Deploy-Tool: Technical Documentation

- Uzaif Ali

Flow Of Content:

1. Introduction

The deploy-tool is a powerful, command-line interface (CLI) designed to automate and simplify the deployment of static React applications to Amazon Web Services (AWS). Inspired by the seamless experience of platforms like Vercel, this tool bridges the gap between frontend development and cloud infrastructure, enabling developers to build, deploy, version, and monitor their applications with a single set of commands, directly from their terminal. It is built to handle the entire deployment lifecycle, from cloning a GitHub repository and building the project to deploying it and setting up automatic health checks.

2. Features

- **One-Command Deployment**: A single command, deploy-tool deploy, orchestrates the entire pipeline, including cloning, building, and uploading the application.
- **Automatic Framework Detection**: The tool intelligently inspects the project's package.json file to identify the framework (e.g., Vite, React) and applies the correct build commands and configurations automatically.
- **Robust Build System**: Employs multiple fallback strategies for dependency installation and building to handle complex or broken projects successfully.
- **Atomic Deployments & Instant Rollbacks**: Leverages a robust versioning strategy on S3 that allows for zero-downtime deployments and the ability to instantly revert to any previous version.
- **Integrated Monitoring Stack**: Includes a full-featured monitoring system (Prometheus, Grafana) that can be managed with simple start and stop commands.

- **Automatic Application Discovery**: The monitoring system automatically detects and begins monitoring the health of any new application deployed by the tool, requiring zero manual configuration.
- **Cost-Efficient by Design**: The monitoring server runs on a separate EC2 instance that can be shut down on demand to save significant costs, and the hosting relies on low-cost S3 storage.

3. Tech Stack

Category	Component	Technology / Service	Purpose & Justification
CLI Tool (deploy-tool)	Language	Python 3	Chosen for its powerful automation capabilities, extensive libraries, and seamless integration with cloud services.
	CLI Framework	Click	A Python library for creating beautiful, composable, and easily maintainable command-line interfaces.
	AWS Interaction	Boto3	The official AWS SDK for Python, used to programmatically manage all AWS resources like S3 and EC2.
	User Interface	Colorama	Used to enhance the terminal experience with cross-platform coloured text.
Deployment Infrastructure	Cloud Provider	Amazon Web Services (AWS)	A reliable, scalable, and feature-rich cloud platform that provides all the necessary building blocks for hosting and monitoring.
	Core Hosting	Amazon S3	Used to store and serve the static files (HTML, CSS, JS) of the deployed applications. It is extremely cost-effective, durable, and scalable.
	Hosting Model	S3 Static Website Hosting	A serverless feature of S3 that allows it to serve web content directly, providing a simple and highly reliable hosting solution.
Monitoring Infrastructure	Compute	Amazon EC2	A virtual server that hosts the monitoring stack. It can be started and stopped on demand via the CLI to save costs.
	Containerization	Docker & Docker Compose	The entire monitoring stack is containerized, ensuring the environment is consistent, portable, and easily managed as a single unit.
	Metrics Collection	Prometheus	The industry-standard, open-source tool for collecting time-series metrics by "scraping" data from various endpoints.
	Visualization	Grafana	An open-source platform for creating powerful, interactive dashboards to visualize metrics collected by Prometheus.

Category	Component	Technology / Service	Purpose & Justification
	Health Checks	Blackbox Exporter	A Prometheus exporter that performs health checks by probing application URLs to measure uptime, latency, etc.
	Auto-Discovery	Python	A key piece of custom automation that automatically finds all deployed applications in S3 and tells Prometheus to monitor them.
Infrastructure as Code	Automation Tool	Terraform	The leading IaC tool used to define all AWS resources as code. This makes the entire infrastructure reproducible, version-controlled, and easy to manage.

4. Architecture

The deploy-tool utilizes a dual-system architecture designed for flexibility, scalability, and control.

