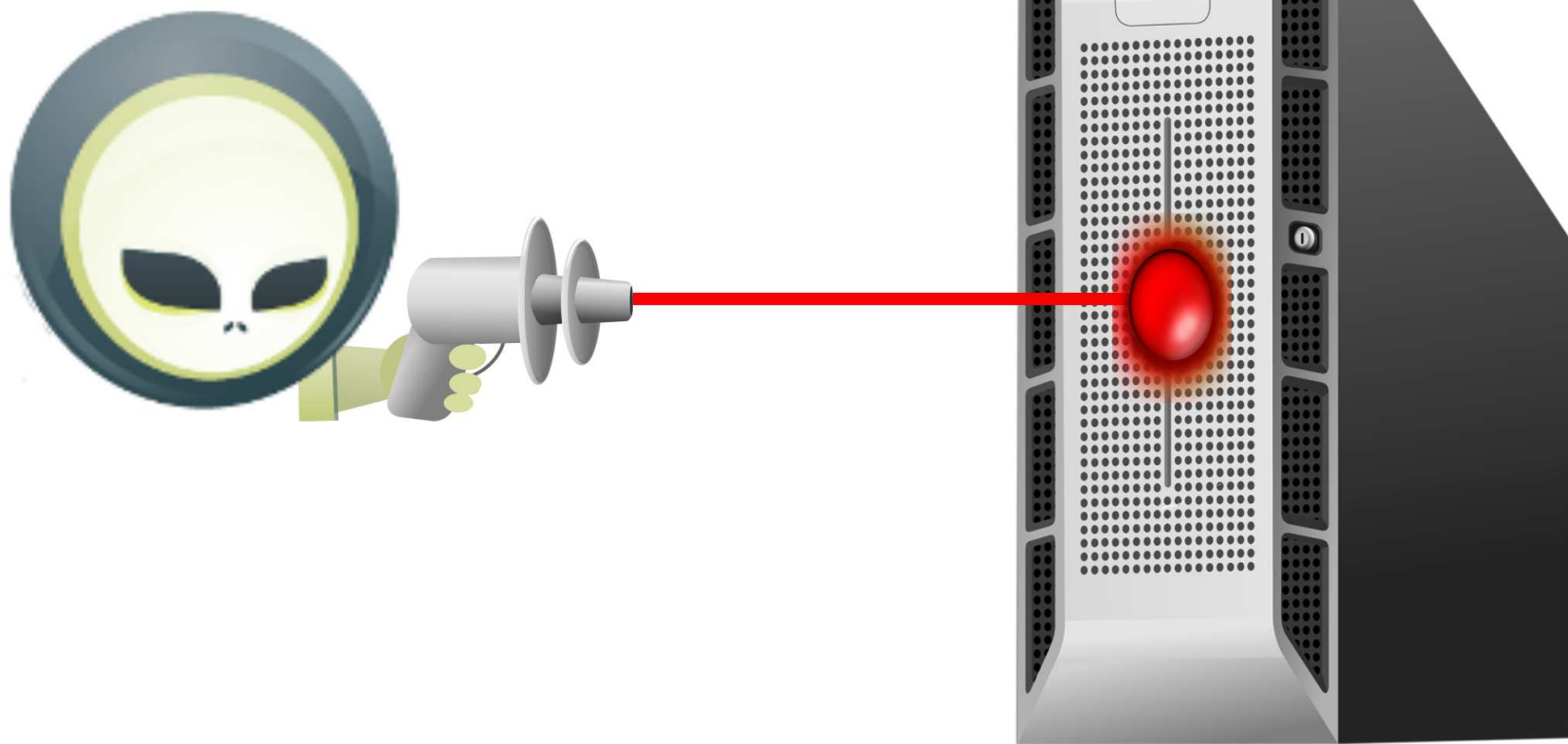
Run test in-container



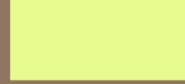


5

Capture and report results



6 Undeploy test archive,
stop or disconnect from container



Can we put these
ideas into practice?

Prove it.





