DEEP-DIVE WITH SELENIUM 4.0

MANOJ KUMAR

@manoj9788

HUSTEF
HUNGARIAN SOFTWARE TESTING FORUM

LAMBDATEST

AGENDA

START TIME	END TIME	DURATION	TOPIC
9:30 AM	9:45 AM	15 Mins	Introductions & Stage Setting
9:45 AM	10:15 AM	30 Mins	Exercise 1: Implement your first Selenium test
10:15 AM	10:45 AM	30 Mins	Test Automation | Continuous Testing | SDLC
10:45 AM	11:00 AM	15 Mins	Coffee Break
11:00 AM	12:00 PM	60 Mins	Exercise 2: Solving for Common Challenges in Selenium tests
12:00 PM	12:30 PM	30 Mins	Overview of Design Patterns & Test Data
12:30 PM	1:00 PM	30 Mins	Overview of Selenium 4.0 features
1:00 PM	1:45 PM	45 Mins	Lunch
1:45 PM	2:30 PM	45 Mins	Exercise 3: Window APIs & Relative Locators
2:30 PM	3:30 PM	60 Mins	Exercise 4: WebDriver Bidi / Chrome DevTools Protocol
3:30 PM	3:45 PM	15 Mins	Break
3:45 PM	4:30 PM	45 Mins	Exercise 5: Setting up your Selenium Grid
4:30 PM	5:00 PM	30 Mins	Exercise 6: Advanced Selenium Grid
5:00 PM	5:30 PM	30 Mins	Q&A: Ask Me Anything

General rules!

- Make in Interactive
- Share your stories
- And.. Please be on time!

Exercise 1

Exercise 1: Implement your first test – 30 mins

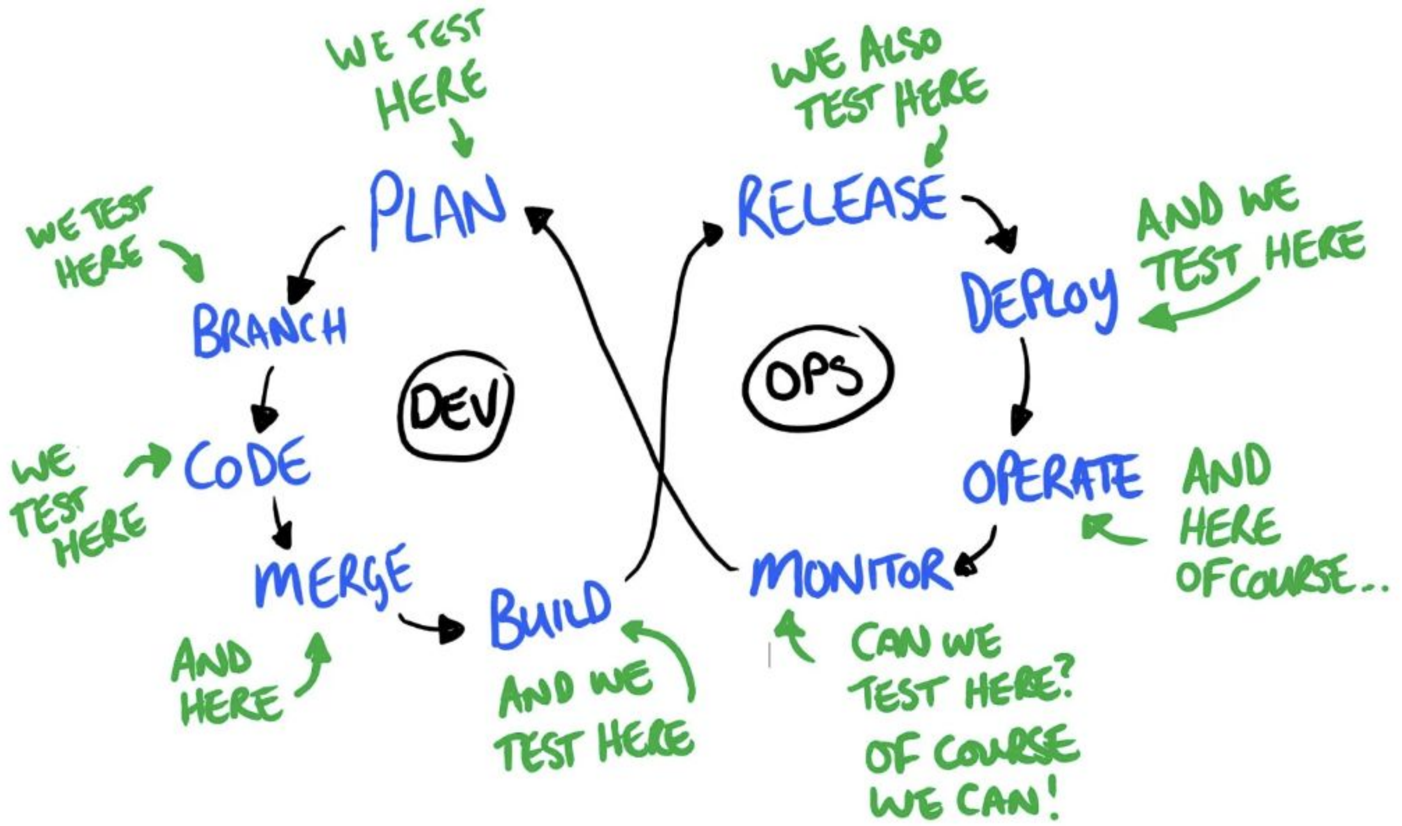
- Navigate to https://www.lambdatest.com/selenium-playground/input-form-demo and automate the form!
- Run the script in any browser of your choice
- Let's discuss the solution

Exercise 1: What did we learn?

- Learnt basic setup – environment setup
- Getting started with simple script

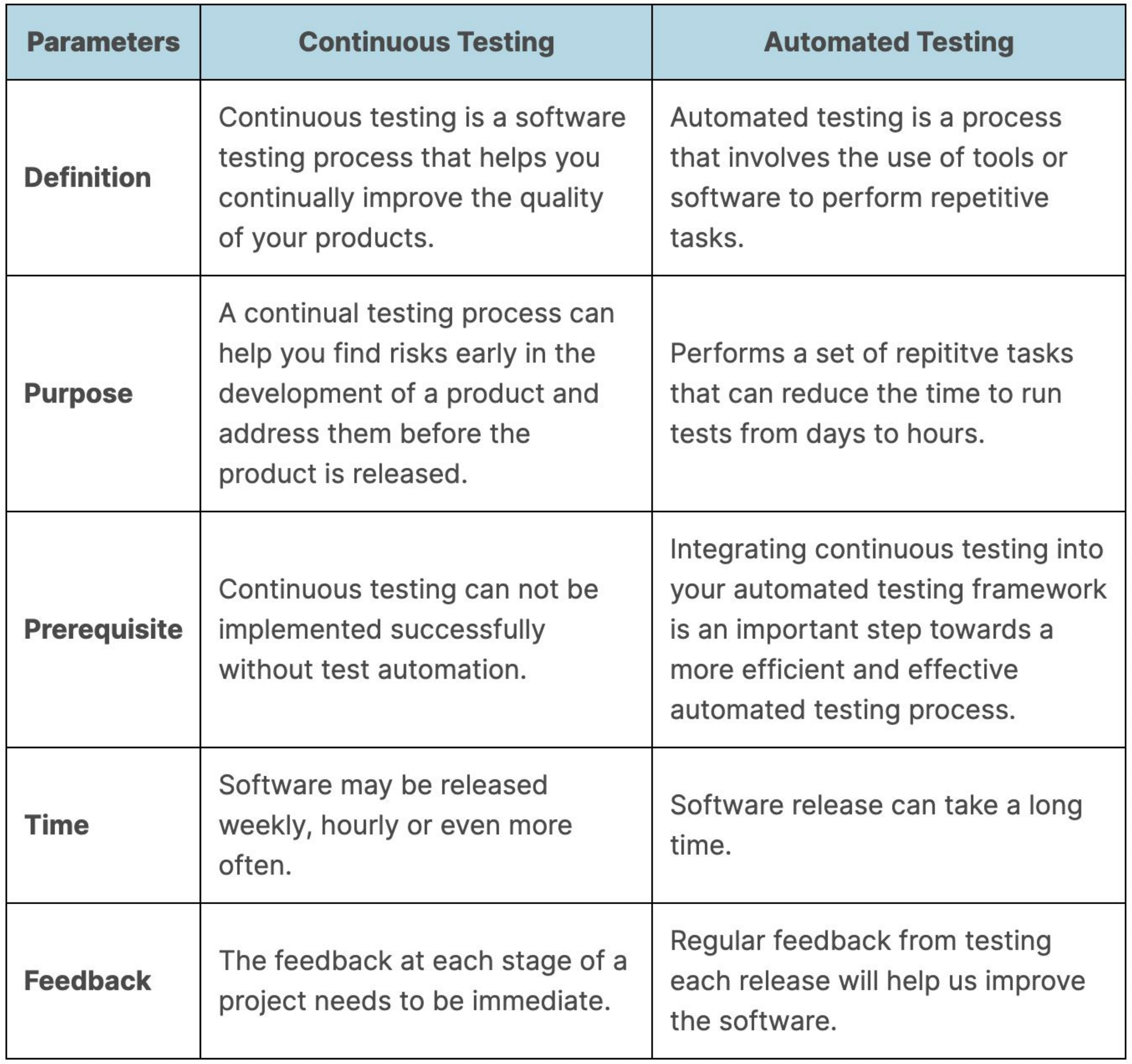
Test Automation | Continuous Testing

Test at all the stages…



@manoj9788

Test Automation <> Continuous Testing



Parameters	Continuous Testing	Automated Testing
Definition	Continuous testing is a software testing process that helps you continually improve the quality of your products.	Automated testing is a process that involves the use of tools or software to perform repetitive tasks.
Purpose	A continual testing process can help you find risks early in the development of a product and address them before the product is released.	Performs a set of repititve tasks that can reduce the time to run tests from days to hours.
Prerequisite	Continuous testing can not be implemented successfully without test automation.	Integrating continuous testing into your automated testing framework is an important step towards a more efficient and effective automated testing process.
Time	Software may be released weekly, hourly or even more often.	Software release can take a long time.
Feedback	The feedback at each stage of a project needs to be immediate.	Regular feedback from testing each release will help us improve the software.

Do you run your tests in the CI?

@manoj9788

Do you run your tests in the CI? If yes, how long?

@manoj9788

What is the test passing percentage?

Do you re-run failing tests?

@manoj9788

Do you deal with flaky tests?

@manoj9788

Do you deal with flaky tests?

@manoj9788

Common Challenges

- Multiple browser & form factors
- Changes in element locators
- Flaky tests
- Test data management
- Test environments*(?)*

Exercise 2

Exercise 2: Solving for Common Challenges

- *Use the same script from Exercise 1 and make changes to your code so it can run on multiple browsers*
- *Change locators*
- *Add assertions*

@manoj9788

Exercise 2: What did we learn?

- Understood how to refactor code for browser options
- Understood usage of element locators

Overview: Design Patterns & Test Data management

Why do we need Design patterns?

What all we have?

- Page Object Pattern
- Screenplay Pattern
- Composition
- Factory
- Singleton

Page Object Patterns

- There is a clean separation between the test code and page-specific code, such as locators (or their use if you’re using a UI Map) and layout.

- There is a single repository for the services or operations the page offers rather than having these services scattered throughout the tests.

Page Object Pattern (contd.)

```
/***
 * Tests login feature
 */
public class Login {

  public void testLogin() {
    // fill login data on sign-in page
    driver.findElement(By.name("user_name")).sendKeys("userName");
    driver.findElement(By.name("password")).sendKeys("my supersecret password");
    driver.findElement(By.name("sign-in")).click();

    // verify h1 tag is "Hello userName" after login
    driver.findElement(By.tagName("h1")).isDisplayed();
    assertThat(driver.findElement(By.tagName("h1")).getText(), is("Hello userName"));
  }
}
```

Page Object Pattern (contd.)