- **The Deployment Pipeline**: This is a client-side automation system, the deploy-tool itself, which runs on the user's local machine. It is responsible for orchestrating the build and deployment of web applications to the AWS S3 bucket.

  code structure:

```
code structure:
 cli-tool/
 ├── .deploy-config.json # generated on `deploy-tool init`
 ├── .gitignore
 ├── README.md
 ├── requirements.txt
 ├── deploy_tool/
 │   ├── init.py
 │   ├── cli.py
 │   ├── commands/
 │   │   ├── init.py
 │   │   ├── config.py
 │   │   ├── deploy.py
 │   │   ├── init.py
 │   │   ├── monitoring.py
 │   │   ├── rollback.py
 │   │   ├── status.py
 │   │   └── versions.py
 │   └── core/
 │       ├── init.py
 │       ├── aws_manager.py
 │       ├── build_manager.py
 │       ├── config_manager.py
 │       ├── git_manager.py
 │       └── utils.py
 ├── terraform/
 ├── monitoring/
 │   ├── terraform/
```

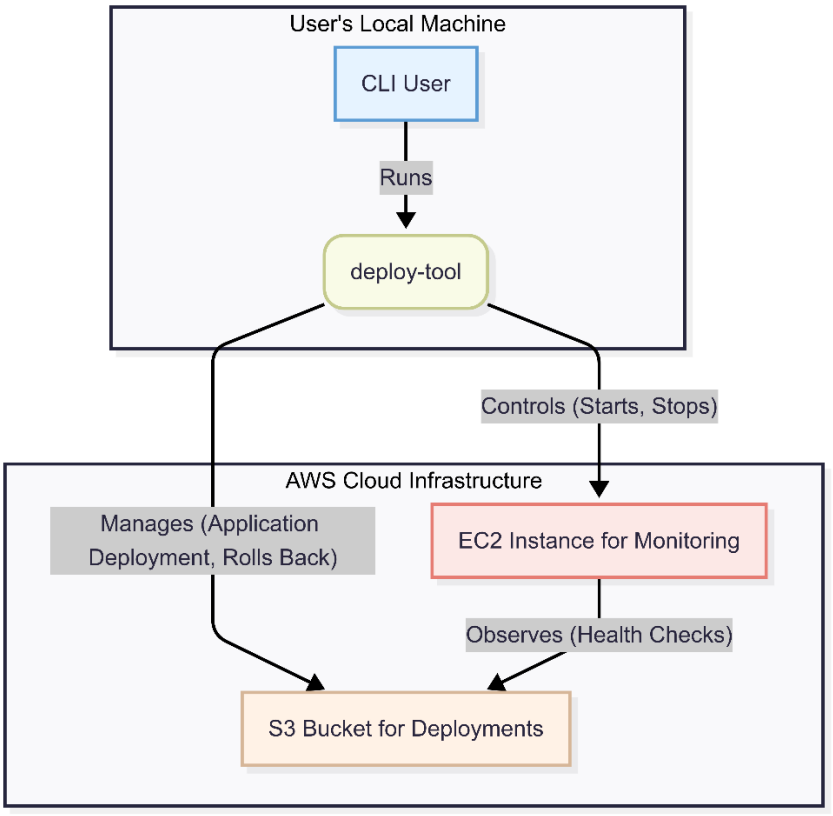
S3 Bucket Structure:

```
s3://your-deployment-bucket/
└── my-app/
    ├── versions/
    │   ├── v20250721-100000/  (Version 1 - Contains index.html, assets, etc.)
    │   └── v20250721-100500/  (Version 2 - Contains its own full set of files)
    │
    └── current/              (This folder serves what is "live" to the user)
```

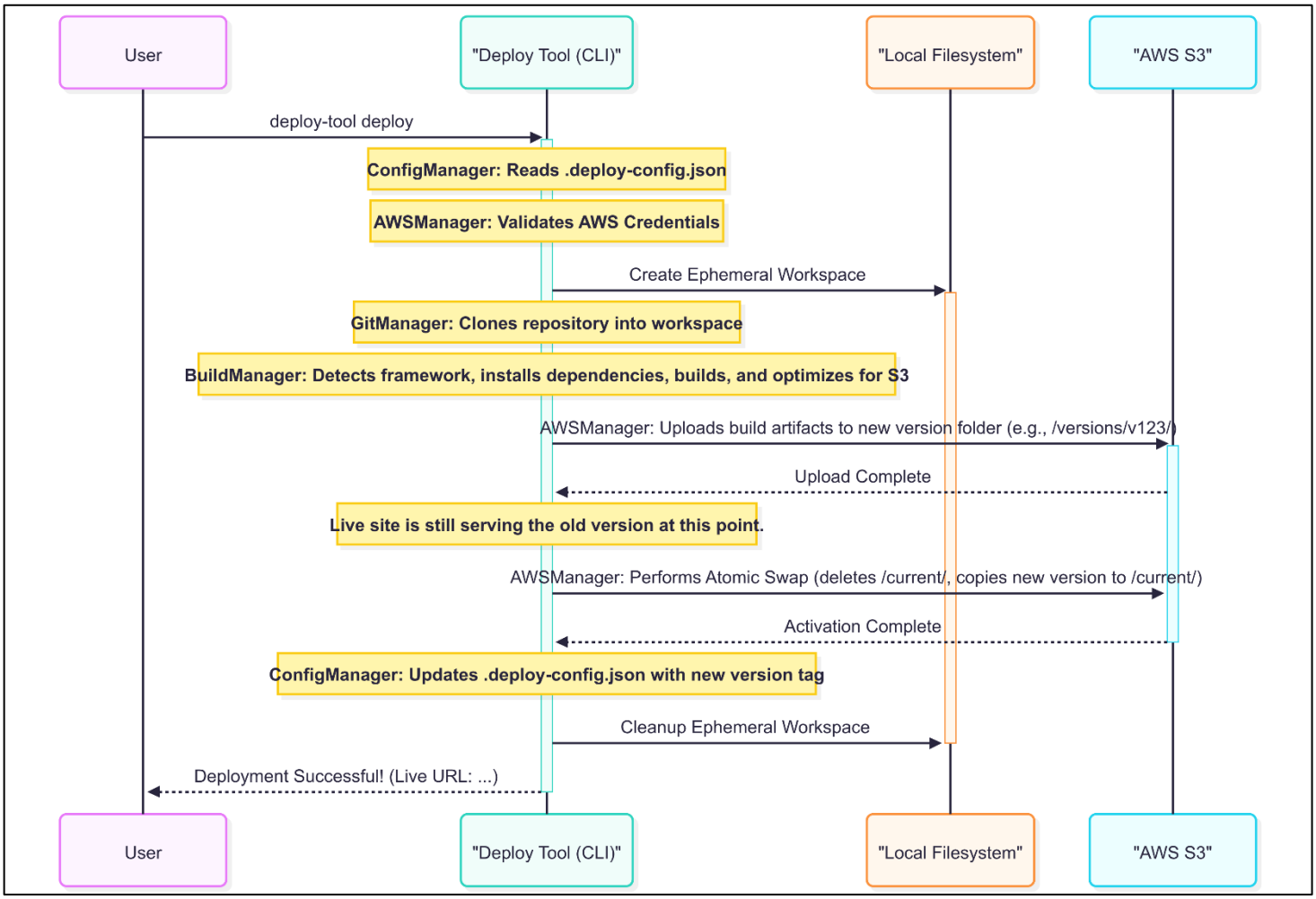
- **The Monitoring Stack**: This is a server-side infrastructure running on a separate AWS EC2 instance. It is responsible for observing the health, uptime, and performance of all deployed applications.

```
code structure:
monitoring-server/
 ├── docker-compose.yml
 │
 ├── prometheus/
 │   ├── prometheus.yml
 │   └── targets/
 │       └── auto_discovered_websites.json
 │
 ├── grafana/
 │   ├── dashboards/
 │   │   └── application_dashboard.json
 │   │
 │   └── provisioning/
 │       ├── datasources/
 │       │   └── prometheus.yml
 │       └── dashboards/
 │           └── dashboard.yml
 │
 ├── alertmanager/
 │   └── alertmanager.yml
 │
 ├── blackbox/
 │   └── blackbox.yml
 │
 ├── discovery-service/
 │   ├── discovery.py
 │   └── requirements.txt
```

5. Workflow



I. The Core Deployment Workflow (deploy-tool deploy)



II. The Rollback Workflow (deploy-tool rollback)

The rollback workflow is designed to be incredibly simple, fast, and safe, leveraging the versioning architecture in S3.

