Snapshot and Frontend Unit Testing in Software

Presenters

Katherine Chen
Software Engineer - Backend

Patrick O'Brien
iOS Software Engineer

Katherine

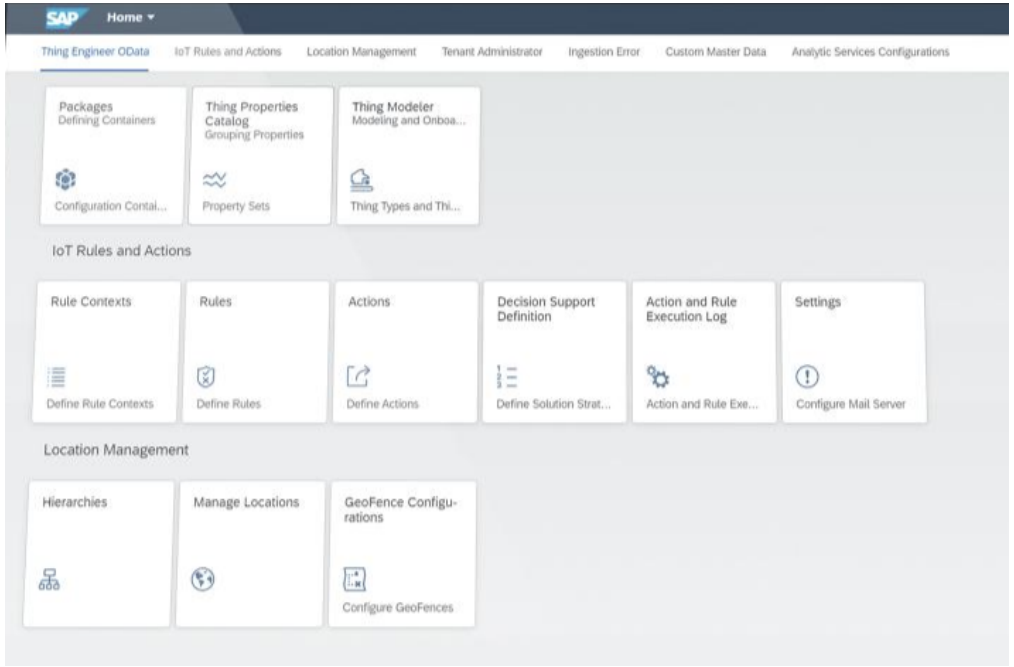
Background:

- Toronto, Canada -> Sacramento, California
- Went to University of California, San Diego (UCSD)
  - Major: Mathematics-Computer Science
  - Minor: Management Science
  - Internship @ CGI - Business Analyst Intern
  - On-campus jobs in tutoring and student wellness centers

Katherine Chen
Backend Software Engineer

Responsibilities

- SAP IoT Team
  - Define rules and event-based actions to enriched IoT data
    - Create rules based on data enriched via a business context
    - Trigger other actions
    - Send notifications about actions
  - Integrate into suite of SAP products
  - Technology and some topics used:
    - Java, Spring, Kafka, Mockito, cloud foundry
    - OOP, multi-threading, backend unit testing



Patrick

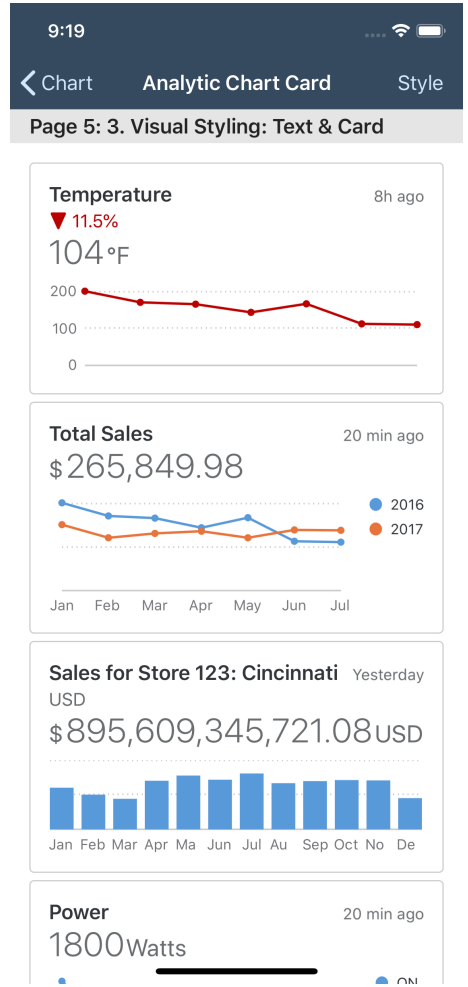
Background:

- Michigan -> Indiana -> Backpack Overseas -> Indiana
- Went to Indiana University, Bloomington
  - Major: Computer Science
  - Minor: Human-Centered Computing
  - Internship @ Liberty Mutual – IT Compliance
  - On-campus jobs: Assistant Java Instructor

Patrick O'Brien
iOS Software Engineer

Responsibilities

- SAP iOS Fiori SDK Team
  - Create a software development kit with built in components
    - Welcome Screens, Logins
    - Charts, Lists,
    - Buttons, Signature Forms
    - Maps, etc
  - Research: Leverage new iOS features
    - Explore Augmented Reality
    - Computer Vision to infer image content
  - Technology and some topics used:
    - Swift, iOS, and XCode. Object Oriented Programming, Design Patterns (MVC, MVVM)
    - Creating new features, Debug existing features, snapshot testing framework



Software Testing

Software Testing:

- Unit, regression, smoke, beta, black box, system integration, and many more
- Why test?
  - Saves time in the long-run
  - Helps prevent old bugs from being re-introduced
  - For developers, helps teach how to write better code
- Unit testing
  - Testing an individual component of unit of software
    - Usually the smallest pieces of code that can be tested individually and independently from the rest of the code
- Snapshot Testing
  - Compare reference image of screen (what the screen should look like) to a new screenshot after code changes

Frontend Unit Testing: Jasmine + Karma

Frontend testing tools:

- Jasmine = testing framework
  - Provides the rules and keywords on how to write/execute tests
- Karma = test runner
  - Provides the environment for you to run your test by usually spawning up browsers or mock servers

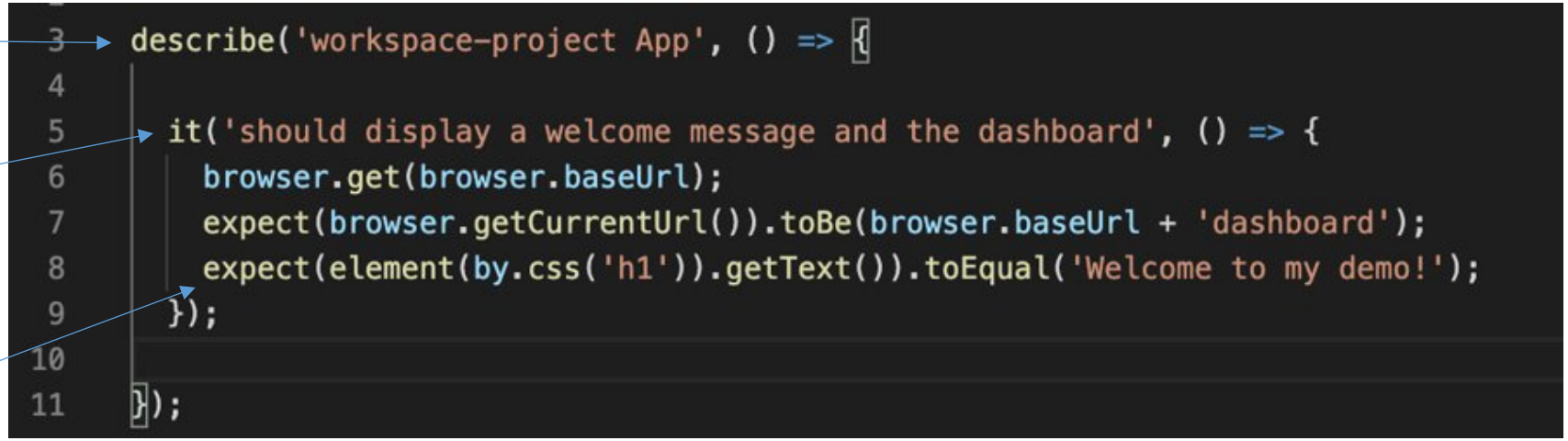
Unit Test Structure

Anatomy of a Jasmine frontend unit test:

A test suite aka a group of specs

An individual unit test aka spec

Expectations on code behavior vs the actual behavior



Unit Test Structure Cont.