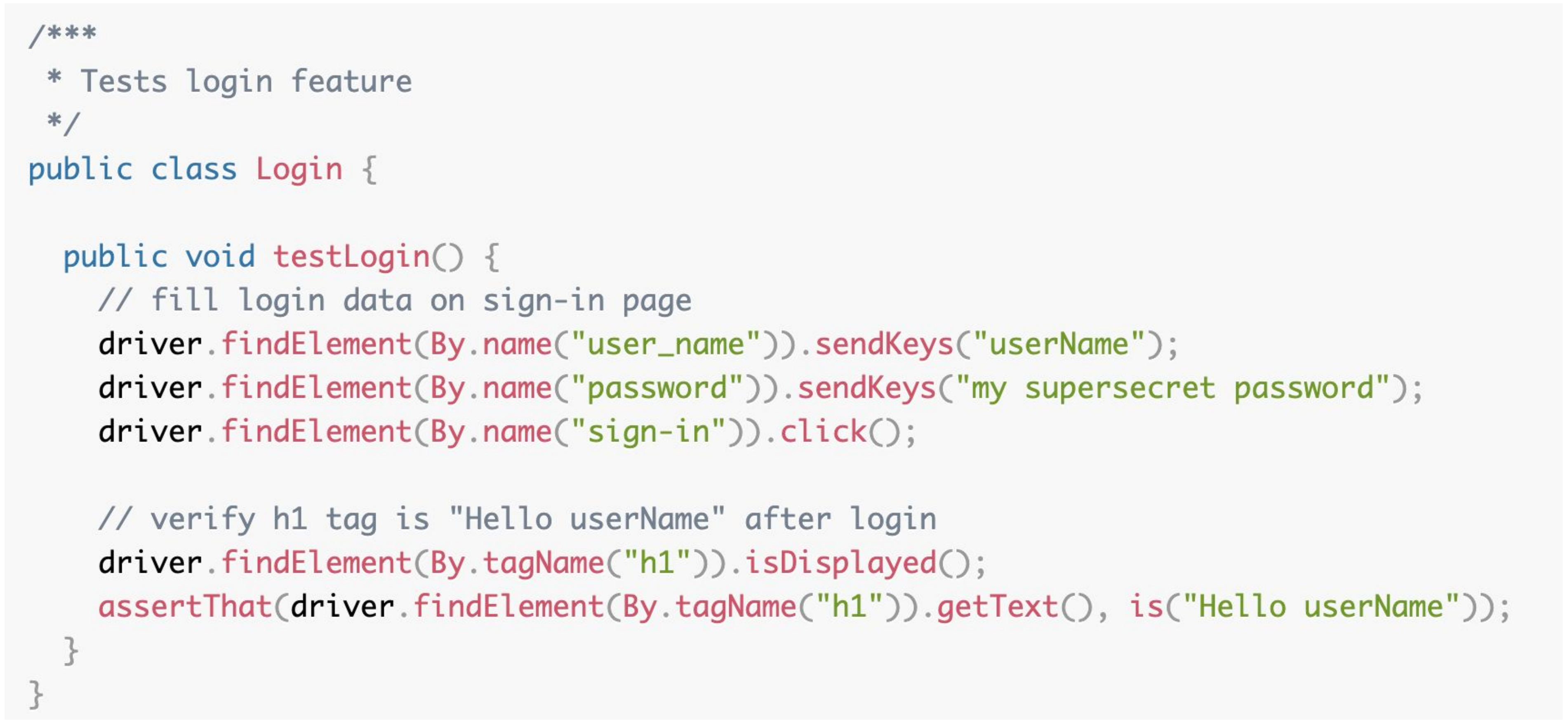
```java
/**
 * Page Object encapsulates the Sign-in page.
 */
public class SignInPage {
  protected WebDriver driver;

  // <input name="user_name" type="text" value="">
  private By usernameBy = By.name("user_name");
  // <input name="password" type="password" value="">
  private By passwordBy = By.name("password");
  // <input name="sign_in" type="submit" value="SignIn">
  private By signinBy = By.name("sign_in");

  public SignInPage(WebDriver driver){
    this.driver = driver;
  }

  /**
    * Login as valid user
    *
    * @param userName
    * @param password
    * @return HomePage object
    */
  public HomePage loginValidUser(String userName, String password) {
    driver.findElement(usernameBy).sendKeys(userName);
    driver.findElement(passwordBy).sendKeys(password);
    driver.findElement(signinBy).click();
    return new HomePage(driver);
  }
```

```java
/**
 * Page Object encapsulates the Home Page
 */
public class HomePage {
  protected WebDriver driver;

  // <h1>Hello userName</h1>
  private By messageBy = By.tagName("h1");

  public HomePage(WebDriver driver){
    this.driver = driver;
    if (!driver.getTitle().equals("Home Page of logged in user")) {
      throw new IllegalStateException("This is not Home Page of logged in user," +
            " current page is: " + driver.getCurrentUrl());
    }
  }

  /**
    * Get message (h1 tag)
    *
    * @return String message text
    */
  public String getMessageText() {
    return driver.findElement(messageBy).getText();
  }

  public HomePage manageProfile() {
    // Page encapsulation to manage profile functionality
    return new HomePage(driver);
  }
  /* More methods offering the services represented by Home Page
  of Logged User. These methods in turn might return more Page Objects
  for example click on Compose mail button could return ComposeMail class object */
}
```

LAMBDATEST

@manoj9788

Page Object Pattern (contd.)



```
/***
 * Tests login feature
 */
public class Login {

  public void testLogin() {
    // fill login data on sign-in page
    driver.findElement(By.name("user_name")).sendKeys("userName");
    driver.findElement(By.name("password")).sendKeys("my supersecret password");
    driver.findElement(By.name("sign-in")).click();

    // verify h1 tag is "Hello userName" after login
    driver.findElement(By.tagName("h1")).isDisplayed();
    assertThat(driver.findElement(By.tagName("h1")).getText(), is("Hello userName"));
  }
}
```