THE **IKE** EXPERIENCE

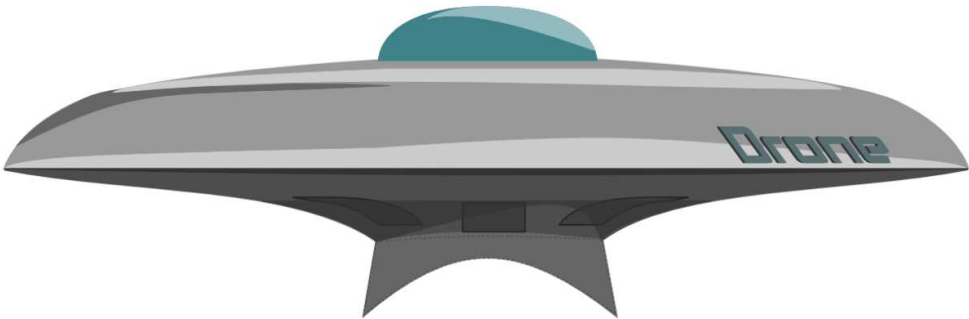
github.com/arquillian/arquillian-showcase



What's on the
horizon for
enterprise testing?





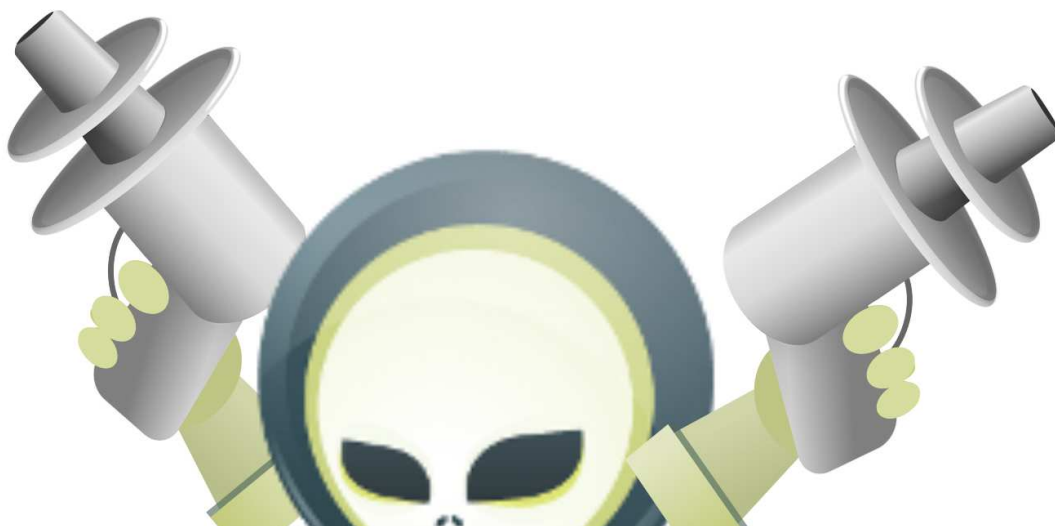


JSFUnit

jclouds

multi - cloud library

Infinispan



Byteman



Less (boilerplate) test code



As much or as little integration as you need



Looks like a unit test, but executes in a true environment



Same test can be run in multiple environments



It's a **learning** environment

“With tools like Arquillian, Shrinkwrap, [and] in-memory databases, [we] are busting through the old barriers of terms like unit and integration tests. Perhaps we need a new term for them: **"real tests"**”



- *Brian Leathem*
Software Engineer, Red Hat



- Rob Williams
jroller.com

“ Arquillian: End of Test
Hell on Earth ”

GO WRITE REAL TESTS!

***BUILD FREE AND IN
CONTAINER!***

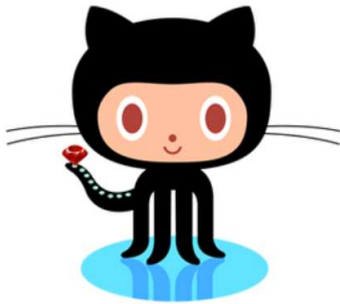




#arquillian

Write for us!

github
SOCIAL CODING



Fork us!

#jboss testing
freenode

Chat with us!



ShrinkWrap

jboss.org/shrinkwrap



arquillian

jboss.org/arquillian



THANK YOU!
(AND DON'T MOCK ME)

1.0.0 Final



arquillian.org

Historical photographs courtesy of the Library of Congress



[flickr.com/photos/library_of_congress/](https://www.flickr.com/photos/library_of_congress/)