Selenium Grid Deployment Alternatives:

Scaling and Adding Video Recording Without Container Orchestration

Eric Frankenberger

Sr. DevOps Engineer, Genesys

3/29/2023

GENESYS™

ABOUT ME

- 2014 – 2021: Chip Ganassi Racing
  - Sysadmin and pit crew member
  - 9x Indy 500’s
  - 3x 24 hour of LeMans
  - 5x 24 hours of Daytona
- 2021 – Present: Genesys
  - Sr. DevOps Engineer
  - Selenium Grid maintainer

GENESYS

ALTERNATIVE GRID DEPLOYMENTS

GENESYS™

Alternative Grid Deployments

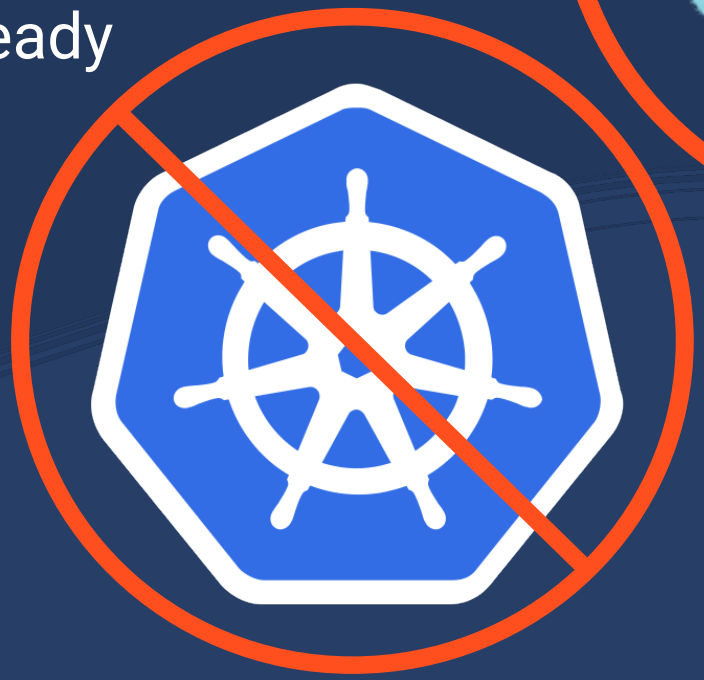
Requirements

- > 15,000 tests/day
- Dynamic video recording
- Per-test node logs
- Autoscaling
- CI/CD compliant
- Without container orchestration
  - Docker Swarm
  - Kubernetes

Alternative Grid Deployments

Why?

- Docker Selenium is robust, mature, and proven
- Dynamic grid can do everything already



However

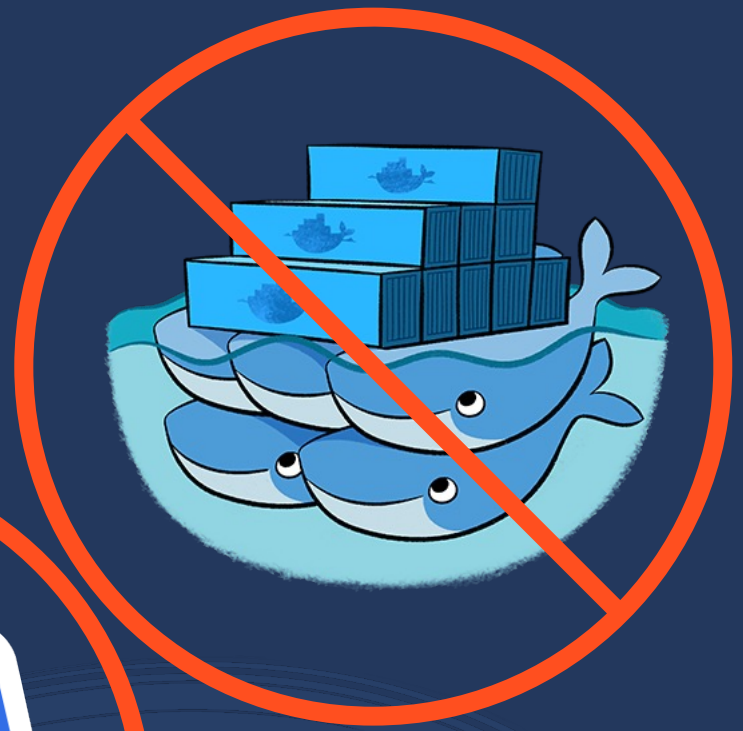
- I can't use
  - Docker Swarm
  - Kubernetes

GENESYS™

Alternative Grid Deployments

So?

- Outside of Dynamic Grid, there is no official support for
  - Video recording
  - Autoscaling
- Eric had to come up with a way to do
  - Video recording
  - Autoscaling



Alternative Grid Deployments

So?

By seeing the inner workings of an alternative grid deployment, your understanding of Selenium as a whole will improve.

THE GRID

GENESYS™

PLAN OF ATTACK

THE GROUND WORK

- Solutions discussed are largely agnostic to:
  - Cloud providers
  - Hardware architecture
  - Programming languages
- To be treated as descriptions of techniques

Plan of Attack

Setting up the grid

- Stand up a fully distributed grid
  - Distributor with event bus
  - Router
  - Session map
  - New session queue

THE GRID

VM 1: ROUTER

```
java -jar selenium-server-<version>.jar router
```

VM 2: DISTRIBUTOR

```
java -jar selenium-server-<version>.jar distributor \
--bind-bus true
```