Gotchas of Page Object Pattern

- Assertions in Page Objects
- A page object does not necessarily need to represent all the parts of a page itself
- Try not to expose the internals of the page
- Methods return other Page-Objects
- Different results for the same action are modelled as different methods

Characteristics of Test Data

- Data is a complex need
  - Needs to mimic real data
  - Needs to be unique

- Data can be shared and reused

- Data sometimes needs to be Dynamic

Characteristics of Test Data

- Test Data in test implementation
- Test Data in page implementation
- Test Data in external sources

Test data file formats

- Excel
- JSON
- YAML
- DataBase

- Usable
- Readable
- Override

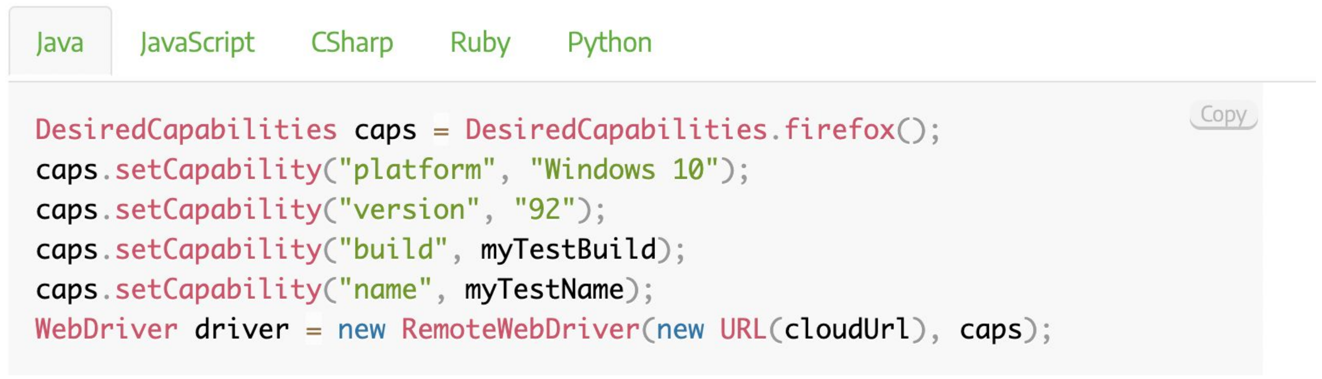
Selenium 4.0

Selenium 4.0 the new stuff!

- Window APIs
- Relative Locators
- WebDriver BiDi <> DevTools Protocol
- Selenium Grid 4.0
- Observability in Selenium Grid

Migrating from Selenium 3.0 to 4.0

Before

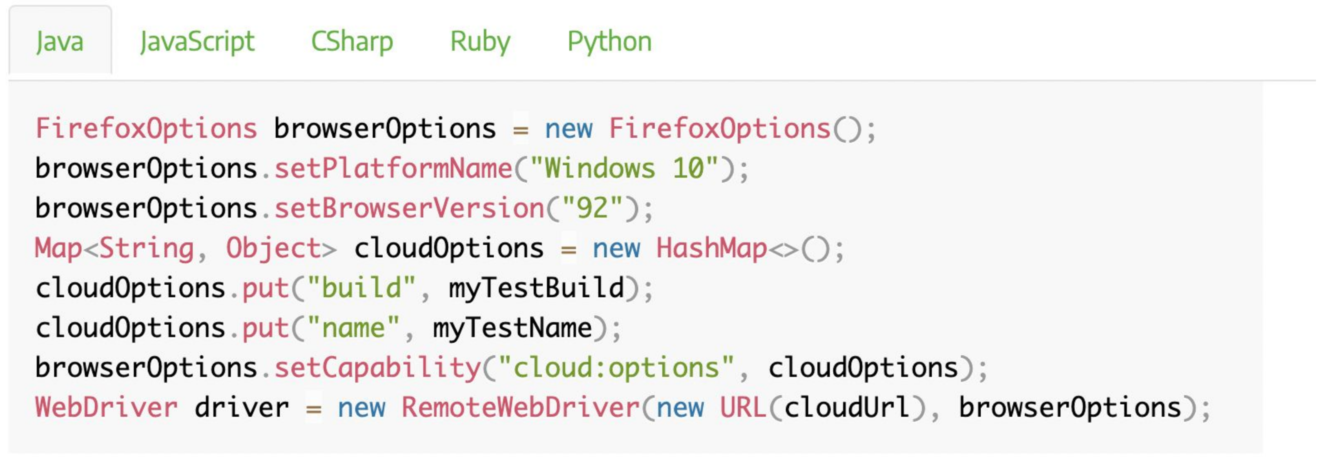


Java JavaScript CSharp Ruby Python

Copy

```
DesiredCapabilities caps = DesiredCapabilities.firefox();
caps.setCapability("platform", "Windows 10");
caps.setCapability("version", "92");
caps.setCapability("build", myTestBuild);
caps.setCapability("name", myTestName);
WebDriver driver = new RemoteWebDriver(new URL(cloudUrl), caps);
```

After



Java JavaScript CSharp Ruby Python

```
FirefoxOptions browserOptions = new FirefoxOptions();
browserOptions.setPlatformName("Windows 10");
browserOptions.setBrowserVersion("92");
Map<String, Object> cloudOptions = new HashMap<>();
cloudOptions.put("build", myTestBuild);
cloudOptions.put("name", myTestName);
browserOptions.setCapability("cloud:options", cloudOptions);
WebDriver driver = new RemoteWebDriver(new URL(cloudUrl), browserOptions);
```

LAMBDATEST

@manoj9788

Exercise 3

Exercise 3: Window APIs & Relative locators

- Write a new test that will open a new window from default session window
- Write a new test that will open a new tab from default session window
- Try to locate via relative locators

Exercise 3: What did we learn?

- How to open & switch between tabs
- How to open & switch between windows
- How to use Relative locators

WebDriver Bidi

- The Chrome DevTools Protocol is developed to enable a debugger inside Chromium-based browsers.
- Selenium 4 now have native support for Chrome DevTools Protocol through “DevTools” interface.
- This helps us getting Chrome Development properties such as Application Cache, Fetch, Network, Performance, Profiler, Resource Timing, Security and Target CDP domains etc.

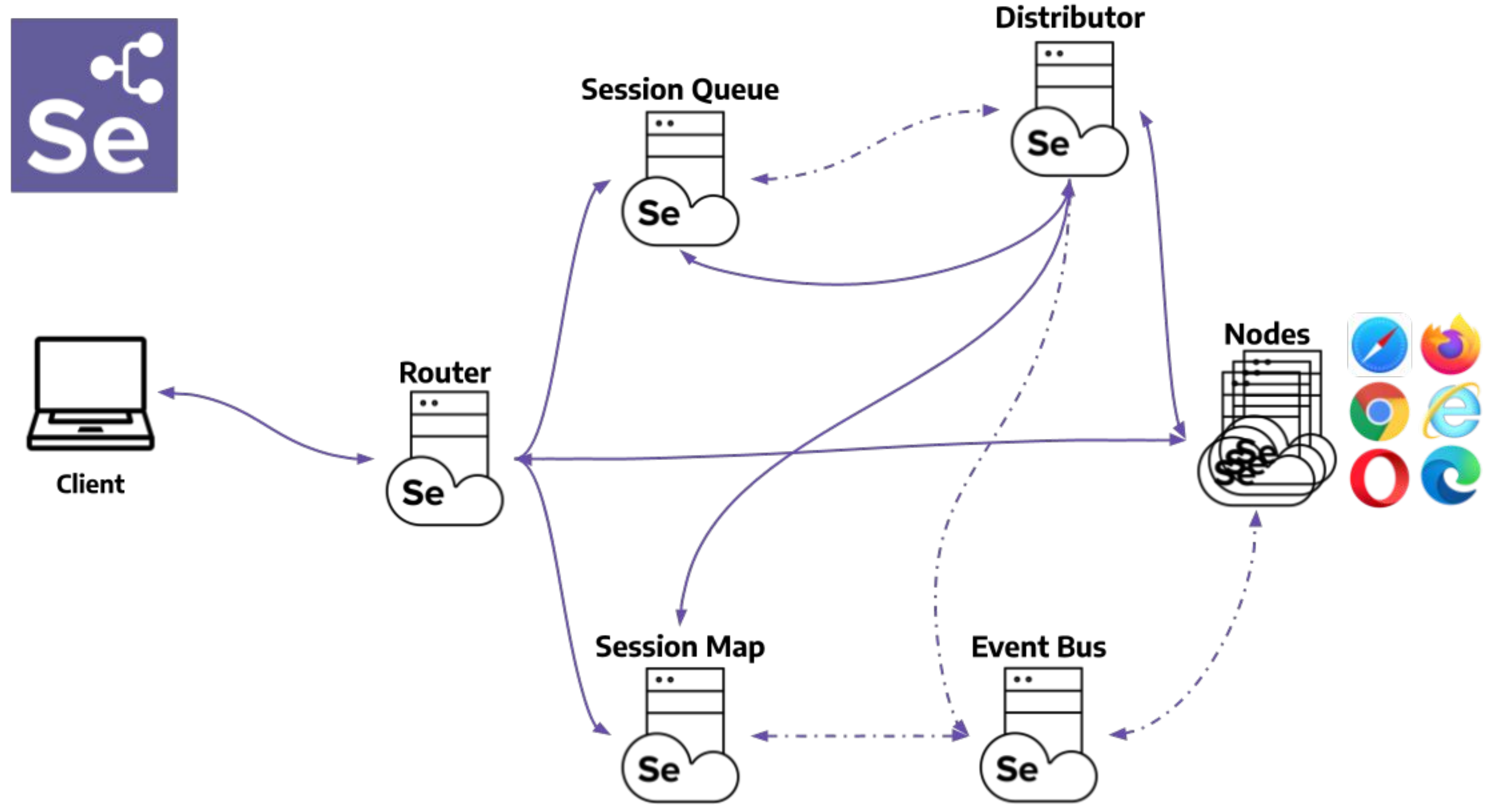
Exercise 4

Exercise 4: WebDriver Bidi

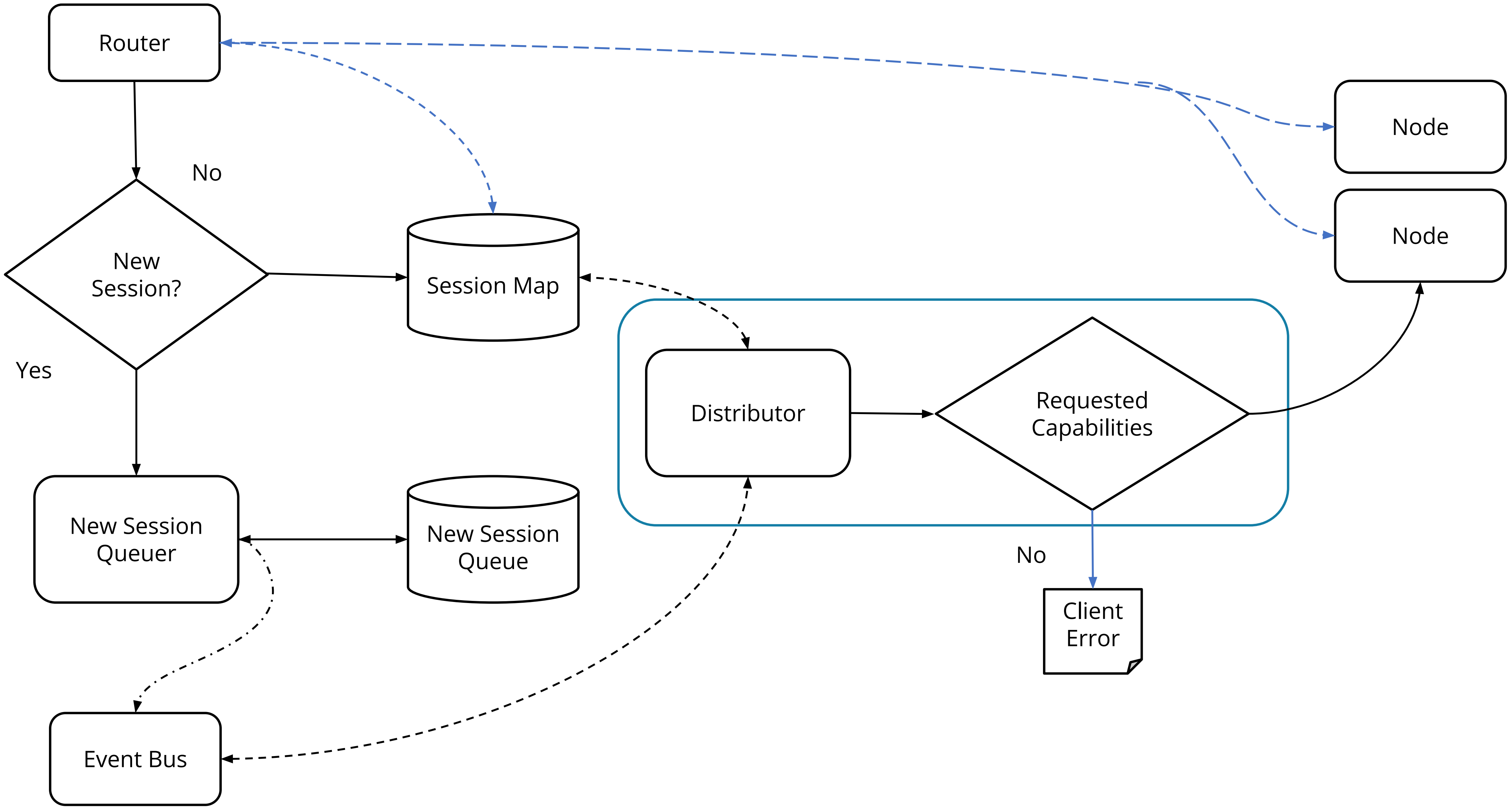
- Capture Performance Metrics
- Perform Geo-location Testing
- Get Console Logs test
- Network Interception test

Selenium Grid

Selenium Grid 4.0

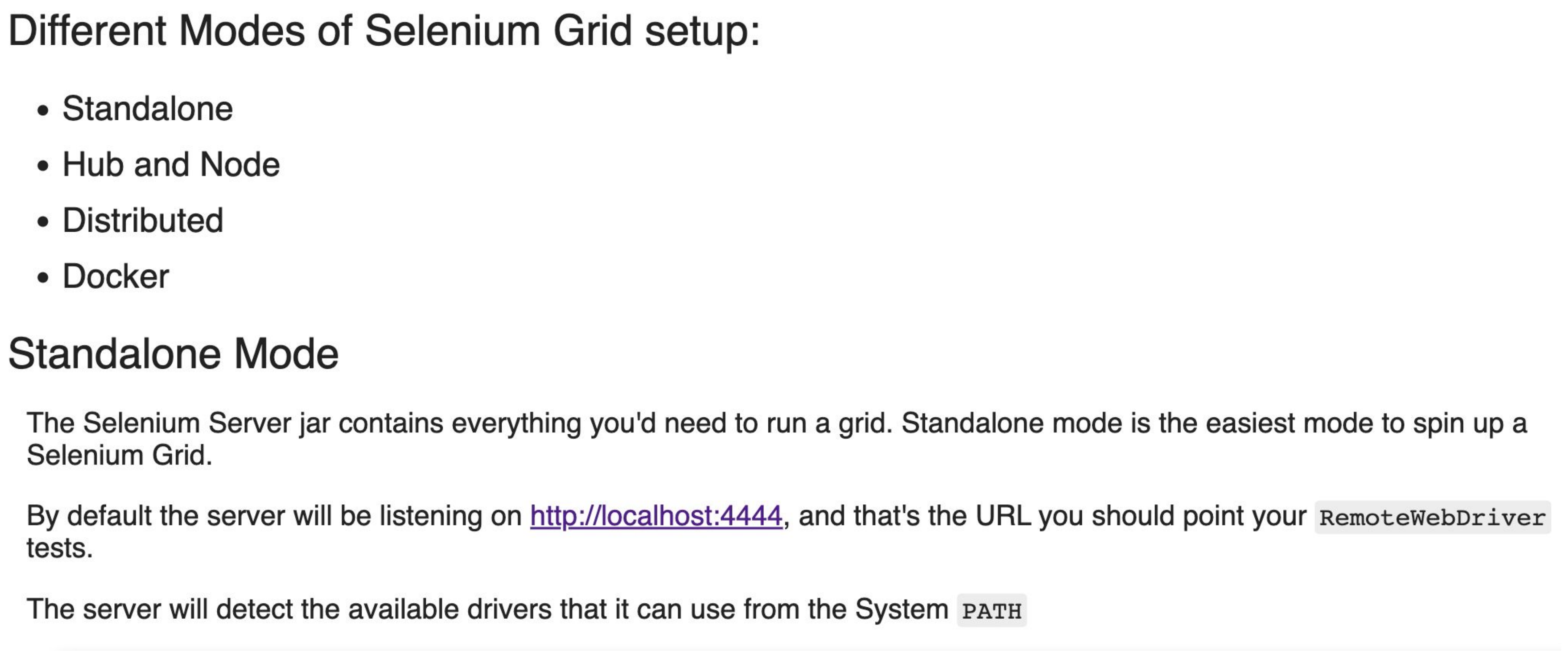


Selenium Grid 4.0 workflow



Exercise 5: Setup Selenium Grid

Setup Selenium Grid using Selenium 4.x Standalone Server



Different Modes of Selenium Grid setup:

- Standalone
- Hub and Node
- Distributed
- Docker

Standalone Mode

The Selenium Server jar contains everything you'd need to run a grid. Standalone mode is the easiest mode to spin up a Selenium Grid.

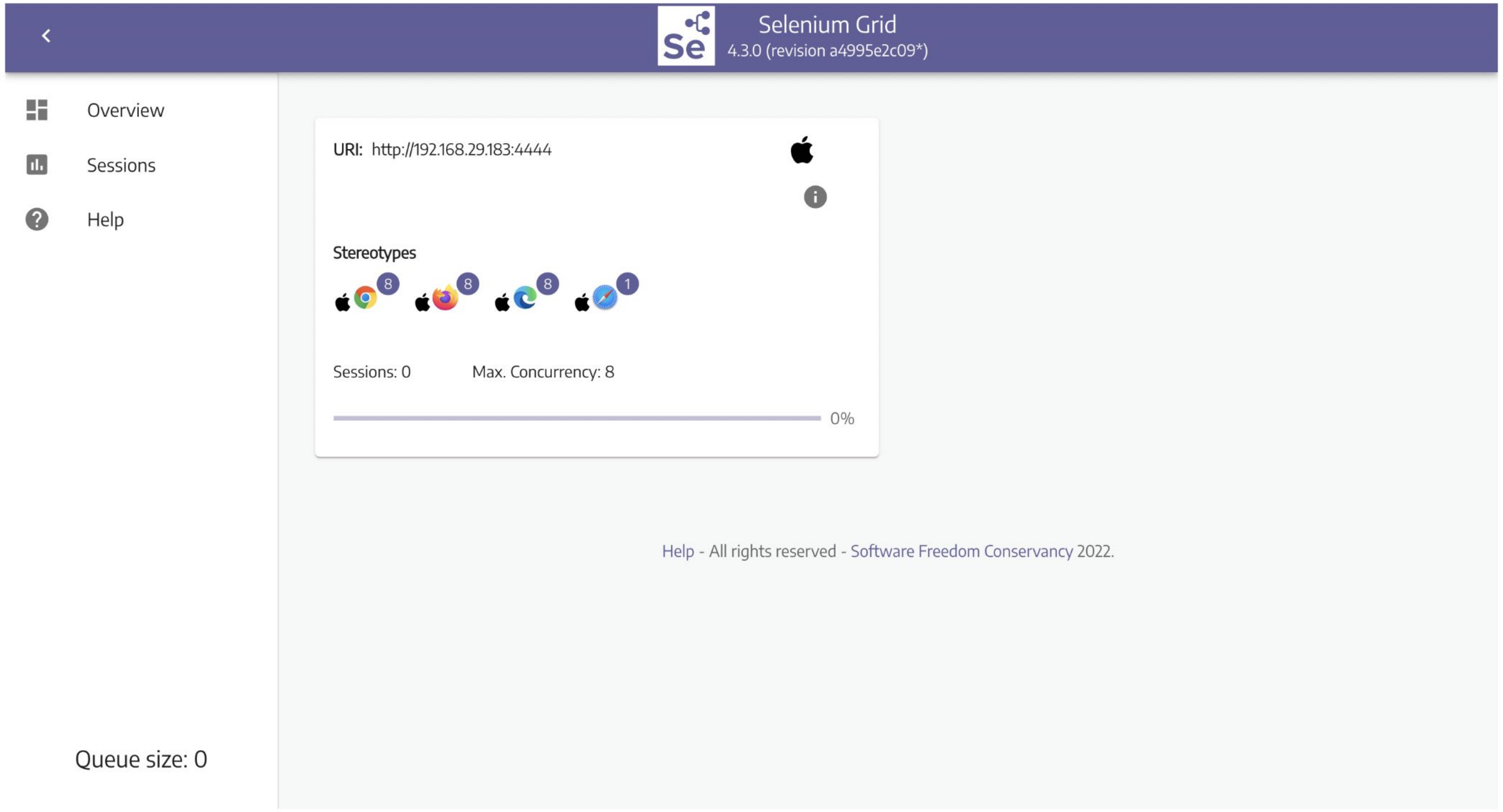
By default the server will be listening on http://localhost:4444, and that's the URL you should point your `RemoteWebDriver` tests.

The server will detect the available drivers that it can use from the System `PATH`

Exercise 5: Setup Selenium Grid

Visit http://localhost:4444/ui/index.html#/

You should now see the Selenium Grid UI!



Selenium Grid
4.3.0 (revision a4995e2c09*)

URI: http://192.168.29.183:4444

Stereotypes

Sessions: 0 Max. Concurrency: 8

0%

Help - All rights reserved - Software Freedom Conservancy 2022.

Queue size: 0

Exercise 5: What did we learn?

- Setting up a basic selenium grid
- Understood the workflow of distributed Selenium Grid
- Understood how to query the grid

Advanced Selenium Grid

Exercise 6

Exercise 6: Advanced Selenium Grid

- Discuss Pain points while remote execution
- Best practices & Maintenance tips
- Overview of Observability

Planning a Grid

Infrastructure as Code

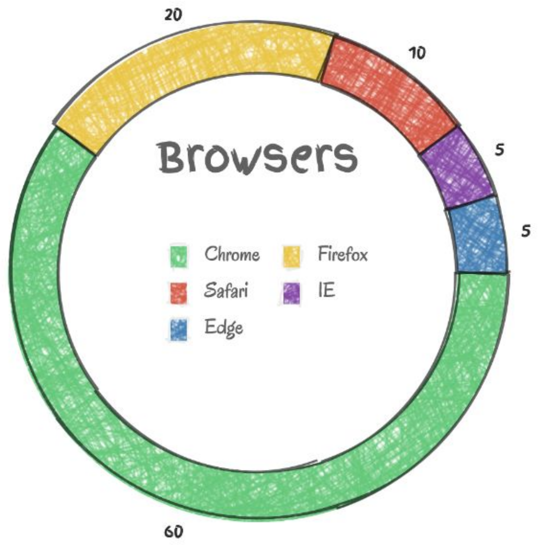
- Avoid manual tasks, prone to error
- Simplify infrastructure deployment by handling configuration files as code through different roles
- Ability to re-build the whole infrastructure when needed
- Hub and Node configuration
- Prerequisites installed
    - Java, Docker, language packs, etc...
- Most popular tools: Puppet / Chef / Ansible

How much RAM & CPU?

- Isolate browsers when possible.
- Rule of thumb, one CPU and one GB of RAM per browser.
  - Holds true for Docker with Chrome & Firefox.
  - Around 3CPU/3GB RAM for Safari/IE/Edge virtualized environments.
- There is not a unique answer.

How many slots are needed?

- What browsers are used?



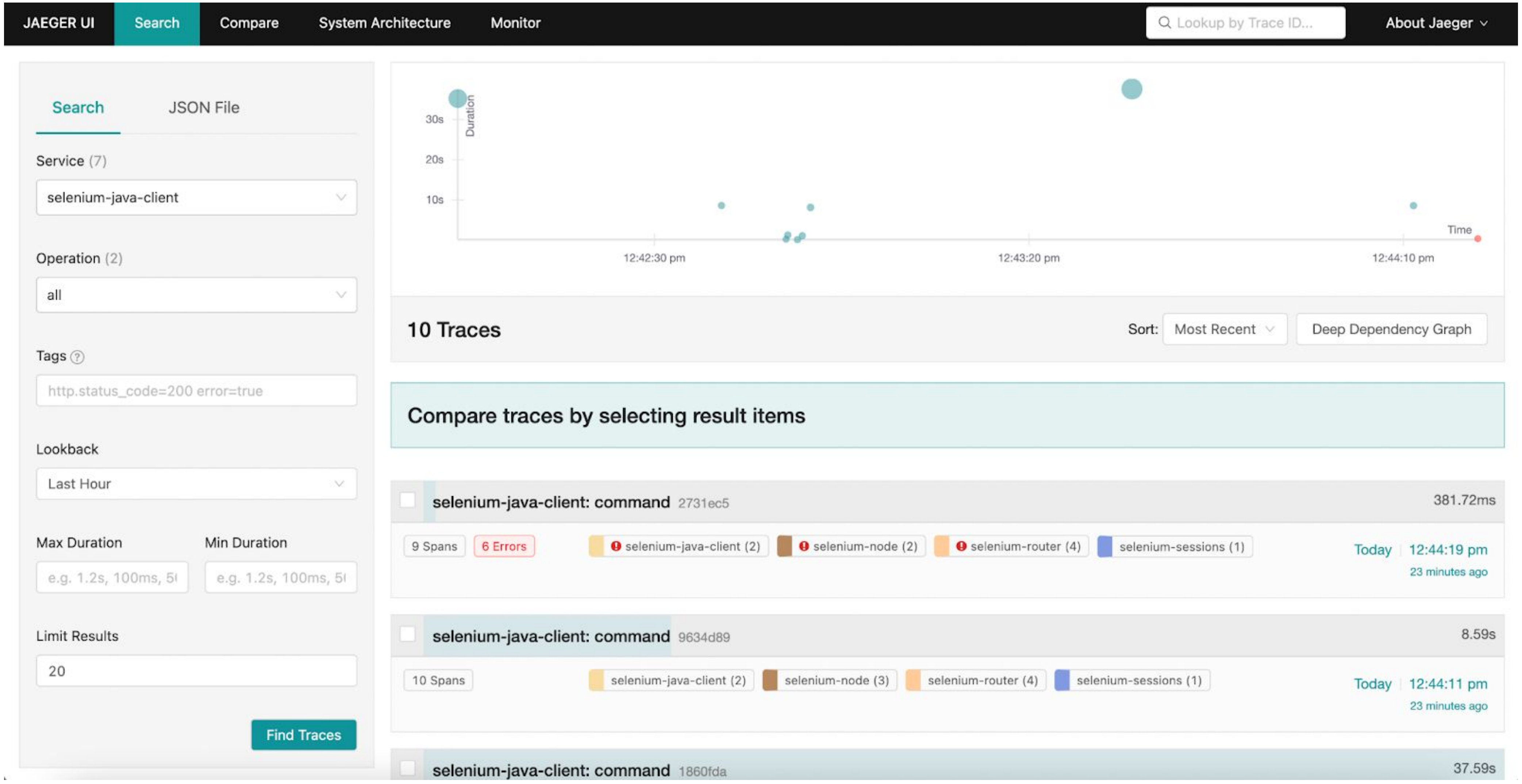
Maintaining Grid: Good concepts to follow!

- Stability -> Speed -> Coverage
    - Stability
        - Use Linux when possible and acceptable
        - Try to run 1 test per host (container/VM) at a time
        - Save your configuration in Git and manage it with Puppet/Chef/Ansible
            - Node/Hub config, cron jobs, versions, etc...
    - Speed
        - Use small VMs/Containers for nodes
        - It is better to have 20 small nodes than 1 big one
    - Coverage
        - Add browsers one by one, depending on your requirements
            - Chrome and Firefox are the easiest ones to start
        - More browsers, more versions and platforms to maintain

Observability in Selenium 4.0

- Selenium server is instrumented with tracing using Open Telemetry.
- Every request to the server is traced from start to end.
- Each trace consists of a series of spans as a request is executed within the server.
- End-to-End tracing : Client & Server

Observability : Visualizing Traces



Service (7)
selenium-java-client

Operation (2)
all

Tags
http.status_code=200 error=true

Lookback
Last Hour

Max Duration
e.g. 1.2s, 100ms, 5

Min Duration
e.g. 1.2s, 100ms, 5

Limit Results
20

10 Traces

Sort: Most Recent

Compare traces by selecting result items

selenium-java-client: command 2731ec5 — 381.72ms
9 Spans | 6 Errors | selenium-java-client (2) | selenium-node (2) | selenium-router (4) | selenium-sessions (1)
Today | 12:44:19 pm
23 minutes ago

selenium-java-client: command 9634d89 — 8.59s
10 Spans | selenium-java-client (2) | selenium-node (3) | selenium-router (4) | selenium-sessions (1)
Today | 12:44:11 pm
23 minutes ago

selenium-java-client: command 1860fda — 37.59s

LAMBDATEST

@manoj9788

Exercise 6: What did we learn?

- How to plan setting up a Grid
- Best practices to setup Grid
- What is Observability and how to trace Selenium Grid

Thank you!

Please share your feedback here: https://forms.gle/Uh3cdYMYxeKF2Cp3A