NPN Training

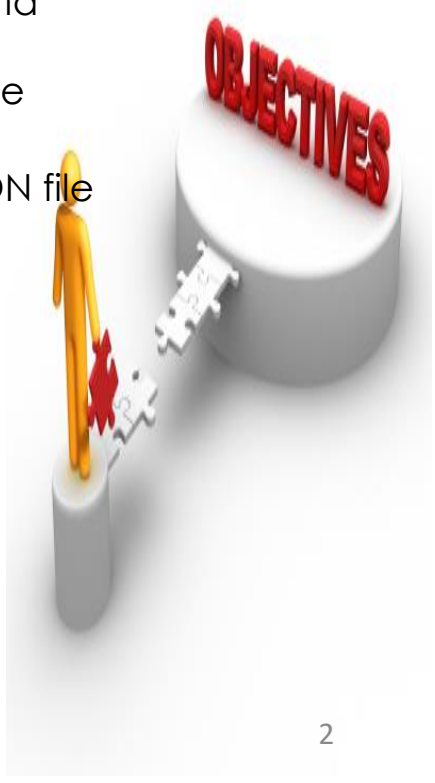
Training is the essence of success and we are committed to it

Selenium-Grid

Topics for the Module

After completing the module, you will be able to understand:

- What is Selenium-Grid?
- Selenium-Grid Architecture
- Uses of Selenium-Grid
- Configuring Hub
- Accessing Selenium-Grid Console
- Configuring Hub with options
- Configuring Node(s)
- Configuring Node(s) with options
- Run Selenium program via Grid
- Configuring Hub with JSON file
- Configuring Node(s) with JSON file
- Exercise/Assignment



What is Selenium -Grid?

Selenium-Grid allows you to run your tests on different machines against different browsers in parallel.

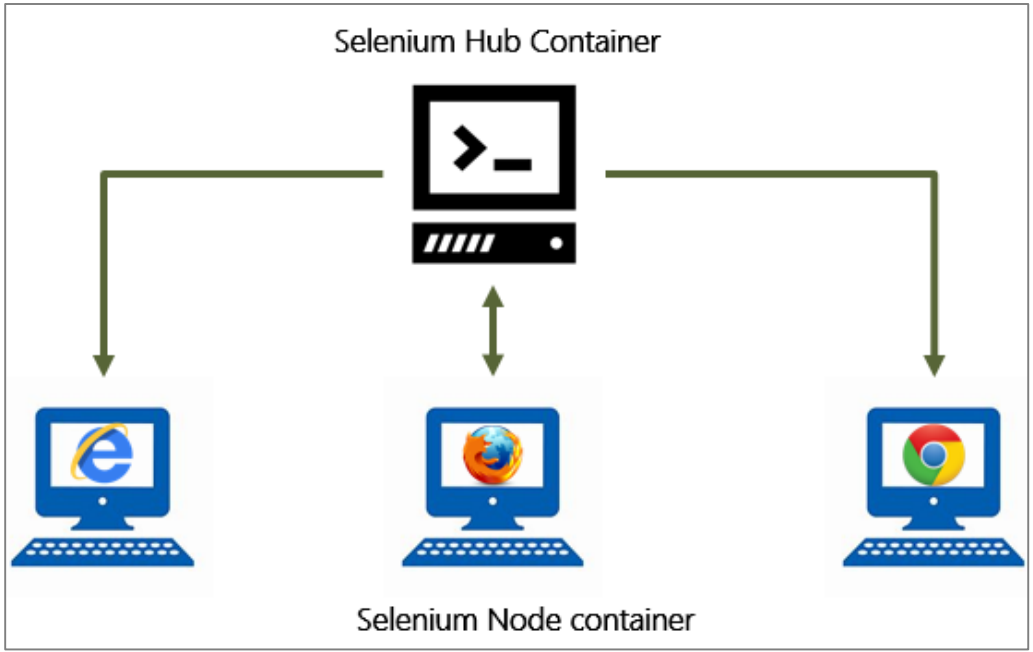
Key features of Selenium-Grid

- ✓ Distributing tests on several machines (parallel execution).
- ✓ Running multiple tests at the same time against different machines running different browsers and operating systems.
- ✓ Essentially, Selenium-Grid support distributed test execution. It allows for running your tests in a distributed test execution environment.

