SWQ IT ACADEMY PROJECT PACK

Practical DevOps / DevSecOps Project Pack for Working Professionals

Eleven projects, ordered by difficulty, built to turn scattered learning into a portfolio you can actually talk about in an interview.

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 project pack is that standard turned into eleven specific things to build.

Built for working IT professionals with some existing background, developers, admins, support engineers, QA engineers, cloud engineers, and security professionals who already know roughly what to learn and want a concrete list of what to build next.

A part of the ENGITRIX group

INTRODUCTION

Projects are what turn knowing into being able to.

Ask most people who've watched a hundred hours of DevOps content to walk you through deploying an application and you'll notice the hesitation. They can describe Docker. They can define CI/CD. What they usually can't do is sit down and build the thing, start to finish, without a tutorial open in the next tab.

That gap is normal, and it's also entirely fixable. Watching content builds recognition, being able to name a concept when you see it again. Building projects builds something different, the ability to produce the thing from nothing, which is the actual skill interviews test for and jobs require.

Projects also do something scattered learning can't: they force the pieces to connect. Reading about Docker and reading about CI/CD separately leaves you with two disconnected mental folders. Building a pipeline that containerises an application and deploys it automatically forces you to understand where one tool's output becomes the next tool's input, which is exactly how these tools actually get used on a job.

WHY SMALL AND COMPLETE BEATS BIG AND HALF FINISHED

A small project you finished, documented, and can explain is worth more in an interview than an ambitious project you're still building six months later. Completion is the skill. Scope it small enough to actually finish.

This pack gives you eleven of those small, complete projects, arranged so each one builds on the last. By the final project, you're not assembling five unrelated exercises, you're extending one coherent body of work.

BEFORE YOU BUILD

How to use this project pack

Four working principles. Read once, then return here whenever you're deciding what to build next.

01 Start at Level 1, even if it feels beneath you

If Level 1 genuinely takes you twenty minutes, that's useful information too, it confirms you're ready to move faster than most. Skipping it to guess is a worse bet than confirming it quickly.

02 Don't try to build everything at once

One project, finished and documented, before the next one starts. Three half built projects in progress at the same time is a worse portfolio than one finished project.

03 Focus on the workflow, not the checkbox

Getting a pipeline to go green matters less than understanding why each stage exists. If you can't explain a step you copied, slow down and figure out why it's there before moving on.

04 Document as you go, not after

A README written while you remember your own decisions is worth far more than one written from memory a month later. Write it alongside the project, not as cleanup afterward.

THE LADDER

The project ladder

Eleven projects across three levels. Each one names who it's for, what roadmap stage it supports, and exactly how to present it once it's done.

LEVEL	PROJECTS	WHAT THEY PROVE
Level 1, Foundation	Dockerize an app, a basic CI workflow, a simple Terraform deployment	You can operate the core tools individually, on your own, without a tutorial open
Level 2, Intermediate	A full CI/CD pipeline, reusable Terraform modules, a combined Docker and CI/CD flow	You can connect tools together into a working process, not just operate them in isolation
Level 3, Advanced Transition	Kubernetes deployment, security scanning, a mini DevSecOps pipeline, IaC and secrets handling, an end to end workflow	You can build and secure a production shaped system, the actual bar for a DevOps or DevSecOps role

LEVEL 1

Foundation projects

Three short projects, each one focused on getting genuinely comfortable with a single tool before anything gets connected together.

LEVEL 1 · PROJECT 1

Dockerize a Simple Application

LEVEL 1

BEST FOR

Anyone starting the pack, especially developers and infra professionals new to containers.

ROADMAP STAGE

Supports Stage 2, Docker and Containers.

WHAT YOU'LL BUILD

A single application, yours or a small sample one, packaged into a Docker image that runs consistently from that image alone.

LEARNING OBJECTIVES

The difference between an image and a container, writing a Dockerfile from scratch, and managing ports and environment variables inside a container.