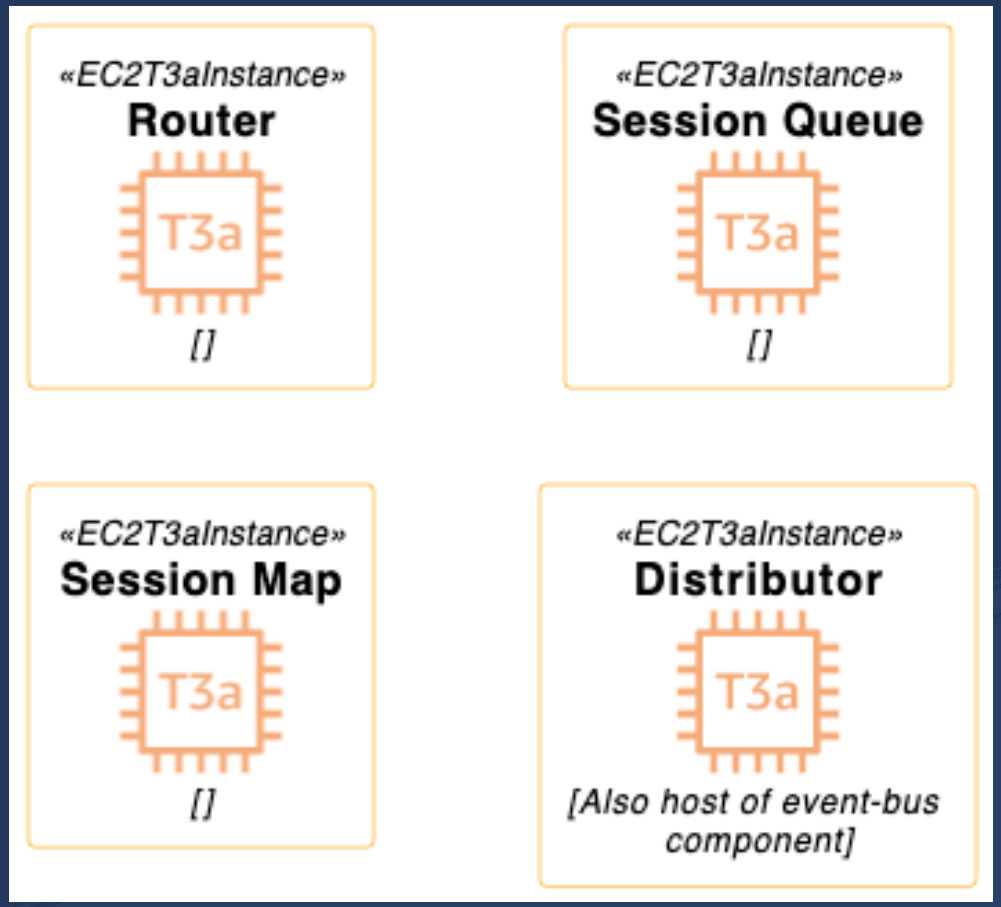
VM 3: SESSION MAP

```
java -jar selenium-server-<version>.jar sessions
```

VM 4: SESSION QUEUE

```
java -jar selenium-server-<version>.jar sessionqueue
```

THE GRID



GENESYS

Making the Grid Communicate

Router needs	Distributor needs	Session map needs	Session queue needs
New session queue	New session queue	Event bus	
Session map	Session map		
Distributor			

MAKING THE GRID COMMUNICATE

ROUTER

```
java -Dwebdriver.http.factory=jdk-http-client \
-jar selenium4.jar \
--ext selenium-http-jdk-client-4.7.1.jar \
router \
--sessions http://$sessions:5556 \
--sessionqueue http://$sessionqueue:5559 \
--distributor http://$distributor:5553 \
--log /var/log/seleniumRouterLogs.log \
--log-level INFO
```

DISTRIBUTOR

```
java -Dwebdriver.http.factory=jdk-http-client \
-jar selenium4.jar \
--ext selenium-http-jdk-client-4.7.1.jar \
distributor \
--sessions http://$sessions:5556 \
--sessionqueue http://$sessionqueue:5559 \
--bind-bus true \
--healthcheck-interval 120 \
--log /var/log/distributorControllerLogs.log \
--log-level INFO
```

SESSION MAP

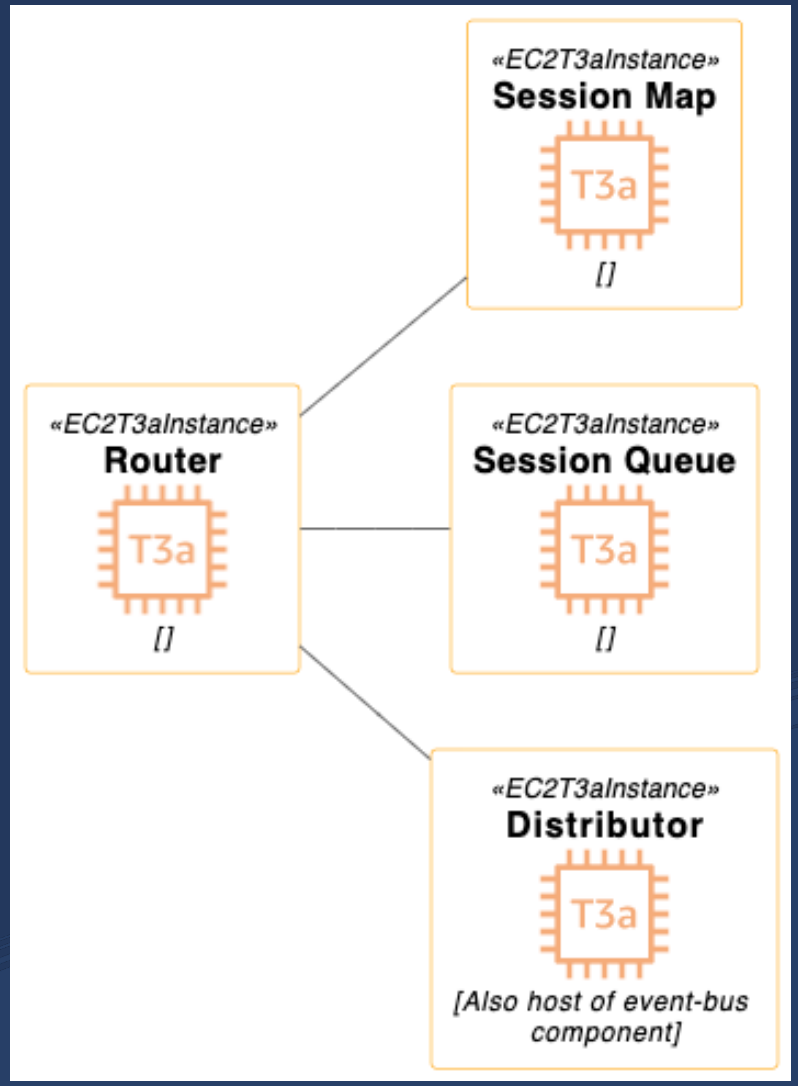
```
java -Dwebdriver.http.factory=jdk-http-client \
-jar selenium4.jar \
--ext selenium-http-jdk-client-4.7.1.jar \
sessions \
--publish-events tcp://$distributor:4442 \
--subscribe-events tcp://$distributor:4443 \
--log /var/log/seleniumSmapLogs.log \
--log-level INFO
```

NEW SESSION QUEUE

```
java -Dwebdriver.http.factory=jdk-http-client \
-jar selenium4.jar \
--ext selenium-http-jdk-client-4.7.1.jar \
sessionqueue \
--session-request-timeout 290 \
--session-retry-interval 3 \
--log /var/log/inin/seleniumSqueueLogs.log \
--log-level INFO
```

GENESYS

MAKING THE GRID COMMUNICATE



GENESYS

SELF HEALING

IP ADDRESS

Pros

- Direct access to component
- Failover as soon as new component is active

Cons

- Requires component's Selenium process to be restarted
- Reinventing DNS
- Multiple scripts required to orchestrate
- External data stores needed
- Restarting components results in

DNS NAME

Pros

- Easy failover
- Simplest method

Cons

- Only one instance of each component can be active
- local DNS caching can lead to long failover times

LOAD BALANCER

Pros

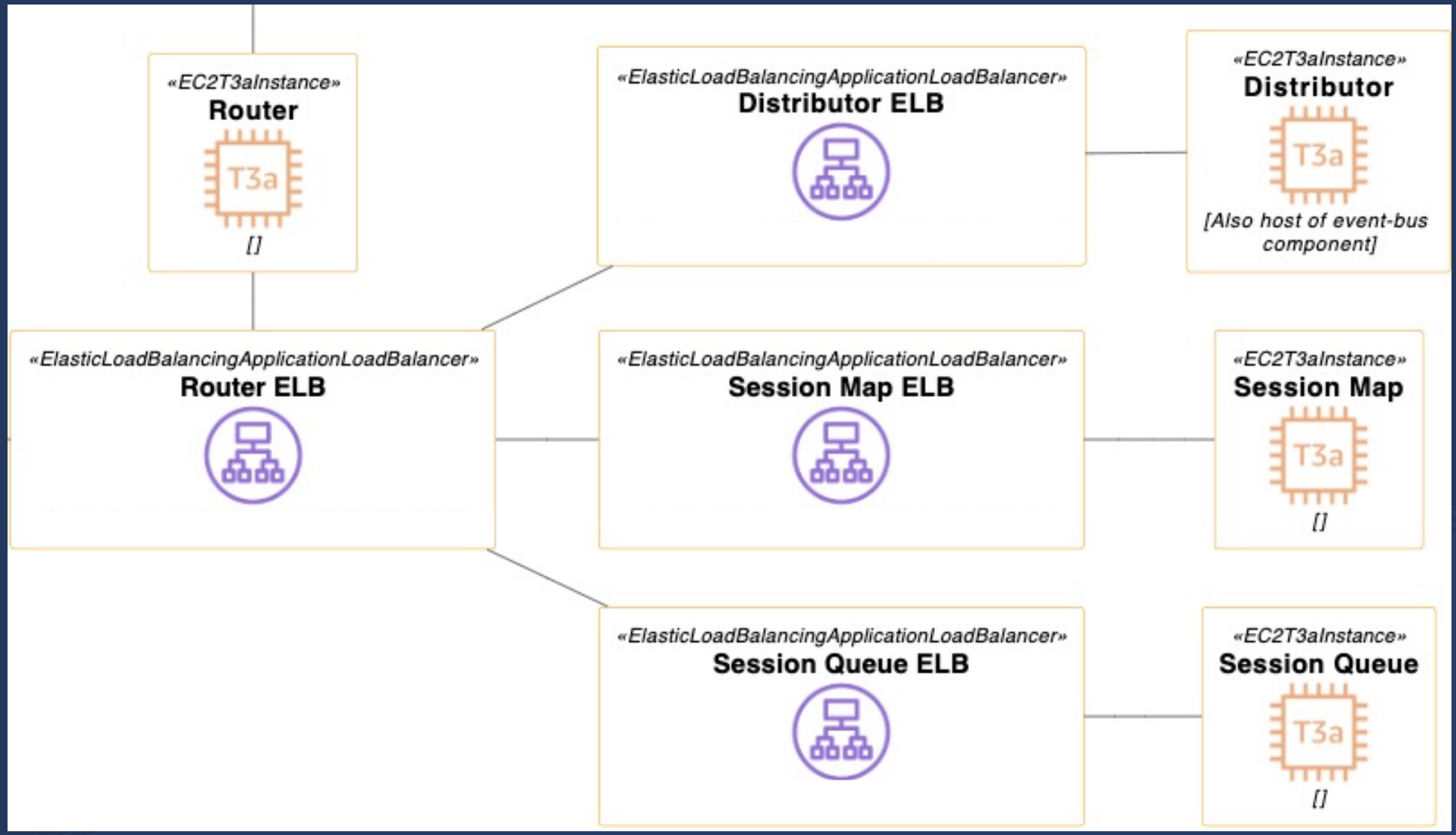
- Easy failover
- Healthchecks can boot failed components

Cons

- Tooling dependent
- Can cause extra work if not automated

GENESYS™

SELF HEALING



GENESYS

REGISTER A SINGLE NODE

GENESYS

REGISTERING A SINGLE NODE

LAUNCH COMMAND

```
java -jar selenium4.jar node \
--grid-url $router \
--detect-drivers true \
--publish-events tcp://$distributor:4442 \
--subscribe-events tcp://$distributor:4443 \
--log /home/selenium4/log-node.txt"
```

