05

CNW-2511 CLOUD NETWORKING

Azure Web Hosting Lifecycle: CLI-Driven Static Sites & Serverless Apps Through a Red Team Lens

Full deploy-modify-teardown of Azure Storage static websites and App Service apps via CLI, mapped to HUMINT/SE adversary tradecraft and OPSEC discipline

COURSE
CNW-2511 Cloud Networking

DATE
April 2025

ENVIRONMENT
Azure for Students subscription (Subscription ID 89c4a4cc-1045-40eb-bc62-77cf81e56e1b)

PAGES (REPORT)
15

STUDENT
Cody Richard · 0005288412

ORGANIZATION
Full Sail University

TECHNICAL BRIEF · EXECUTIVE SUMMARY

Azure Web Hosting Lifecycle: CLI-Driven Static Sites & Serverless Apps Through a Red Team Lens

For Milestone 3 of CNW-2511 (the "3.5 Activity - Content Delivery" lab), I executed the full lifecycle of Azure-hosted web infrastructure — deployment, in-place content modification, and complete teardown — entirely through command-line tooling on an Azure for Students subscription spanning West US and Central US. I stood up a resource group (CNW-WEB), a StorageV2 LRS storage account (crichardweb) with static website hosting, then deployed and pivoted a static index.html (purple to yellow theme), and provisioned a serverless tier with an F1 Free App Service Plan and the cnw1crichard App Service, overwriting hostingstart.html via the Kudu API to serve a controlled lab page. No attacks were conducted; instead I framed each phase against real-world Red Team tradecraft — HUMINT-informed resource naming, social engineering plausibility, and OPSEC reflexes like blob overwrites over new-object creation to minimize storage telemetry. The engagement closed with a surgical, CLI-only resource group deletion that wiped all artifacts, simulating adversary cover-tracks behavior.

▸ Objectives

- Deploy, configure, and operationalize Azure cloud web services simulating early-stage infrastructure establishment
- Execute the full lifecycle (build, modify, teardown) of static websites and serverless apps entirely via CLI to minimize portal telemetry
- Host static content from Azure Storage and validate external exposure via the storage account's primary blob endpoint
- Stand up a serverless App Service tier and replace its default content using the Kudu file-management API
- Map each technical phase to real-world Red Team tradecraft (HUMINT, social engineering, OPSEC) without conducting any actual attacks
- Achieve complete environmental cleanup, leaving no residual cloud artifacts post-operation

▸ Environment

`Azure for Students subscription (Subscription ID 89c4a4cc-1045-40eb-bc62-77cf81e56e1b)`

`Regions: West US and Central US (resources provisioned in Central US)`

`Resource Group: CNW-WEB`

WALKTHROUGH & EVIDENCE

For Milestone 3 of CNW-2511, I drove the full lifecycle of Azure-hosted web infrastructure — deploy, modify, teardown — entirely from the Azure CLI on an Azure for Students subscription, then mapped each phase to real-world Red Team tradecraft (Initial Access through Cover Tracks). The build spans a StorageV2 static website (crichardweb) and an F1 Free Tier App Service (cnw1crichard), with in-place blob overwrites and Kudu edits chosen deliberately to suppress portal and storage telemetry. No attacks were conducted; the engagement closes with a single-command resource group deletion that wipes every artifact.

STEP 01 — INITIAL ACCESS

Storage Account & Static Website Hosting

Created resource group CNW-WEB in Central US, deployed the crichardweb storage account (StorageV2, Locally Redundant Storage), and enabled static website hosting on the $web container with index.html as the index document.

