Technical Documentation: CLI Deployment Tool

1. Project Summary

This CLI tool automates the end-to-end deployment of React frontend applications on AWS, simplifying complex manual processes into intuitive commands. Designed with a strong focus on automation and developer experience, the tool detects project types automatically, manages infrastructure setup on AWS—including storage and monitoring—provisioning industry-standard monitoring with Prometheus and Grafana. Integrated alerting via Gmail SMTP ensures timely notifications for system health issues.

By abstracting away AWS complexity and embracing modular, reusable components, this deployment tool empowers development teams to rapidly build, deploy, observe, and rollback React-based websites with zero manual AWS configuration knowledge.

2. Technical Stack

Component	Technologies / Tools	Purpose & Reasoning
Programming Language	Python 3.12	Provides cross-platform compatibility, rich standard libraries, and mature third-party packages for AWS integration (boto3) and CLI design.
CLI Framework	Custom CLI using Python's built-in argparse and base classes	Avoids external dependencies, allows full customization of CLI behavior and help generation, ensuring a lightweight, portable tool.
AWS SDK	boto3	Official Python SDK for AWS to programmatically manage S3, EC2, IAM, and other AWS resources.
Build System	Node.js, npm/yarn	Industry-standard tooling for building React and Vite projects, underlying frontend framework compatibility.
Monitoring Stack	Prometheus, Grafana, Alert manager	Selected for their widespread use in production systems for detailed metrics collection, rich visualization, and robust alerting workflows.
Email Alerting	Gmail SMTP	Leverages common, secure email services to notify developers of critical system alerts without complex custom setup.

3. Use Cases

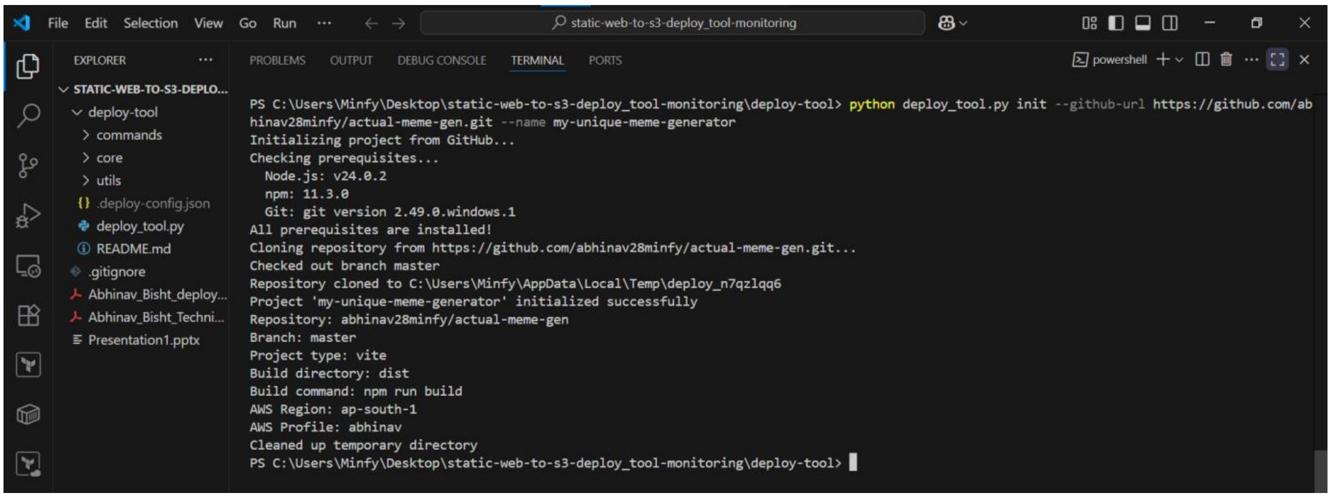
- **Project On-boarding:** Initialize deployment config and AWS infra from source repo with init.
- **Code Deployment:** Build and deploy React apps with a single deploy command.
- **Monitoring Setup:** Provision monitoring stack and configure alerts with monitoring init.
- **Deploy Verification:** Use status to monitor deployment and health check statuses.
- **Deployment Reversion:** Rollback to previous deployment snapshots using rollback.
- **Configuration Management:** Dynamically change deployment settings with config commands.
- **Pre-deployment Checks:** Validate prerequisites and environment integrity using check.

4. Features & CLI Commands

Command	Description
deploy_tool.py init --github-url <url>	Initializes the project, detects React type, creates .deploy-config.json, provisions AWS S3 buckets.
deploy_tool.py rollback [--deployment N]	Rolls back deployment to a previous successful snapshot/version.
deploy_tool.py monitoring init	Provisions Prometheus and Grafana on EC2, configures alerting via Gmail SMTP.
deploy_tool.py monitoring status	Displays monitoring stack health, alert rules, and target statuses.
deploy_tool.py monitoring update	Updates monitoring targets or configurations.
deploy_tool.py monitoring destroy	Tears down monitoring infrastructure (EC2 instance, etc.).
deploy_tool.py config --set key=value	Updates configuration keys in .deploy-config.json, supports nested keys.
deploy_tool.py config --list	Lists current configuration in hierarchical format.
deploy_tool.py status	Shows overall deployment, AWS resources, and health check status.
deploy_tool.py check	Checks system and AWS prerequisites: Node.js, AWS credentials, Git.
deploy_tool.py help or deploy_tool.py -h	Displays contextual usage and options globally or per command.

