



## **T3**

Test Techniques

Thursday, October 24th, 2019 10:15 AM

# **Testing in Production**

Presented by:

**Talia Nassi**

WeWork

Brought to you by:



888-268-8770 · 904-278-0524 - [info@techwell.com](mailto:info@techwell.com) - <http://www.starcanada.techwell.com/>

# Talia Nassi

Talia Nassi is a quality-driven Test Engineer at WeWork with a passion for breaking and rebuilding software to be the highest possible quality. She started interning in QA when she was studying at UC San Diego and immediately knew that she had found her calling. From UCSD she was recruited to work at Visa, where she tested the payment processing system for the Prepaid Cards. After Visa, Talia started at WeWork, where she continues to innovate and do what she lovesâ€”deliver high quality software!

# Testing In Production

Talia Nassi





**Think about the last feature  
your team deployed.**

**Is it working?**

**Right Now?**

**In Production?**

**How do you know?**

**Testing in production is the only way  
to know that your features  
are working in production right now.**

What is it

**Testing your features in the environments where  
your features will live**

# What is it **not**

**It's not a replacement for all testing**

**It's not the end of the world**





# Talia Nassi

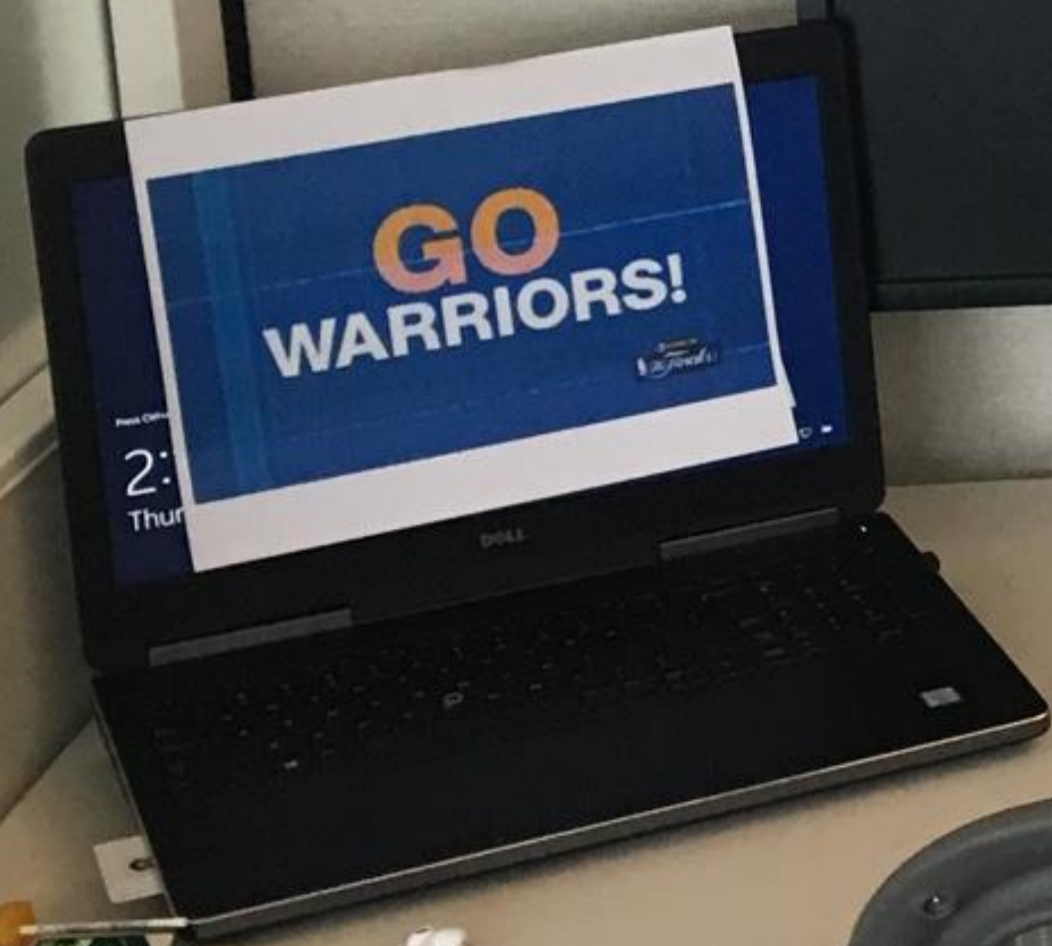
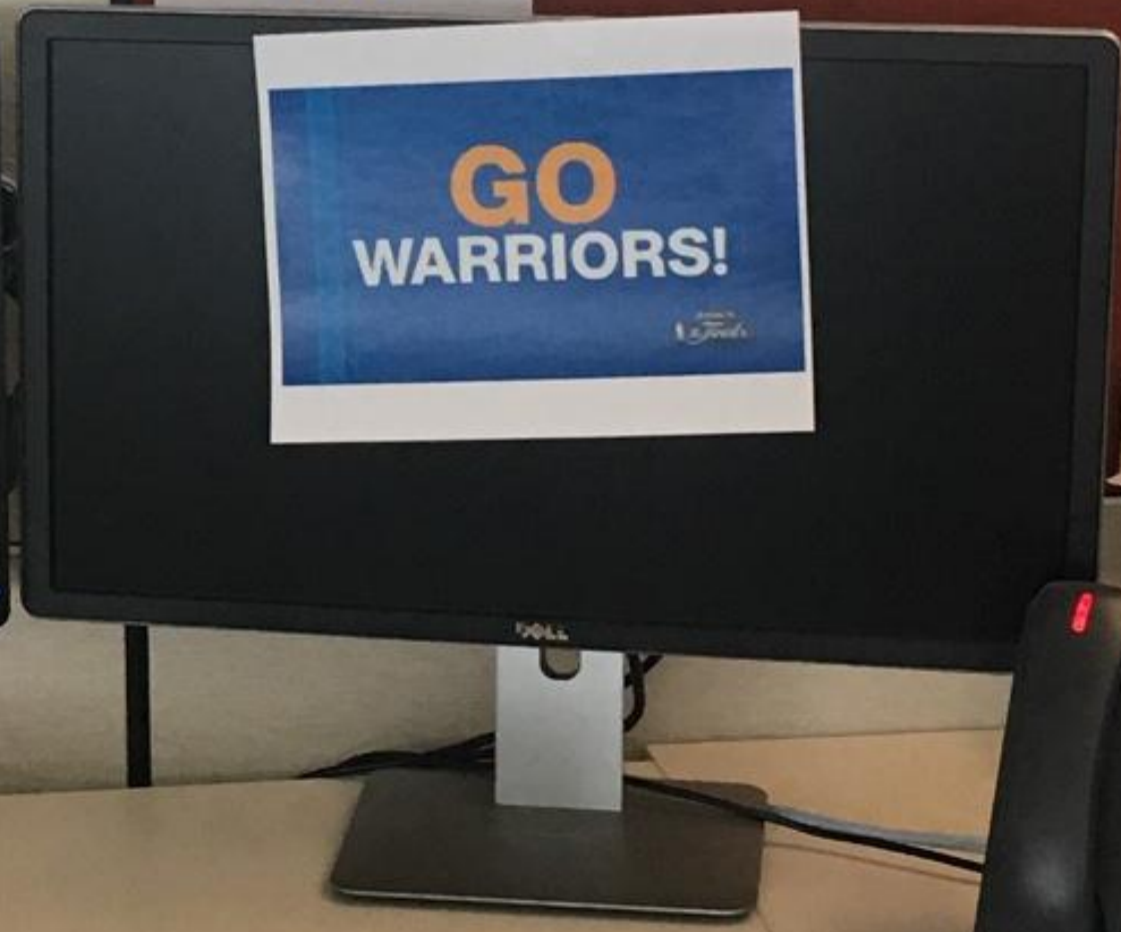
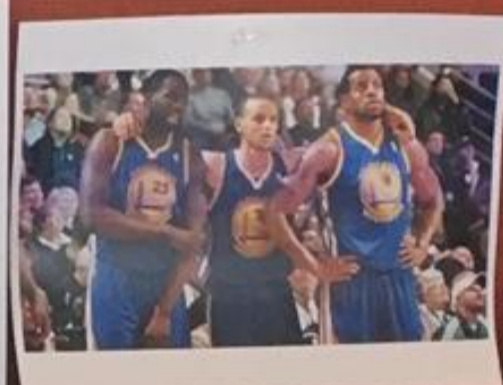
## Software Engineer in Test