1. Identify Target Version: The user runs deploy-tool rollback. The tool checks the command for a specific version tag to roll-back to that particular version of the application.
2. Execute Atomic Copy: The AWSManager performs the same atomic swap as a deployment, but in reverse. It deletes the contents of the current/ folder and copies the contents from the desired *previous* version's folder into current/.
3. Update Configuration: The .deploy-config.json file is updated to reflect that the rolled-back version is now the current_version.

III. The Monitoring Management Workflow (deploy-tool monitoring start/stop)

This workflow allows the user to control the server-side monitoring infrastructure directly from their CLI.

1. User Issues Command: The user runs deploy-tool monitoring start.
2. EC2 Instance Management: The AWSManager makes an API call to AWS to start the specified EC2 instance. It waits until the instance is in the "running" state.
3. SSH Connection: The tool securely connects to the EC2 instance using the provided SSH key.
4. Docker Compose Execution: Once connected, it executes the docker-compose up -d command on the server, which starts all the monitoring services (Prometheus, Grafana, etc.) in the background.

The monitoring stop command performs the reverse: it runs docker-compose down on the server and then issues an API call to AWS to stop the EC2 instance, thus saving costs.

6. AWS Permissions and Security Rules

For the deploy-tool to function correctly and securely, specific AWS permissions are required. These should be attached to the IAM role that the user assumes when they log in via AWS SSO.

The IAM role used by your AWS profile needs the following permissions:

Service	Actions	Resource	Purpose
Amazon S3	s3:PutObject, s3:GetObject, s3:ListBucket, s3:DeleteObject, s3:PutBucketWebsite	arn:aws:s3:::your-bucket-name/*	To upload, manage, and serve the application files.
Amazon EC2	ec2:StartInstances, ec2:StopInstances, ec2:DescribeInstances	arn:aws:ec2:*:*:instance/your-instance-id	To start, stop, and check the status of the monitoring server.
AWS STS	sts:GetCallerIdentity	*	To validate the user's current authenticated session.

Monitoring Server Security Group Rules

The EC2 instance for the monitoring server requires a Security Group with the following inbound rules to allow access to its services:

Port Range	Protocol	Source	Description
22	TCP	Your IP Address	Allows you to SSH into the server for maintenance.
3000	TCP	0.0.0.0/0 (Anywhere)	Public access to the Grafana dashboard.
9090	TCP	0.0.0.0/0 (Anywhere)	Public access to the Prometheus dashboard.
9093	TCP	0.0.0.0/0 (Anywhere)	Public access to the Alertmanager UI.
8082	TCP	0.0.0.0/0 (Anywhere)	Public access to the custom Discovery Service.

7. Setup Guide

Prerequisites:

AWS Account
AWS CLI
Python 3.8 or higher
Git

It is assumed that the user will have node and npm installed in his system.

Step-by-Step Setup

1. Configure AWS SSO
   Before using the tool, you must configure your AWS profile. Open a terminal and run:

   ```
   aws configure sso
   ```

   Follow the on-screen prompts. When asked for a "Profile name," use “Uzaif”.

2. Clone the Github Repository

   ```
   git clone https://github.com/Uzaif-Minfy/Capstone-DevOps
   ```

3. Install the Deploy Tool
   Install the deploy-tool using pip.

   ```
   cd Uzaif-DevOps-Capstone/cli-tool
   pip install .
   ```

4. Place Your SSH Key (only for monitoring)
   The tool needs an SSH key to manage the monitoring server. Here my SSH key is used which is placed in: "C:\Users\Minfy\Desktop\capstone-devops\minfy-uzaif-capstone-key-pair.pem"

5. Initialize Your First Project
   You are now ready to set up your first project. Create a directory for your project, navigate into it, and run the init command:

```
mkdir my-first-project
cd my-first-project
deploy-tool init --github-url https://github.com/your-username/your-repo
```

This will create a .deploy-config.json file in the directory, and you are ready to deploy.