REGISTERING A SINGLE NODE - NOTES

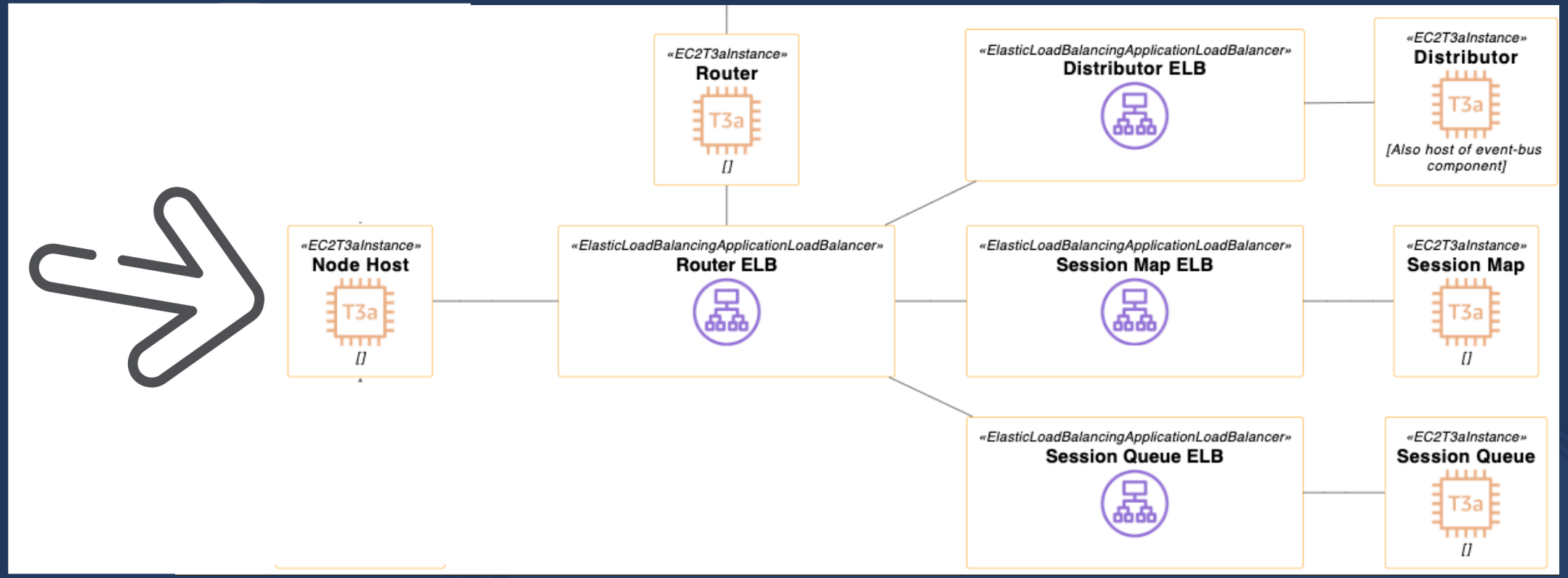
```
--detect-drivers true \
```

- If set to false, node will never send a registration event
- At least one browser + browser driver should be installed

Other concerns

- Make sure the rest of the grid is talking
- If components aren't talking to one another properly, registration will not occur

REGISTERING A SINGLE NODE



GENESYS

BUILDING FEATURES ON OUR NODES

Utilizing Docker Selenium

Running 1 test per node

Reusing nodes

Video recording

Scaling

I THOUGHT YOU SAID WE WEREN'T USING DOCKER

I SAID DOCKER ***SWARM***

Utilizing Docker Selenium

Instead of having to:

- download browser(s)
- download browser driver(s)
- download selenium
- download vnc
- download FFMPEG
- script and install browsers
- script and install browser drivers
- script and run selenium
- script and run vnc
- script and run FFMPEG
- configure XVFB on node
- script FFMPEG to XVFB connection
- AND MORE

We do this

- `docker run selenium/node-chrome`
- `docker run selenium/video`

GENESYS™

Utilizing Docker Selenium

Node launch command

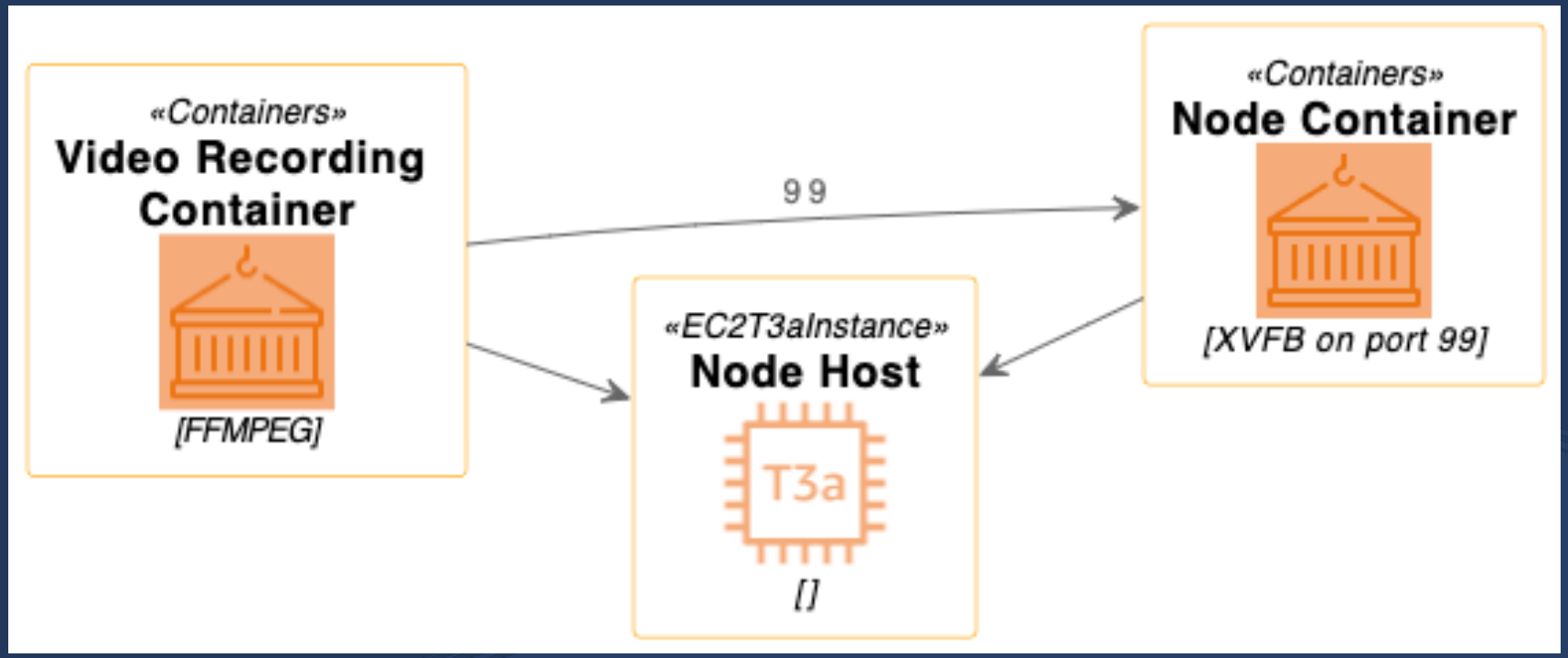
```
docker run \
-d \
-p 5555:5555 \
-p 5900:5900 \
-e SE_DRAIN_AFTER_SESSION_COUNT=1 \
-e SE_EVENT_BUS_PUBLISH_PORT=4442 \
-e SE_EVENT_BUS_SUBSCRIBE_PORT=4443 \
-e SE_NODE_SESSION_TIMEOUT=600 \
-e SE_NODE_HOST=$machineIP \
-e SE_SCREEN_WIDTH=$width \
-e SE_SCREEN_HEIGHT=$height \
-e SE_OPTS="--log-level CONFIG" \
--name node \
--net grid \
--shm-size="2g" \
selenium4/node
docker logs -f node &> /var/log/node-`date +"%d_%T"`.log &
```

Video launch command

```
docker run \
-d \
-p 9000:9000 \
-e SE_SCREEN_WIDTH=$width \
-e SE_SCREEN_HEIGHT=$height \
--name video \
--net grid \
--shm-size="2g" \
selenium4/video
docker logs -f video &>
/var/log/video-`date +"%d_%T"`.log &
```

GENESYS

Utilizing Docker Selenium



BUILDING FEATURES ON OUR NODES

Utilizing Docker Selenium

Running 1 test per node

Reusing node hosts

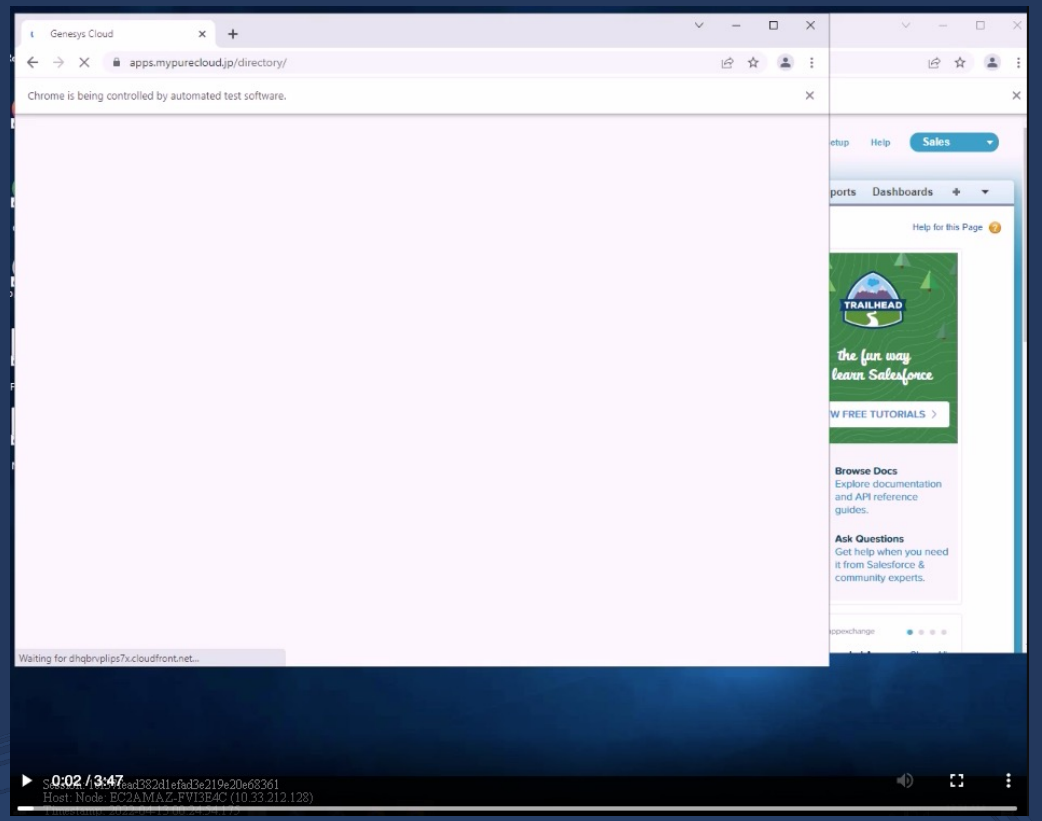
Video recording

Scaling

1 Test Per Node

Why?

- Sterile test environment
- Makes video recording a little easier
- Cattle not pets



How?

- `-e SE_DRAIN_AFTER_SESSION_COUNT=1`
- `--drain-after-session-count 1`

GENESYS

Reusing nodes

- When node is drained, container is killed after test
- Without reuse logic, entire VMs need to be relaunched
- Very wasteful and inefficient
- We already have a sterile environment with containers
- Let’s reuse them!

REUSING NODES

But how?