- Test suites (describe()) can be nested within each other, but unit tests cannot (it())
- You can have multiple expectations within a single unit test
- You can suppress unit tests and test suites from being run by placing an 'x' in front of the keyword:
  - Ignored unit tests: xit()
  - Ignored test suites: xdescribe()

```
describe("A spec", function() {
  var foo;

  beforeEach(function() {
    foo = 0;
    foo += 1;
  });

  afterEach(function() {
    foo = 0;
  });

  it("is just a function, so it can contain any code", function() {
    expect(foo).toEqual(1);
  });

  it("can have more than one expectation", function() {
    expect(foo).toEqual(1);
    expect(true).toEqual(true);
  });

  describe("nested inside a second describe", function() {
    var bar;

    beforeEach(function() {
      bar = 1;
    });

    it("can reference both scopes as needed", function() {
      expect(foo).toEqual(bar);
    });
  });
});
```

Using Matchers

For asserting unit tests:

- Format is expect(actual value).matcher(expected value)
  - example: expect(something).toBe(true)
- To invert assertions, add .not in front of the matcher
  - example: expect(something).not.toBe(true)
- Popular matchers:
  - .toBeDefined()
  - .toBeFalse(), .toBeTrue()
  - .toBeInstanceOf(Class)
  - .toContain(something)
  - .toThrow(exception)
  - .toBeGreaterThan(num), .toBeLessThan(num)
- Or create your own, by overriding the compare() function

```
it("and has a positive case", function() {
  expect(true).toBe(true);
});
```

```
it("and can have a negative case", function() {
  expect(false).not.toBe(true);
});
```

JavaScript - Truthy and Falsy

Two matchers: .toBeTruthy(), .toBeFalsy()

- more lenient versions of .toBeTrue() and .toBeFalse()
- Falsy values in JavaScript (when encountered in boolean context):
  - false
  - 0, -0
  - 0n (BigInt)
  - “”
  - null
  - undefined
  - NaN (not a number)
- On the flip side, any value evaluates to truthy if it is not one of the items from the list above

Using Spies

For mocking function calls:

- any object and one object method
  - spyOn(object, method)
  - You can mock behavior upon creation by chaining an **.and** with some common methods:
    - .callFake(function)
    - .returnValue(something)
    - .returnValues(...something)
    - .throwError(exception)
- an object with no functions to spy on
  - jasmine.createSpy(objectName)
  - This is a bare spy – no implementation behind it
- an object with multiple functions that need spying on
  - jasmine.createSpyObj(objectName, [method1, method2, etc...])

```
describe("a spy test", () => {
  var foo = null;

  beforeEach(() => {
    foo = {
      bar: "bar",
      setBar: function(value) {
        bar = value;
      }
    };

    spyOn(foo, 'setBar');
  });

  it("checks the spy", () => {
    foo.setBar("test");
    expect(foo.setBar).toHaveBeenCalled();
  })

})
```

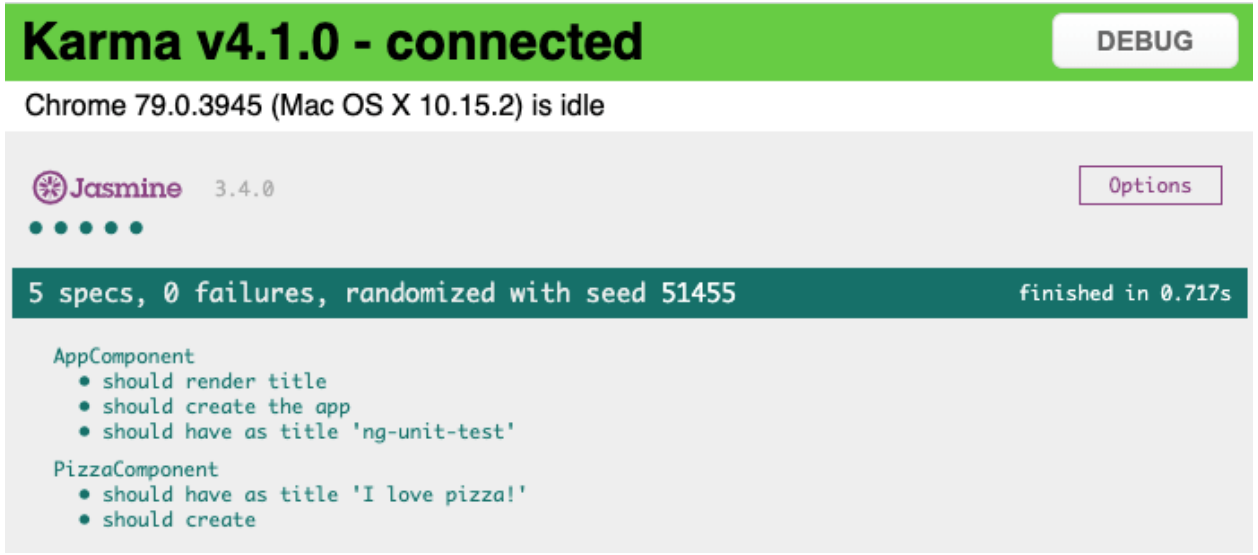
Test Doubles

- Jasmine
  - Spies
    - Stubs functions and tracks its calls
    - SpyOn
    - CreateSpy
    - CreateSpyObj
- Commonly used alongside Angular
  - Angular provides additional testing capabilities for its components
    - TestBed API – provides testing modules and methods to set up unit tests in Angular
    - Jasmine is used to provide the unit test keywords and assertions (or expectations)
- When and when not to use them?

Frontend Unit Testing (continued)

Jasmine testing alongside Karma:

- Reading unit test results
- Tests run linearly but can be run in parallel using additional plugins like "karma-parallel" (available via npm)



Jasmine vs JUnit

JUnit (+ Mockito)	Jasmine
Java testing framework (Mockito = mocking framework)	JavaScript testing framework that also provides some mocking capabilities for function calls
Test driven development	Behavior driven development*
Unit tests denoted with @Test	Unit tests (aka “specs”) denoted with the function it() - test suites are denoted with the function describe()
Test fixtures denoted with @Before, @After	Fixtures use beforeEach(), beforeAll(), afterEach(), afterAll()
Assertions made with JUnit’s Assert class	Assertions (or “expectations”) made with the function expect() (takes in the actual value) - chained alongside a Matcher function which takes in the expected value

Behavior-Driven Development Advantages

- English-like syntax
  - Better collaboration
  - Allows more people from different parts of the project to contribute to the tests (I.e. business, developers, QA, etc.)
  - Defines end-user expected behavior
- Supports agile processes
  - Encourages teams to work in quick iterations based on their user stories/use cases
- Stems from other testing methodologies like Test-Driven Development which it actively advocates for

```
describe("musicBox", function() {

    it('plays',function(){
        var musicBox = new MusicBox();
        musicBox.play();
        expect(musicBox.isPlaying).toBe(true);
    });
});
```

Snapshot Testing vs Unit Testing

Snapshot Testing

- Create a reference Image
- New Images are created on pull request after code changes
- Images are compared

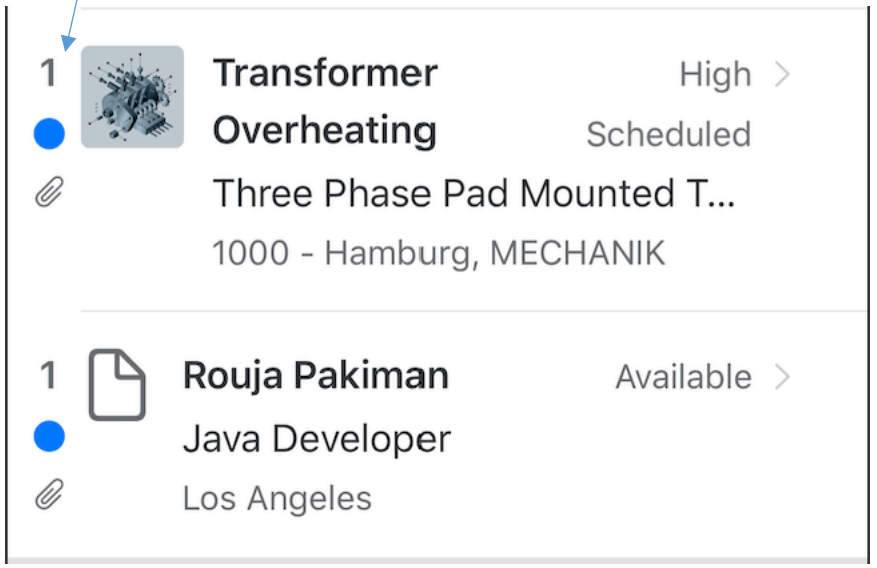
Unit Test to check icons are assigned as expected

```
640         let icons = ["1", "dot", "lock"]
641         cell.iconImages = icons
642         for (i, icon) in cell.iconImages.enumerated() {
643             XCTAssertEqual(icon.attributedText.string, icons[i])
644         }
```

Function to add styling to icons

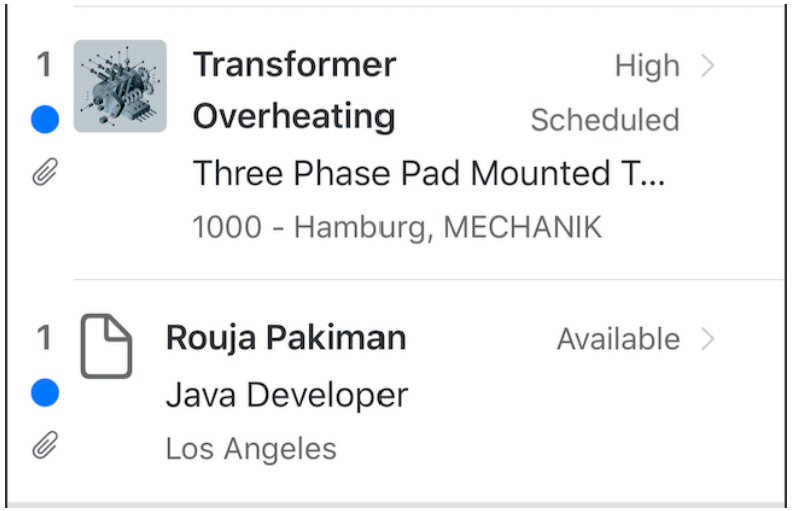
```
case is String:
    attributes.updateValue(UIFont.preferredFont(forTextStyle: .headline), forKey: .font)
    attributes.updateValue(UIColor.preferredFioriColor(forStyle: .primary3), forKey: .foregroundColor)
```

Icons are styled

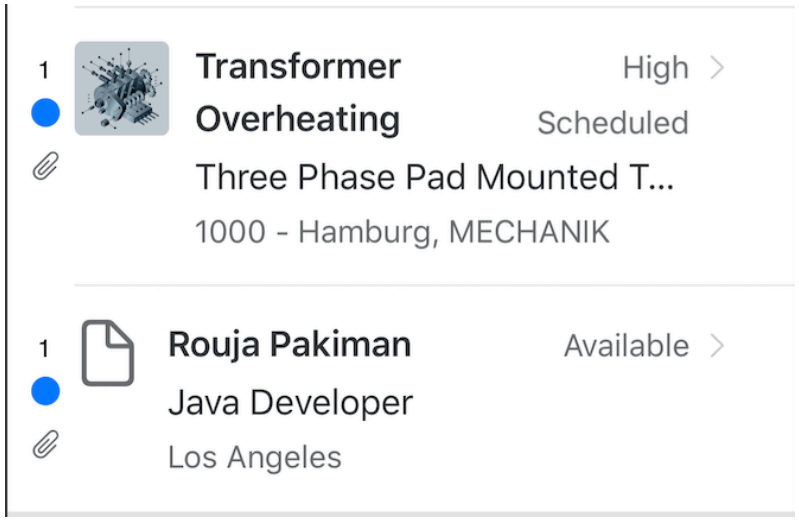


Snapshot Testing (continued)

Reference Image

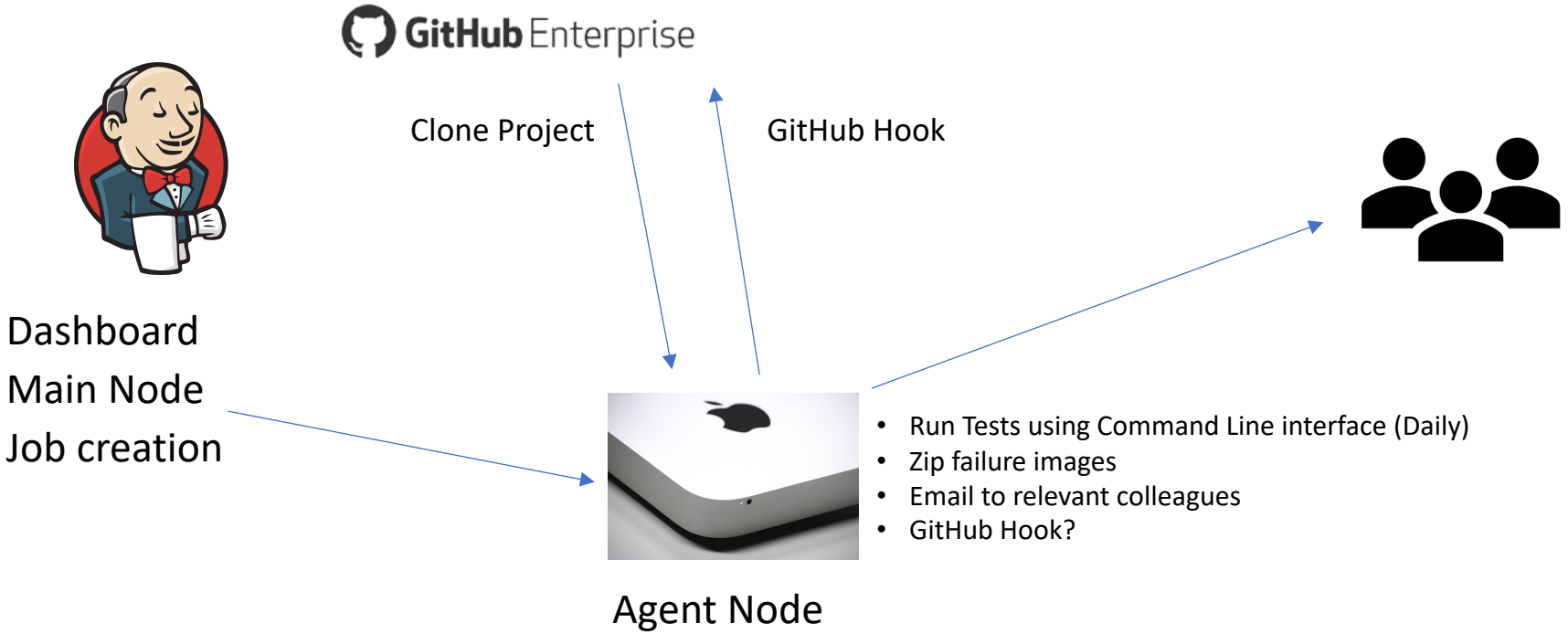


Failure Image



Diffing Image

Continuous Integration



iOS Snapshot Testing

- SAP has Jenkins As A Service (JAAS)
- Jenkins Job pulls repository from Github
- Mac Mini is needed to build the project
- Will eventually be updated to Github Actions, why not yet?

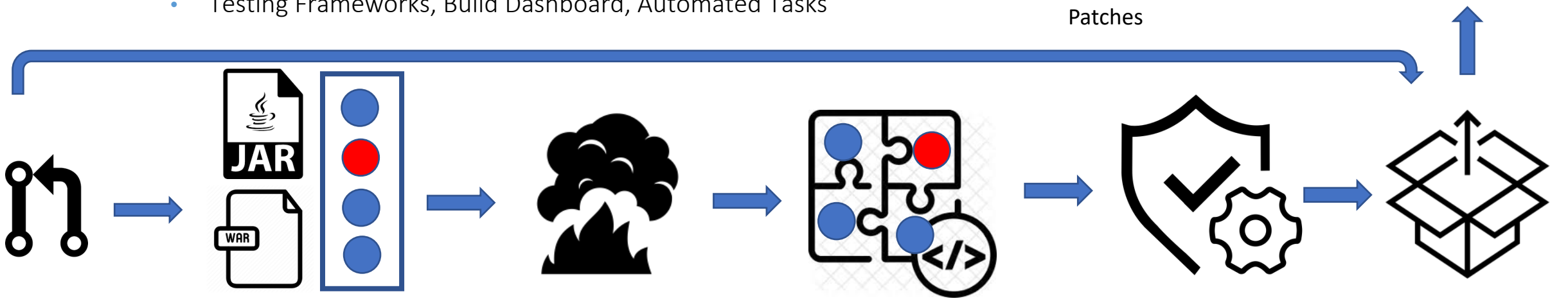
Continuous Integration - SuccessFactors

SAP SuccessFactors: Human Capital Management Company

- E.g. E-learning, job application portal, organization chart, time tracking

DevOps Platform

- Facilitate the build pipeline and create tools to support the developer
- Technology and tool examples
  - Java, Groovy, Jenkins, Virtual Machines, Gradle, Jfrog Artifactory
  - Testing Frameworks, Build Dashboard, Automated Tasks



Consequences of not Testing

- More bugs are reported/created
- Less time for new feature development
- Stakeholders have to wait for bug fixes
- Pushes critical releases to customers past deadlines

Thanks!

Q & A