Founder of Women Who Test Tel Aviv.  
Testing in prod skeptic to practitioner.

### My Superpowers

- Turning product requirements into test cases
- Breaking features prior to launch
- Testing my coworkers







**Forest**  
**Environment**



# What's wrong with staging environments?

01.

Staging environments are expensive to maintain

02.

Staging test results do not always match production test results

03.

Production includes data that staging doesn't have

04.

No one cares if staging is broken

05.

The load in staging does not match production

“I love my staging  
environment”

~No one Ever  
Ever.

**TEST IN PRODUCTION**

**WHAT COULD GO WRONG**



**What is the first thing you do  
right after you deploy a feature?**

**You go to production and test it.**

# Testing in production only works when..

01



The whole team  
owns product  
quality

02



You test early and  
often

03



You trust your team  
and your product

**Fear.**



# I know it's risky.

\_Can affect real end users

\_Can affect reporting and analytics > business decisions

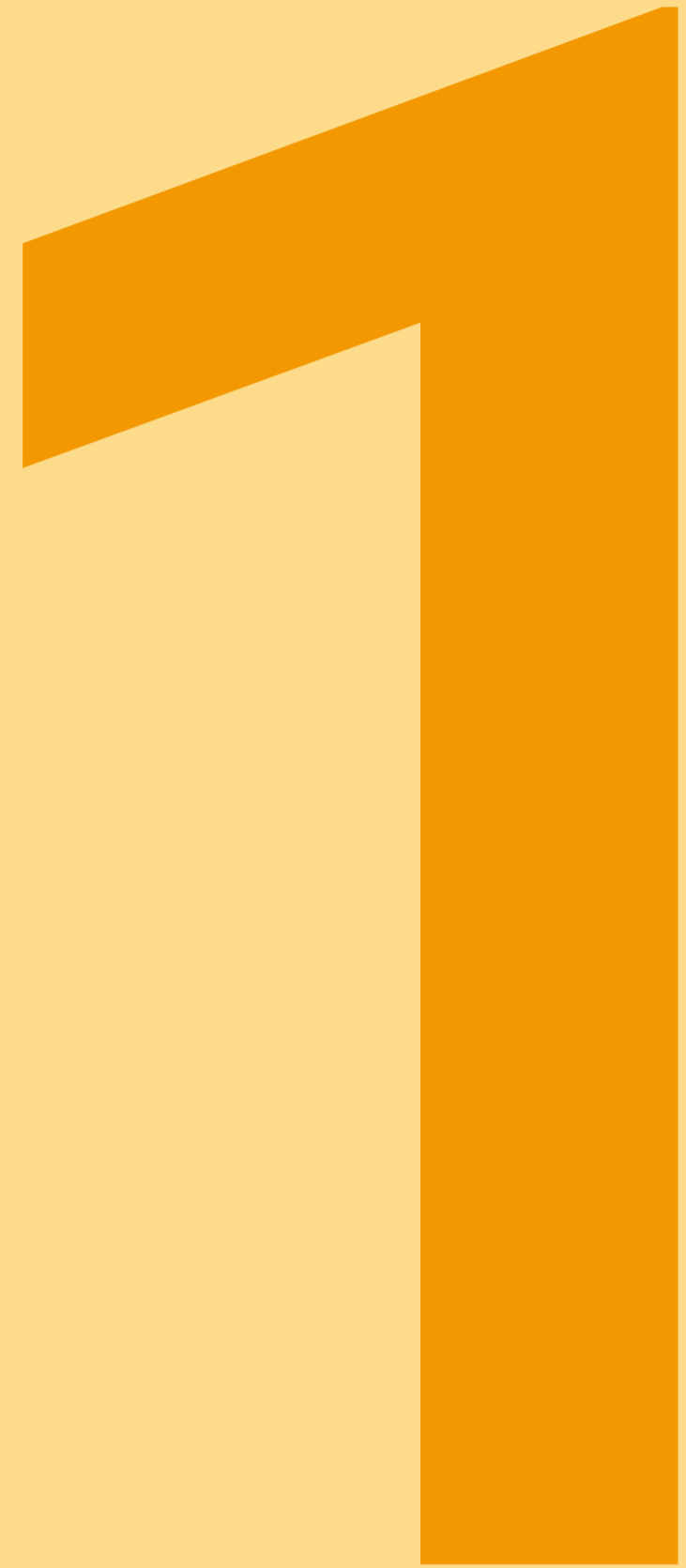
\_Can affect third parties that your software is integrated with

**Let me show you how  
to do it safely**

**We will address the  
risks and set this up  
together**

A meme featuring a man with a shocked expression, wearing a blue shirt and a brown jacket, with the text "TELL ME HOW!" overlaid at the bottom.

**TELL ME HOW!**



**Install the  
necessary  
tools**



# Feature Flagging

- \_ It is a way to decide who sees which features
- \_ It's used to hide, enable, or disable the feature during run time



# PRODUCTION

Only these people see the changes

Feature Flag



Target Users:  
Tester  
Devs  
Product  
Design  
Bots



These people (users) do not see any changes



# PRODUCTION

While the feature flag is off :

- \_ Test requirements
- \_ Open defects
- \_ Write automation scripts
- \_ Verify design
- \_ **Validate proper functionality**

Feature Flag



Target Users:

- \_ Tester
- \_ Devs
- \_ Product
- \_ Design
- \_ **Bots**



# PRODUCTION

Feature Flag



Target Users:

- \_ Tester
- \_ Devs
- \_ Product
- \_ Design
- \_ **Bots**



These people see the bugs and fix them

These people do not see the bugs

# PRODUCTION

Feature Flag



Target Users:

- \_ Tester
- \_ Devs
- \_ Product
- \_ Design
- \_ **Bots**



# PRODUCTION



Feature Flag



Target Users:

- \_ Tester
- \_ Devs
- \_ Product
- \_ Design
- \_ Bots**

Everyone sees the new feature



# PRODUCTION

## Feature Flag



### Target Users:

- \_ Tester
- \_ Devs
- \_ Product
- \_ Design
- \_ Bots




# Let's do it in Split.io

✕


## Splits

Create a feature flag or experiment in your application.


Create Split




OWNED BY ME

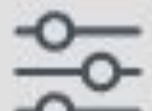



STARRED BY ME



SEARCH

**Sort:** Split Name A→Z 

 Filter 

# Create a split

Once saved, the split name and traffic type cannot be changed.

**Name** Use this name in your code.

## Traffic Type

Select traffic type



device

employee

location

user

visitors

Start typing a tag. Use tags to organize your splits by team, feature, or however you'd like.


**Description** (optional)

Add a description. As best practice, it's good to always add a description to explain the functionality controlled by this split.



# Really\_Cool\_Freature

This is a test.

☆ | Traffic Type: Location | Tags: None 

**Environment**  Production 

- Targeting rules
- Live impressions
- Metrics impact

Permissions:  On | Editors: 2 Groups 



**No targeting rules have been defined**


Add targeting rules to the **Production** environment to set your treatments and define your targeting criteria for this Split.

Define new targeting rules

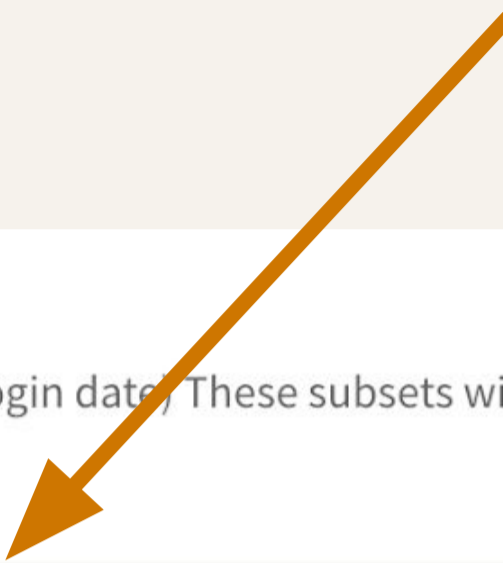
[Add Rules](#)

or

Copy targeting rules from

Select environment 

Notice the test user  
inside the feature  
flag



### Set Targeting Rules

Target specific subsets of your customers based off of any attribute you want. (e.g. location or last-login date) These subsets will be placed into a specific treatment or we'll take these subsets and randomly distribute customers in that subset between your treatments based off of percentages you decide.

IF user  is in list

serve  on

Add Rule

### Set The Default Rule

For any of your customers that weren't assigned a treatment in the sections above, use this section to place them into a treatment or randomly distribute these customers between your treatments/ variations based off of percentages you decide.

serve  off

# The Code

```
try {  
  // this code points to a feature flag where we check  
  // if the user is able to perform a certain action  
  if (Really_Cool_Feature === 'on') {  
    do something  
  
  };  
} else {  
  do something else  
}  
} catch (error) {  
  
}
```

# The Test

```
user.login(robot1@wework.com);
```

```
this.page.goTo(www.google.com);
```

```
this.page.waitFor(searchBar);
```

```
...
```

```
...
```

```
...
```

```
expect(really_cool_feature).toWork();
```

The best part?



**AUTOMATE**



# Automation Framework

**Why?**

**It's not scalable to manually test every feature in prod after you deploy it**

**AUTOMATE**



# Automation Framework

- \_ Easy to adopt
- \_ Easy to debug
- \_ Good reporting
- \_ Support community

# Job Scheduler



I will run the  
tests every  
30 minutes  
for you



# Job Scheduler

Why not just run the tests in a loop?

## It's Overkill

You can set up the tests to run in your build pipeline as well

## Trust your tests

Run them 20 times in production and watch them.

# Alerting

- \_ Choose an alerting tool that can be integrated with your job scheduler**
- \_ Have a rotation of who is on call to handle alerts**

# Tools

Feature flagging



Automation framework



+



Job Scheduler



Alerting



2

**Carefully  
create  
test data**

# Problem

We need a way to create and manipulate test data in production **without affecting real end users** or any data and analytics

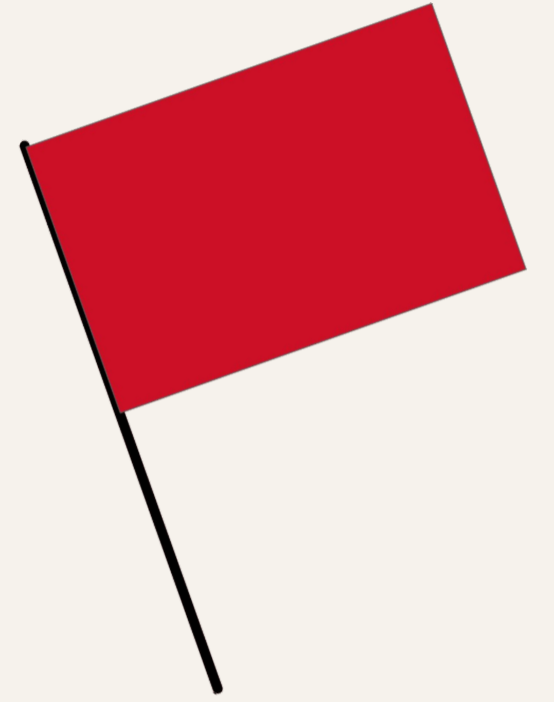
# Solution

Create a consistent naming convention for test users



# Solution

Backend flagging system used to identify test entities



# Solution

**\_In your automation scripts, you make sure that your test entities should only interact with each other**

**\_ We were able to create and manipulate testing entities in production by following these specific predetermined guidelines.**



