SWQ IT ACADEMY LEARNING PATHS

# Role Based DevOps / DevSecOps Learning Paths for Working Professionals

*Four starting points, one for developers, infra and support, security, and cloud or operations professionals, each with its own sequence and reasoning.*

SWQ IT Academy trains working IT professionals with a simple standard: skills you can actually demonstrate, not skills you have merely watched someone else use. The path that gets you there depends on where you're actually starting from, which is exactly what this resource is built around.

Built for working IT professionals with an existing background in development, infrastructure, support, security, or cloud operations, who want a transition plan shaped around what they already know, not a generic list that ignores it.


A part of the ENGITRIX group

INTRODUCTION

# The best next step for you depends on where you're standing.

Hand the same DevOps roadmap to a developer, a support engineer, and a security analyst, and you'll get three very different experiences of it. The developer breezes through the Git and scripting sections and stalls on infrastructure. The support engineer already thinks in servers and networks but has never written a pull request. The security analyst understands risk deeply but has rarely built the pipeline they'd eventually be securing.

One roadmap, three completely different experiences, because a roadmap only describes the destination. It says nothing about where each person is actually standing when they start reading it.

That's the gap this resource is built to close. Instead of one generic sequence, it gives you four, one for each common starting point, developer, infra or support, security, and cloud or operations. Each path names what you already bring, what's actually missing, and the order that gets you from where you are to a genuine transition, without wasting weeks relearning things you're already strong at or skipping things you've quietly never touched.

**THE PATTERN WE SEE MOST OFTEN**

*People don't usually fail this transition because they can't learn the material. They fail because they followed a sequence built for someone else's background, and burned three months on the wrong things first.*

Role based paths fix that by being honest about the asymmetry. Everyone eventually needs the same core skills, but the order you acquire them in, and how much time each one deserves, should look different depending on what you already carry into it.

BEFORE YOU CHOOSE

# How to use this resource

*Three principles before you jump to a path.*

### 01 Pick the path closest to your current role, not your dream role

Choose based on what you do today, developer, infra, security, or cloud and operations, not based on where you're headed. The path accounts for the destination already, it needs an honest starting point.

### 02 Follow one path, don't blend all four

Reading all four paths for context is fine. Trying to follow all four's sequences at once recreates the exact scattered problem this resource exists to prevent.

### 03 Pay attention to the sequence and the reasoning, not just the tool names

Two paths might both mention Terraform, but at different points and for different reasons. The order and the "why" matter more than the list of names, that's the actual content of a role based path.

> **A NOTE ON THE PROJECTS REFERENCED IN THIS DOCUMENT**
>
> Each path points to specific projects from SWQ IT Academy's Practical DevOps / DevSecOps Project Pack. If you don't have that pack yet, the project descriptions here are still enough to build from, the full pack simply gives you the complete step by step outline for each one.

PATH 1

# Developer → DevOps

*You already think in systems and logic. The gap is almost entirely on the operations side, infrastructure, deployment, and the environment code actually runs in.*

## What you already have

*Real strengths, not starting from zero*

Fluency in code, comfort with debugging, an existing relationship with Git, and an instinct for breaking problems into logical steps. You've likely written some tests and touched a build process, even if you've never owned it end to end.

## What you need to add

*The genuine gap, and it's consistent*

Comfort operating in Linux outside an IDE, infrastructure as code, container orchestration, and a working understanding of the cloud environment your applications actually run in. None of this is harder than what you already know, it's just unfamiliar territory.

### LEARNING SEQUENCE

| ORDER | FOCUS | WHY HERE, AND NOT SOMEWHERE ELSE |
|---|---|---|
| **1** | Linux and shell comfort | A quick refresh, not a deep dive, most of this is closer than you think from your existing terminal use |
| **2** | Docker | The most natural bridge from code to operations, packaging something you already understand |
| **3** | CI/CD with GitHub Actions | Automating a build and test process similar to ones you've already used, just now owned by you |
| **4** | Terraform | The real mindset shift, treating infrastructure as something declared in code, not clicked into existence |
| **5** | Kubernetes | Where your container and pipeline experience finally combine into something production shaped |

| | | |
|---|---|---|
| 6 | Cloud security baseline | Enough IAM and configuration awareness to not be the reason a deployment gets flagged |

### WHAT TO POSTPONE

Deep security specialisation, service mesh, and multi cloud architecture. All genuinely useful eventually, none of it belongs in your first year of this transition.

### PRACTICAL PROJECTS

- ✓ **Dockerize a simple application**, ideally one of your own, not a sample app.
- ✓ **A full CI/CD pipeline** that tests, builds, and deploys that same application automatically.
- ✓ **A Kubernetes deployment** of the same application once the pipeline is solid.

### 60 TO 90 DAY STARTING PLAN

#### First 30 days

*Linux refresh, then Docker*

**WEEKS 1 TO 2**

Linux command line and networking basics, faster than most since you already think in systems.

**WEEKS 3 TO 4**

Docker fundamentals, ending with your own application fully containerised.

#### Days 31 to 90

*Pipelines, infrastructure, and orchestration*

**DAYS 31 TO 50**

GitHub Actions, building the pipeline around your containerised application.

**DAYS 51 TO 70**

Terraform, provisioning the infrastructure your application will actually run on.

**DAYS 71 TO 90**

Kubernetes, bringing the container, pipeline, and infrastructure together into one deployment.

PATH 2

# Infra / Support → DevOps

*You already live in production. The gap is on the software side, code, automation, and treating infrastructure as something written rather than operated.*

## Existing strengths

*Genuinely hard won, not easily taught*

Real production instincts, comfort with servers and networking, incident response experience, and an understanding of what actually breaks in the real world. This is exactly the operational judgment many developers moving into DevOps are missing.

## Missing pieces

*Where the actual work is*

Scripting and light programming ability, a genuine Git habit rather than occasional use, and the shift from manually operating infrastructure to defining it in version controlled code. Automation as a default, not an afterthought.

### LEARNING SEQUENCE

| ORDER | FOCUS | WHY HERE, AND NOT SOMEWHERE ELSE |
|---|---|---|
| 1 | Git and scripting | The genuine foundation gap, worth real time here even though it feels basic |
| 2 | Docker | A natural extension of packaging and running services, familiar territory with a new interface |
| 3 | Terraform | Your infrastructure knowledge transfers directly, the shift is expressing it as code instead of clicks |
| 4 | CI/CD with GitHub Actions | Usually the real gap for this background, automating what you currently do by hand |
| 5 | Kubernetes | An extension of orchestration concepts you likely already understand at the infrastructure level |
| 6 | Cloud security baseline | |

Formalising security instincts you likely already apply informally in production

## PROJECT SUGGESTIONS

- ✓ **A simple Terraform deployment**, provisioning something you'd normally set up manually.
- ✓ **A basic CI workflow** in GitHub Actions, your first real scripting and automation habit.
- ✓ **Reusable Terraform modules**, turning your infrastructure knowledge into genuinely reusable code.

## COMMON MISTAKES

- × **Treating Terraform as just another way to do what you already do manually.** The state and code review discipline around it is the actual shift, not the syntax.
- × **Skipping real Git discipline.** Occasional commits won't hold up in a role that expects branches, pull requests, and code review as a daily habit.
- × **Underestimating the scripting gap.** "I can find my way around a script" and "I can write one from scratch" are different skills, and interviews test the second one.

## 60 TO 90 DAY STARTING PLAN

### First 30 days

*Scripting and Git, then Docker*

**WEEKS 1 TO 2**

Bash or Python fundamentals and a genuine Git branching and pull request habit.

**WEEKS 3 TO 4**

Docker fundamentals, containerising something you'd normally deploy manually.

## Days 31 to 90

*Infrastructure as code and automation*

**DAYS 31 TO 50**

Terraform, starting with a single resource and moving to a reusable module.

**DAYS 51 TO 70**

GitHub Actions, automating a process you currently do by hand.

**DAYS 71 TO 90**

Kubernetes basics, connecting your infrastructure and pipeline work into one deployment.

PATH 3

# Security → DevSecOps

*You already understand risk. The gap is hands on engineering, you need to be able to build the pipeline before you can credibly secure it.*

## Transferable strengths

*The part most engineers have to learn the hard way*

Risk thinking, attack surface awareness, IAM and access control instincts, and comfort with compliance and audit processes. This is precisely the judgment a DevSecOps role needs and most engineering backgrounds arrive without.

## Engineering and DevOps skills to add

*Where the real effort goes*

Hands on scripting, Docker, and CI/CD pipeline construction. Not to become a full time developer, but enough that you can build the systems you're meant to secure, not just review them after the fact.

### LEARNING SEQUENCE

| ORDER | FOCUS | WHY HERE, AND NOT SOMEWHERE ELSE |
|---|---|---|
| 1 | Linux, Git, and scripting | The same foundation everyone needs, don't skip it assuming security knowledge covers this gap |
| 2 | Docker | You can't meaningfully scan or secure a container you've never built yourself |
| 3 | Terraform | Infrastructure as code, the layer most cloud misconfigurations actually originate from |
| 4 | CI/CD with GitHub Actions | The pipeline has to exist before there's anything real to secure inside it |
| 5 | Cloud security, deepened | Now applied hands on against infrastructure and pipelines you actually built |

| | | |
|---|---|---|
| **6** | DevSecOps, pipeline security | Scanning, gating, and secrets handling, integrated directly into the pipeline from step four |
| **7** | Kubernetes, lighter depth | Enough to secure a cluster, not necessarily to administer one at deep operational depth |

### SECURITY, PIPELINE, AND CLOUD PRIORITIES

Weight your time toward pipeline security and cloud configuration over deep Kubernetes administration. A DevSecOps role generally cares more about what's gating a deployment and what an IAM policy actually permits than about advanced cluster operations, those usually stay with a dedicated platform team.

### PROJECT SUGGESTIONS

- ✓ **Add security scanning to CI/CD**, your first real proof that you can secure a pipeline, not just describe one.
- ✓ **Add IaC scanning and secrets handling** to a Terraform project, applying your risk instincts to infrastructure code specifically.
- ✓ **Build a mini DevSecOps pipeline**, combining several checks into one deliberately sequenced, gated flow.

### COMMON MISTAKES

- × **Staying theoretical.** Understanding OWASP categories doesn't demonstrate you can add a working scanning stage to a real pipeline. Build the thing.
- × **Skipping hands on pipeline building.** Jumping straight to security tooling without first building a working CI/CD pipeline leaves you securing something you don't fully understand.
- × **Assuming security knowledge alone is sufficient.** It's necessary, not sufficient, the engineering half of DevSecOps is still genuinely half the job.

## 60 TO 90 DAY STARTING PLAN

### First 30 days

*Foundations and Docker*

**WEEKS 1 TO 2**

Linux, Git, and scripting fundamentals, treated as seriously as any security certification.

**WEEKS 3 TO 4**

Docker fundamentals, building and running your first container from scratch.

### Days 31 to 90

*Pipelines, then security integrated in*

**DAYS 31 TO 50**

Terraform and a basic CI/CD pipeline, the system you'll spend the rest of this path securing.

**DAYS 51 TO 70**

Security scanning added into that pipeline, plus an IaC scan and secrets cleanup on your Terraform project.

**DAYS 71 TO 90**

A mini DevSecOps pipeline, combining what you've built into one deliberately gated flow.

PATH 4

# Cloud / Operations → DevOps or Cloud Security

*This background genuinely opens two reasonable directions. Worth deciding deliberately rather than defaulting to whichever one you heard about first.*

| MOVE TOWARD DEVOPS IF | MOVE TOWARD CLOUD SECURITY IF |
|---|---|
| You enjoy building and automating things, and the idea of owning a delivery pipeline sounds satisfying rather than tedious. | You're drawn to risk, governance, and understanding how systems get compromised, more than how they get built. |
| You'd rather write a script to solve a repeated problem than document a manual process to hand off. | Audit, compliance, and incident response work sound genuinely interesting rather than like a box to check. |
| You want deployment speed and reliability to be your core measure of success. | You want exposure reduction and control effectiveness to be your core measure of success. |

### HOW TO DECIDE, IF IT'S STILL UNCLEAR

Think back to the last time you fixed something in production. If your instinct afterward was "how do we automate this so it never happens again," that's a DevOps instinct. If your instinct was "how did this configuration end up exposed in the first place," that's a cloud security instinct. Neither answer is more valid, they're just different starting points.

### WHAT TO LEARN FIRST IN EACH DIRECTION

| DIRECTION | START WITH | WHY |
|---|---|---|
| **DevOps** | Docker, then CI/CD with GitHub Actions | The fastest way to start owning delivery rather than just operating around it |
| **Cloud Security** | IAM and cloud security posture basics, then DevSecOps pipeline integration | Builds directly on your existing cloud and operations knowledge without a large detour |

> **YOU DON'T HAVE TO CHOOSE FOREVER**
>
> Six months into either direction, most of the other path's fundamentals are still accessible to you. This decision shapes your next quarter, not your entire career.

ALL FOUR PATHS, SIDE BY SIDE

# Comparison table

*The full picture in one view, useful for a quick reference once you've read your own path in detail.*

| CURRENT ROLE | STRONGEST TRANSFERABLE SKILLS | BIGGEST KNOWLEDGE GAPS | BEST FIRST TOPIC | RECOMMENDED NEXT 3 STEPS | BEST FIRST PROJECT |
|---|---|---|---|---|---|
| **Developer** | Code fluency, Git basics, debugging, logical problem solving | Linux operations depth, infrastructure as code, orchestration | Docker | Docker, then GitHub Actions, then Terraform | Dockerize one of your own applications |
| **Infra / Support** | Production instincts, networking, incident response, server operations | Scripting, Git habit, infrastructure as code mindset | Git and scripting | Scripting, then Docker, then Terraform | A simple Terraform deployment |
| **Security** | Risk thinking, IAM instincts, compliance and audit awareness | Hands on engineering, Docker, pipeline construction | Linux, Git, and scripting | Docker, then a CI/CD pipeline, then security scanning inside it | Add security scanning to an existing CI/CD pipeline |
| **Cloud / Operations** | Cloud platform fluency, operational judgment, console and service knowledge | Depends on direction, automation depth for DevOps, security depth for Cloud Security | Docker, or IAM basics depending on direction | Decide direction first, then follow that path's sequence | A basic CI workflow, or an IAM and security posture review |

CHOOSING

# Which path is right for me

*A short self assessment. Answer honestly based on your current day to day work, not your aspirations.*

| ASK YOURSELF | IF THAT'S MOSTLY TRUE | YOUR PATH |
|---|---|---|
| Do you spend most of your day writing or reviewing application code | You already think in logic and structure, the gap is operational | Path 1, Developer to DevOps |
| Do you spend most of your day managing servers, responding to incidents, or handling support tickets | You already understand production, the gap is automation and code | Path 2, Infra / Support to DevOps |
| Do you spend most of your day assessing risk, reviewing access, or handling compliance and audits | You already understand what can go wrong, the gap is building the systems themselves | Path 3, Security to DevSecOps |
| Do you spend most of your day working inside a cloud platform, provisioning or monitoring services | You have a genuine choice ahead of you, worth deciding deliberately | Path 4, Cloud / Operations, choose a direction |

**IF MORE THAN ONE FELT TRUE**

*Pick the one that describes the majority of your actual working week, not the one that sounds most impressive. The honest answer gets you a better starting point than the aspirational one.*

WHERE TO GO FROM HERE

# Four paths is a good start. One conversation makes it specific.

These paths cover the common starting points well, but they can't account for your specific years of experience, the tools your current job already has you using, or a background that sits between two of these categories. That's worth ten minutes of context, not a guess.

### Message us with three things

Your current role, your years of experience, and whether you're aiming for DevOps or DevSecOps. That's enough for us to suggest the right next learning sequence for you specifically, not just the closest matching path.

### Not sure which path fits

If your background genuinely straddles two of these paths, tell us both halves. Most hybrid backgrounds have a clear better starting point once we know the details.

### Join the SWQ IT Academy community

Connect with other professionals following the same path as you, compare progress, and stay accountable to the sequence rather than drifting back to scattered learning.


SWQ IT ACADEMY
Real Skills. Real Outcomes.

A part of the ENGITRIX group

Email: itacademy@swqgroup.com
WhatsApp and calls: +91 8123999170
Web: itacademy.swqgroup.com


*We build industry ready professionals. The job is yours to earn, we make sure you are fully equipped to earn it.*