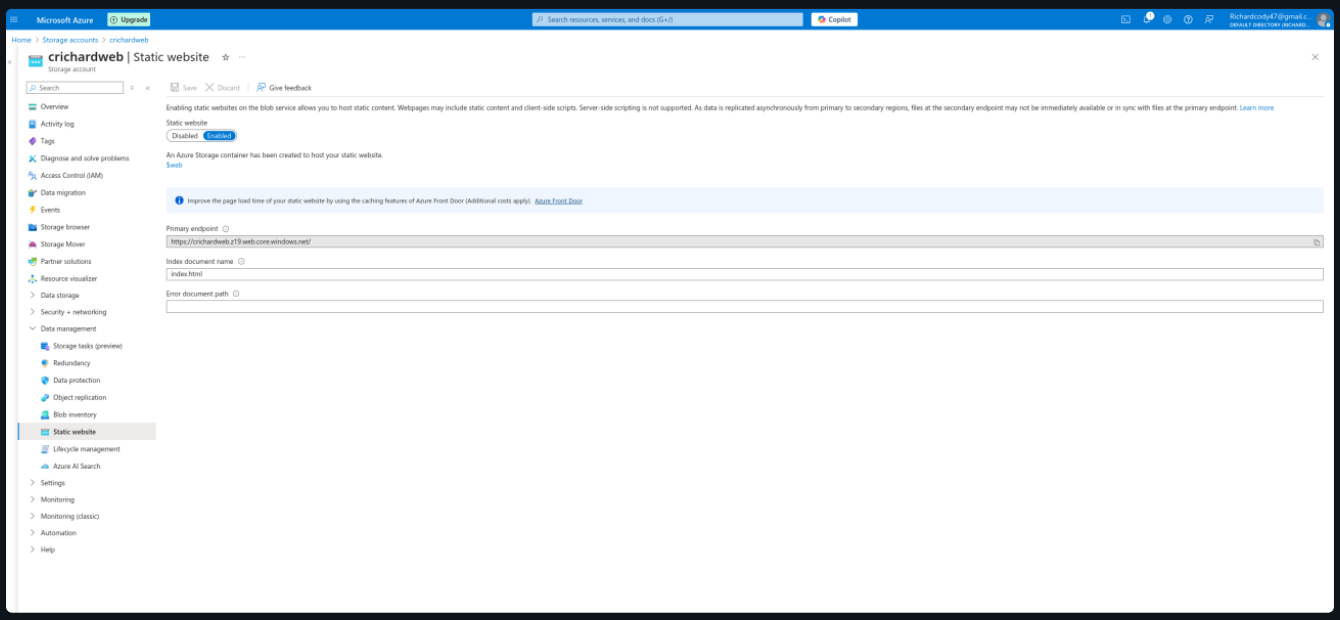

FIG 01 — crichardweb storage account — Static website blade set to Enabled, with the auto-created $web container and primary endpoint https://crichardweb.z19.web.core.windows.net/ serving index.html as the index document.