8. Comprehensive Command Reference

deploy-tool –help/ deploy-tool <command> --help: displays rules/options to run the tool

Core Deployment Commands

Command	Description	Example Usage
deploy-tool init	Initializes a new project in the current directory.	deploy-tool init --github-url <url>
deploy-tool deploy	Builds and deploys the application to AWS S3.	deploy-tool deploy
deploy-tool status	Displays the current status and active version.	deploy-tool status
deploy-tool rollback	Reverts to a previous version.	deploy-tool rollback --version <tag>
deploy-tool versions	Lists the last 10 available deployment versions and version clean-up.	deploy-tool versions –cleanup –keep <no.>

Configuration Management

Command	Description	Example Usage
deploy-tool config show	Displays the current project configuration.	deploy-tool config show
deploy-tool config set	Sets a configuration value using dot notation.	deploy-tool config set project.name new-app
deploy-tool config get	Retrieves a specific configuration value.	deploy-tool config get project.name
deploy-tool config reset	Deletes the configuration file (requires confirmation).	deploy-tool config reset

Monitoring Management

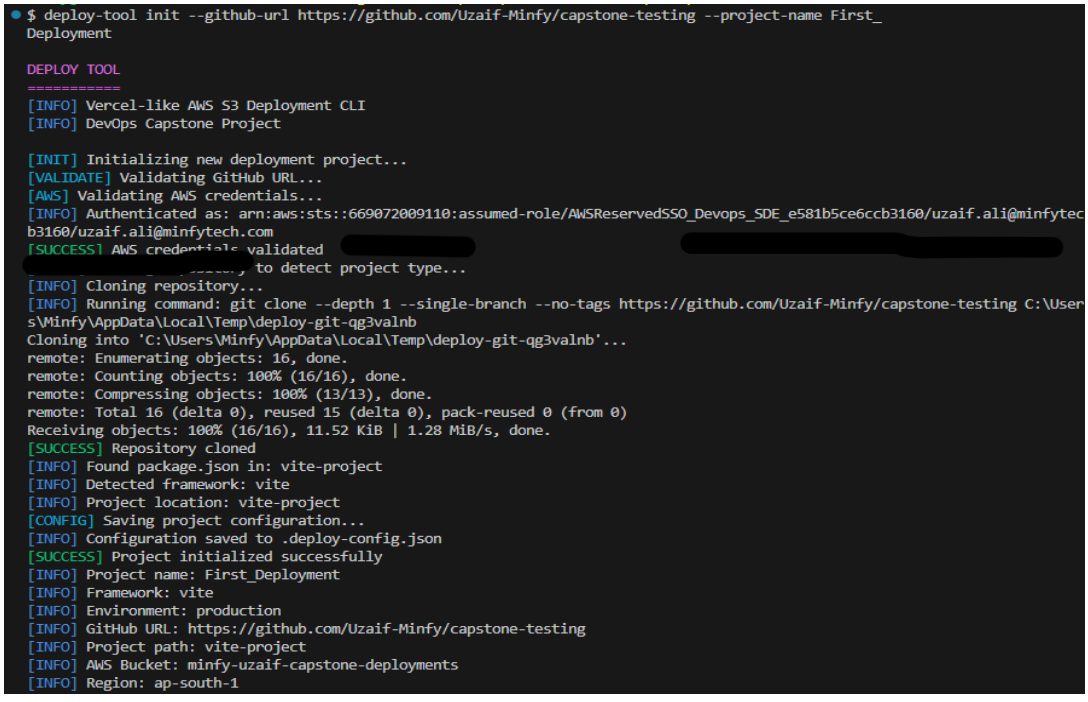
Command	Description	Example Usage
deploy-tool monitoring start	Starts the EC2 instance and monitoring services.	deploy-tool monitoring start
deploy-tool monitoring stop	Stops the EC2 instance to save costs.	deploy-tool monitoring stop
deploy-tool monitoring status	Checks the status of the monitoring server.	deploy-tool monitoring status
deploy-tool monitoring urls	Displays access URLs for Grafana and Prometheus.	deploy-tool monitoring urls
deploy-tool monitoring dashboard	Opens the Grafana dashboard in a web browser.	deploy-tool monitoring dashboard
deploy-tool monitoring discovered	Lists all apps automatically found by the service.	deploy-tool monitoring discovered