TOOLS

Docker, and a container registry such as Docker Hub.

PREREQUISITES

Comfortable with the command line and basic familiarity with the application you're containerising.

STEP BY STEP OUTLINE

1. Pick a small application you already have, or a simple sample app in a language you know.
2. Write a Dockerfile that installs dependencies and runs the app.
3. Build the image locally and run it, confirming it behaves the same as running it directly.
4. Reduce the image size with a slimmer base image and cleaner build layers.
5. Push the image to a registry and pull it down elsewhere to confirm it truly runs anywhere.

DELIVERABLE

A public repository with the Dockerfile, the application, and a README explaining how to build and run it.

> **HOW TO PRESENT THIS**
>
> Pin the repository with a one line description like "Containerised this app, reduced image size by X percent." A concrete number stands out more than "learned Docker."

LEVEL 1 · PROJECT 2

A Basic CI Workflow in GitHub Actions

LEVEL 1

BEST FOR

Developers and QA professionals comfortable with Git, new to automation.

ROADMAP STAGE

Supports Stage 4, CI/CD with GitHub Actions, introductory level.

WHAT YOU'LL BUILD

A GitHub Actions workflow that automatically runs your test suite every time code is pushed.

LEARNING OBJECTIVES

Workflow syntax, triggers, jobs and steps, and reading pipeline logs to diagnose a failure.

TOOLS

GitHub Actions, and a testing framework appropriate to your project's language.

PREREQUISITES

A repository with at least a few basic automated tests already written.

STEP BY STEP OUTLINE

1. Add a workflow file that triggers on every push to the main branch.
2. Configure it to install dependencies and run your test suite.
3. Intentionally break a test and confirm the workflow fails and reports why clearly.
4. Add a status badge to your README showing the current build state.
5. Fix the broken test and confirm the workflow goes green again.

DELIVERABLE

A repository with a passing GitHub Actions workflow and a visible status badge in the README.

> **HOW TO PRESENT THIS**
>
> The badge itself is the pitch. A green passing badge on a public repository is instant, visible proof, no one needs to read a word to believe it.

LEVEL 1 · PROJECT 3

A Simple Terraform Deployment

LEVEL 1

BEST FOR

Infra and support professionals moving toward infrastructure as code, and developers wanting cloud exposure.

ROADMAP STAGE

Supports Stage 3, Terraform and Infrastructure as Code, introductory level.

WHAT YOU'LL BUILD

A small, real piece of cloud infrastructure, provisioned and destroyed entirely through Terraform code.

LEARNING OBJECTIVES

Terraform syntax and providers, what state actually does, and safely applying and destroying infrastructure.

TOOLS

Terraform, and a free tier account on one cloud provider, AWS, Azure, or GCP.

PREREQUISITES

A free tier cloud account and basic familiarity with that provider's console.

STEP BY STEP OUTLINE

1. Write a Terraform configuration that provisions one resource, for example a small virtual machine or a storage bucket.
2. Run a plan and read through it carefully before applying, understanding exactly what will be created.
3. Apply the configuration and confirm the resource exists in the cloud console.
4. Change one variable and re apply, observing how Terraform calculates the difference.
5. Destroy the infrastructure entirely through Terraform and confirm nothing is left behind.

DELIVERABLE

A repository with working Terraform files and a README documenting what gets created and how to tear it down.

> **HOW TO PRESENT THIS**
>
> State plainly that you provisioned and destroyed real cloud infrastructure with code. That specific claim carries more weight than a console screenshot ever will.

LEVEL 2

Intermediate projects

Three projects that connect the tools from Level 1 into working processes, the actual shape of the job, not isolated exercises.

LEVEL 2 · PROJECT 4

A Full CI/CD Pipeline for a Sample Application

LEVEL 2

BEST FOR

Anyone who has completed the Level 1 Docker and CI projects.

ROADMAP STAGE

Supports Stage 4, CI/CD with GitHub Actions, full depth.

WHAT YOU'LL BUILD

A pipeline that tests, builds a container image, and deploys it automatically on every push.

LEARNING OBJECTIVES

Multi stage pipelines, passing artifacts between jobs, and securely handling registry credentials as secrets.

TOOLS

GitHub Actions, Docker, a container registry, and a simple deployment target such as a small server or hosting service.

PREREQUISITES

Completed Level 1's Dockerize and basic CI projects, or equivalent comfort with both.

STEP BY STEP OUTLINE

1. Extend your Level 1 CI workflow to also build a Docker image after tests pass.
2. Push the built image to a container registry using securely stored credentials.
3. Add a deployment step that pulls the new image and runs it on your chosen target.
4. Trigger the full pipeline end to end and confirm a code change reaches a running deployment automatically.
5. Introduce a failing test on purpose and confirm the pipeline stops before deployment, not after.

DELIVERABLE

A repository with one pipeline covering test, build, and deploy, and a short written explanation of each stage.

> **HOW TO PRESENT THIS**
>
> Record a thirty second screen capture of a commit triggering the full pipeline. A short clip in a LinkedIn post earns more attention than a static screenshot.

LEVEL 2 · PROJECT 5

Provision Infrastructure with Reusable Terraform Modules

LEVEL 2

BEST FOR

Infra and support professionals ready to move past single resource Terraform files.

ROADMAP STAGE

Supports Stage 3, Terraform and Infrastructure as Code, module design.

WHAT YOU'LL BUILD

A Terraform module you can reuse to provision the same type of environment more than once, with different settings each time.

LEARNING OBJECTIVES

Module structure, variables and outputs, and calling the same module for more than one environment.

TOOLS

Terraform, and one cloud provider.

PREREQUISITES

Completed the Level 1 Terraform project, or equivalent comfort with basic Terraform.

STEP BY STEP OUTLINE

1. Take your Level 1 Terraform configuration and extract it into a reusable module with input variables.
2. Call that module twice with different variable values, for example a small and a slightly larger environment.
3. Add sensible outputs so calling code can reference what the module created.
4. Provision both environments together and confirm they're genuinely independent of each other.
5. Destroy both cleanly and document the module's inputs and outputs in a README.

DELIVERABLE

A Terraform module with clear input and output documentation, called from at least two separate configurations.

> **HOW TO PRESENT THIS**
>
> Frame it plainly as "built a reusable module." That phrase signals you're thinking about infrastructure the way a team actually works, not writing one off scripts.

LEVEL 2 · PROJECT 6

Deploy an App with a Combined Docker and CI/CD Flow

LEVEL 2

BEST FOR

Developers who want to see one of their own applications move through a full delivery flow, not just a sample app.

ROADMAP STAGE

Bridges Stage 2, Docker, and Stage 4, CI/CD, into one connected flow.

WHAT YOU'LL BUILD

One of your own applications, containerised and deployed through a pipeline you built yourself, treated as real software rather than a demo.

LEARNING OBJECTIVES

Applying containerisation and pipeline design to a project you actually care about, and handling environment specific configuration cleanly.

TOOLS

Docker, GitHub Actions, and a deployment target of your choice.

PREREQUISITES

Completed Level 1's Docker and CI projects, and Level 2 Project 4.

STEP BY STEP OUTLINE

1. Pick an application of your own, with a little more complexity than a sample app.
2. Containerise it properly, including clean handling of configuration and environment variables.
3. Build a pipeline that tests, builds, and deploys it, reusing what you learned in Project 4.
4. Add basic health checks so you can confirm the deployment actually works, not just that it ran.
5. Write a short architecture note explaining how the pieces fit together.

DELIVERABLE

A live or easily runnable deployment of your own application, with a documented pipeline and an architecture note.

> **HOW TO PRESENT THIS**
>
> This is your first project that isn't a sample app. Present it as such: "took my own project from local code to an automated deployment."

LEVEL 3

Advanced transition projects

Five projects at the actual bar for a DevOps or DevSecOps role, orchestration, security, and one capstone that ties everything together.

LEVEL 3 · PROJECT 7

Deploy a Containerized App to Kubernetes

LEVEL 3

BEST FOR

Anyone ready to move past single container deployments into orchestration.

ROADMAP STAGE

Supports Stage 5, Kubernetes.

WHAT YOU'LL BUILD

The application from your CI/CD project, running on a local Kubernetes cluster with proper configuration management.

LEARNING OBJECTIVES

Pods, deployments, and services, managing configuration through ConfigMaps and secrets, and troubleshooting with kubectl.

TOOLS

Kubernetes, a local cluster tool such as Minikube or Kind, kubectl, and optionally Helm.

PREREQUISITES

Completed the Docker and CI/CD projects from Levels 1 and 2.

STEP BY STEP OUTLINE

1. Set up a local Kubernetes cluster and confirm kubectl can reach it.
2. Write a deployment manifest for your containerised application.
3. Expose it through a service and confirm you can reach it from outside the cluster.
4. Move any configuration or secrets out of the container image and into ConfigMaps and secrets properly.
5. Deliberately break the deployment, for example a bad image tag, and practice diagnosing it with kubectl describe and logs.

DELIVERABLE

A set of Kubernetes manifests, or a basic Helm chart, that deploys your application with configuration handled correctly.

> **HOW TO PRESENT THIS**
>
> Include a short "how I debugged this" note about the deliberate failure you fixed. Troubleshooting stories are what interviewers actually want to hear.

LEVEL 3 · PROJECT 8

Add Security Scanning to CI/CD

LEVEL 3

BEST FOR

Anyone on the DevSecOps path, and DevOps focused professionals who want baseline security literacy.

ROADMAP STAGE

Supports Stage 7, DevSecOps, introductory level.

WHAT YOU'LL BUILD

An upgraded version of your CI/CD pipeline that scans for vulnerabilities before deployment is allowed to proceed.

LEARNING OBJECTIVES

Container image scanning, dependency scanning, and configuring a pipeline to fail on real risk rather than every minor warning.

TOOLS

A container image scanner such as Trivy, and a dependency scanning action for your language ecosystem.

PREREQUISITES

A working CI/CD pipeline from Level 2.

STEP BY STEP OUTLINE

1. Add an image scanning step to your pipeline that runs after the build stage.
2. Configure it to fail the build only on high or critical severity findings, not every low severity note.
3. Intentionally use a dependency with a known vulnerability to confirm the scan actually catches something real.
4. Fix the vulnerability and confirm the pipeline passes cleanly afterward.
5. Document what the scanner checks and what your pipeline does when it finds a problem.

DELIVERABLE

A pipeline with a correctly tuned security scanning stage and a documented example of it catching a real issue.

> **HOW TO PRESENT THIS**
>
> The story of catching and fixing a real vulnerability is more compelling than the scanning step itself. Lead with that story, not the tool name.

LEVEL 3 · PROJECT 9

Build a Mini DevSecOps Pipeline

LEVEL 3

BEST FOR

Professionals actively targeting a DevSecOps role, as a synthesis project.

ROADMAP STAGE

Supports Stage 7, DevSecOps, full depth.

WHAT YOU'LL BUILD

A pipeline where security is a genuine gate, not an afterthought bolted onto the end, combining several checks into one coherent flow.

LEARNING OBJECTIVES

Sequencing multiple security checks sensibly, deciding what blocks a deployment versus what only warns, and treating a pipeline as a security control.

TOOLS

Your existing CI/CD pipeline, an image scanner, a dependency scanner, and a static analysis tool appropriate to your language.

PREREQUISITES

Completed Project 8, security scanning in CI/CD.