- Initializing your project:

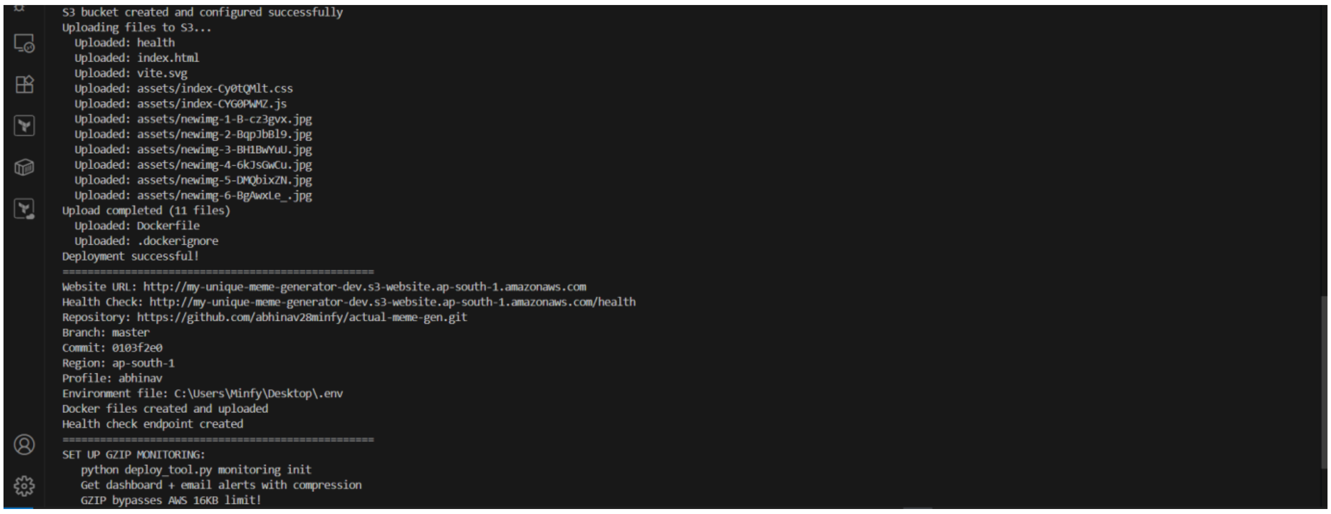
python deploy_tool.py init --github-url https://github.com/abhinav28minfy/actual-meme-gen.git --name my-unique-meme-generator



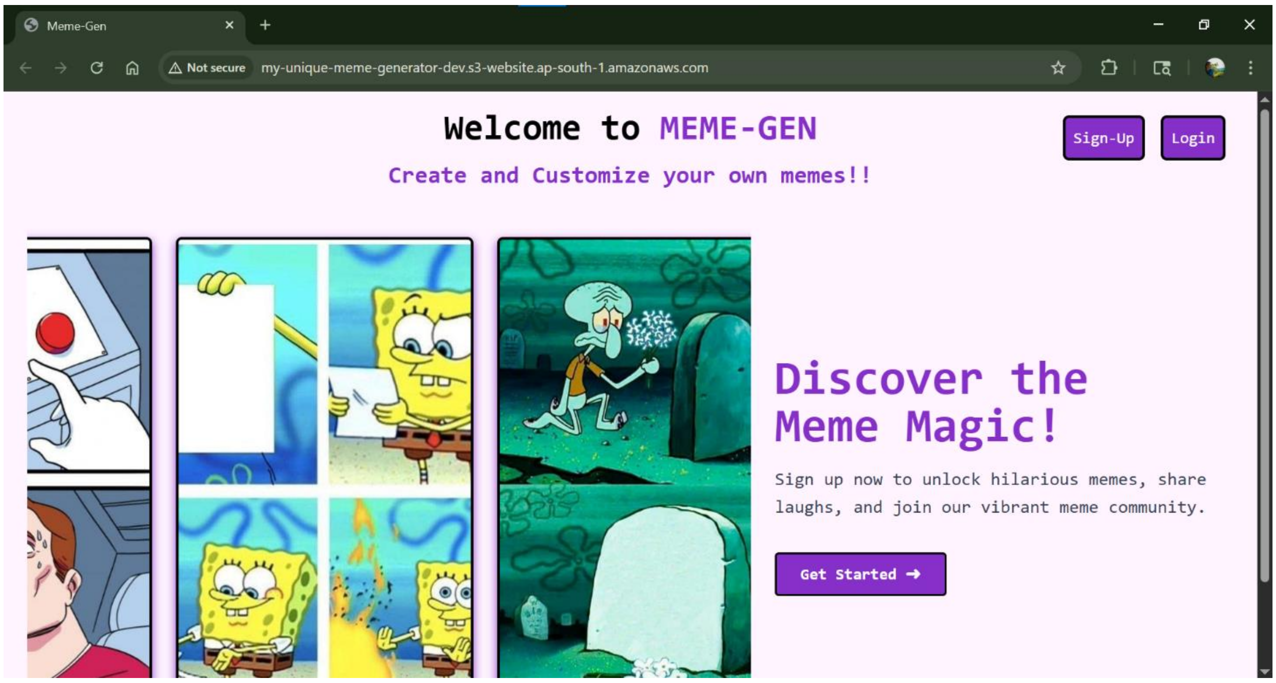
```
PS C:\Users\Minfy\Desktop\static-web-to-s3-deploy_tool-monitoring\deploy-tool> python deploy_tool.py init --github-url https://github.com/ab
hinav28minfy/actual-meme-gen.git --name my-unique-meme-generator
Initializing project from GitHub...
Checking prerequisites...
  Node.js: v24.0.2
  npm: 11.3.0
  Git: git version 2.49.0.windows.1
All prerequisites are installed!
Cloning repository from https://github.com/abhinav28minfy/actual-meme-gen.git...
Checked out branch master
Repository cloned to C:\Users\Minfy\AppData\Local\Temp\deploy_n7qzlqq6
Project 'my-unique-meme-generator' initialized successfully
Repository: abhinav28minfy/actual-meme-gen
Branch: master
Project type: vite
Build directory: dist
Build command: npm run build
AWS Region: ap-south-1
AWS Profile: abhinav
Cleaned up temporary directory
PS C:\Users\Minfy\Desktop\static-web-to-s3-deploy_tool-monitoring\deploy-tool> 
```