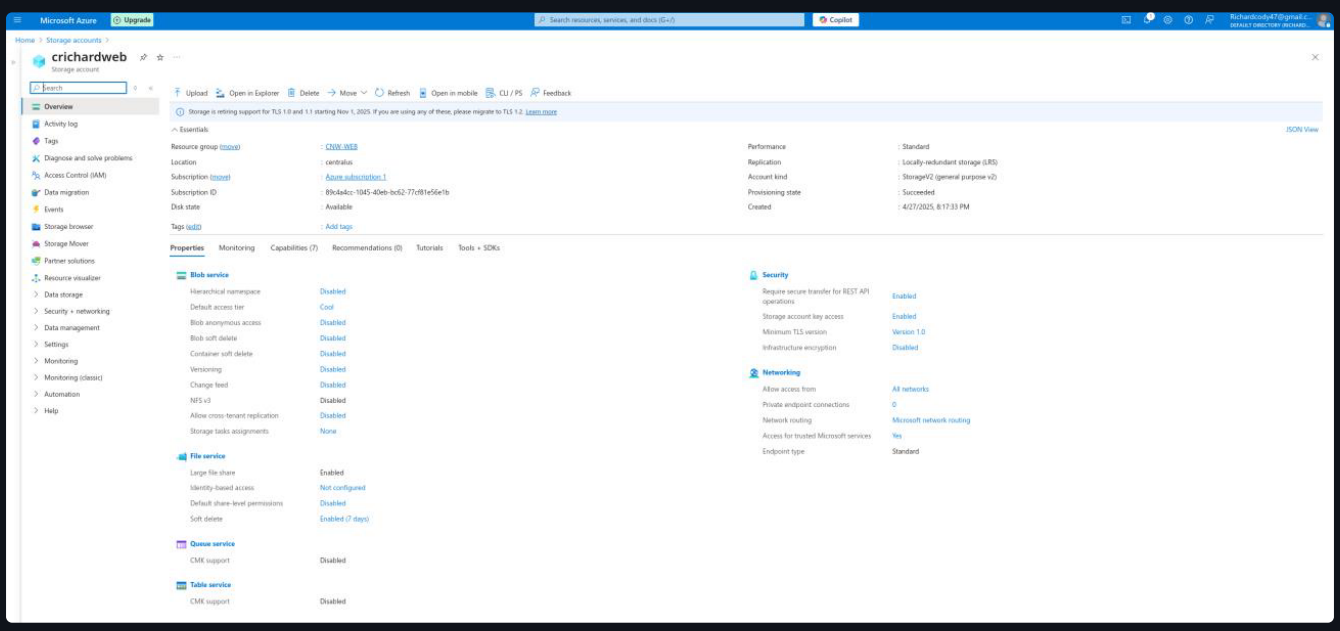

FIG 02 — crichardweb Overview — StorageV2 (general purpose v2) account, Locally Redundant Storage (LRS) replication, in resource group CNW-WEB, confirming the provisioned hosting tier.

STEP 02 — EXPOSURE

Static Content Deployment & In-Place Pivot

Uploaded the initial index.html to the $web container and validated external reachability through the storage blob endpoint, then overwrote the blob in place to pivot the theme — minimizing storage-activity anomalies versus creating a new object.

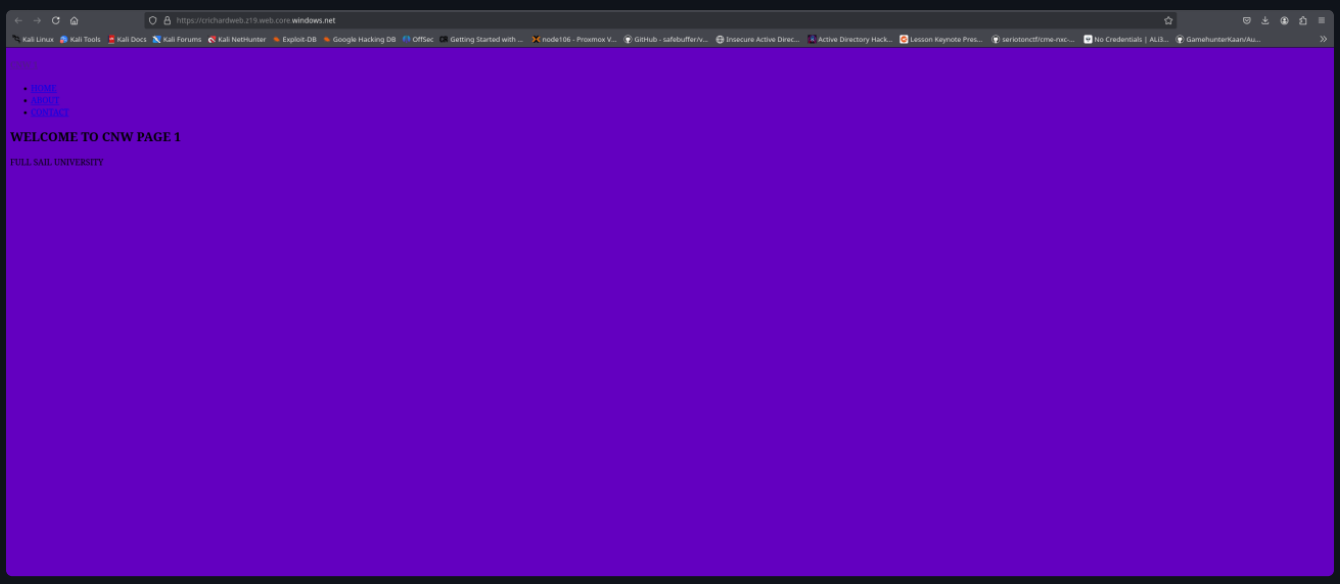

FIG 03 — Initial deployment served from https://crichardweb.z19.web.core.windows.net/ — purple-themed 'WELCOME TO CNW PAGE 1' index.html with HOME/ABOUT/CONTACT nav, confirming external static hosting.