9. Tutorial- Your First Deployment

Aim: To deploy a simple static application

GitHub URL of the application: https://github.com/Uzaif-Minfy/capstone-testing

- **Initialize Your Project**
  deploy-tool init --github-url https://github.com/Uzaif-Minfy/capstone-testing --project-name First_Deployment



This creates a .deloy-config.json file for your project

Uzaif-DevOps-Capstone > cli-tool > {} .deploy-config.json > ...


```
{
  "aws": {
    "bucket": "minfy-uzaif-capstone-deployments",
    "profile": "Uzaif",
    "region": "ap-south-1"
  },
  "build": {
    "build_command": "npm run build",
    "install_command": "npm ci",
    "output_dir": "dist"
  },
  "deployment": {
    "auto_cleanup": true,
    "versions_to_keep": 10
  },
  "project": {
    "created_at": "2025-07-23T15:52:03.555106",
    "current_version": "v20250723-155452",
    "environment": "production",
    "framework": "vite",
    "github_url": "https://github.com/Uzaif-Minfy/capstone-testing",
    "last_deployed": "2025-07-23T15:55:02.296322",
    "name": "First_Deployment",
    "project_path": "vite-project"
  }
}
```


- **Deploy Your Application**

  deploy-tool deploy

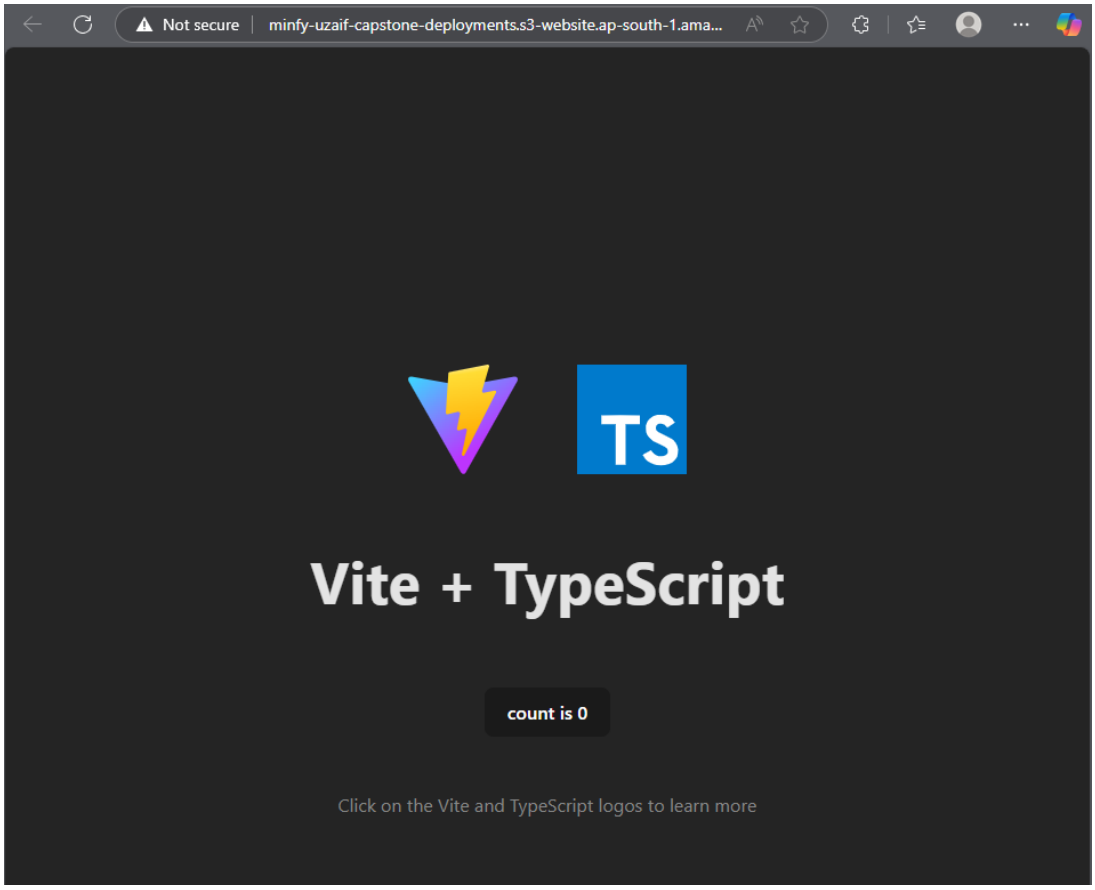
```
$ deploy-tool deploy

DEPLOY TOOL
==========
[INFO] Vercel-like AWS S3 Deployment CLI
[INFO] DevOps Capstone Project

[DEPLOY] Starting deployment process...
[AUTH] Validating AWS credentials...
[INFO] Authenticated as: arn:aws:sts::[redacted]umed-role/AWSReserved[redacted]c
h.com
[INFO] Deploying version: v20250723-155452
[1/4] Preparing repository...
[INFO] Cloning repository...
[INFO] Running command: git clone --depth 1 --single-branch --no-tags https://github.com/Uzaif-Minfy/capstone-testing C:\User
s\Minfy\AppData\Local\Temp\deploy-workspace\deploy-First_Deployment-e74b6b2b
Cloning into 'C:\Users\Minfy\AppData\Local\Temp\deploy-workspace\deploy-First_Deployment-e74b6b2b'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
✓ built in 165ms
[SUCCESS] Standard build completed
[SUCCESS] Fixed asset paths in 1 HTML files
[SUCCESS] Build completed: 5 files, 6.1KB
[3/4] Deploying to S3...
[UPLOAD] Uploading files to S3...
[SUCCESS] Uploaded 4 files (6.1KB)
[4/4] Activating new version...
[ACTIVATE] Making version live...
[SUCCESS] Website is now live (4 files)
[INFO] Configuration saved to .deploy-config.json
[SUCCESS] Deployment completed successfully!
[INFO] Version: v20250723-155452
[INFO] Live URL: http://minfy-uzaif-capstone-deployments.s3-website.ap-south-1.amazonaws.com/First_Deployment/current/
[INFO] Deployed: 2025-07-23 15:55:02
```

This generated the URL for your project.

The application is now live



- **Check the Status**

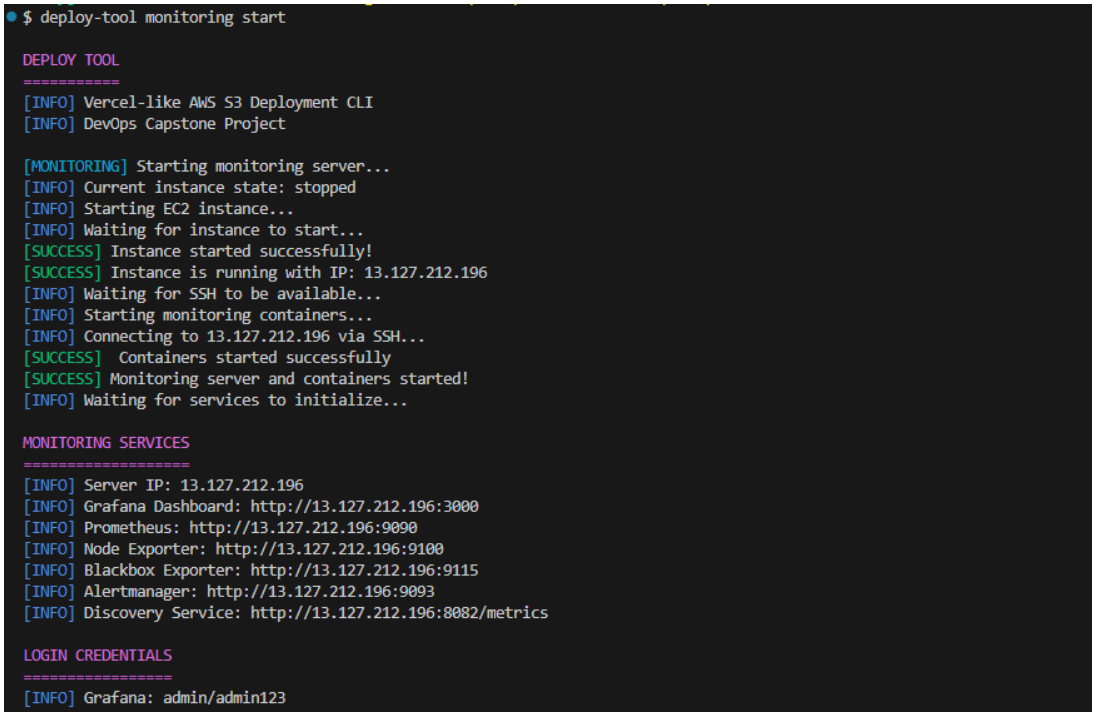
```
$ deploy-tool status

DEPLOY TOOL
===========
[INFO] Vercel-like AWS S3 Deployment CLI
[INFO] DevOps Capstone Project


PROJECT STATUS
==============
[INFO] Project Name:     First_Deployment
[INFO] Framework:        vite
[INFO] GitHub URL:       https://github.com/Uzaif-Minfy/capstone-testing
[INFO] AWS Bucket:       minfy-uzaif-capstone-deployments
[INFO] Region:           ap-south-1
[INFO] Current Version:  v20250723-155452
[INFO] Live URL:         http://minfy-uzaif-capstone-deployments.s3-website.ap-south-1.amazonaws.com/First_Deployment/current
/
[INFO] Created:          2025-07-23T15:52:03.555106


AVAILABLE VERSIONS
==================
* v20250723-155452 (current)
```

- **Optional - Enable monitoring**
  If you want to use monitoring service, run
  deploy-tool monitoring start



Later to stop monitoring, you need to run
`deploy-tool monitoring stop`

10. Troubleshooting Guide

- Problem: AWS Authentication Error ("Access Denied")
    - Cause: Your AWS SSO session has likely expired, or your IAM role lacks the necessary permissions.
    - Solution:
        1. Run aws sso login --profile Uzaif (here profile name Uzaif is used) to refresh your session.
        2. Ensure your IAM role has the permissions listed in the "AWS Permissions" section of this document.
- Problem: Build Fails with "npm" or "vite" Errors
    - Cause: The project may have dependency conflicts or a broken build script.
    - Solution:
        1. The deploy-tool has fallback mechanisms, but if they fail, try cloning the repository locally and running npm install && npm run build to debug the issue directly.
        2. Check for and resolve any vulnerabilities reported by npm audit.
- Problem: Monitoring Commands Fail with SSH Errors
    - Cause: The SSH key may not be found or may have incorrect permissions.
    - Solution:
        1. Ensure your .pem key file is located in your C:\Users\Minfy\Desktop\capstone-devops\ directory.
        2. Ensure the key has the correct permissions by running
           `chmod 400  minfy-uzaif-capstone-key-pair.pem`

11. Upcoming Features/Updates

The deploy-tool is under active development. Planned future enhancements include:

- Build with .env: Utilize .env file provided with ‘deploy-tool deploy’ command for streamlined configuration during deployment.
- Webhook Automation: A listener service to automatically trigger deployments on every git push to the main branch, creating a true CI/CD pipeline.
- Web-Based User Interface: Develop a web dashboard that provides a graphical interface for managing deployments, viewing status, and rolling back versions, offering an alternative to the CLI.