3

**Write**  
**your**  
**tests**

# BDD



```
erkin.txt
*** Settings ***
Library CalculatorLibrary

*** Test Cases ***
Addition
    Given calculator has been cleared
    When user types "1 + 1"
    and user pushes equals
    Then result is "2"

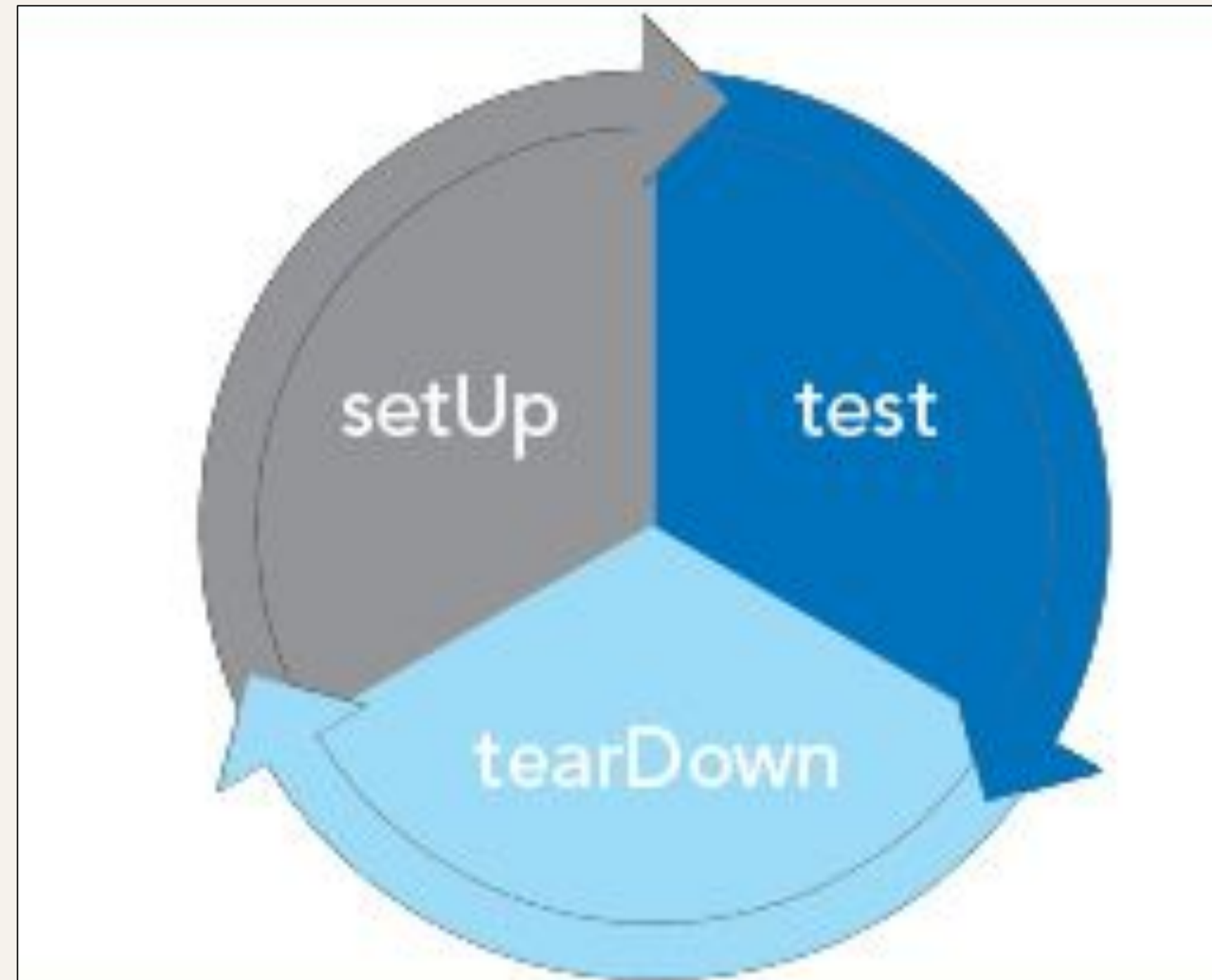
*** Keywords ***
Calculator has been cleared
    Push button C

User types "${expression}"
    Push buttons    ${expression}

User pushes equals
    Push button    =

Result is "${result}"
    Result should be    ${result}
```

# Setup & Teardown



4

**Deploy to  
a production  
canary**

# What's a production canary?



It's when you slowly roll out a change to a small subset of users before rolling it out to the entire infrastructure to minimize impact if something goes wrong

# Why use a production canary?

- \_ Canary launches provide risk mitigation
- \_ Do you want 100% of your users to encounter the issue or 1%?

# Why use a production canary?

- \_ Quickly identify the issue that might impact your entire user base
- \_ Roll back easily to a good version
- \_ Fix the issue in a controlled environment

# Production Canaries



**Production  
Canary**



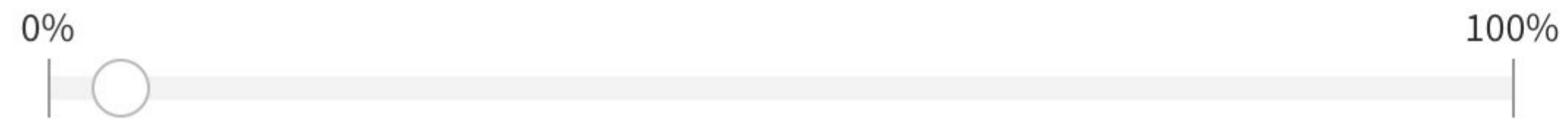


# Canary Setup

## Split.io

### ▼ Allocate Traffic

Advanced: If you're running an experiment and want to limit the number of customers that are exposed to it, you can select a specific percentage to include in the experiment.



% of total User in this Split

## Netlify

### Split test #1 Running

Split Testing is on. Last update on Sep 10 (a day ago)

Branch	<input type="text" value="production"/>		Split	<input type="text" value="90"/>	%	
Branch	<input type="text" value="really_cool_feature"/>		Split	<input type="text" value="10"/>	%	