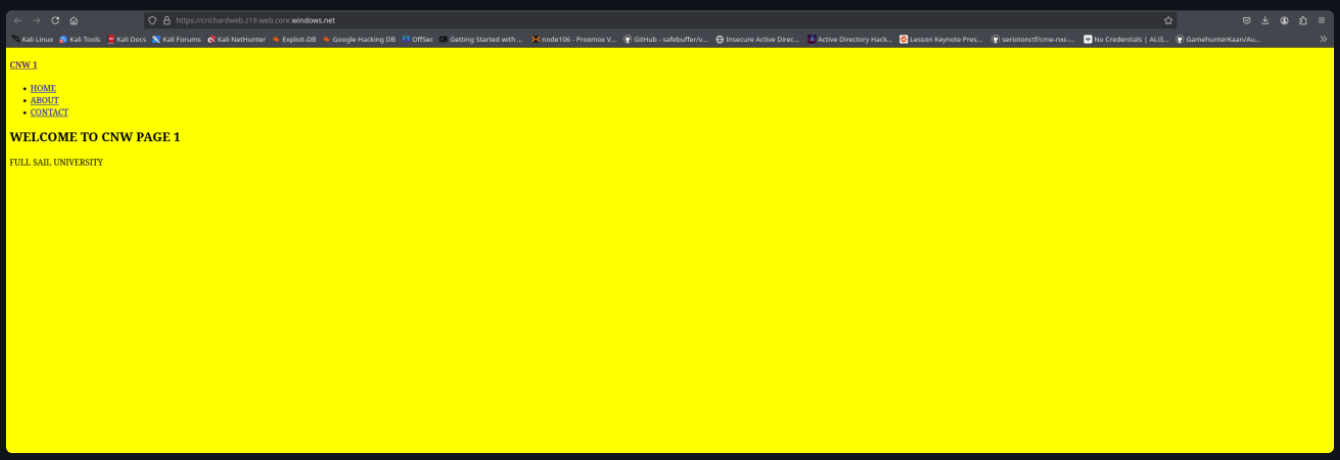

FIG 04 — Same endpoint after an in-place blob overwrite — yellow-themed Full Sail University page, demonstrating live content pivoting without uploading a new object.

STEP 03 — PERSISTENCE

Serverless App Service Provisioning

Created the crichardweb App Service Plan (F1 Free Tier, Windows) and the cnw1crichard App Service instance, then served a controlled lab page at its default domain.