- Deploying your project:

python deploy_tool.py deploy --env dev

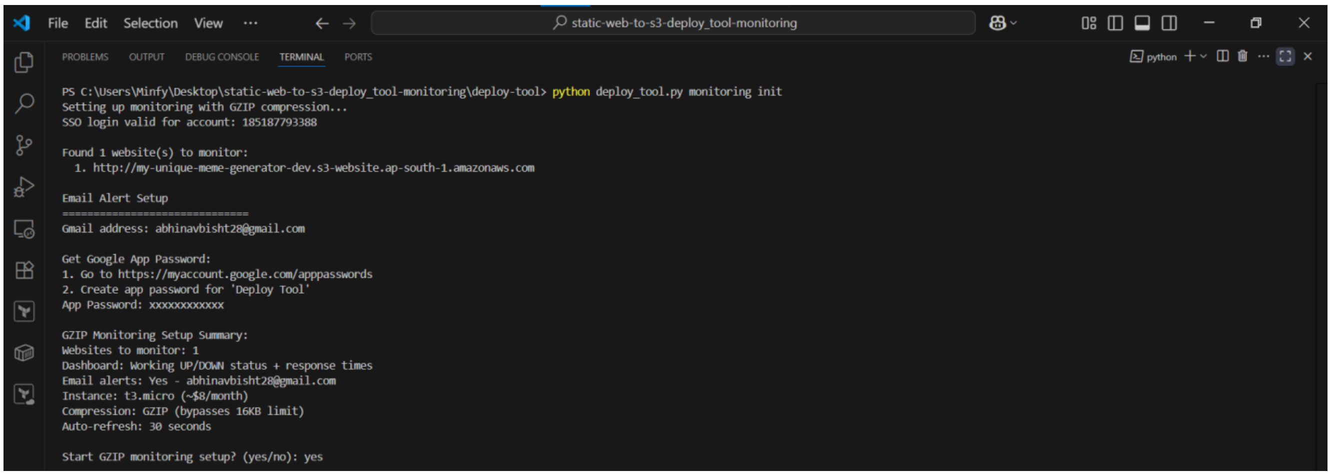


```
S3 bucket created and configured successfully
Uploading files to S3...
  Uploaded: health
  Uploaded: index.html
  Uploaded: vite.svg
  Uploaded: assets/index-Cy0tQMlt.css
  Uploaded: assets/index-CYG0PWMZ.js
  Uploaded: assets/newimg-1-B-cz3gvx.jpg
  Uploaded: assets/newimg-2-BqpJbBl9.jpg
  Uploaded: assets/newimg-3-BH1BwYuU.jpg
  Uploaded: assets/newimg-4-6kJsGwCu.jpg
  Uploaded: assets/newimg-5-DMQbixZN.jpg
  Uploaded: assets/newimg-6-BgAwxLe_.jpg
Upload completed (11 files)
  Uploaded: Dockerfile
  Uploaded: .dockerignore
Deployment successful!
==================================================
Website URL: http://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com
Health Check: http://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com/health
Repository: https://github.com/abhinav28minfy/actual-meme-gen.git
Branch: master
Commit: 0103f2e0
Region: ap-south-1
Profile: abhinav
Environment file: C:\Users\Minfy\Desktop\.env
Docker files created and uploaded
Health check endpoint created
==================================================
SET UP GZIP MONITORING:
   python deploy_tool.py monitoring init
   Get dashboard + email alerts with compression
   GZIP bypasses AWS 16KB limit!
```

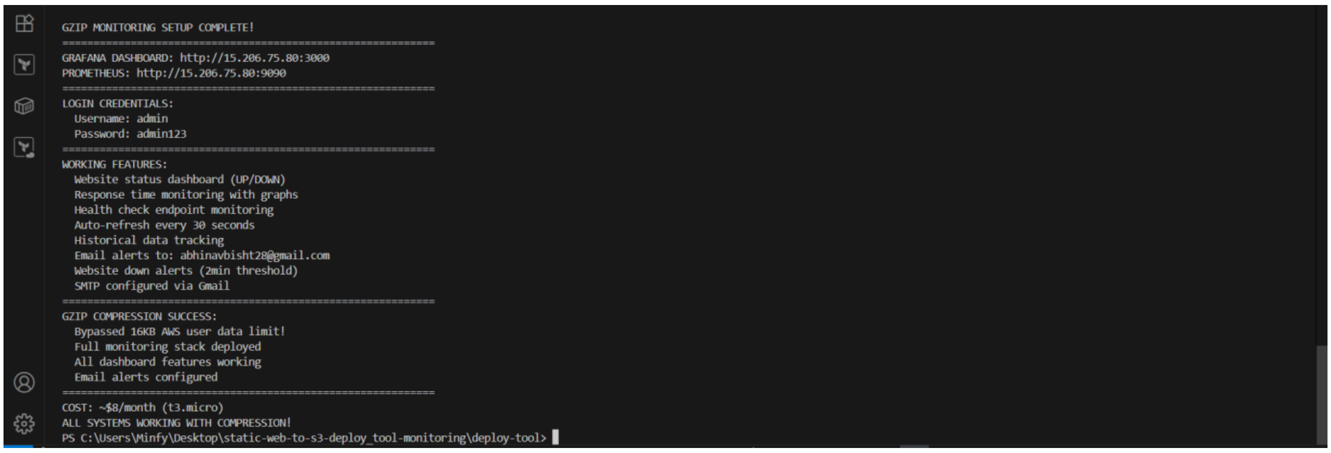

Meme-Gen
Not secure
my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com
Welcome to MEME-GEN
Sign-Up
Login
Create and Customize your own memes!!
Discover the Meme Magic!
Sign up now to unlock hilarious memes, share laughs, and join our vibrant meme community.
Get Started ➜

- Setting up monitoring:

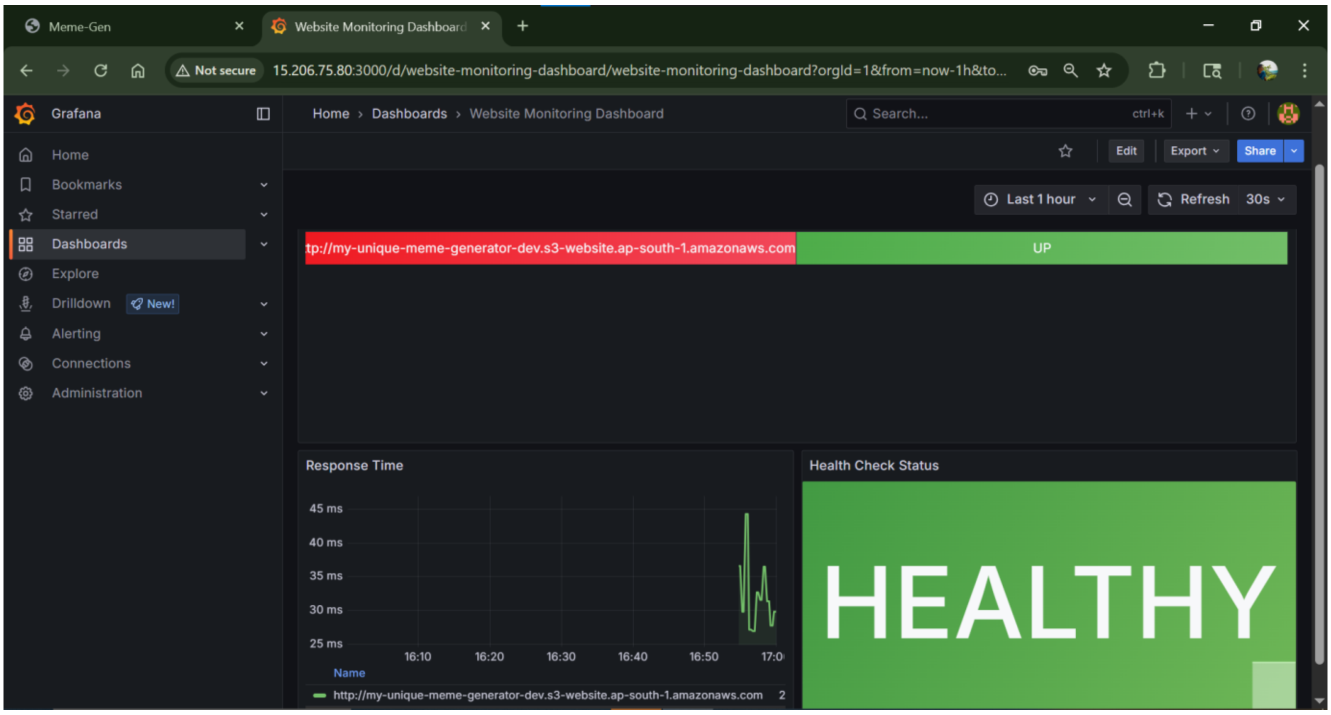
python deploy_tool.py monitoring init



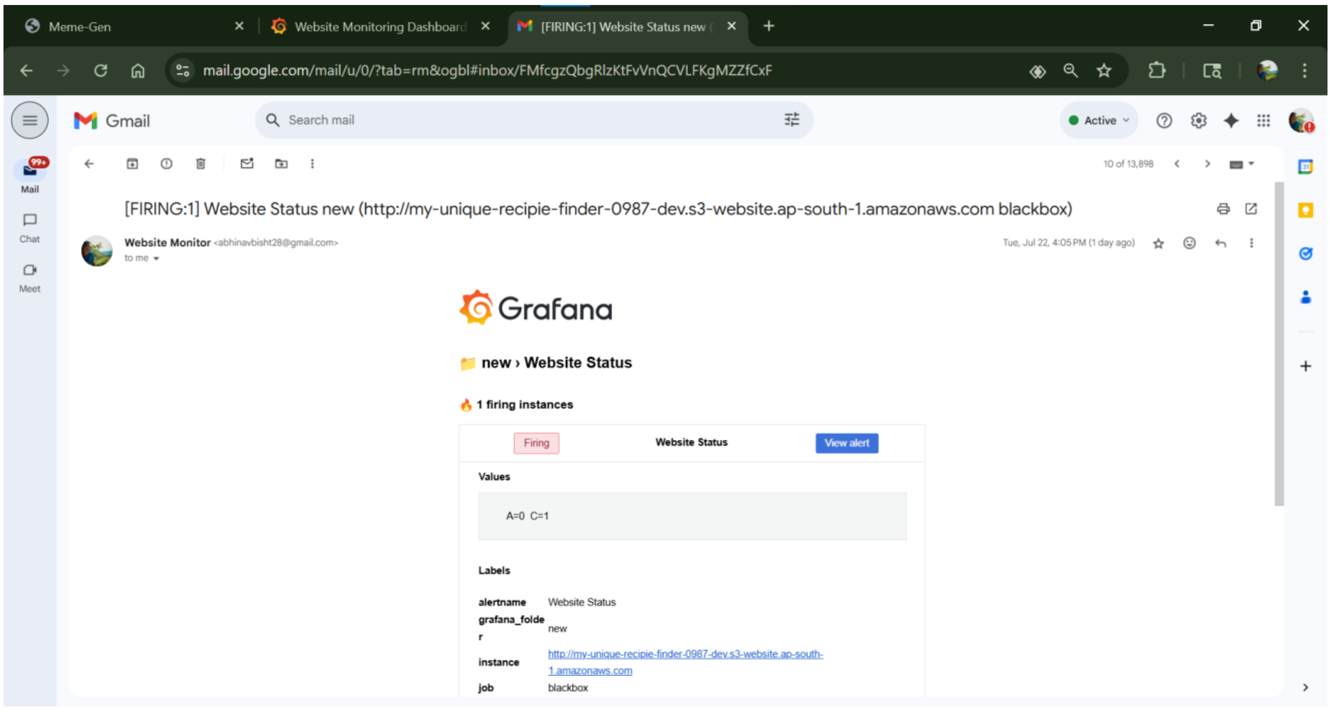
```
PS C:\Users\Minfy\Desktop\static-web-to-s3-deploy_tool-monitoring\deploy-tool> python deploy_tool.py monitoring init
Setting up monitoring with GZIP compression...
SSO login valid for account: 185187793388

Found 1 website(s) to monitor:
  1. http://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com

Email Alert Setup
==============================
Gmail address: abhinavbisht28@gmail.com

Get Google App Password:
1. Go to https://myaccount.google.com/apppasswords
2. Create app password for 'Deploy Tool'
App Password: xxxxxxxxxxxx

GZIP Monitoring Setup Summary:
Websites to monitor: 1
Dashboard: Working UP/DOWN status + response times
Email alerts: Yes - abhinavbisht28@gmail.com
Instance: t3.micro (~$8/month)
Compression: GZIP (bypasses 16KB limit)
Auto-refresh: 30 seconds

Start GZIP monitoring setup? (yes/no): yes
```



```
GZIP MONITORING SETUP COMPLETE!
============================================================
GRAFANA DASHBOARD: http://15.206.75.80:3000
PROMETHEUS: http://15.206.75.80:9090
============================================================
LOGIN CREDENTIALS:
  Username: admin
  Password: admin123
============================================================
WORKING FEATURES:
  Website status dashboard (UP/DOWN)
  Response time monitoring with graphs
  Health check endpoint monitoring
  Auto-refresh every 30 seconds
  Historical data tracking
  Email alerts to: abhinavbisht28@gmail.com
  Website down alerts (2min threshold)
  SMTP configured via Gmail
============================================================
GZIP COMPRESSION SUCCESS:
  Bypassed 16KB AWS user data limit!
  Full monitoring stack deployed
  All dashboard features working
  Email alerts configured
============================================================
COST: ~$8/month (t3.micro)
ALL SYSTEMS WORKING WITH COMPRESSION!
PS C:\Users\Minfy\Desktop\static-web-to-s3-deploy_tool-monitoring\deploy-tool>
```


Meme-Gen
Website Monitoring Dashboard
Not secure
15.206.75.80:3000/d/website-monitoring-dashboard/website-monitoring-dashboard?orgId=1&from=now-1h&to...
Grafana
Home
Bookmarks
Starred
Dashboards
Explore
Drilldown
New!
Alerting
Connections
Administration
Home › Dashboards › Website Monitoring Dashboard
Search...
Edit
Export
Share
Last 1 hour
Refresh
30s
tp://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com
UP
Response Time
45 ms
40 ms
35 ms
30 ms
25 ms
16:10
16:20
16:30
16:40
16:50
17:0
Name
http://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com
Health Check Status
HEALTHY

- Alerts: By setting up contact point and alerting rules



[FIRING:1] Website Status new (http://my-unique-recipie-finder-0987-dev.s3-website.ap-south-1.amazonaws.com blackbox)

Website Monitor <abhinavbisht28@gmail.com>
to me

Tue, Jul 22, 4:05 PM (1 day ago)

Grafana

new › Website Status

1 firing instances

Firing | Website Status | View alert

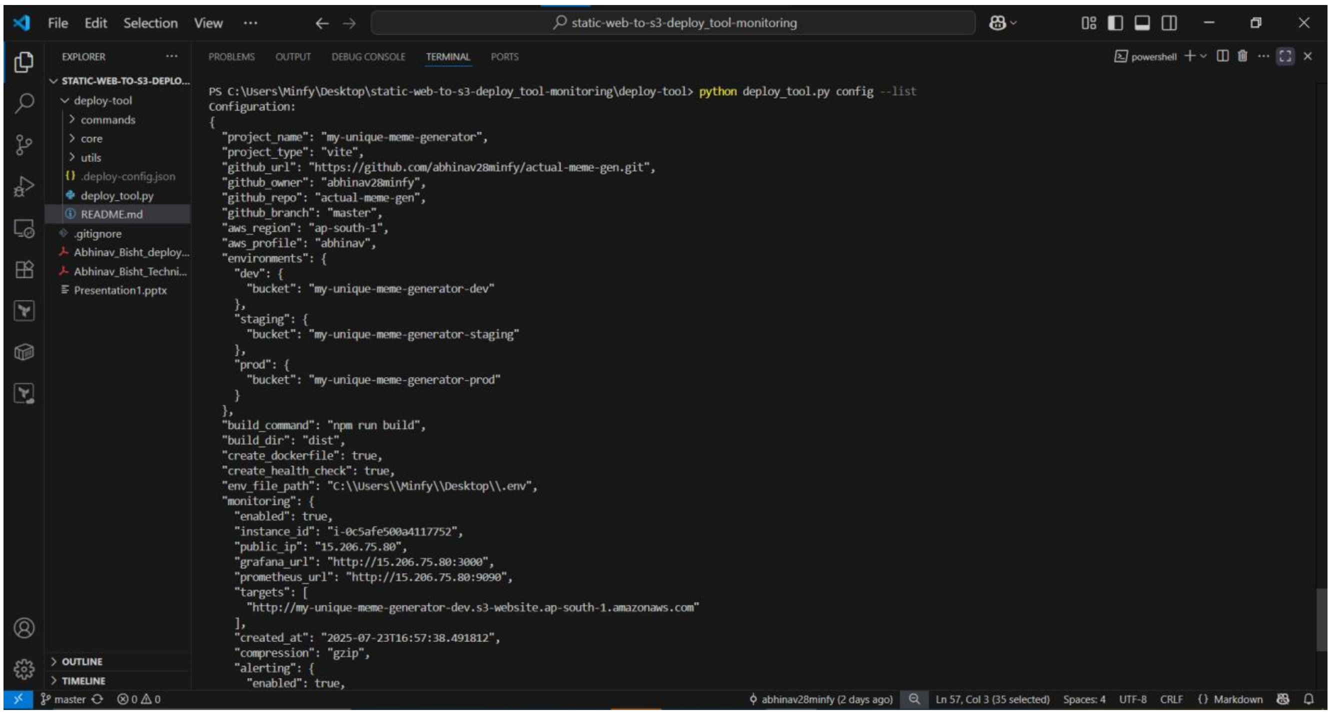
Values

A=0 C=1

Labels

alertname	Website Status
grafana_folder	new
instance	http://my-unique-recipie-finder-0987-dev.s3-website.ap-south-1.amazonaws.com
job	blackbox

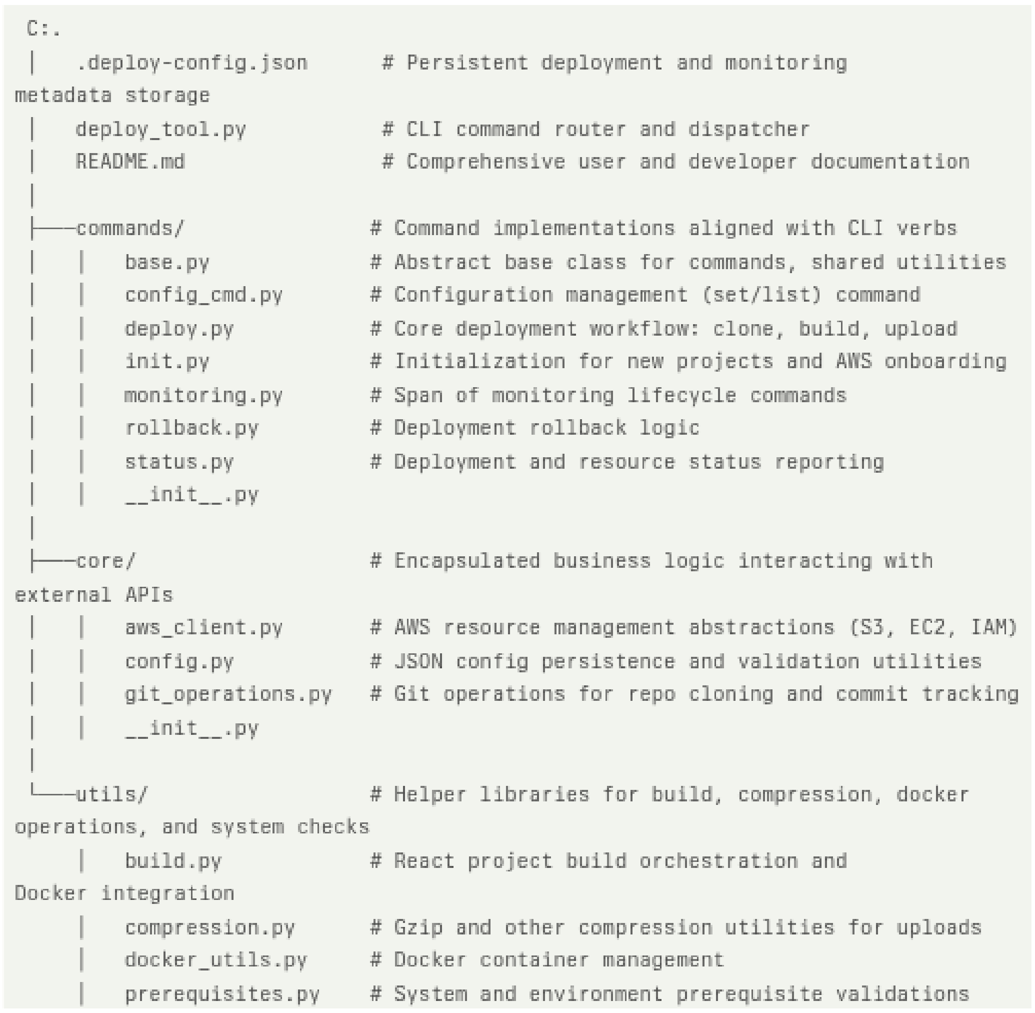
- Checking status



```
PS C:\Users\Minfy\Desktop\static-web-to-s3-deploy_tool-monitoring\deploy-tool> python deploy_tool.py config --list
Configuration:
{
  "project_name": "my-unique-meme-generator",
  "project_type": "vite",
  "github_url": "https://github.com/abhinav28minfy/actual-meme-gen.git",
  "github_owner": "abhinav28minfy",
  "github_repo": "actual-meme-gen",
  "github_branch": "master",
  "aws_region": "ap-south-1",
  "aws_profile": "abhinav",
  "environments": {
    "dev": {
      "bucket": "my-unique-meme-generator-dev"
    },
    "staging": {
      "bucket": "my-unique-meme-generator-staging"
    },
    "prod": {
      "bucket": "my-unique-meme-generator-prod"
    }
  },
  "build_command": "npm run build",
  "build_dir": "dist",
  "create_dockerfile": true,
  "create_health_check": true,
  "env_file_path": "C:\\Users\\Minfy\\Desktop\\.env",
  "monitoring": {
    "enabled": true,
    "instance_id": "i-0c5afe500a4117752",
    "public_ip": "15.206.75.80",
    "grafana_url": "http://15.206.75.80:3000",
    "prometheus_url": "http://15.206.75.80:9090",
    "targets": [
      "http://my-unique-meme-generator-dev.s3-website.ap-south-1.amazonaws.com"
    ],
    "created_at": "2025-07-23T16:57:38.491812",
    "compression": "gzip",
    "alerting": {
      "enabled": true,
```

5. High-Level Architecture & File System Overview

The tool's architecture emphasizes modularity, separation of concerns, and extensibility to accommodate complex deployment workflows while maintaining ease of use.



```
 C:.
 │   .deploy-config.json      # Persistent deployment and monitoring
metadata storage
 │   deploy_tool.py           # CLI command router and dispatcher
 │   README.md                # Comprehensive user and developer documentation
 │
 ├───commands/               # Command implementations aligned with CLI verbs
 │   │   base.py             # Abstract base class for commands, shared utilities
 │   │   config_cmd.py       # Configuration management (set/list) command
 │   │   deploy.py           # Core deployment workflow: clone, build, upload
 │   │   init.py             # Initialization for new projects and AWS onboarding
 │   │   monitoring.py       # Span of monitoring lifecycle commands
 │   │   rollback.py         # Deployment rollback logic
 │   │   status.py           # Deployment and resource status reporting
 │   │   __init__.py
 │
 ├───core/                   # Encapsulated business logic interacting with
external APIs
 │   │   aws_client.py       # AWS resource management abstractions (S3, EC2, IAM)
 │   │   config.py           # JSON config persistence and validation utilities
 │   │   git_operations.py   # Git operations for repo cloning and commit tracking
 │   │   __init__.py
 │
 └───utils/                  # Helper libraries for build, compression, docker
operations, and system checks
     │   build.py            # React project build orchestration and
Docker integration
     │   compression.py      # Gzip and other compression utilities for uploads
     │   docker_utils.py     # Docker container management
     │   prerequisites.py    # System and environment prerequisite validations
```

Architectural Layers Explained:

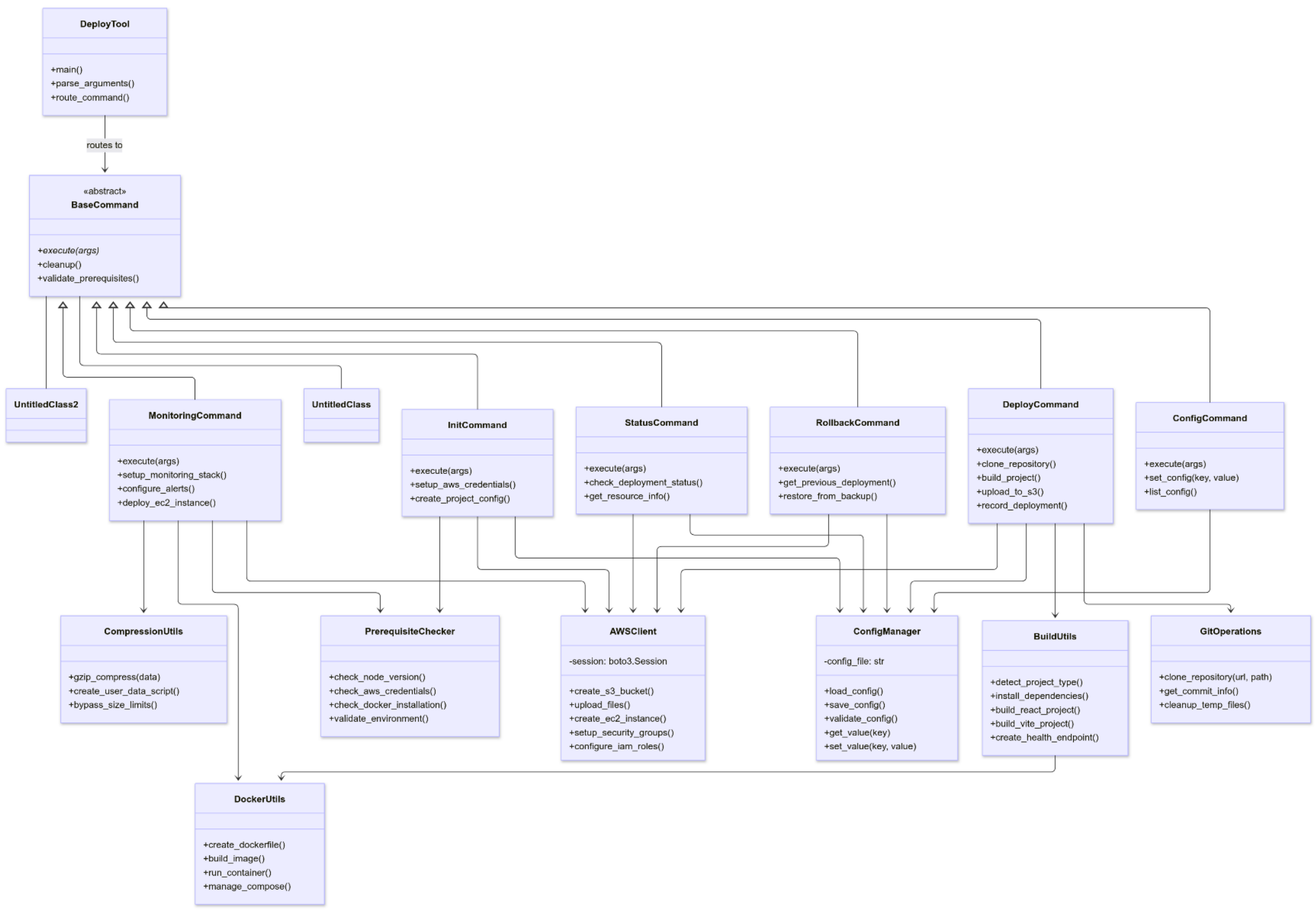
- **CLI Layer** (deploy_tool.py + commands/):
  Responsible for user interaction, parsing CLI commands, and invoking appropriate business logic modules. Each command encapsulates related operations, enabling clear code separation and easier testing.
- **Core Logic Layer** (core/):
  Serves as an abstraction boundary for external service interaction. AWS SDK interactions (S3, EC2, IAM) and Git operations are wrapped to provide battle-tested, consistent APIs used across commands. This promotes code reuse and simplification.
- **Utility Layer** (utils/):
  Houses auxiliary functionalities such as facilitating builds, compressing data payloads efficiently, and validating the presence of required tools and configurations on the host machine.

Data and State Management:

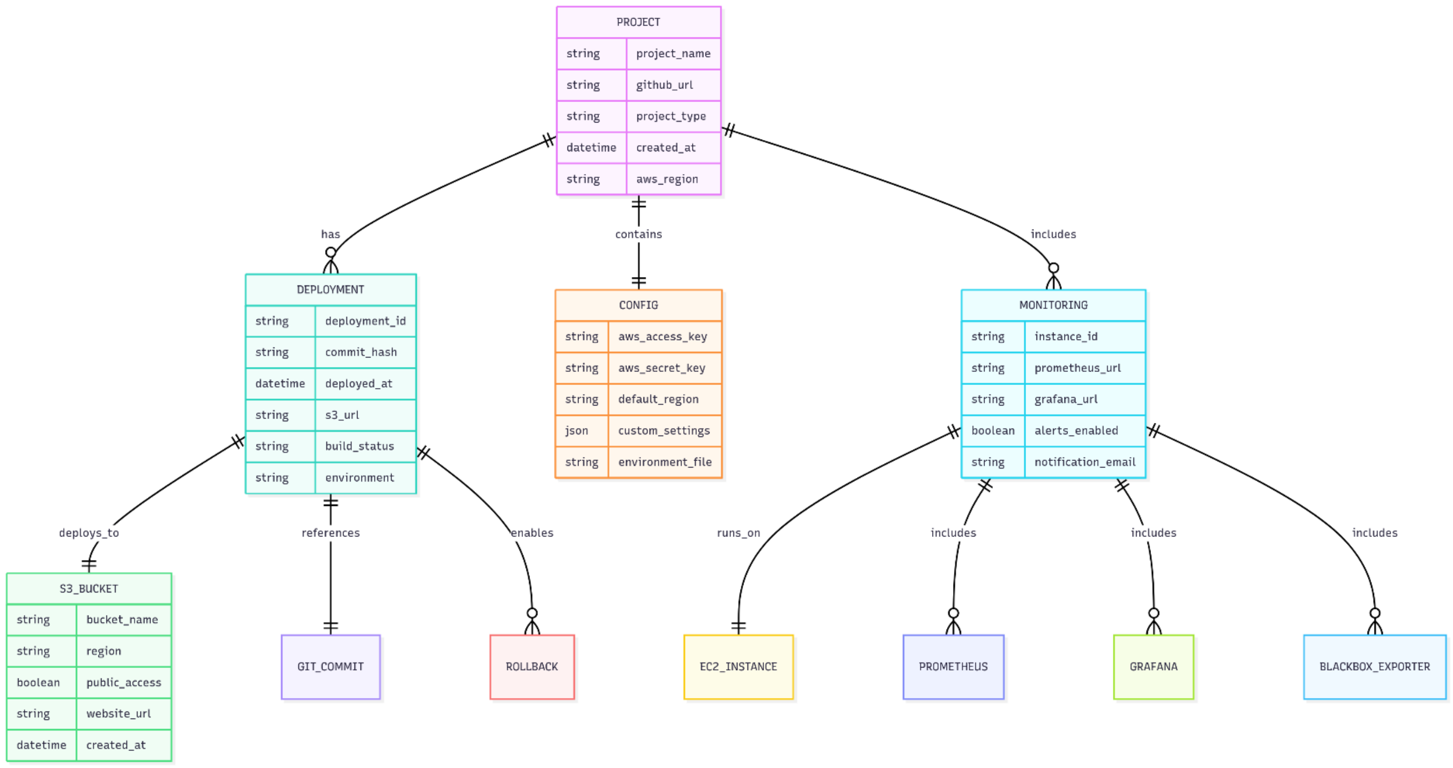
- The .deploy-config.json file acts as a canonical store of deployment metadata, configuration options, environment variables, and resource identifiers.
- It enables consistent state between runs and supports nested, environment-specific config via dot notation keys.

Infrastructure Flow:

1. **Initialization (init) creates AWS resources** such as versioned S3 buckets configured for static web hosting, IAM roles with least privileges, and prepares the deployment environment.
2. **Deployment (deploy) clones the repository**, detects the project type, builds code in isolated Docker containers, and uploads static artifacts to S3, updating deployment state and triggering health checks.
3. **Monitoring stack setup (monitoring init) provisions an EC2 instance** hosting Prometheus and Grafana, configures scraping and dashboards, and creates alert email contact points.
4. **Rollback, configuration, and status** commands interact with stored metadata and live AWS resource states to provide operational control and observability.



6. Low-Level Design



6.1 CLI Command Modules (commands/)

- init.py validates environment prerequisites and clones the user's GitHub repo. It detects React project type and auto-generates the initial .deploy-config.json with AWS bucket information and deployment environment details.
- deploy.py orchestrates cloning/pulling the repo, running builds for detected React types inside using utils.build, uploading static assets to configured S3 buckets, and registering deployment info for rollback capabilities.
- rollback.py reads prior deployments from configuration and switches the "current" deployment pointer back to a previously successful snapshot on S3.
- monitoring.py provisions an EC2 instance for Prometheus and Grafana, writes configuration files for alerting via Gmail SMTP, configures Grafana contact points, and manages monitoring lifecycle including status reporting and teardown.
- config_cmd.py supports dynamic configuration of deployment parameters, allowing hierarchical keys (e.g., monitoring.alert.email) and listing current config states.
- status.py aggregates and presents core deployment status, checking AWS resources, deployment history, and deployment health (such as successful health checks).
- base.py provides shared CLI parsing utilities, error handling, and logging facilities across all commands.

6.2 Core Libraries (core/)

- aws_client.py wraps bare boto3 calls, simplifying bucket creation and upload, EC2 instance spinning up (with tagging), and IAM role creation with strict permissions per least privilege principles.
- config.py reads and persists the .deploy-config.json file, validating new inputs, supporting dot notation keys for nested config updates, and providing safe transactional writes.
- git_operations.py abstracts Git CLI commands: repository cloning, branch checkout, latest commit hash retrieval, and cleanup of temporary repositories.

6.3 Utilities (utils/)

- build.py auto-detects React framework (CRA or Vite) through package.json scripts, installs dependencies, and runs production build commands inside Docker containers for build determinism.
- docker_utils.py manages spinning up Docker containers and executing commands/scripts within them while safely mounting volumes and cleaning containers post-build to maintain environment hygiene.
- compression.py applies gzip compression to built artifacts and monitoring data uploads, reducing transfer sizes and improving speeds.
- prerequisites.py guarantees all required tools (node, git, awscli, docker, python) are installed, and AWS credentials/config are set up correctly, guiding users with informative errors.

7. Help and Usage System

- The CLI uses Python's built-in argparse combined with a base command class to handle help text generation.
- Running deploy_tool.py -h or deploy_tool.py --help shows global usage and lists available commands.
- Each command supports --help showing command-specific options, descriptions, and examples.
- Help texts are maintained in docstrings and argparse metadata for synchronization.
- Example usage:

  bash
  ```
  python deploy_tool.py --help
  python deploy_tool.py deploy --help
  ```

8. Why This Design?

- Modular command classes allow clean separation of concerns, easy testing, and future enhancements.
- Core AWS and Git abstractions in core/ foster reusable API wrappers and facilitate maintenance.
- Using Python native CLI techniques avoid external heavy dependencies, ensuring cross-platform compatibility.
- Incorporation of industry-standard monitoring and alerting stack provides full production readiness.

9. Future Enhancements

- Support additional frontend frameworks: Next.js, Angular.
- Integrate with multi-cloud backends beyond AWS.
- Enhance CI/CD integration with webhook handling and pipeline triggers.
- Expand monitoring to support high-availability Prometheus and Grafana clusters.
- Add richer metrics and logging dashboards, including deployment metrics inside Grafana.

10. Summary

This CLI deployment tool embodies a comprehensive automation framework designed to democratize React frontend AWS deployments. By capitalizing on Python's scripting flexibility, it reliably builds projects in isolated environments, handling diverse React build pipelines with minimal user intervention.

The modular architecture, clearly divided into CLI commands, core logic, and utilities, fuels extensibility and maintainability while ensuring separation of concerns that facilitate testing and future feature additions.

By integrating monitoring tools and automated alerting, it offers observability essential for modern applications, packaged in a simple CLI interface. This balance of ease and power transforms complex AWS deployment pipelines into stress-free developer experiences.

Future expansions will extend framework support, enhance deployment strategies, and deepen cloud-provider integrations to maintain agility and relevance in evolving cloud ecosystems.