**Introduction**:

**Setup**:

pipenv shell 🡪 to create a pip environment shell

pip install locust 🡪 to install locust in this environment

Test Scenarios in plain old python:

1. Looping behavior of users.
2. Perform some conditional behavior
3. Calculations

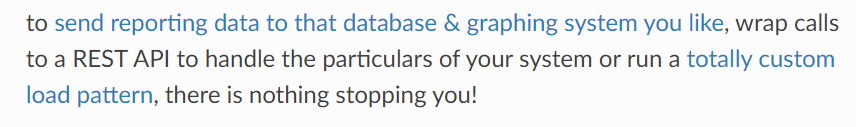
Locust runs every user inside its own *greenlet* (lightweight process/ co-routine).

Q>What is even-based framework (*gevent*) 🡪Makes it possible for single process to be handled by many thousands concurrent users.

Web based UI of Locust

Q> Locust can run without UI as well, making it easy for CI/CD testing, How?

Q>Just write a client for what you want to test, How?

Q> how?

Used for generating realistic load against dynamic websites where most pages have different content for different users.

Limitations of JMeter:

🡪 It is thread bound which means for every user to stimulate you need a separate thread. Needless to say benchmarking thousands of users on a single machine is not feasible.

*Topic*: Thread based process, benchmarking users

**Video information**:

🡪 Can test any system/protocol (WebSockets, selenium/WebDriver, Kafka, Postgres)

*Topic*: system/protocol integrated with Locust.

🡪 We can easily modify/extend Locust capabilities using Hooks. Like sending your data to a certain database.

*Topic*: Locust – Hooks, Connect database with locust.

🡪 from locust import HttpUser

HttpUser is similar to virtual users in load runner or thread groups in JMeter.

*Topic*: virtual users, thread groups.

🡪 A user defines a behavior which you can run in Locust.

*Topic*: Users in Locust

🡪 class DemoUser(HttpUser):

@task

Here our user is DemoUser and tasks are something that users can run.

*Topic*: Tasks in Locust

🡪 @task

def hello\_world(self):

self.client.get(“/hello”)

self.client.get(“/world”)

This task is called hello\_world. It is using the client which is an http session built on python requests framework which we use to call to end points.

*Topic*: self, client, http session in Python request framework.

🡪 @task(3)

def view\_item(self):

Task with higher weight(3). This means this task will be executed three times as much as previous task.

*Topic*: weights in locust task.

🡪 def view\_item(self):

for item\_id in range(10):

self.client.get(“/item?item\_id={item\_id}, name=”/item”)

time.sleep(1)

This loop is to show how we can use regular programming constructs in the locust file.

*Topic*: Regular Python programming in Locust file.

🡪 def on\_start(self):

self.client.post(“/login”, json={“username”: “foo”, “password”: “bar” })

here on\_start function will be called before this user does any tasks. For example Performing login or get some data from a specific user.

locust -f demo.py –H http://localhost:8000

name of the file

Where I want to direct my load

Create a fake service and introduce a performance bug into it.

*Topic*: How to use Django to create a local request.

🡪 You can use web UI for locust or you can do everything from command line.

*Topic*: Locust Web UI and command line

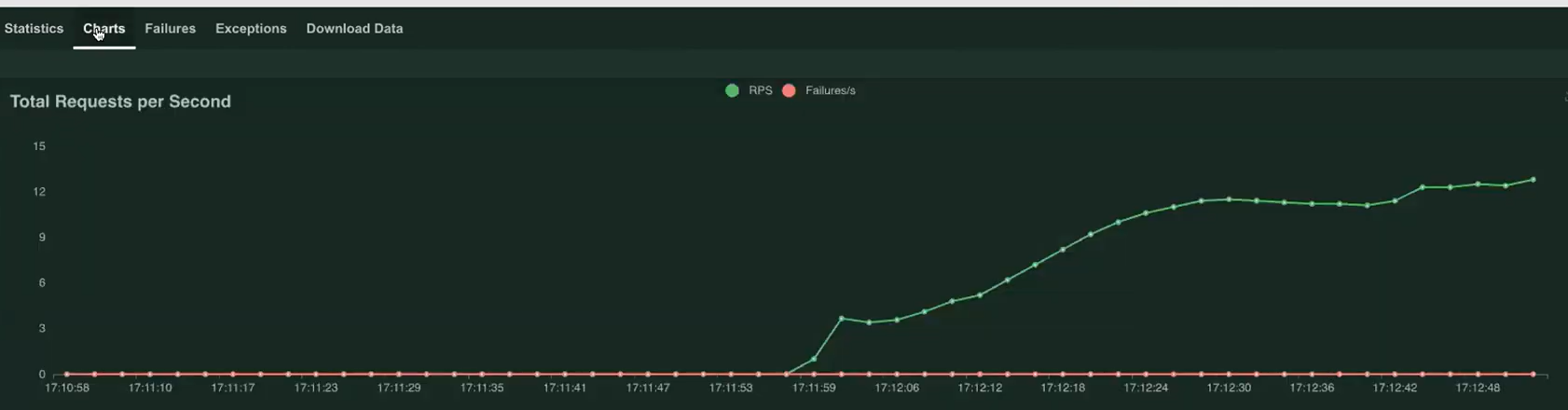
🡪 “We send our logs in to a time scaled database then use grafana to visualize everything”

*Topic*: Sending locust logs to database.

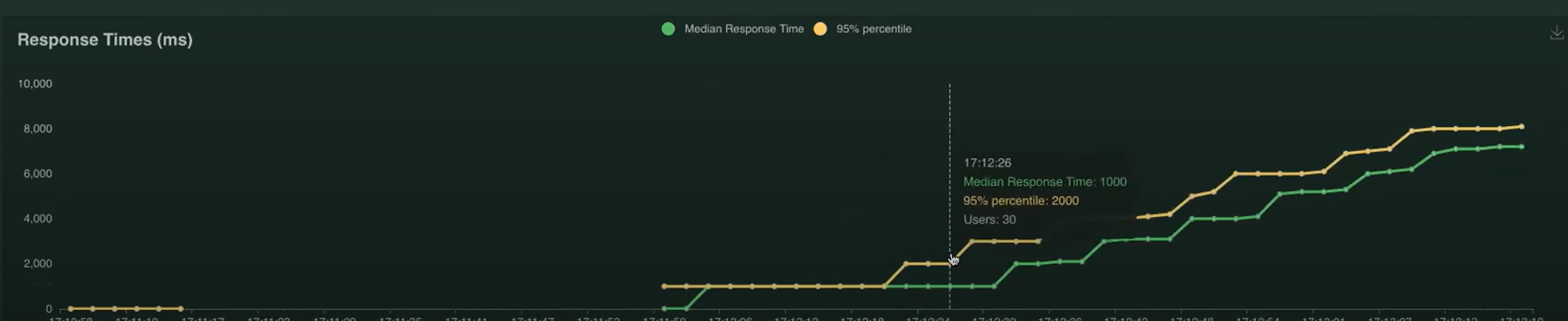
🡪 Statistics



🡪 In chart under Total Requests per second, we can see total number of requests increasing as we ramp up the load



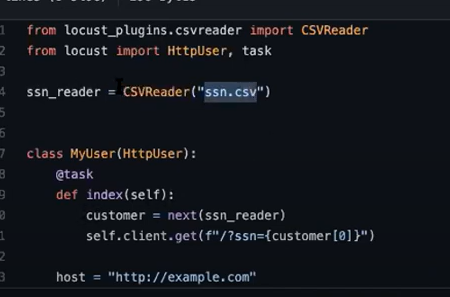
🡪 Response Times(ms) is increasing as well



What is the best practice for data driven testing, how to pull data that I need for my test? How to query test data?

A> Several ways to handle test data in Locust. Basic one is just use some sort of CSV file which we can read directly.

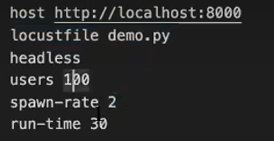
We use locust-plugins



*Topic*: locust-plugins, CSVReader.

🡪 We can configure Locust not just by command line parameters but also by using locust configuration file.

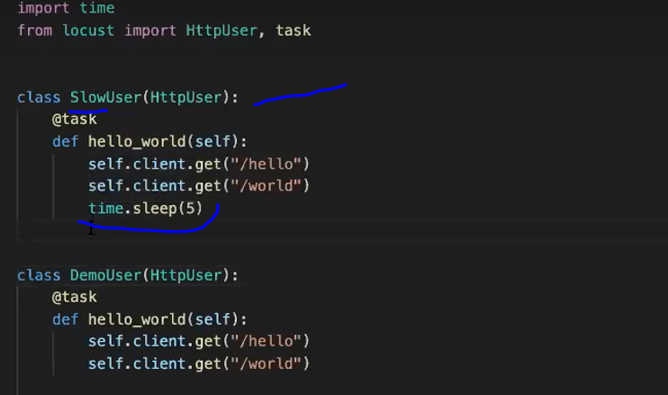
*Topic*: locust.conf



run-time limit means after 30 seconds it will stop.

Just type locust in command line and execution with these parameters in configuration file will begin.

🡪 Simulate slow user and normal user:



There are two users in this locust file, so requests will be divided 50-50.

SlowUser has time.sleep(5) so it will be slower than DemoUser.

🡪 How to simulate slow network connection?

A>Some Linux tool or Firewall or Selenium WebDriver client.

**Distributed Load generation**:

