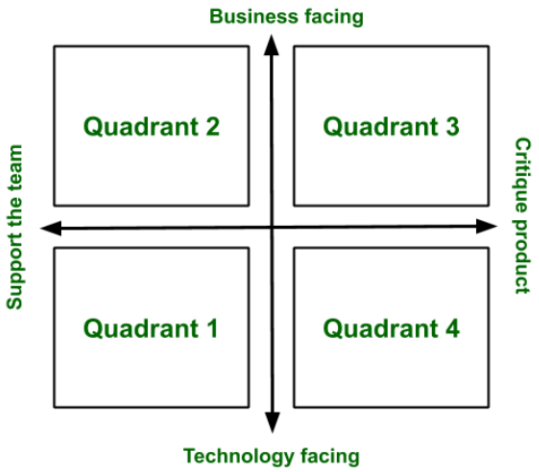
Assignment:
At Hadrian, we're developing an API first platform as many of our customers will use Hadrian through their own in-house build dashboards or through our integrations. We would like you to write a short proposal on how to test this platform. How would you structure your E2E tests, how to cover multiple microservices, what type of tools would you use, and how to check the quality of tests might be interesting areas to cover.

One of the core principles of Agile QA methodology is Agile Testing Quadrants:



Let me introduce my proposal using that principle. I've added 3 steps to each Quadrant, which should be in my opinion top priorities in building the platform.
It is an important part of the principle, that a quadrant number is an arbitrary number and not a priority. So depending on what Hadrian already has, priorities could vary on urgency and ROI.

Quadrant 2: **Regression e2e suite** To know if the tracks are not lost, nightly run to check if e2e processes are intact. **Smoke pre-release e2e suite** To avoid rollbacks, quick pre-release deployment checks as a smoke alarm. **Production dashboards** To know when things go wrong before it matters, dashboards with critical values from production.	Quadrant 3: **QA Review** To identify problems early, QA should participate in the development on early stages. **Tests as Documentation** To maintain system knowledge, if we do not have proper documentation tests should be as close as they can be to be used as such. **Exploration testing** To look outside the frame, spend some time trying to break things.
Quadrant 1: **Component-based unit tests integrated into CI pipeline** To do the heavy lifting. With coverage metrics merge gates. **Fast results** To avoid respite, automated tests should return results before the developer switches to the next task. **Contract testing** To prevent the test environment from looking like a zoo, contract testing allows cross-checks of dependencies in microservices.	Quadrant 4: **Ourselves as a client** To make sure we do things right, we need to use our own product ourselves. QA assuming the role of a client. **Load and performance testing** To avoid scaling problems, we should create and maintain weekly performance and load test suites **Recovery and Rollback exercise** To be prepared in case of things go bananas, we should try out the worst scenarios.

I deliberately did not specify any tools, because there is a high variety of them. And the decision on which ones to use mostly depends on whatever tools developers are using.