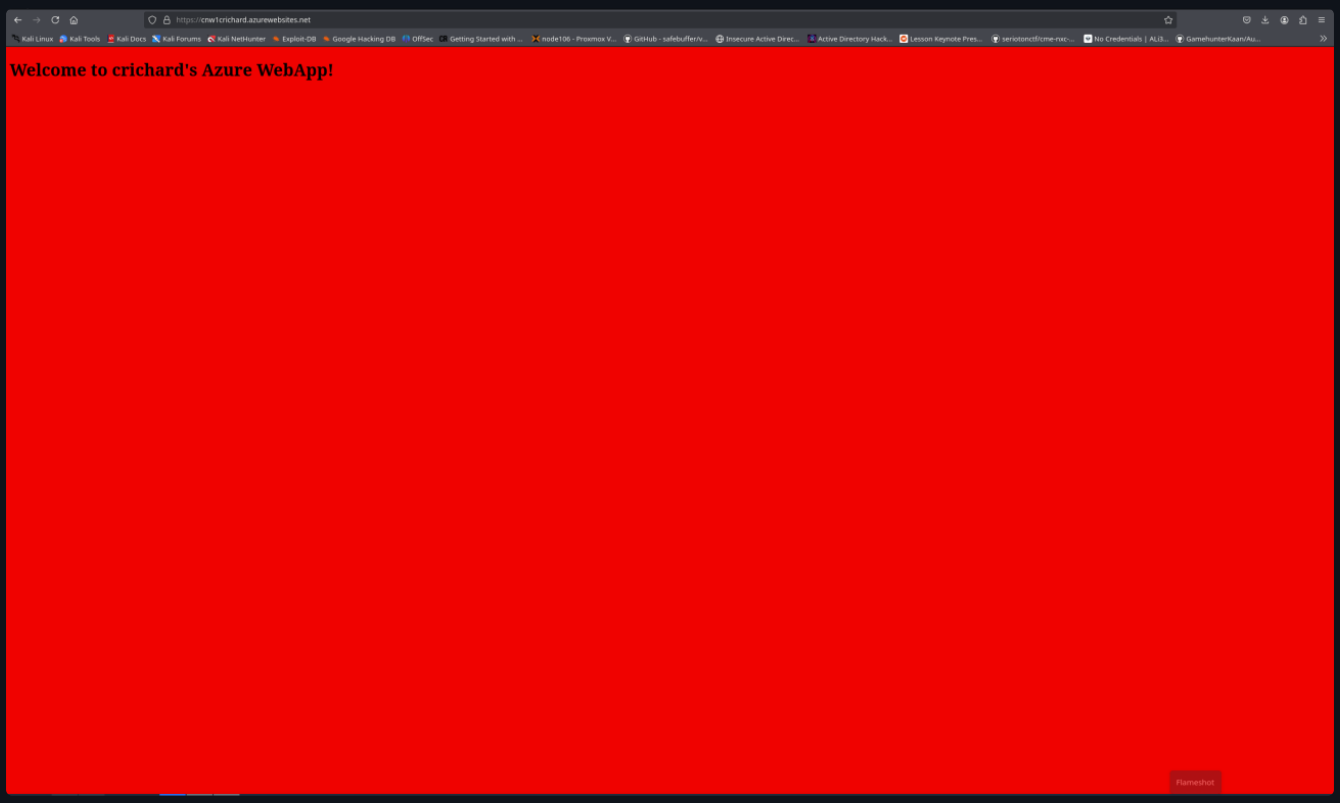

FIG 05 — Live App Service at https://cnw1crichard.azurewebsites.net — red-themed 'Welcome to crichard's Azure WebApp!' page serving as a benign web-shell-equivalent for the lab.

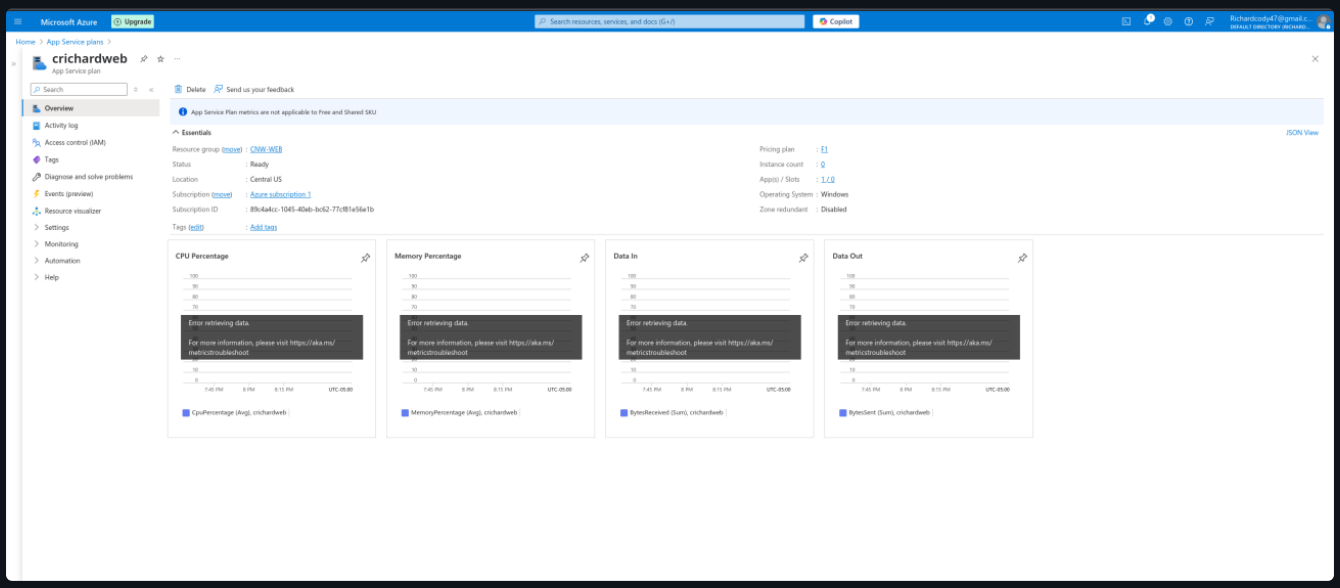

FIG 06 — crichardweb App Service Plan Overview — F1 pricing tier, Windows OS, Central US, status Ready, in resource group CNW-WEB (metrics N/A on Free/Shared SKU).

STEP 04 — PERSISTENCE / KUDU EDIT

Kudu API Content Overwrite

Used the Kudu (SCM) App Service Editor to directly overwrite hostingstart.html under wwwroot, replacing the default landing content rather than redeploying — an in-place edit consistent with low-footprint operator discipline.