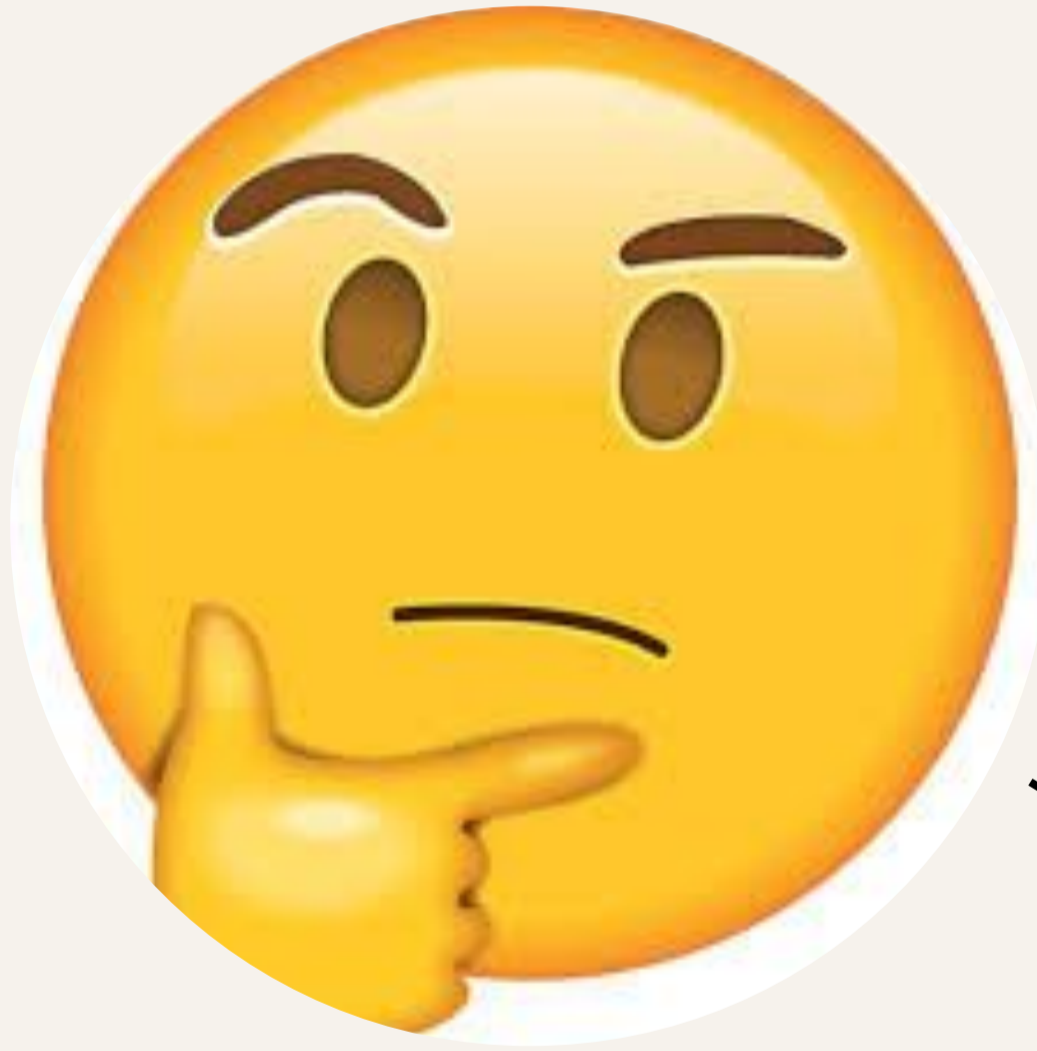
Add another branch

Stop test

# What if there's an issue with your canary code?



Analyze the issue



Is it ok if my users see this?

Yes

Keep the canary as is and fix the bug

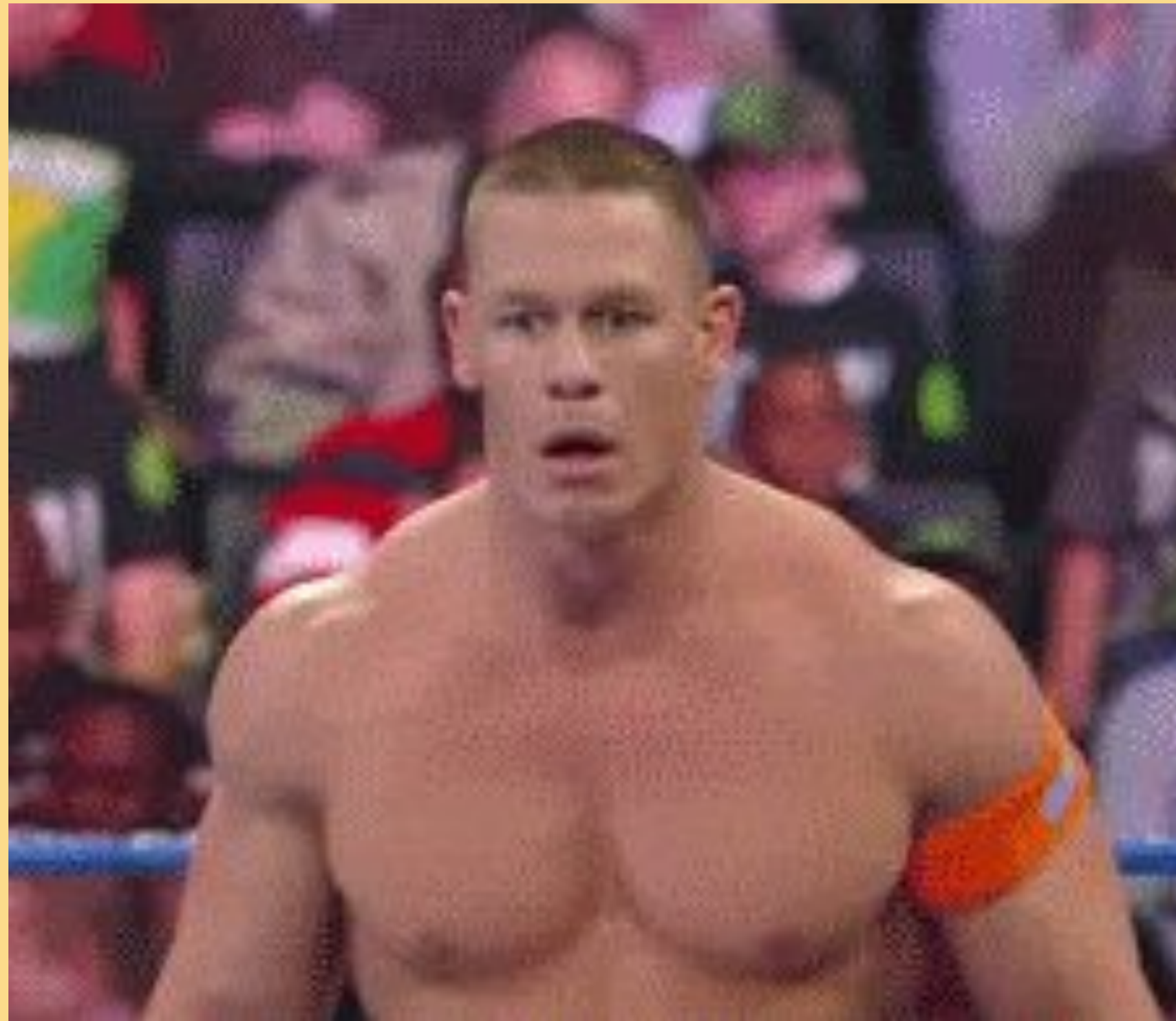
No

Return the canary to 0% traffic and fix the bug

# Risk Mitigation

- \_ Before launch, use Feature Flagging to target users**
- \_ After launch, use Production Canaries to limit audience**