Selenium-Grid Architecture

Selenium-Grid creates a network HTTP servers with two roles

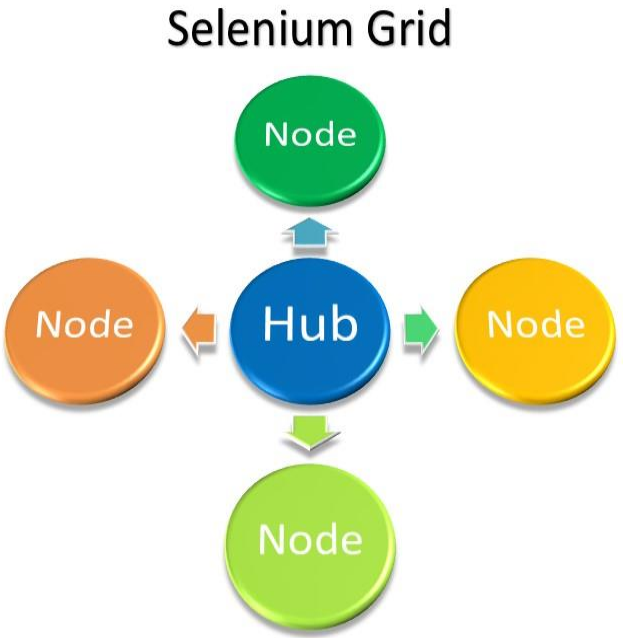
- ✓ Hub
  - ✓ The Hub is the central point that will receive all the test request and distribute them the right nodes.
- ✓ Node
  - ✓ A node is a machine that you register with your hub. By registering with the hub, the hub now knows about the node and any configuration information you used when you registered the node.



Uses of Selenium-Grid

Uses of Selenium-Grid…

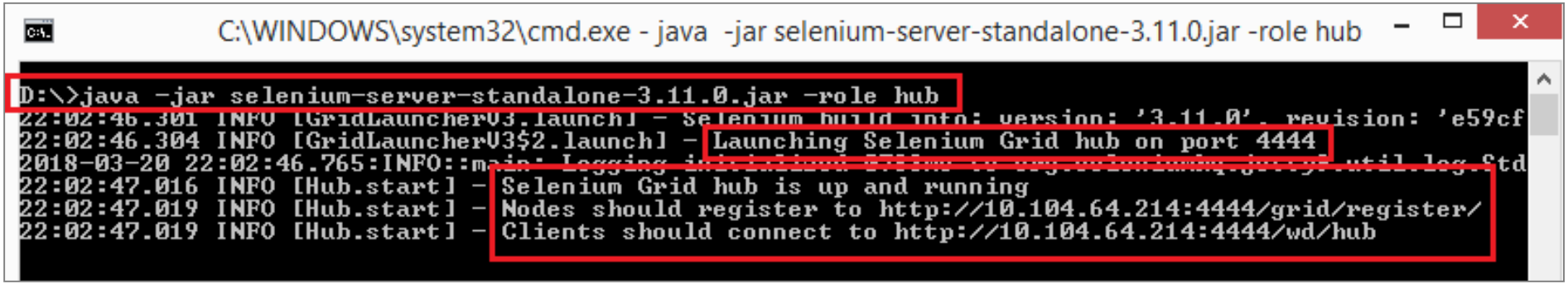
- ✓ Using a Selenium grid allows you to save time by spreading your test across multiple machines.
- ✓ This means you can run your test in parallel, which reduces the amount of total time to run your full automation test suite.
- ✓ To run against multiple browser(s) versions by allowing the hub to automatically manage that for you.



Configuring Hub [Hands-on]

- ✓ Download Selenium Server (https://www.seleniumhq.org/download/)
- ✓ Start the command prompt and navigate to the location, where the Selenium server jar file was saved.
- ✓ From command prompt, run the jar file according to below syntax

```
java –jar <selenium server jar>.jar –role hub
```



Accessing Selenium-Grid console [Hands-on]

Key points to observe…

- ✓ By default the Selenium-Grid will run on port 4444.
  We can change this port by passing ***-port*** parameter.

- ✓ The URL for registering nodes
  http://<IP>:<port>/grid/register

- ✓ The URL for connecting hub.
  http://<IP>:<port>/wd/hub

- ✓ The URL for Selenium-Grid console will be
  http://<IP>:<port>/grid/console

Configuring Hub with options

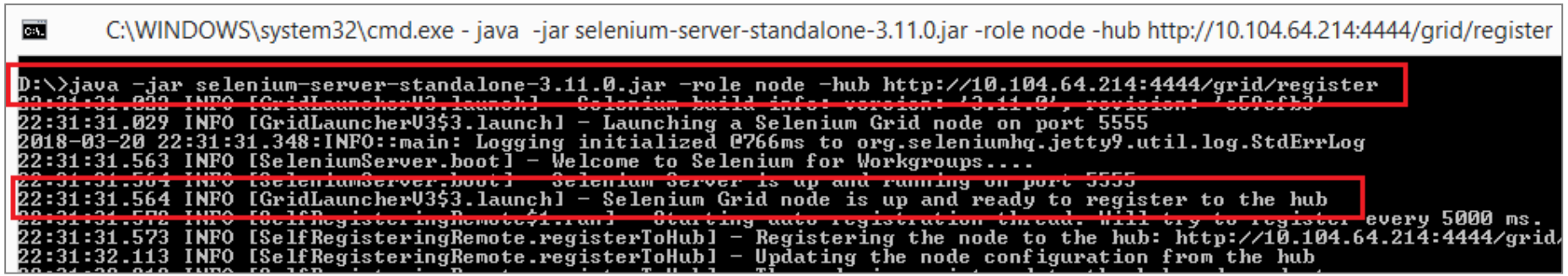
[Hands-on]

- We can configure the Hub with different options
  - **-port <Integer>** : the port number the server will use. Default value is 4444.
  - **-browserTimeout <Integer>** in seconds: number of seconds a browser session is allowed to hang while a WebDriver command is running (example: driver.get(url)). Minimum value is 60. An unspecified, zero, or negative value means wait indefinitely.
  - **-host <String>** : IP or hostname of the system
  - **-maxSession <String>** : max number of tests that can run at the same time on the node, irrespective of the browser used
  - **-timeout, -sessionTimeout <Integer>** in seconds**:** Specifies the timeout before the server automatically kills a session that hasn't had any activity in the last X seconds.
  - **-cleanUpCycle <Integer>** in ms**:** specifies how often the hub will poll running proxies for timed-out (i.e. hung) threads.
  - **-throwOnCapabilityNotPresent <Boolean>** in true or false**:** If true, the hub will reject all test requests if no compatible proxy is currently registered. If set to false, the request will queue until a node supporting the capability is registered with the grid.

Configuring Node(s) [Hands-on]

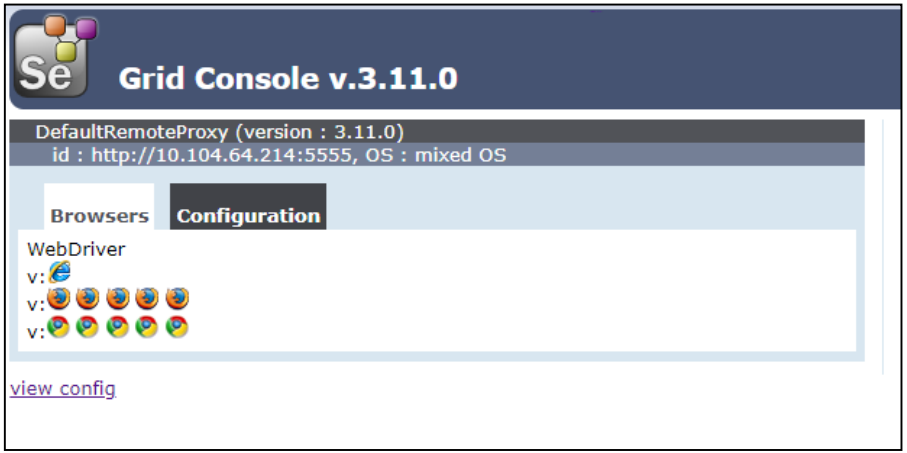
- ✓ In Node machine(s), download Selenium Server (https://www.seleniumhq.org/download/)
- ✓ In Node machine(s), start the command prompt and navigate to the location, where the Selenium server jar file was saved.
- ✓ From command prompt, run the jar file according to below syntax

```
java -jar <selenium server jar>.jar -role node -hub http://<IP>:<port>/grid/register
```



Configuring Node(s) [Hands-on]

- ✓ Now, navigate to Selenium Grid console and verify the node(s) information.
- ✓ If the node is created successfully, then the node information will be displayed in Grid console (As shown below).



Configuring node(s) with options

[Hands-on]

- We can configure the Hub with different options
  - **-port <Integer>** : the port number the server will use. Default value is 5555.
  - **-browserTimeout <Integer>** in seconds: number of seconds a browser session is allowed to hang while a WebDriver command is running (example: driver.get(url)). Minimum value is 60. An unspecified, zero, or negative value means wait indefinitely.
  - **-host <String>** : IP or hostname of the system
  - **-maxSession <String>** : max number of tests that can run at the same time on the node, irrespective of the browser used
  - **-timeout, -sessionTimeout <Integer>** in seconds**:** Specifies the timeout before the server automatically kills a session that hasn't had any activity in the last X seconds.
  - **-cleanUpCycle <Integer>** in ms**:** specifies how often the hub will poll running proxies for timed-out (i.e. hung) threads.
  - **-nodePolling <Integer>** in ms**:** specifies how often the hub will poll to see if the node is still responding.

Configuring node(s) with options

[Hands-on]

- We can configure the Hub with different options
  - **-nodeStatusCheckTimeout <Integer>** in ms: connection/socket timeout, used for node "nodePolling“ check.
  - **-registerCycle <String>** in ms : specifies how often the node will try to register itself again. Allows administrator to restart the hub without restarting or risk orphaning) registered nodes.
  - **-downPollingLimit <String>** : node is marked as "down" if the node hasn't responded after the number of checks specified in [downPollingLimit].
  - **-capabilities, -browser < String > :** comma separated Capability values.

Example: -capabilities browserName=firefox,maxInstances=20,platform=LINUX
-browser browserName=chrome,maxInstances=5,platform=WINDOWS

  - **-log <String>** filename**:** The filename to use for logging.

Configuring Hub with JSON file

[Hands-on]

```
{
      "browserName": "internet explorer",
      "maxInstances": 1,
      "platform": "WINDOWS",
      "webdriver.ie.driver": "C:/Program Files (x86)/Internet Explorer/iexplore.exe"
    }
  ],
  "configuration": {
    "_comment" : "Configuration for Node",
    "cleanUpCycle": 2000,
    "timeout": 30000,
    "proxy": "org.openqa.grid.selenium.proxy.WebDriverRemoteProxy",
    "port": 5555,
    "host": ip,
    "register": true,
    "hubPort": 4444,
    "maxSessions": 5
  }
}
```

Run Selenium program via Grid [Hands-on]

- ✓ DesiredCapablities()/BrowserOptions()
- ✓ RemoteWebDriver()

```
FirefoxOptions options = new FirefoxOptions();
WebDriver driver = new RemoteWebDriver(new URL("http://<IP>:<PORT>/wd/hub"),options);
driver.get("http://npntraining.com");
```

Configuring Hub with JSON file

[Hands-on]

- We can configure the Hub with JSON file
  - **-hubConfig <String>** filename: a JSON file (following grid2 format), which defines the hub properties

```
{"_comment" : "Configuration for Hub - hubConfig.json",
  "host": google.com,
  "maxSessions": 5,
  "port": 4444,
  "cleanupCycle": 5000,
  "timeout": 300000,
  "newSessionWaitTimeout": -1,
  "servlets": [],
  "prioritizer": null,
  "capabilityMatcher":
"org.openqa.grid.internal.utils.DefaultCapabilityMatcher",
  "throwOnCapabilityNotPresent": true,
  "nodePolling": 180000,
  "platform": "WINDOWS"}
```

Configuring Node(s) with JSON file [Hands-on]

- We can configure the Node(s) with JSON file
  - **-nodeConfig <String>** filename: JSON configuration file for the node. Overrides the default values

```
{
  "capabilities": [
    {
      "browserName": "firefox",
      "acceptSslCerts": true,
      "javascriptEnabled": true,
      "takesScreenshot": false,
      "firefox_profile": "",
      "browser-version": "27",
      "platform": "WINDOWS",
      "maxInstances": 5,
      "firefox_binary": "",
      "cleanSession": true
    },
    {
      "browserName": "chrome",
      "maxInstances": 5,
      "platform": "WINDOWS",
      "webdriver.chrome.driver": "C:/Program Files (x86)/Google/Chrome/Application/chrome.exe"
    },
```