- ITS EASY
- IT'S ONLY LIKE 13 LINES OF CODE

The Script

- Run a loop that checks if node and video containers are running
- if they are, no action
- if they aren't, run the bash script that launches the nodes

REUSING NODES - THE CODE

```
client = docker.from_env()
containers = ['video', 'node']


while True:
        running_containers = []
        for container in client.containers.list():
                if container.name in containers:
                        running_containers.append(container.name)
        if set(containers).issubset(set(running_containers)):
                print('Containers are running, no action taken')
        else:
                print('Containers are not running, launching')
                for container in containers:
                        if container not in running_containers:
                                client.containers.run(container, detach=True)
        time.sleep(5)
```

GENESYS

BUILDING FEATURES ON OUR NODES

Utilizing Docker Selenium

Running 1 test per node

Reusing nodes

Video recording 

Scaling

VIDEO RECORDING

Out of the box:*

- Video container connects to Node
- Records the lifetime of the container
- Names the file video.mp4
- May or may not crash

What I need:

- Video container connects to Node
- Records only while test runs
- Make a unique file per test
- Names the file $sessionId.mp4
- Upload to remote storage
- Terminate the container on completion

*this is not a shortcoming of Docker Selenium

VIDEO RECORDING

But how?

- Utilize existing Docker Selenium video recording container
- Write a script to replace video.sh
  - Connect to node's XVFB
  - Query it's API
  - Start and stop video
  - Upload or move the video
  - Terminate the container upon completion

VIDEO RECORDING

Doesn’t this only fulfill a narrow use case?

- no
- The principle and scripting are very simple
- FFMPEG can be ran on the node itself
- Upload to cloud component can be adapted to anything
- At its base, this is nothing more than a technique

VIDEO RECORDING

OK WELL HOW DOES IT WORK

Video containers are launched 1:1 with nodes

Video container's script gets 3 variables, set by querying node API

- SESSION_ID
- PRE_SESSION -initialized to True
- RUNNING_TEST -initialized to False

These 3 combined are enough to set up our entire operation

VIDEO RECORDING

Every 1 second, in a loop:

IF

- SESSION_ID is None
- PRE_SESSION is True
- RUNNING_TEST is False

AND

- FFMPEG can connect to Node

THEN

- Hang out for a bit

IF

- SESSION_ID is not None

THEN

- set PRE_SESSION to False
- set RUNNING_TEST to True

AND

- Start FFMPEG
- begin recording, video name = SESSION_ID

IF

- SESSION_ID is None
- PRE_SESSION is False

THEN

- Set RUNNING_TEST to False

AND

- stop recording
- upload file to s3
- terminate container

GENESYS

BUILDING FEATURES ON OUR NODES

Utilizing Docker Selenium

Running 1 test per node

Reusing nodes

Video recording

Scaling

GENESYS

SCALING

Get the data first. Every 3 seconds:

Query the router

```
ROUTER_QUERY =
requests.get("http://127.0.0.1:4444/status", timeout=1).json()
```

Query the session queue

```
QUEUE_QUERY =
requests.get("http://127.0.0.1:4444/se/grid/newsessionqueue/queue",
timeout=1).json()
```

GENESYS

SCALING - NODE DATA

Get total nodes

```
len(ROUTER_QUERY['value']['nodes'])
```

Get used nodes

```
for nodes in ROUTER_QUERY['value']['nodes’]:
    for slots in nodes['slots’]:
        if slots['session'] is not None:
            used_count += 1
```

GENESYS™

SCALING - QUEUE DATA

Get sessions in queue

```
for items in QUEUE_QUERY:
    if items is not None:
        queue_count += 1
    else:
        queue_count = 0
```

SCALING - OTHER DATA

Set node buffer

```
NODE_BUFFER = 10
```

Define scale up and scale down times

```
SCALE_UP_TIME = 30 //seconds
SCALE_DOWN_TIME = 60 //seconds
```

Define query interval

```
QUERY_INTERVAL = 3 //seconds
```

SCALING - USING THE DATA

Now we have all the data, how do we use it?

Every 3 seconds:

- Query router
- Query session queue

if queue = 0

```
    proposed_desired = used_nodes + node_buffer
```

if queue > 1

```
    proposed_desired = used_nodes + node_buffer + queue

return proposed_desired
```

SCALING - EVENTS

Every 3 seconds

- `if queue > 0`
  - `scale_up(proposed_desired)`

Every 30 seconds

- `if proposed_desired > reported_desired`
  - `scale_up(proposed_desired)`

Every 60 seconds

- `if proposed_desired < reported_desired`
  - `incremental_scale_down(proposed_desired)`

BUILDING FEATURES ON OUR NODES

Utilizing Docker Selenium

Running 1 test per node

Reusing nodes

Video recording

Scaling

BONUS FEATURE!!!!

MULTIPLE BROWSERS PER NODE

GENESYS

Multiple browsers per node

With video recording, nodes only run 1 test at a time

Instead of having a group of nodes for

- Chrome tests
- Firefox tests
- Edge tests

Let’s make a single set of nodes with

- Chrome, Edge, and Firefox!

Multiple Browsers per Node

Step 1 - Combine the Docker Selenium `dockerfiles`

From NodeChrome:

- Chrome Launch Script Wrapper section
- Chrome webdriver section
- wrap_chrome_binary file

From NodeFirefox:

- GeckoDriver section

From NodeEdge:

- Edge Launch Script Wrapper section
- Edge webdriver section
- wrap_edge_binary file

Multiple Browsers per Node

Step 2 – Modify the `dockerfile` to echo the correct `browser_name`

- `RUN echo "chrome,firefox,edge" > /opt/selenium/browser_name`

```
[[node.driver-configuration]]
display-name = "chrome"
stereotype = '{"browserName": "chrome", "browserVersion": "110.0", "platformName": "Linux"}'
max-sessions = 1

[[node.driver-configuration]]
display-name = "firefox"
stereotype = '{"browserName": "firefox", "browserVersion": "110.0", "platformName": "Linux"}'
max-sessions = 1

[[node.driver-configuration]]
display-name = "edge"
stereotype = '{"browserName": "MicrosoftEdge", "browserVersion": "110.0", "platformName": "Linux"}'
max-sessions = 1
```

