CloudTruth Integration Task Monitor

This Bash script monitors the status of a CloudTruth integration action task (such as a push or pull) until it completes, and provides actionable logging and output for DevOps, platform, and software engineers.

Features

- **Monitors CloudTruth integration tasks** (push or pull) for AWS, GitHub, Azure Key Vault, and more.
- **Polls the CloudTruth API** until the task completes (success or failure).
- **Logs all relevant task and step details** (including error codes and details) to a log file for troubleshooting.
- **Outputs concise status to the console** for CI/CD and automation use.
- **Handles authentication** via the `CLOUDTRUTH_API_KEY` environment variable.
- **Supports Azure Key Vault's unique API path**.

Usage

```
./monitor_task.sh PROVIDER INTEGRATION_TYPE INTEGRATION_PK ACTION_PK
```

Arguments

- `PROVIDER` – The integration provider (e.g., `aws`, `github`, `azure`)
- `INTEGRATION_TYPE` – The type of action (`push` or `pull`)
- `INTEGRATION_PK` – The unique ID (PK) of the integration
- `ACTION_PK` – The unique ID (PK) of the push or pull action

Options

- `-h`, `--help` – Show usage information and exit

Environment Variables

- `CLOUDTRUTH_API_KEY` – **Required.** Your CloudTruth API key.
- `MONITOR_TASK_LOG` – Optional. Path to log file (default: `/tmp/monitor_task.log`).

Example

Monitor an AWS push action:

```
CLOUDTRUTH_API_KEY=xxxx ./monitor_task.sh aws push 12e647a5-4d6f-4559-aabc-1b8ff4ec9bdc 960e
```

Monitor an Azure Key Vault pull action:

```
CLOUDTRUTH_API_KEY=xxxx ./monitor_task.sh azure pull 12345678-aaaa-bbbb-cccc-123456789abc 87
```

Example Use Cases in DevOps/SDLC

- **CI/CD Pipelines:** Use this script as a step in your pipeline to block further deployment until a CloudTruth integration task completes, ensuring configuration syncs or secrets pushes succeed before proceeding.
- **Automated Rollbacks:** Integrate with your automation to trigger rollbacks or alerts if a CloudTruth task fails, using the script's exit code and log file for diagnostics.
- **Observability & Auditing:** Pipe the log file to your observability platform or archive it for compliance, capturing all relevant error details for failed tasks and steps.

Output

- **Console:**
  - Shows task state transitions and only the IDs of failed steps (if any).
- **Log File:**
  - Contains detailed breadcrumbs for all task and step failures, including error codes and descriptions.

Requirements

- Bash (tested on macOS and Linux)
- `curl`
- `jq`

For more details, see the script's `usage` function or run: