SWQ IT ACADEMY ACTION PLAN

30-60-90 Day DevOps / DevSecOps Action Plan for Working Professionals

A practical execution plan, not another roadmap image. Built for people who already know what to learn and need a plan for actually doing it.

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. This plan turns that standard into ninety days of specific, weekly action.

> Built for working IT professionals with some existing background, developers, admins, support engineers, QA engineers, cloud engineers, and security professionals. This is an execution plan, meant to sit next to a roadmap, not replace the judgment of learning fundamentals first.

A part of the ENGITRIX group

INTRODUCTION

A roadmap tells you what. It doesn't tell you what to do on a Tuesday.

Most people trying to move into DevOps or DevSecOps already have a roadmap saved somewhere, a screenshot, a PDF, a mental list of seven or eight things to learn in order. What they don't have is a plan for what that actually looks like on a random Tuesday evening after a nine hour workday.

That gap is where most transitions quietly stall. A roadmap tells you the destinations. It doesn't tell you how many hours to spend this week, what to build before you move on, or how to know if you're actually on track. Without that layer, even a good roadmap turns into an open ended browsing session.

Ninety days is long enough to build real depth in several connected areas and short enough to stay a genuine commitment rather than a vague someday goal. It's also roughly the natural rhythm of a job search, long enough to prepare, short enough to keep momentum. That's why this plan is built in three thirty day blocks instead of one long, undifferentiated stretch.

WHY THIS MATTERS

A plan you follow for ninety days beats a roadmap you admire for a year. This document is built to be followed, week by week, not just read once and saved.

Every thirty day block in this plan tells you the goal, the weekly focus, the specific tasks, and the deliverable that proves you did the work. It also tells you, deliberately, what to ignore for now. That last part matters as much as the rest.

BEFORE YOU START

Before you start

A few things worth getting right before day one, so the first thirty days go toward building, not toward setup and confusion.

BASELINE KNOWLEDGE THAT HELPS

You don't need to be an expert in anything to start, but comfort with the command line, a basic sense of how the internet and networking work, and some exposure to writing or reading code will make the first thirty days considerably smoother. If you're missing one of these, this plan still works, it just means Days 1 to 30 will feel a little denser.

ENVIRONMENT TO PREPARE

A laptop that can run Docker and a local Kubernetes cluster comfortably, at least sixteen gigabytes of memory is a reasonable target. A free tier account on one cloud provider, AWS, Azure, or GCP, any one is fine, this plan does not require multiple. A GitHub account you're willing to use publicly, since your commit history becomes part of your evidence.

MINDSET TO BRING

Treat this as skill building with evidence attached, not content consumption. If a week goes by and you have nothing built or changed in a repository, that week didn't count, regardless of how many videos you watched.

REALISTIC WEEKLY HOURS

> **AIM FOR EIGHT TO TEN HOURS A WEEK**
>
> That's roughly ninety minutes on four weekday evenings plus a few hours across the weekend. More hours will move you faster. Fewer hours means stretching this plan past ninety days rather than cutting the deliverables, the deliverables are what make the ninety days worth something.

PHASE ONE

Days 1 to 30

Main goal: get fluent in Linux, Git, and Docker, the three things almost everything later in this plan depends on.

WEEK	FOCUS	SPECIFIC TOPICS	PRACTICAL TASK
Week 1	Linux comfort	File systems, permissions, processes, package managers, basic networking commands	Set up a Linux environment (VM or WSL) and do every task that week from the terminal, not a GUI
Week 2	Git as a habit	Branching, merging, pull requests, resolving conflicts, writing real commit messages	Create a public GitHub repository and use branches and pull requests for every change, even solo
Week 3	Scripting	Bash or Python fundamentals, reading input, writing output, basic error handling	Automate one task you currently do by hand, at work or at home, using a script
Week 4	Docker fundamentals	Images versus containers, writing a Dockerfile, Docker Compose, container networking	Containerise an application you already have, with a database if possible, using Docker Compose

MINI DELIVERABLES BY DAY 30

- ✓ **A public GitHub repository** with a working automation script and a clear README explaining what it does.
- ✓ **A demonstrated Git workflow**, branches and pull requests used deliberately, not just a single commit history.
- ✓ **A containerised application** running through Docker Compose, with at least two connected services.

> **WHAT NOT TO WORRY ABOUT YET**
>
> Kubernetes, Terraform, cloud consoles, CI/CD pipelines, and security tooling. None of it yet. Every one of those becomes noticeably easier once this month's foundation is genuinely solid, and noticeably harder if you rush ahead without it.