STEP BY STEP OUTLINE

1. Map out the checks you want: dependency scanning, image scanning, and basic static analysis.
2. Decide deliberately which checks block deployment and which only report, and write down why.
3. Implement the checks in a logical order, cheapest and fastest checks first, so failures are caught early.
4. Run the full pipeline against a version of your app with at least two different classes of issue seeded in on purpose.
5. Write a short security gate document explaining what the pipeline checks and why each gate exists.

DELIVERABLE

A pipeline with multiple, deliberately sequenced security checks, and a short document explaining the reasoning behind the gates.

> **HOW TO PRESENT THIS**
>
> That reasoning document is the differentiator. Most candidates can add a scanner, few can explain why they ordered and gated it the way they did.

LEVEL 3 · PROJECT 10

Add IaC Scanning and Secrets Handling

LEVEL 3

BEST FOR

Infra and security professionals wanting to secure the infrastructure layer specifically, not just the application layer.

ROADMAP STAGE

Bridges Stage 6, Cloud Security, and Stage 7, DevSecOps.

WHAT YOU'LL BUILD

A secured version of your Terraform project, scanned for misconfiguration and free of any hardcoded secrets.

LEARNING OBJECTIVES

Static analysis for infrastructure code, proper secrets management patterns, and recognising common Terraform misconfigurations.

TOOLS

An IaC scanner such as tfsec or Checkov, and a secrets manager appropriate to your cloud provider.

PREREQUISITES

Completed the Terraform projects from Levels 1 and 2.

STEP BY STEP OUTLINE

1. Run an IaC scanner against your existing Terraform project and read through every finding.
2. Fix at least three real findings, don't just silence the warnings.
3. Find any hardcoded credentials or secrets in your configuration, there's a good chance there's at least one.
4. Move those secrets into a proper secrets manager and reference them securely from Terraform instead.
5. Add the IaC scan as an automated step in your CI/CD pipeline so future changes are checked automatically.

DELIVERABLE

A Terraform project with a documented before and after scan result, and secrets fully moved out of the codebase.

> **HOW TO PRESENT THIS**
>
> A concrete before and after finding count is a strong, specific proof point. "Reduced findings from twelve to zero" reads far better than "added security scanning."

LEVEL 3 · PROJECT 11, CAPSTONE

Create an End to End Deployment Workflow

LEVEL 3

BEST FOR

Anyone finishing this pack, as the capstone project that ties everything together.

ROADMAP STAGE

Draws on all seven stages of the core roadmap at once.

WHAT YOU'LL BUILD

One complete system, infrastructure, containerised application, pipeline, orchestration, and security, working together as a single coherent project.

LEARNING OBJECTIVES

Designing a full system rather than an isolated exercise, and being able to explain the whole chain end to end under interview conditions.

TOOLS

Everything from the previous ten projects, brought together in one repository or a small set of connected repositories.

PREREQUISITES

Completed at least one project from each earlier level.

STEP BY STEP OUTLINE

1. Sketch the full architecture on paper first, infrastructure, application, pipeline, orchestration, and security checks, before touching any code.
2. Provision the infrastructure with Terraform, including anywhere your application and cluster will run.
3. Connect your CI/CD pipeline so a commit triggers testing, building, scanning, and deployment to Kubernetes automatically.
4. Confirm the whole chain works end to end with one real commit, from code change to a running, secured deployment.
5. Write a single architecture document that explains the whole system, this becomes your primary portfolio piece.

DELIVERABLE

One working system covering infrastructure, containers, CI/CD, orchestration, and security, plus an architecture document explaining it end to end.

> **HOW TO PRESENT THIS**
>
> This is the project to lead with. Put the architecture document front and center on your GitHub profile and reference it directly in applications.

ADJUST FOR YOUR SITUATION

Suggested project paths by background

The full ladder works for everyone. Here's the order that gets you to a strong portfolio fastest, depending on where you're starting from.

Developers moving to DevOps

Your code skills carry over, infrastructure is the real gap

SUGGESTED ORDER

Project 2 (CI workflow) first, since it's closest to what you already know, then Project 1 (Dockerize), Project 4 (full CI/CD), Project 6 (your own app), then Project 7 (Kubernetes).

DON'T SKIP

Project 3 and Project 5, the Terraform projects. Developers moving into DevOps most often try to skip infrastructure as code, and it shows in interviews.

Infra and support professionals moving to DevOps

Terraform will feel natural, pipelines are the real gap

SUGGESTED ORDER

Project 3 (Terraform) first, then Project 1 (Dockerize), Project 2 (CI workflow), Project 5 (Terraform modules), then Project 4 (full CI/CD).

DON'T SKIP

Project 2 and Project 4. Pipeline automation, not infrastructure, is usually the actual gap between an infra background and a DevOps role.

Security professionals moving to DevSecOps

Build the DevOps foundation first, then layer in security

SUGGESTED ORDER

Projects 1, 2, and 3 in order, then Project 4 (full CI/CD), then move directly into Project 8, Project 9, and Project 10, the security focused Level 3 projects.

DON'T SKIP

Projects 1 through 4. Security professionals often underestimate how much hands on pipeline experience the DevSecOps title actually assumes, don't jump to Project 8 before the pipeline it protects actually exists.

MAKING IT FIT A REAL WEEK

Thirty day project execution model

A realistic way to complete two to three of these projects a month around a full time job, without it becoming another source of guilt.

WEEK	FOCUS	WHAT "DONE" LOOKS LIKE
Week 1	Pick one project and scope it before writing anything. Read the outline, confirm the prerequisites, set up the environment.	Environment ready, first two steps of the outline complete
Week 2	Build the core of the project, working through the remaining steps in the outline in order, not skipping ahead.	The project technically works, even if it's rough
Week 3	Refine, break it on purpose and fix it, and write the README while the decisions are still fresh.	The project is documented and you can explain every part of it
Week 4	Publish it properly, write the portfolio note, and start scoping the next project on the ladder.	Live on GitHub, presented, and you're ready to start the next one

> **PACE EXPECTATION**
>
> At eight to ten hours a week, one project roughly every three to four weeks is realistic and sustainable. That's eight to eleven projects across nine to twelve months, the entire ladder, at a pace you can actually keep up alongside a job.

AVOID THESE

Common project mistakes

Five ways a good project idea turns into a weak portfolio piece.

- × **Copying without understanding.** Following a tutorial line by line proves you can follow instructions, not that you understand the system. If you can't rebuild it from memory a week later, it hasn't landed yet.
- × **Building too big, too early.** An ambitious first project usually ends up half finished. Scope small enough to actually complete, complexity can come in the next project.
- × **Not documenting.** An undocumented project asks an interviewer to take your word for what it does. A clear README lets the work speak for itself before you say anything.
- × **Not connecting the project to a real use case.** A project with no clear reason to exist is harder to talk about convincingly. Even a small, invented scenario gives you something concrete to explain.
- × **Skipping reflection and cleanup.** The last twenty minutes, tidying the repository, writing what you'd do differently, is what turns a finished exercise into a genuine portfolio piece.

WHERE TO GO FROM HERE

Eleven projects is a lot of options. You don't have to pick alone.

The right starting project depends on your background, your goal, and honestly, what you've already half built somewhere. A short conversation usually saves more time than staring at this list trying to decide.

Get a role based project path

Tell us your background and target role. We'll tell you which projects to prioritise, and in what order, based on where you're actually starting from.

Ask which project to start with

If you're stuck choosing, send us what you already know and what you're aiming for. We'll give you a straight answer on where to begin.

Join the SWQ IT Academy community

Share progress on these projects with other working professionals building the same ladder, and get feedback before you publish.

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.