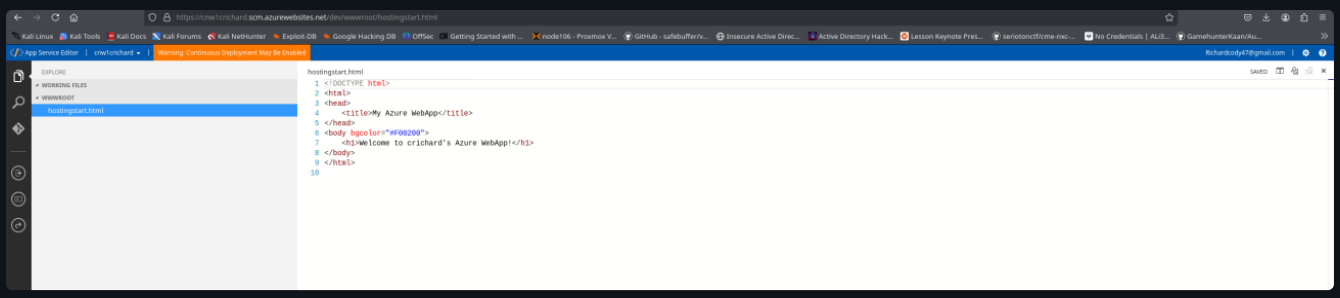

FIG 07 — Kudu App Service Editor at cnw1crichard.scm.azurewebsites.net — hostingstart.html source showing title 'My Azure WebApp', body bgcolor #F00200, and the h1 'Welcome to crichard's Azure WebApp!'.

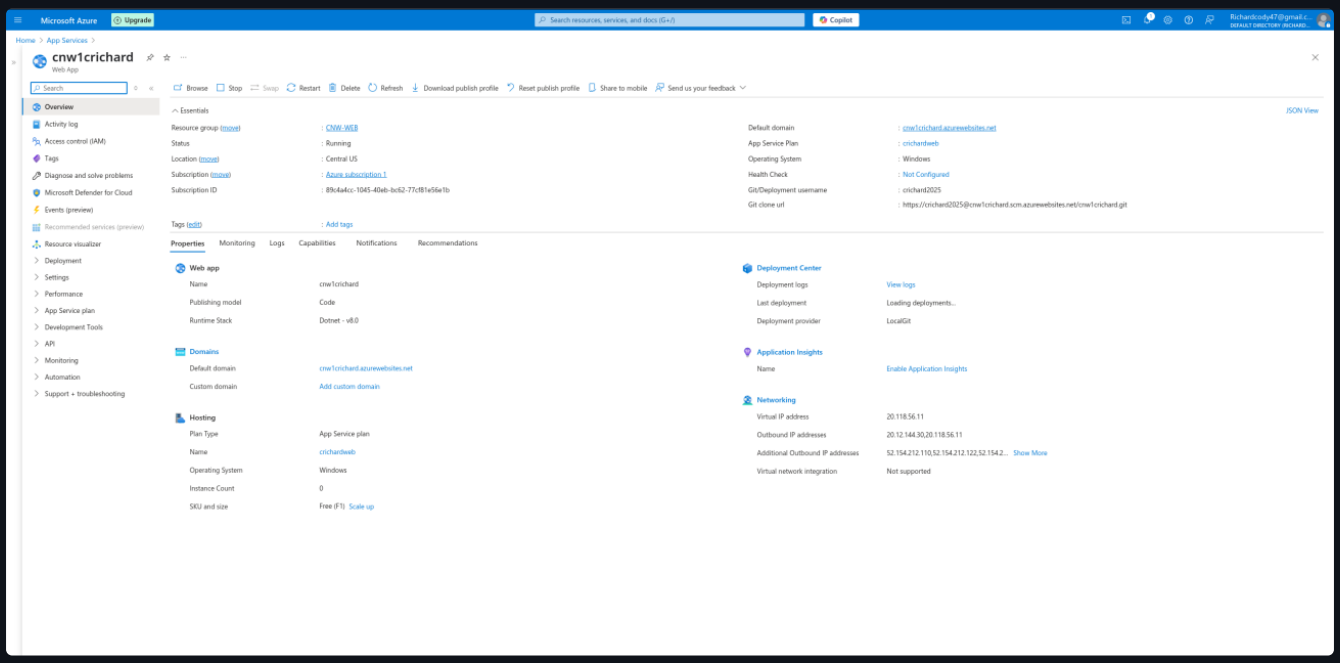

FIG 08 — cnw1crichard App Service Overview — Running, default domain cnw1crichard.azurewebsites.net, App Service Plan crichardweb, Dotnet v8.0 runtime, SKU Free (F1), LocalGit deployment provider.

STEP 05 — COVER TRACKS

Resource Group Teardown

Deleted the entire CNW-WEB resource group via the CLI, wiping the storage account, App Service, and App Service Plan in a single command and leaving no residual artifacts.

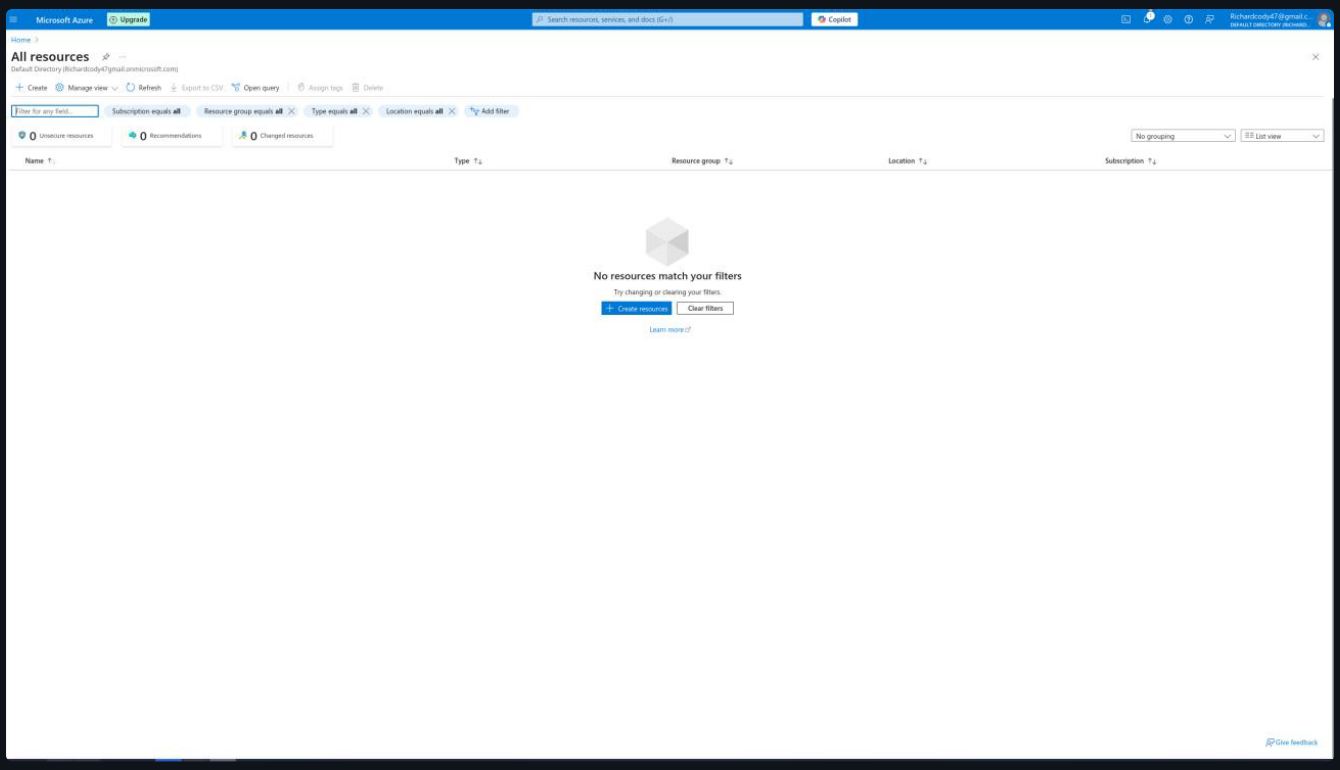

FIG 09 — Azure 'All resources' view post-teardown — 'No resources match your filters', confirming full environmental cleanup after the CLI resource group deletion.