PHASE TWO

Days 31 to 60

Main goal: move from packaging an application to defining and automating the infrastructure and pipeline around it.

WEEK	FOCUS	SPECIFIC TOPICS	PRACTICAL TASK
Week 5	Cloud basics	Core services on your chosen provider, IAM fundamentals, the console versus the command line interface	Manually provision one small resource, then delete it, purely to understand the console you'll soon automate away from
Week 6	Terraform	Providers, resources, variables, and what Terraform state actually does	Rebuild last week's resource using Terraform instead, and destroy it cleanly with code
Week 7	Terraform in practice	Modules, outputs, and provisioning more than one connected resource together	Provision a small real environment, for example a virtual machine plus basic networking, entirely through Terraform
Week 8	CI/CD basics	GitHub Actions workflow syntax, triggers, build and test stages, secrets handling	Build a pipeline that automatically tests and builds the container from Phase One on every push

HANDS ON DELIVERABLES BY DAY 60

- ✓ **A working Terraform project** that provisions and can cleanly destroy a real environment.
- ✓ **An early CI/CD pipeline** in GitHub Actions that builds and tests your containerised application automatically.

HOW TO KNOW YOU'RE ON TRACK

You should be able to explain, out loud and without notes, what Terraform state is and why editing infrastructure manually after provisioning it with code causes problems. You should also be able to look at a failed GitHub Actions run and diagnose it from the logs alone, without guessing.

PHASE THREE

Days 61 to 90

Main goal: bring everything together into one working deployment, and add the security layer your goal path actually needs.

WEEK	FOCUS	SPECIFIC TOPICS	PRACTICAL TASK
Week 9	Pipeline completion	Deploying a built image automatically, not just testing and building it	Extend last month's pipeline so a successful build also deploys somewhere real, even a simple server
Week 10	Kubernetes basics	Pods, deployments, services, ConfigMaps, and a local cluster tool like Minikube or Kind	Deploy your containerised application to a local Kubernetes cluster for the first time
Week 11	Kubernetes in practice	Secrets management in a cluster, troubleshooting with kubectl, basic Helm structure	Break your own deployment on purpose, then fix it using only logs and describe output
Week 12	Security layer	IAM review, secrets handling, and either cloud security basics or pipeline scanning, based on your goal path	Audit your own infrastructure and pipeline, and fix at least three real issues you find

BRINGING IT TOGETHER

By the final week, you're not learning four separate things, you're running one workflow: a code change triggers a pipeline, the pipeline tests and builds an image, the image deploys to Kubernetes, and a security check sits somewhere in that chain. That combined workflow, not any single tool, is what a working interview demonstration actually looks like.

READINESS INDICATORS FOR MOVING BEYOND NINETY DAYS

You can walk someone through your entire pipeline, commit to deployment, without notes. You can explain a failure in any part of the chain using logs rather than guesswork. And you have at least one project you'd be comfortable sharing as a portfolio link in a job application, not a private repository you're embarrassed to show.

THE PROOF

Ninety day deliverables checklist

What you should ideally have by the end, in one list. If several of these are missing, that's not a failure, it's a sign of exactly where to spend the next few weeks.

- ✓ **An active GitHub repository** with a real commit history, branches, and pull requests, not a single upload commit.
- ✓ **A containerised application** with a working Dockerfile and a multi service Docker Compose setup.
- ✓ **A Terraform project** that provisions a real environment and can be destroyed cleanly with the same code.
- ✓ **A working CI/CD workflow** in GitHub Actions that tests, builds, and deploys automatically on a push.
- ✓ **A basic Kubernetes deployment** of your application on a local cluster, with services and configuration handled properly.
- ✓ **Some form of security integration**, either a cloud security review of your own infrastructure or a scanning step inside your pipeline, depending on your goal path.

> **A NOTE ON THIS LIST**
>
> These six items are not six separate side projects. Ideally they are all part of the same application, the same pipeline, and the same repository, so you can talk about them as one coherent piece of work instead of six disconnected exercises.

ADJUST FOR YOUR SITUATION

Different versions by career goal

The default plan above works as written for most people. Here's how to bend it depending on where you're starting from and where you're headed.

Targeting DevOps

Follow the default plan largely as written

ADJUSTMENT

Spend extra time in Phase Three on Kubernetes rather than security. Let the Week 12 security work stay at baseline, IAM and secrets hygiene, without going deeper into pipeline scanning.