Multiple Browsers per Node

Step 3 – combine `generate_config`

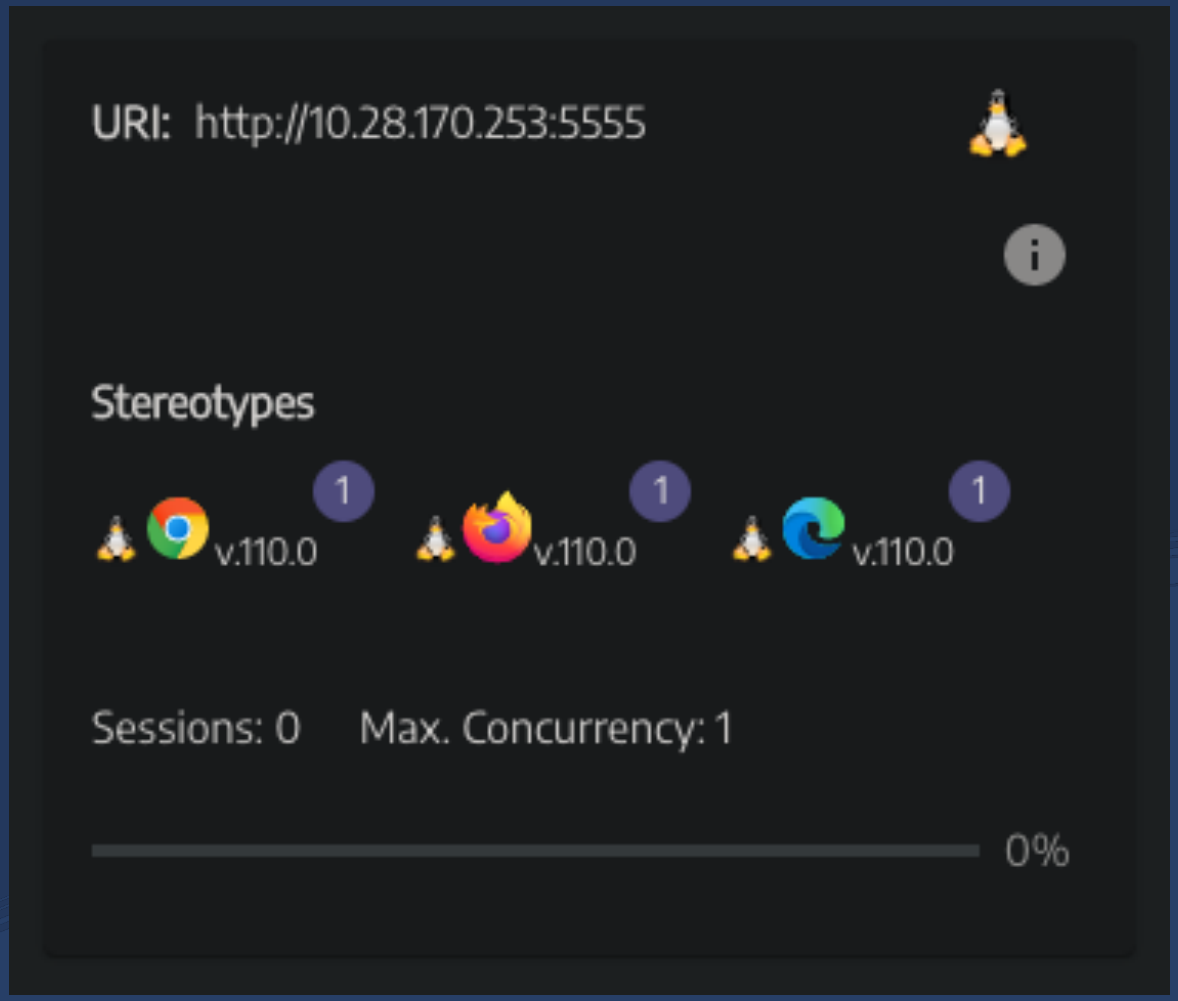
- Copy SE_NODE_STEREOTYPE for each browser type
- Copy section that writes to config.toml for each browser type

```
elif [[ "${SE_NODE_BROWSER_NAME}" == "chrome,firefox,edge" ]]; then
SE_NODE_STEREOTYPE_CHROME="{\"browserName\": \"${SE_NODE_BROWSER_NAME_CHROME}\", \"browserVersion\":\"${SE_NODE_BROWSER_VERSION_CHROME}\",
\"platformName\": \"Linux\"}"

SE_NODE_STEREOTYPE_FIREFOX="{\"browserName\": \"${SE_NODE_BROWSER_NAME_FIREFOX}\", \"browserVersion\": \"${SE_NODE_BROWSER_VERSION_FIREFOX}\",
\"platformName\": \"Linux\"}"

SE_NODE_STEREOTYPE_EDGE="{\"browserName\": \"${SE_NODE_BROWSER_NAME_EDGE}\", \"browserVersion\": \"${SE_NODE_BROWSER_VERSION_EDGE}\", \"platformName\":
\"Linux\"}”

echo "[[node.driver-configuration]]" >> "$FILENAME”
echo "display-name = \"chrome\"" >> "$FILENAME”
echo "stereotype = '${SE_NODE_STEREOTYPE_CHROME}'" >> "$FILENAME”
echo "max-sessions = ${SE_NODE_MAX_SESSIONS}" >> "$FILENAME"

echo "[[node.driver-configuration]]" >> "$FILENAME”
echo "display-name = \"firefox\"" >> "$FILENAME”
echo "stereotype = '${SE_NODE_STEREOTYPE_FIREFOX}'" >> "$FILENAME”
echo "max-sessions = ${SE_NODE_MAX_SESSIONS}" >> "$FILENAME"

echo "[[node.driver-configuration]]" >> "$FILENAME”
echo "display-name = \"edge\"" >> "$FILENAME”
echo "stereotype = '${SE_NODE_STEREOTYPE_EDGE}'" >> "$FILENAME”
echo "max-sessions = ${SE_NODE_MAX_SESSIONS}" >> "$FILENAME"
```

GENESYS

Multiple browsers per node

Step 4 – Run your container!



LET'S HAVE A CHAT.

GENESYS

HAVING A CHAT

Our grid is now:

- Deployed
- Self healing
- Scalable
- Capable of video recording
- CICD compliant

That means we’re finished, right????

- Maybe
- It depends

Common Complications

Out of the box

- Queueing just works

The complications come from

- Tests being run from automation tools
- Cloud based load balancers
- Built in Selenium timeouts
- Testing frameworks
- Other hidden timeouts that you find out about months into development

COMMON COMPLICATIONS

Timeouts 

ZOMBIE NODES

Timeouts and Queueing

Timeouts internal to Selenium

- –session-request-timeout on new session queue
  - Defaults to 300
  - I set it to 290

- --session-timeout on node
  - Defaults to 300
  - I set it to 600

Timeouts and Queueing

Timeouts external to Selenium

Load balancers

- AWS ELB/ALB, GCP Cloud Load Balancing, Azure Load Balancer
- 30-60 second default timeouts
- I changed to 350 seconds

Others

- Internal timeouts
- Frameworks
- CI/CD tooling
- Retry logic

Timeouts and Queueing

“These things all seem kinda simple”

“There’s only a couple of them”

“What’s the big deal”

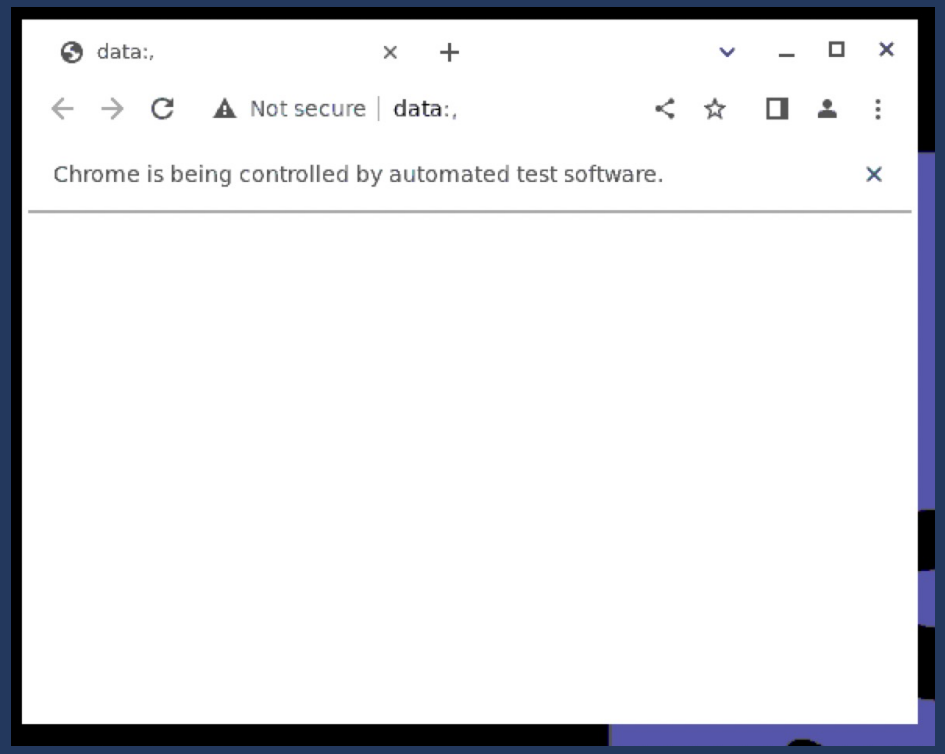
”Who cares if a node gets tied up for a bit”

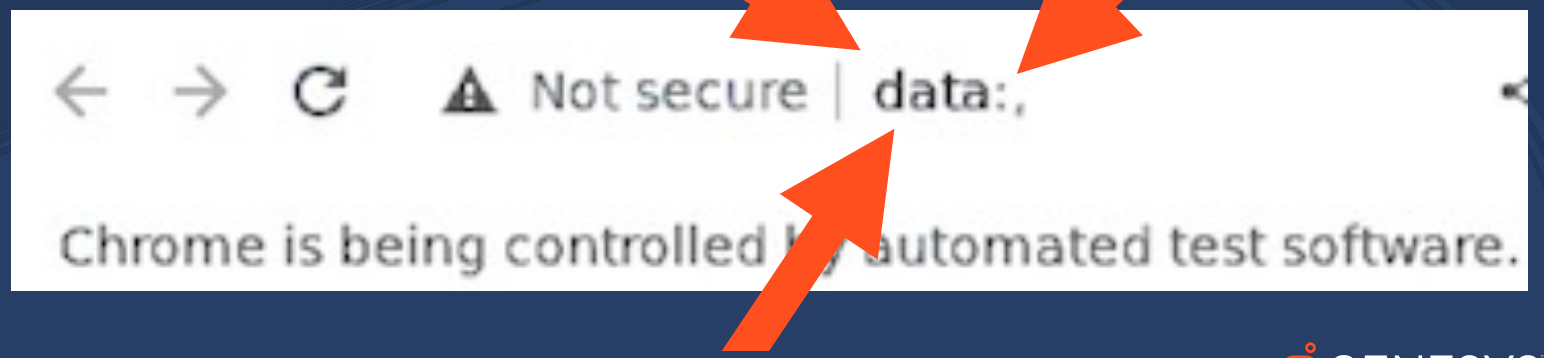
COMMON COMPLICATIONS

Timeouts

ZOMBIE NODES 

ZOMBIE NODES





GENESYS

ZOMBIE NODES

Occurs when:

- Test enters queue
- Test is abandoned before –session-request-timeout
- Test remains in queue
- Node becomes available
- Abandoned test takes a node
- Node is a ZOMBIE for duration of --session-timeout

ZOMBIE NODES

How can this play out?
Assuming 3x retry after 60s logic is in place:

- Grid is at capacity
- Test comes in, is abandoned before it gets a node
- Test retries, fails
- Test retries, fails
- Test retries, fails

In 4 minutes, 1 test made 4 zombie nodes
As more tests come in, grid is rendered useless

ZOMBIE NODES

So what do we do?

- Identify all timeouts that may affect you
  - If unsure, note the elapsed time between failures
- Make sure to set them in a responsible manner
- --session-request-timeout is always less than:
  - Load balancer timeout
  - Framework timeouts
  - Job timeouts
  - etc

ZOMBIE NODES

Take a deep breath

Q&A

GENESYS