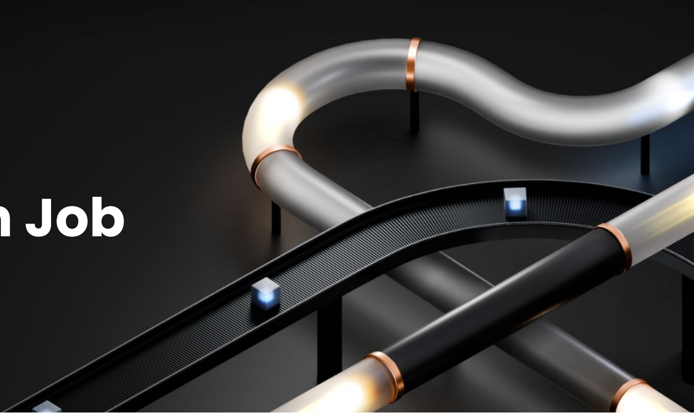
harness

Implementation Job Task Analysis



Purpose

This Job Task Analysis outlines the responsibilities, workflows, skills, and decision-making expected of a DevOps Implementation Engineer working in a consulting or professional services environment.
The goal is to capture the work engineers actually perform during implementations so organizations can build better onboarding, training, and certification programs around real job tasks.
The information reflects common practices across DevOps consulting teams and platform implementation roles.

1. Role Overview

DevOps Implementation Engineers help organizations adopt DevOps platforms and practices. Their work typically involves configuring tooling, guiding teams through implementation, and helping customers successfully integrate DevOps capabilities into their environments.
The role sits somewhere between engineering and consulting. Engineers need strong technical skills, but they also spend a lot of time working directly with customer teams to understand their environments and guide them toward workable solutions.

Typical areas of work include:

- Continuous Integration (CI)
- Continuous Delivery / Deployment (CD)
- Infrastructure as Code
- Cloud integrations
- DevOps automation
- Security and compliance workflows

Implementation engineers often work with multiple groups during an engagement, including development teams, platform teams, infrastructure engineers, and security teams.

2. Implementation Lifecycle

Most implementations follow a similar pattern from start to finish.

Discovery

The first step is understanding the customer's environment and goals. This usually involves reviewing their architecture, identifying dependencies, and clarifying what they want the platform to accomplish.
Typical activities include:

- Reviewing infrastructure architecture
- Understanding repository structures
- Assessing cloud environments
- Identifying integration requirements

Design

After discovery, engineers work with the customer to design how the platform will be implemented.
This can include:

- Designing pipeline structures
- Defining access control models
- Planning repository and workflow organization
- Identifying any architecture gaps that need to be addressed

Implementation

Once the design is agreed on, the engineer helps configure and deploy the platform components.
This often involves:

- Setting up agents, runners, or delegates
- Creating pipelines and templates
- Integrating source control and artifact repositories
- Configuring cloud environments and infrastructure access

Validation

After implementation, engineers work with the customer to verify everything is functioning correctly.
Common activities include:

- Running test pipelines
- Verifying integrations
- Troubleshooting configuration issues
- Confirming expected deployment behavior

Closeout

The final phase focuses on documentation and transition.
Typical activities include:

- Providing architecture documentation
- Reviewing implementation decisions
- Knowledge transfer sessions
- Transitioning ongoing ownership to the customer team

3. Day to Day Work

A typical week for an implementation engineer includes a mix of technical work and collaboration. Common activities include:

- Running implementation sessions or technical workshops
- Helping teams design or refine pipelines
- Troubleshooting environment or integration issues
- Documenting architecture and configuration decisions
- Coordinating with project managers or delivery leads
- Participating in internal design discussions or escalations

Much of the work happens live with customer teams, often during working sessions where configuration and troubleshooting happen together.

4. Complex Implementation Scenarios

Not every implementation follows a clean path.

Engineers often run into situations where they need to adapt quickly, such as:

- Organizations with limited DevOps maturity
- Legacy environments that don't align well with modern workflows
- Highly customized infrastructure
- Teams with conflicting technical priorities

In these cases, the engineer has to balance best practices with practical constraints. The goal is usually to move the organization forward while still respecting the realities of their environment.

5. Key Responsibilities

5.1 Internal/External Customer-Facing Work

A large part of the role involves working directly with internal or external customer teams.

Common responsibilities include:

- Running discovery sessions
- Clarifying technical requirements
- Leading design and architecture discussions
- Delivering technical walkthroughs
- Helping teams troubleshoot issues
- Documenting implementation outcomes

Good communication skills are essential, since engineers often work with both technical and non-technical stakeholders.

5. Key Responsibilities, continued

5.2 Cross-Org Collaboration

Implementation engineers also coordinate with cross-org teams to keep projects moving forward.

This may include:

- Updating project progress
- Flagging risks or scope issues
- Participating in design reviews
- Escalating complex technical problems
- Sharing lessons learned with other engineers

6. Technical Skills

Implementation engineers need a broad mix of infrastructure, automation, and DevOps knowledge.

Some of the most common technical areas include:

Infrastructure Platforms
- Kubernetes and container orchestration
- Cloud infrastructure across AWS, Azure, or GCP
- Environment configuration and lifecycle management

Infrastructure as Code
- Terraform or similar tools
- Template-based infrastructure provisioning

Automation and Scripting
- Shell scripting
- Python
- YAML configuration

DevOps Tooling
- Git-based source control
- CI/CD platforms
- Artifact management systems

Systems Administration
- Linux administration
- SSH access and permissions
- Identity and access management

Networking
- Diagnosing connectivity issues
- Load balancing and DNS configuration
- Secure service communication

7. Troubleshooting Skills

Troubleshooting is a large part of the job.
Engineers frequently need to diagnose issues across multiple systems at once. This may include:

- Analyzing pipeline execution logs
- Identifying integration failures
- Tracking down configuration issues
- Debugging automation scripts
- Designing rollback strategies when deployments fail

Strong troubleshooting instincts are often what separate experienced engineers from newer ones.

8. Decision-Making

Some parts of the job follow repeatable procedures, but many situations require independent judgment.

Examples of routine tasks include:

- Setting up integrations
- Configuring access control
- Creating pipeline templates

More complex work involves:

- Adapting implementations to unusual architectures
- Deciding when to escalate technical blockers
- Recommending architectural changes
- Managing tradeoffs between speed and long-term maintainability

9. Common Risks During Implementations

There are a few areas where mistakes can have a big impact.

Misconfiguration

- Incorrect settings can cause pipeline failures or deployment outages.

Over-customization

- Too much customization can make environments difficult to maintain.

Legacy migrations

- Simply copying legacy processes into a new platform often carries forward existing problems.

Incomplete discovery

- If the customer environment isn't fully understood early on, implementation issues tend to appear later in the project.

10. Core Competencies

Successful DevOps implementation engineers usually demonstrate strength in several areas.

Domain	Competency	Level
Infrastructure	Kubernetes and cloud platforms	Advanced
Infrastructure as Code	Terraform or similar tools	Advanced
Automation	Shell, Python, YAML	Intermediate
DevOps Platforms	CI/CD tooling and repository management	Advanced
Communication	Working with technical and business stakeholders	Advanced
Process	Discovery, scoping, documentation	Intermediate
Troubleshooting	Log analysis and issue diagnosis	Advanced
Project Coordination	Tracking progress and deliverables	Intermediate

11. How the Role is Evolving

As DevOps practices mature, the implementation engineer role is starting to overlap more with platform engineering.
Many organizations are shifting toward internal developer platforms, which means engineers increasingly help design reusable systems that support multiple teams.
Areas that are becoming more important include:

- Platform engineering
- Internal developer portals
- DevSecOps automation
- Multi-team pipeline architectures
- AI-assisted development workflows

In many cases, the role is evolving from pure implementation toward long-term platform design and enablement.

How the Job Task Analysis (JTA) Was Developed

This Job Task Analysis was developed through structured interviews with subject matter experts and managers who perform or oversee the role. The goal was to document the real work involved in the position; what people actually do, the skills they rely on, and the decisions they make; so training, certification, and job aids can better reflect day-to-day responsibilities.

This Job Task Analysis was developed by Kendra Marquart, Principal Technical Program Manager for Product Education at Harness. Interviews were conducted with subject matter experts and managers currently working in or overseeing the role to capture real-world responsibilities, skills, and workflows.