EXTRA EMPHASIS

CI/CD and Kubernetes, the two areas tested most heavily in DevOps interviews.

Targeting DevSecOps

Pull security forward, push Kubernetes back slightly

ADJUSTMENT

In Phase Three, prioritise Week 12's security work over deep Kubernetes exploration. Add a scanning step to your pipeline as the core Phase Three deliverable, and treat Kubernetes as something to deepen after day ninety.

EXTRA EMPHASIS

IAM, secrets management, and pipeline level scanning, the actual substance of a DevSecOps role.

Coming from a development background

Compress Phase One, lean into automation thinking

ADJUSTMENT

Weeks 1 to 3 will likely move faster since Git and scripting are probably already familiar. Use the extra time to go deeper into Docker in Week 4, and don't skip Terraform even if infrastructure feels unfamiliar, it's the biggest genuine gap for developers moving into this field.

WATCH FOR

The instinct to treat infrastructure as someone else's job. That instinct is exactly what this plan is meant to change.

Coming from infrastructure or support

Compress the Linux week, lean into pipelines

ADJUSTMENT

Week 1 will likely move quickly. Use the extra time in Phase Two on CI/CD, since pipeline automation, not infrastructure, is usually the real gap for this background. Git discipline in Week 2 deserves real attention if your current work hasn't required much branching or pull request habit.

WATCH FOR

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.

Coming from security

Reorder toward DevSecOps, keep the technical base

ADJUSTMENT

Follow the DevOps foundation in Phases One and Two closely, security professionals often underestimate how much hands on Docker, Terraform, and CI/CD experience the DevSecOps title actually assumes. Don't skip to Week 12 early, the pipeline needs to exist before you can meaningfully secure it.

EXTRA EMPHASIS

Weeks 4, 7, and 8, the technical foundation that most differentiates a working DevSecOps engineer from a security reviewer.

WHAT DERAILS THIS

Common failure points

Five ways this plan quietly falls apart, all avoidable, all common.

- × **Trying to learn too many tools at once.** Every week in this plan names one tool on purpose. Alternatives can wait until a job or interview specifically asks for them.
- × **Not building anything.** A week without a commit, a deployment, or a change to something real is a week that didn't move the plan forward, regardless of how much was read or watched.
- × **Never revising.** Without a short weekly review, week three's lessons quietly fade by week seven. The weekly rhythm template later in this document includes a slot for exactly this.
- × **Spending most hours consuming content.** Videos and articles are useful for the first exposure to a concept. They are not evidence of a skill, and interviewers can generally tell the difference quickly.
- × **Using a tool without understanding why it exists.** If you can operate Terraform but can't explain what problem it solves compared to clicking through a console, that gap will surface in an interview at the worst possible moment.

MAKING IT REPEATABLE

Weekly learning rhythm template

The same structure, week after week, for all twelve weeks of this plan. Predictability is what makes a plan survivable across a full time job.

SLOT	PURPOSE	TYPICAL LENGTH
Monday and Tuesday evenings	Theory block, reading documentation and working through that week's specific topics, not passive video watching	45 to 60 minutes each
Wednesday and Thursday evenings	Practice block, hands on time inside that week's tool, working toward the week's practical task	45 to 60 minutes each
Saturday or Sunday	Project time, finishing the week's practical task and connecting it to the broader deliverable for that phase	2 to 3 hours
One weekly slot, any day	Review, writing a few lines about what you learned and what's still unclear, and updating your GitHub README where relevant	20 to 30 minutes

WHY THE REVIEW SLOT MATTERS MORE THAN IT LOOKS

Twenty minutes of honest review each week is what turns twelve separate weeks into one connected quarter of progress, instead of twelve weeks you'd struggle to summarise afterward.

WHERE TO GO FROM HERE

A generic ninety days is a starting point. Yours doesn't have to be generic.

This plan works as written for most people. It works better once it accounts for your actual background, your actual hours, and the actual role you're aiming for. That adjustment is quick to make with the right context, and usually not worth guessing at alone.

Get a role specific version of this plan

Tell us your current background and target role. We'll tell you honestly which weeks to compress, which to slow down on, and what this plan should look like adjusted for you specifically.

Ask what your next best step is

If you've already started somewhere, don't restart from Week 1 by default. Tell us where you actually are, and we'll tell you where this plan says you should pick up.

Join the SWQ IT Academy learning community

Work through these same twelve weeks alongside other working professionals doing the same thing, comparing progress and staying accountable instead of doing it alone.

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.