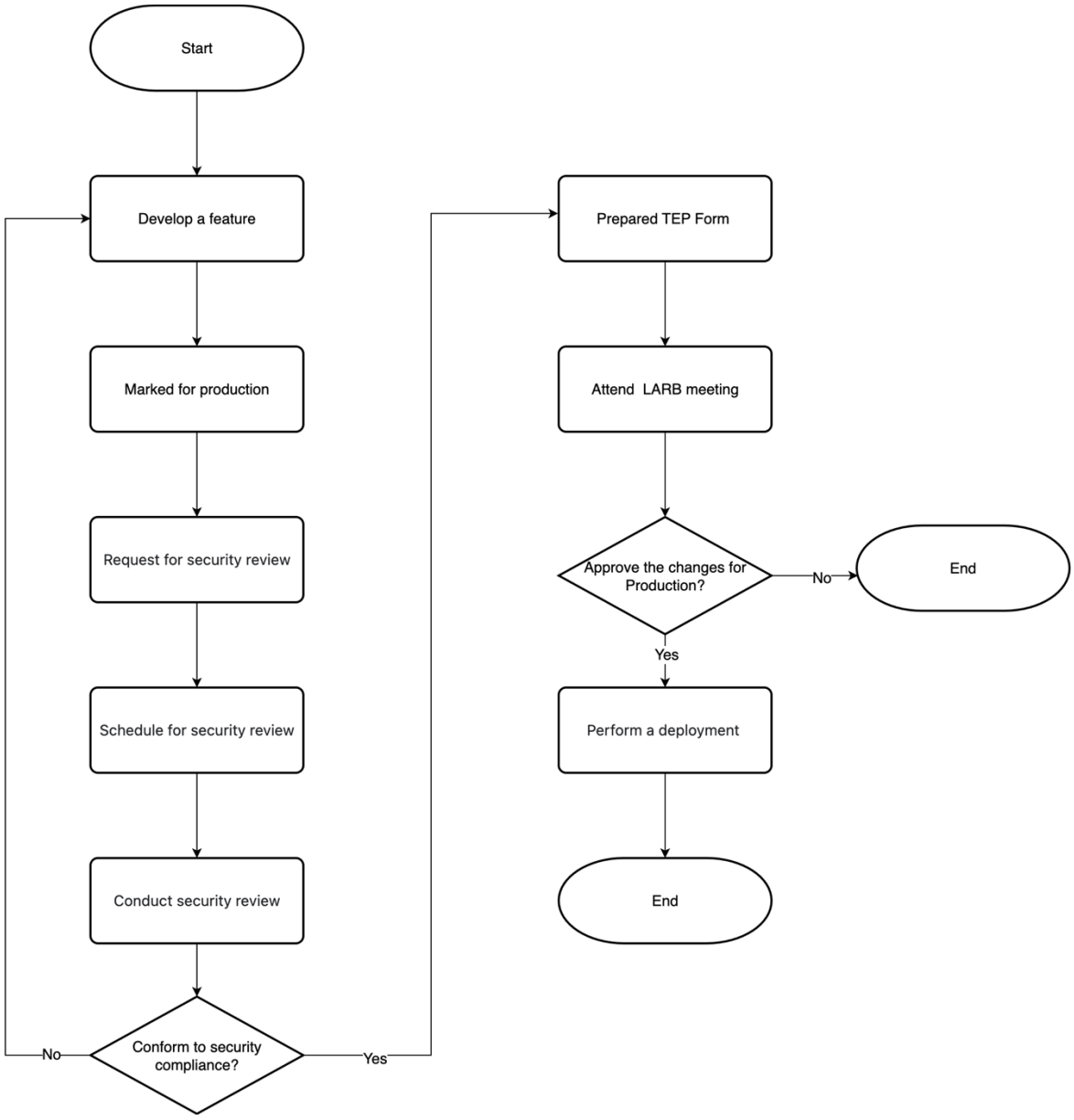
1. **List and describe the constraints that Maxine discovers in the deployment process?**

   In the deployment process, Maxine discovers that the greatest constraints come from the Security department and the TEP-LARB (Technology Evaluation Process and Lead Architecture Review Board). Even though the Data Hub team can now quickly develop, test, and prepare features for release, they still cannot deploy them to production without approval from these two powerful groups.

   The Security department acts as a mandatory gatekeeper. Every significant change must undergo a comprehensive security audit and compliance review, which can take weeks or months. Security insists on performing its own due diligence, requiring extensive documentation and refusing to prioritize urgent business needs. This delay prevents the team from deploying updates on demand.

   The TEP-LARB represents an even deeper organizational constraint. It's an outdated committee created decades ago to "review new technologies," but in practice, it has become a bureaucratic obstacle to innovation. To get approval, teams must fill out a 50-page form, attend long meetings, and wait months for a decision. The LARB's culture is focused on saying "no" to change rather than enabling progress. When Maxine and her team propose containerized environments and automated deployments, the LARB refuses to approve them, showing how rigid governance processes limit agility and innovation.

2. **Create a low-fidelity diagram of the current deployment process.**



3. **What are the DevOps processes and behaviours currently implemented in the deployment process?**

   First, the team establishes a continuous integration system that automatically builds and tests the code whenever a developer checks in changes. Developers and QA engineers collaborate to create automated tests stored in the same source code repository. When a test fails, the system provides instant feedback, and developers usually fix the issue within minutes. Maxine observes the new workflow in action as the continuous integration server runs builds and automated tests for every check-in, and developers correct failed tests within ten minutes. This practice greatly improves feedback speed, code quality, and deployment readiness.

Second, the team implements containerization to achieve environment consistency. Data Hub has been fully migrated to Docker containers, allowing any developer to deploy an identical environment on their laptop within seconds. As Tom explains, “instead of waiting weeks to get access to one of the scarce QA environments, you can just run this Docker image on your laptop.” This ensures that “if it works in Dev, it works in Test,” eliminating environment drift and long provisioning delays across environments.

Another key DevOps practice in the deployment process is the “You build it, you run it” principle. Developers are no longer only responsible for writing code; they also take ownership of deployment and ongoing operations. When Tom asks whether developers will have to take production on-call duties, Brent responds, “Hell, yes. You build it, you run it.” This marks a fundamental cultural shift, demonstrating the team’s acceptance of full end-to-end responsibility for the systems they create.

In addition, Chris issues a formal memo stating that the team “will be exempt from the normal rules and regulations around changes, able to test their own code, deploy it, and operate it in production themselves.” This leadership support enables true deployment autonomy, aligning organizational policy with DevOps principles.

4. **What are the potential improvements to the deployment process that Maxine identifies? How do they support DevOps behaviour and processes?**

   First, she recognizes that deployment is the current constraint in the value stream. While Development can quickly get features ready, the Operations team’s slow release cycles delay production deployments by weeks. To improve this, Maxine and her team aim to decouple Data Hub from the larger Phoenix system, allowing them to deploy changes independently and on demand. This approach supports continuous delivery, a core DevOps principle, by enabling rapid, safe, and frequent deployments without waiting for the traditional release schedule.

   Another improvement involves creating fully reproducible environments across Development, Test, and Production using containers and automated configuration management tools. By standardizing environments and automating builds, tests, and deployments, Maxine ensures that each stage behaves consistently, reducing errors and rollbacks—a key DevOps practice that emphasizes reliability and feedback loops. Additionally, by lifting restrictions and allowing the team to operate production services themselves, the team can implement rapid experimentation, respond quickly to customer needs, and iterate on features more effectively. This aligns with DevOps culture by fostering collaboration between Development and Operations, breaking down silos, and introducing a sense of shared responsibility for both software quality and system stability.

   Maxine also identifies improvements in communication and accountability as part of the deployment process. By bringing the product manager closer to the team, questions and clarifications are answered immediately, reducing waiting time and

enabling faster development cycles. This change reflects the DevOps focus on optimizing the entire value stream rather than isolated activities, enabling teams to deliver value to customers continuously and efficiently.