From 0 to 1:
How to start with load testing

1 IDENTIFY WHAT YOU NEED TO TEST

Load testing everything at once is unnecessary and often counterproductive. Instead, focus on the application or API that handles critical traffic or revenue.

- ☐ **Identify the application or service you want to evaluate**
- ☐ **List the most important user journeys (login, search, checkout, etc.)**
- ☐ **Identify the APIs or endpoints used in those journeys**
- ☐ **Determine which flows generate the most traffic in production**
- ☐ **Confirm which failures would have the largest business impact**

Done when: You have clearly defined the application, APIs, and user journeys that your first load test will focus on.

2 DEFINE HOW YOU WILL EVALUATE PERFORMANCE

Before running a load test, define what "good performance" means for your system. Without clear targets, test results are difficult to interpret and teams may disagree on whether performance is acceptable.

- ☐ **Define the service level objectives (SLOs) for the system, including response time targets**
- ☐ **Estimate the traffic volume the system must support in production**
- ☐ **Define the acceptable error rate under load**
- ☐ **Define how long the system must sustain this traffic level**
- ☐ **Write a clear performance objective that combines these metrics**

Done when: The team agrees on the defined performance targets, expected traffic, and acceptable error rate that will determine whether the test passes or fails.

3 DEFINE OWNERSHIP AND PROCESS

Load testing is most effective when responsibility is clearly assigned. Without ownership, tests are often created once and never updated. Defining who owns the tests and how they fit into the development workflow ensures the practice becomes sustainable.

- ☐ **Identify the team responsible for load testing**
- ☐ **Store test scripts in a shared Git repository**
- ☐ **Define how simulations will be maintained over time**
- ☐ **Decide when tests should run in the delivery pipeline**
- ☐ **Define how results will be shared with other teams**

Done when: A team owns the tests, scripts are stored in version control, and the testing process is documented.

From 0 to 1:
How to start with load testing

4 DESIGN YOUR PERFORMANCE TEST

Now translate your performance objectives into a test scenario. The goal is to simulate realistic traffic patterns so the system behaves as it would in production.

- ☐ **Define the scenario based on user journeys or API requests**
- ☐ **Estimate the traffic volume to simulate**
- ☐ **Define the distribution of user actions and API requests during the test**
- ☐ **Select the injection pattern (load test, spike test, stress test...)**
- ☐ **Define how traffic will ramp up and down during the test**

Done when: You have a documented test scenario describing the traffic volume, user actions, and load pattern.

5 PREPARE THE TEST ENVIRONMENT

Load testing results are only reliable if the environment closely resembles production. This step ensures the infrastructure and monitoring tools are ready to support realistic testing.

- ☐ **Run tests in a staging or pre-production environment**
- ☐ **Ensure infrastructure configuration matches production as closely as possible**
- ☐ **Verify databases and dependent services are accessible**
- ☐ **Configure monitoring and observability dashboards**
- ☐ **Confirm the environment can support the expected traffic**

Done when: Your test environment is ready, monitoring tools are active, and the infrastructure can support the planned test load.

6 VALIDATE THE TEST FLOW

Before generating large amounts of traffic, verify that the scenario behaves correctly. Many first load tests fail because authentication, sessions, or dynamic parameters are not configured properly.

- ☐ **Run a smoke test with a small number of virtual users**
- ☐ **Verify requests reach the correct endpoints**
- ☐ **Confirm responses contain the expected data**
- ☐ **Check authentication and session handling**
- ☐ **Ensure monitoring tools capture the test traffic**

Done when: The scenario runs successfully with a small number of users and generates valid responses.

From 0 to 1:
How to start with load testing

7 RUN THE FIRST BASELINE LOAD TEST

Once the test script works correctly, run the full test under realistic traffic conditions. This first run will establish a baseline for future performance comparisons.

- [ ] **Gradually ramp traffic up to the target load**
- [ ] **Maintain steady traffic for the planned duration**
- [ ] **Capture latency percentiles (p50, p95, p99)**
- [ ] **Record throughput and error rates**
- [ ] **Monitor infrastructure metrics such as CPU, memory, and database usage**

✓ **Done when:** You have recorded the first baseline metrics describing how the system performs under expected load.

8 ANALYZE THE RESULTS

The goal of this step is to understand how the system behaves under load and identify potential bottlenecks. Interpreting the results helps teams prioritize performance improvements.

- [ ] **Compare results with the defined performance objectives**
- [ ] **Identify endpoints with high latency**
- [ ] **Review latency percentiles instead of averages**
- [ ] **Correlate application metrics with infrastructure metrics**
- [ ] **Document performance issues and potential improvements**

✓ **Done when:** The scenario runs successfully under expected load

9 AUTOMATE THE PROCESS

To make load testing sustainable, integrate it into your development workflow. Automation ensures performance regressions are detected before they reach production.

- [ ] **Integrate load tests into your CI/CD pipeline**
- [ ] **Define automated pass or fail thresholds**
- [ ] **Run regular performance tests across releases**
- [ ] **Compare results against the established baseline**
- [ ] **Share performance dashboards with engineering teams**

✓ **Done when:** Load tests run automatically as part of the delivery pipeline and help detect regressions before deployment.