# Organizing for DevOps

—--------------------------------------------------------

This module is all about organizing for DevOps. You will learn about the organizational impact of DevOps, how DevOps teams are structured, and the importance of everyone being responsible for success. You will discover the optimal organization for DevOps teams. You will review a variety of perspectives on DevOps and explore misconceptions about DevOps. This module addresses the importance of consequences, that is, allowing teams to feel the consequences of their actions on others who are involved in the work. You will see how a shared mindset empowers everyone to deliver customer value.

### **Learning Objectives**

* Describe the organizational impact of DevOps.
* Describe team structure in DevOps.
* Describe how making everyone responsible for success leads to success.

—----------------------------------------------------------------------------------------------------------------------------

## Video 1: Organizational Impact of DevOps

—---------------------------------------------------------------------------------

This lecture emphasizes the significance of organizational structure in implementing DevOps effectively. Here are the main takeaways:

1. **Transition to Agile**: An Agile mindset is essential for DevOps, as Agile's principles—small, focused, and cross-functional teams—support DevOps objectives. Teams should be dedicated to a single project to ensure focus, efficiency, and cohesion.
2. **Cross-Functional, Self-Organizing Teams**: Effective DevOps requires small, cross-functional teams with all necessary roles (developers, testers, operations, etc.) working collaboratively within the same team rather than in isolated silos. This setup minimizes dependencies on external teams and ticketing systems, fostering quicker progress and accountability.
3. **Conway's Law**: This principle, which states that a system's design will mirror the organization’s communication structure, illustrates why team organization impacts software design. For example, if teams are split by function (UI, backend, database), the resulting software will likely mirror these layers (e.g., a three-tier architecture). This law suggests that restructuring teams to mirror desired software architecture, like microservices, can lead to more efficient development aligned with business goals.
4. **Domain-Based Team Organization**: Optimal DevOps teams are organized around business domains (e.g., account management, personalization, warehouse operations) rather than technical layers. Domain-based teams possess end-to-end responsibility—from front-end UI to backend logic and database—allowing them to operate independently without needing other teams to fulfill dependencies. This structure supports the development of microservices, with each team functioning like a "mini-startup," accountable for their domain’s complete lifecycle, from development to maintenance.
5. **Long-Term Mission and Autonomy**: To maximize DevOps success, teams should have a long-term mission within a single business domain. Stability fosters ownership and pride, reducing turnover and helping teams build expertise in their domain. Team autonomy is crucial, allowing teams to manage their features from conception to deployment and ongoing support.

### **Practical Application and Real-Life Examples**

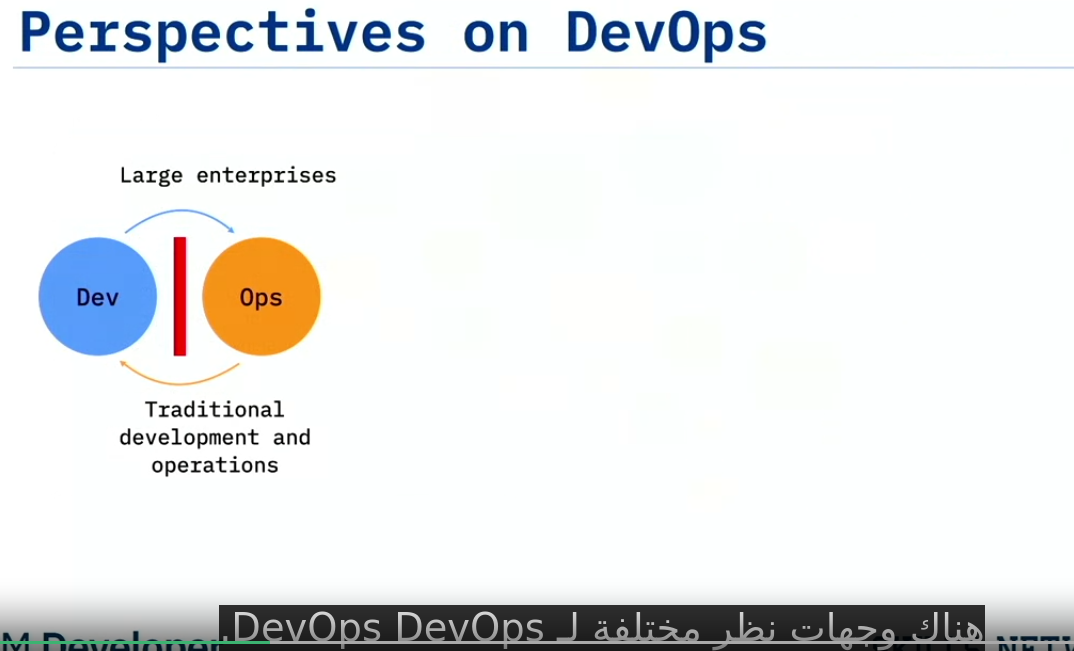
* **Real-Life Example**: Consider an e-commerce company. Instead of organizing teams by functions (e.g., a front-end team, backend team, and database team), the company could create teams around specific business functions—like the "checkout" team, "user account" team, and "product catalog" team. The checkout team, for instance, would have developers, testers, and operations staff focused on optimizing the checkout experience without external dependencies. This structure enables faster releases, reduces bottlenecks, and fosters ownership, directly aligning with the goals of DevOps and Conway’s Law.
* **Real-Life Example in a Software Company**: A company aiming to adopt microservices could reorganize its teams around microservice boundaries, such as billing, customer profiles, and analytics. Each team would handle everything necessary to support its service, from frontend to backend and database tasks. This arrangement not only speeds up development but also enables teams to specialize in their service, improving both quality and innovation over time.

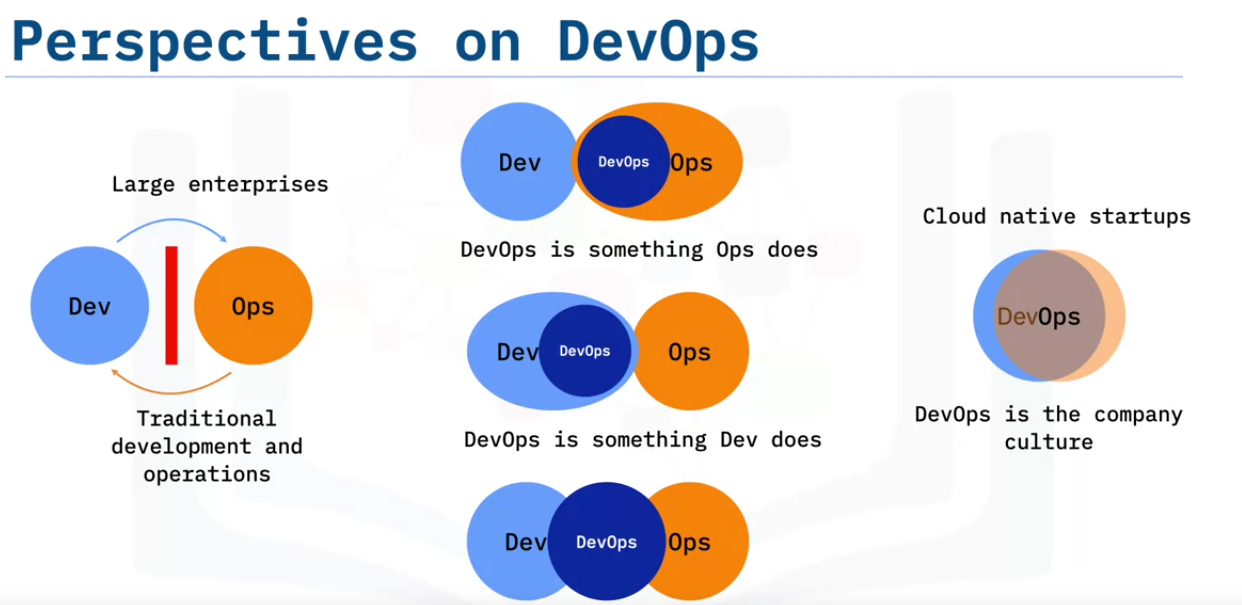
By aligning teams around business goals and reducing dependency on external processes, organizations can achieve more seamless DevOps integration, facilitating faster delivery, higher quality, and greater flexibility in adapting to changes.

—----------------------------------------------------------------------------------------------------------------------------

## Video 2: There is No DevOps Team

—---------------------------------------------





This lecture clarifies DevOps as a collaborative organizational culture rather than a distinct job or team function. Here are the key insights:

1. Understanding DevOps: DevOps is a mindset that combines development (Dev) and operations (Ops), promoting collaboration throughout the software lifecycle to improve software quality, stability, and delivery speed. It’s not just a technical operations role but an integrated approach where both development and operations share goals and work in sync.
2. Common Misconceptions: Many organizations misunderstand DevOps, viewing it as:
   * A function solely for Ops teams.
   * A responsibility only for developers.
   * A separate "DevOps team" that bridges development and operations.
3. However, these views create silos and misunderstand the core purpose of DevOps. Creating a standalone "DevOps team" actually contradicts DevOps principles, as it introduces another functional silo rather than fostering cross-functional collaboration.
4. DevOps as an Organizational Culture: DevOps should be a shared company culture, similar to Agile, where the entire organization embraces the principles of openness, collaboration, and continuous improvement. Both development and operations teams should have aligned goals, metrics, and responsibilities, working on the same team to avoid the traditional “wall of confusion,” which divides development's focus on new features from operations' focus on system stability.
5. **Implementing DevOps**: Success with DevOps requires that development and operations staff work together, ideally within cross-functional teams, following Lean and Agile principles. Emphasizing transparency, trust, and flexibility helps create an environment for high-quality, stable software delivery.

### **Practical Application and Real-Life Examples**

* **Startups and Small Teams**: Startups often naturally adopt DevOps because everyone works closely across different roles. For instance, developers and operations staff may collaborate on deployment and troubleshooting, avoiding silos by necessity. This collaborative setup aligns perfectly with DevOps values, leading to quicker, continuous releases.
* **Real-Life Example in Enterprises**: Consider a company implementing a new product feature. Traditionally, developers would pass completed code to an operations team for deployment, potentially resulting in delays or errors. By adopting DevOps, the same team could manage the entire lifecycle from code to deployment, with shared accountability for smooth delivery and operation. This end-to-end responsibility minimizes handoffs, ensures faster time-to-market, and reduces the risk of deployment issues.

By shifting to a collaborative DevOps culture, companies can reduce friction, deliver quality updates quickly, and respond to customer needs with greater agility.

—---------------------------------------------------------------------------------------------------------

## Video 3 Everyone is Responsible for Success

—------------------------------------------------------------

The lecture explains the importance of connecting teams with the consequences of their actions in a DevOps environment to maintain accountability and empathy across functional areas. Jez Humble’s quote—“Bad behavior arises when you abstract people away from the consequences of their actions”—highlights that when functional silos isolate team members, they lose sight of how their actions impact others. This separation leads to reduced ownership and apathy.

An example given was a company that faced quality issues in their software. They hired a dedicated QA team to improve quality, but this led to developers distancing themselves from testing. As a result, they focused solely on pushing features, expecting the QA team to handle all testing responsibilities. This approach reduced overall quality since developers stopped testing their code thoroughly, and the QA team became overwhelmed with the volume.

To prevent these issues, the following strategies are recommended:

1. **Create Cross-Functional Teams:** Allow members from different functions (e.g., development, operations, testing) to work closely, fostering ownership and collaboration. For example, a development team that includes QA and operations members would understand the full scope of quality requirements.
2. **Shared Responsibilities:** Have developers rotate through operations roles or put them on pager duty, making them responsible for the systems they build. This helps developers gain empathy for the challenges operations teams face, encouraging better code quality. For example, if a developer receives a call at 3 a.m. due to system issues, they are more likely to write stable code to avoid future disruptions.
3. **Shared Consciousness with Local Control:** While everyone should understand the organization’s objectives, teams should have autonomy to decide how to achieve their goals. This allows for flexibility and empowers teams to make decisions that directly benefit customer satisfaction.

A real-life example of this approach is Amazon’s “You build it, you run it” principle, where developers are responsible for both developing and maintaining their services. This responsibility motivates teams to prioritize quality and customer satisfaction from the beginning, aligning Dev and Ops under the shared goal of delivering value to the customer.

—---------------------------------------------------------------------------------------------------------

# **Summary and Highlights**

Congratulations! You have completed this lesson. At this point in the course, you know:

* Organizations need to have small, dedicated, cross-functional, self-organizing teams to successfully implement DevOps.
* Conway’s Law implies that a company’s design results are a direct reflection of the company’s communication structure.
* Instead of the traditional structure organized around technology, successful DevOps teams should be organized around business domains. Each team should have its own mission that aligns with a business domain.
* DevOps is a mindset that the whole organization adopts.
* DevOps solves problems caused by siloed teams.
* DevOps is the practice of development and operations engineers working together during the entire software lifecycle, following lean and Agile principles that allow them to deliver high-quality results.
* Actions without consequences can lead to apathy.
* Allowing teams to feel the effect of their actions fosters empathy, resulting in higher-quality work.
* The organizational objective of DevOps is to attain a shared mindset and empower everyone to deliver customer value.