# ShuftiPro DevOps Plan

## Highlighting the Issues

- Our current infrastructure exhibits varying resource utilization patterns, with many CPU threads, significant RAM, and ample disk space frequently remain unused. However, there are instances where certain components heavily utilize these resources, leading to potential server congestion. The imbalance in resource consumption across different aspects of our system can result in inefficiencies and occasional server chokepoints. Addressing these disparities in resource allocation and optimizing the utilization of available CPU threads, RAM, and disk space will be a key focus to ensure consistent and efficient performance across all aspects of our infrastructure.
- Currently, our infrastructure relies on multiple servers, but we lack the redundancy and failover mechanisms provided by multiple clusters. In the event of a misfortune leading to the unavailability of one cluster, our system may experience service disruptions or downtime. To enhance the resilience and reliability of our infrastructure, it is crucial to implement a multi-cluster architecture with load balancing and failover capabilities.
- In our current organizational structure, we lack dedicated teams responsible for ensuring the availability of our applications to the world. This absence of clear ownership has resulted in a potential blame game between our DevOps and development teams. DevOps technologies, this has led to an atmosphere where responsibilities are unclear and accountability is diffuse. Without a streamlined approach to managing availability and reliability, workplace dynamics have the potential to become strained, fostering an environment that could be described as challenging and counterproductive.

# Road Map

**Minimum Resources Maximum Output:**

- To optimize resource utilization and maximize output, we propose the implementation of multiple clusters across a subset of servers, excluding those designated as backup servers. By distributing our workload among these clusters, we aim to achieve not only a more efficient use of resources but also to ensure a higher level of availability for our application. The establishment of multiple clusters serves as a strategic approach to guaranteeing near 100% uptime while concurrently mitigating infrastructure costs. This model allows us to harness the full potential of our existing resources, enhancing the overall efficiency and reliability of our application deployment.

**Roles & Responsibilities:**

- The delineation of responsibilities between the development and DevOps teams is essential for operational clarity. The development team is exclusively tasked with the creation and refinement of code, focusing on the innovation and functionality of our applications. Concurrently, the DevOps team assumes sole responsibility for the deployment and continuous availability of these applications. This division of labor ensures a streamlined and efficient workflow, allowing each team to specialize in their respective domains. By articulating these distinct roles, we aim to foster collaboration, enhance accountability, and achieve optimal outcomes in both the developmental and operational facets of our application lifecycle.
- The DevOps team bears the responsibility of facilitating a seamless integration and deployment process for the development team. This involves providing interfaces that empower developers to efficiently push and merge their code into the system. Moreover, the DevOps team ensures that the development team has access to a well structure environment that supports rigorous testing. This collaborative approach is designed to optimize the code delivery pipeline and enhance the overall efficiency of the development process.
- In collaboration with the development team, the DevOps team is instrumental in creating interfaces that enable testers to conduct thorough testing of the application code. This includes providing a testing environment conducive to comprehensive testing procedures. Furthermore, the DevOps team works to streamline the generation of detailed reports for test cases, offering valuable insights into the quality and functionality of the code. This collaborative effort ensures a robust testing framework and contributes to the overall reliability and performance of our applications.

**Toolchain selection:**

The selection of appropriate tools is integral to seamlessly integrating deployment and testing processes into our workflow. The judicious choice of tools can significantly streamline these aspects, making deployment and testing a seamlessly orchestrated and efficient operation. This strategic selection contributes to a cohesive and well-integrated software development lifecycle, ensuring that the complexities of deployment and testing are navigated with precision and effectiveness.

| | | |
|---|---|---|
| **Continuous Integration (CI) tools** | Jenkins, CircleCI, GitLab CI/CD | Automate code integration, build, and testing as developers commit code changes. |
| **Continuous Delivery (CD) tools** | Jenkins, CircleCI, GitLab CI/CD | Automate and manage the deployment of applications and infrastructure to various environments. |
| **Containerization and orchestration tools** | Docker, Kubernetes | Create, package, and manage applications in lightweight containers for consistent deployment. |
| **Monitoring and logging tools** | ELK Stack | Monitor application and infrastructure health, collect logs and generate insights for troubleshooting and optimization. |
| **Version control tools** | GitLab | Manage and track changes to source code, enabling collaboration, branching, and version history. |
| **Testing and Quality assurance tools** | Selenium, JUnit, Cyprus. Depends on QA Teams | Automate testing processes, manage test cases, and ensure software quality and reliability. |
| **Security and Compliance tools** | HashiCorp Vault | Implement security measures, vulnerability scanning, and compliance checks throughout the development lifecycle. |
| **Automation and scripting tools** | Ansible, Puppet | Automate repetitive tasks, configuration management, and infrastructure provisioning. |

**Continuous Improvement:**

- Instil within the organizational fabric a culture centered around continuous improvement, wherein teams systematically engage in the periodic review and enhancement of their operational processes. Foster a conducive environment for constructive feedback loops and retrospectives, empowering teams to discern and address areas ripe for refinement. This proactive approach promotes an ethos of perpetual enhancement, ensuring that operational procedures evolve in tandem with evolving demands and industry best practices.