# Outcome



- \_ Higher confidence
- \_ Increased developer velocity

~~Reactive~~

Proactive



# Long term effects



Tests would fail



Immediately get alerted

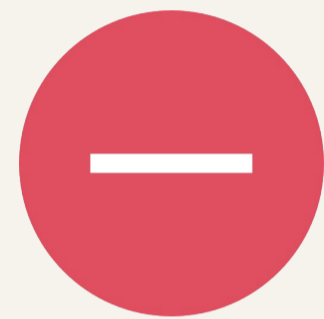


Analyze the issue right away



Resolve it ASAP

# Long term effects



**Minimize user interaction with bugs and defects**



**Ensures a great user experience**

# Performance Testing in Production

Yes, it's possible.

Do it at a time with low traffic  
and be prepared for outages

It's better to do it on purpose  
and control the outcome



# Testing in Production with third parties

# What if I can't test in prod?

**\_Spin up a containerized environment  
per build**

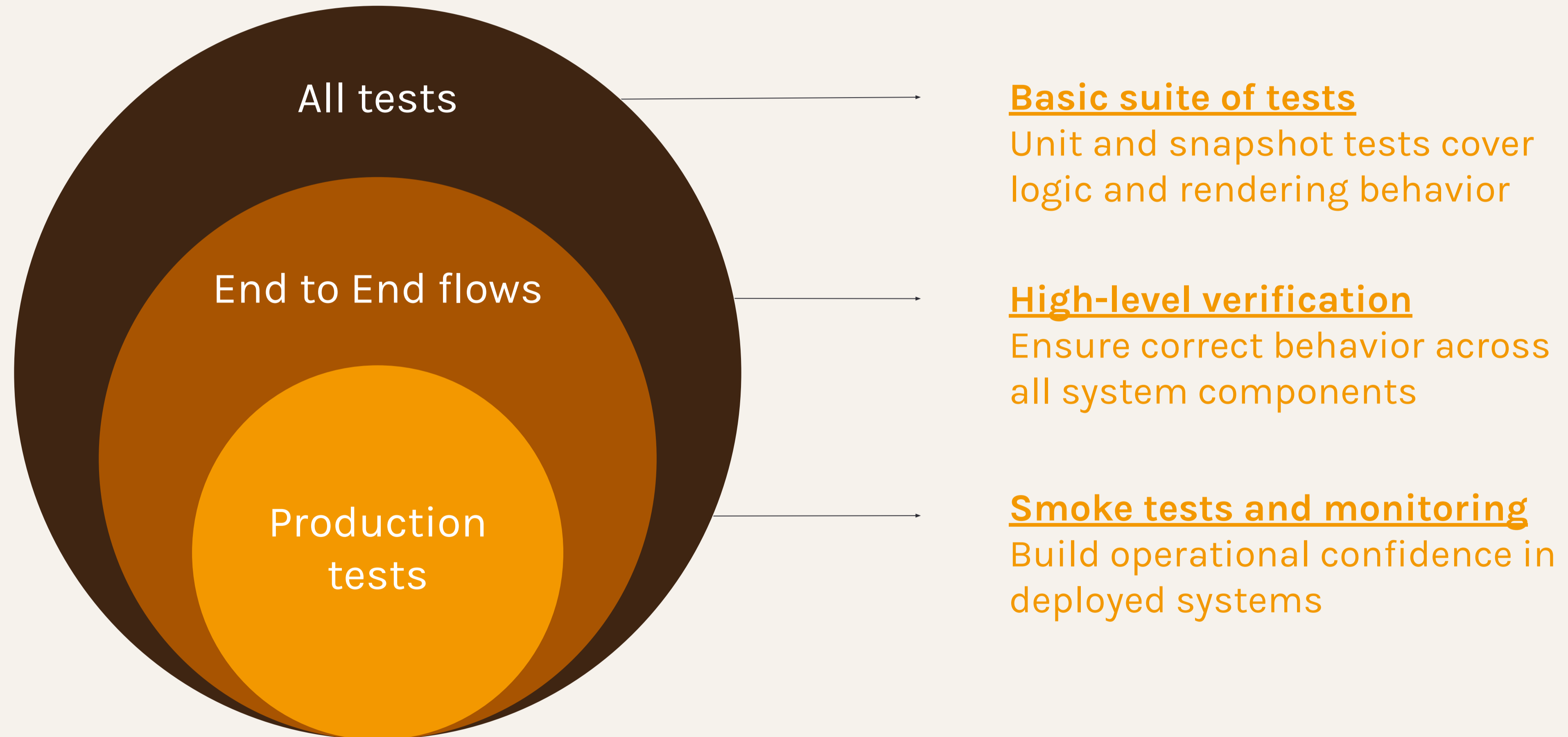
**\_Use docker to spin up microservices  
(Remember, the more dependencies, the  
more complicated)**



# What about other non production tests?

- \_ Staging environments can still be used for pre-production testing
- \_ Very important for financial systems that abide by SOX compliance
- \_ These should be tested ahead of time and not in production
- \_ Privacy-related efforts should also be tested here to minimize data breaches

# Where do prod tests operate?



# Shifting your company's testing culture



## Explain why the pros outweigh the cons

\_ Is your staging environment unreliable?

\_ Are there frequently issues that you think could have been caught if you were testing in prod?

## Use examples from the past

\_ Do you remember when we merged xyz and it caused this issue in production?

\_ Do you think if we tested xyz in production that that issue could have been caught?

\_ Do you remember when we tested xyz inside and out in staging and then we deployed to prod and it broke?

## Propose a path forward

\_ Have you heard about this cool thing called feature flagging? Can I take some time in the next sprint to see if we could benefit from it?

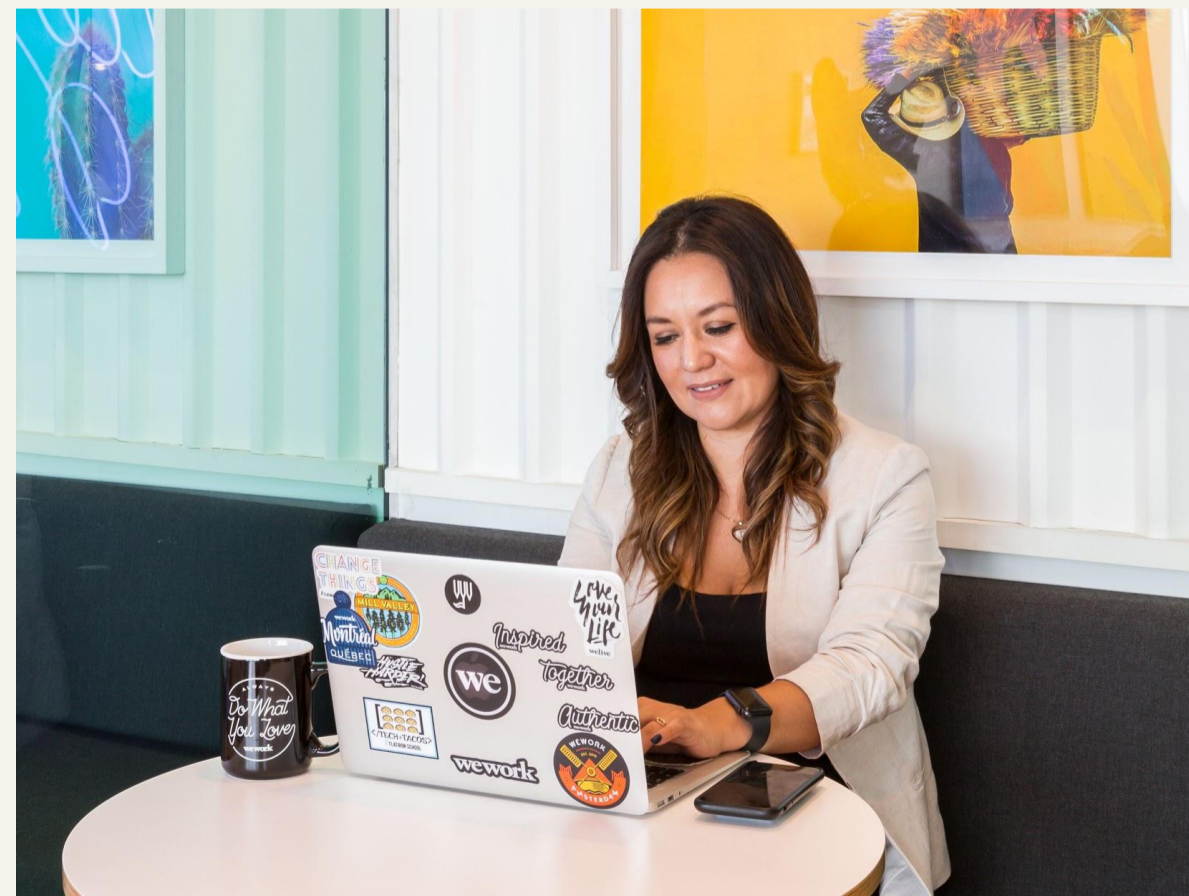
\_ If we were to start testing in prod, which tests do you think would bring us the most value?

# Shifting my company's testing culture

\_ We used to be scared of deploying new features. We used to have debates of whether or not to deploy code on Fridays

\_ Once we started moving more and more things to testing in prod, these discussions and this fear stopped

\_ The lead time to know if something was wrong was reduced and the confidence in the releases increased



# How to deal with naysayers



- \_ Staging will never fully represent prod
- \_ Staging is a sunk cost
- \_ They're not your target audience

# Try it!

## Step 1



Install Tools -  
Feature  
flagging,  
automation  
framework,  
job  
scheduler,  
alerting tool

## Step 2



Create test  
data - Make  
sure your test  
entities only  
interact with  
each other

## Step 3



Write Your  
Tests - Make  
sure your  
teardown  
cleans up  
your test

## Step 4



Launch your  
feature  
behind a  
feature flag

## Step 5



Deploy To  
Production  
Canary

## Step 6



Have a drink!



**No one cares if your  
feature is working  
in staging, we care  
if its working in  
prod**

**The only way to  
know if its working  
in prod is to test it  
in prod**

**LOONEY TUNES**

**Talia Nassi**

**[talia.nassi@wework.com](mailto:talia.nassi@wework.com)**