STEP 06 — CLI TRADECRAFT

Behind the Scenes: CLI Operations

All deployment and validation ran from the Azure CLI to avoid portal-generated events — blob uploads to $web, an external endpoint query for validation, and an --overwrite upload for the in-place content pivot.

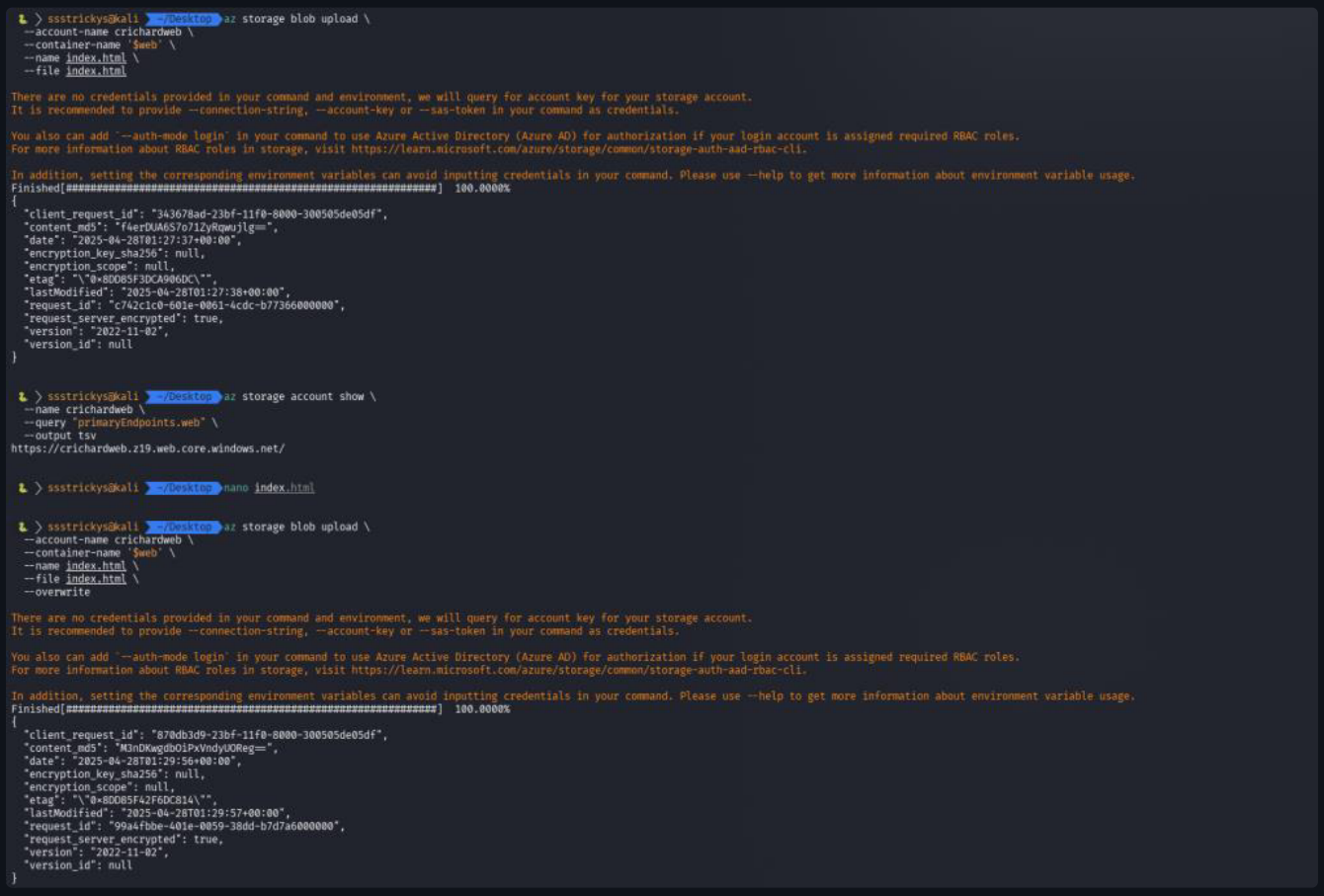

FIG 10 — Azure CLI session: az storage blob upload to the $web container, az storage account show querying primaryEndpoints.web (returning https://crichardweb.z19.web.core.windows.net/), and a follow-up az storage blob upload --overwrite of index.html — validating exposure and pivoting content without touching the Portal UI.