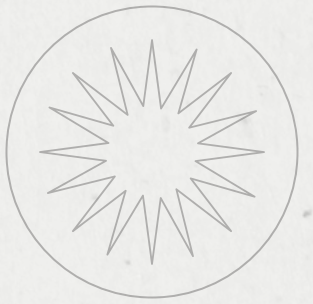
Automating Microservices Releases

MOHAMED AZARUDEEN M
SR. DEVOPS ENGINEER

Overview



tag_releases.sh is a Bash script designed to automate tagging and pushing releases for multiple Git repositories, specifically for microservices deployed on Google Cloud Run with tag-based triggers in Google Cloud Build. It uses a JSON file (e.g., **tags.json**) to map repositories to their respective tags, ensuring reliable and efficient releases.

Problems Solved

1. **Manual Effort:** Eliminates the need to manually tag 12+ repositories, reducing 50+ commands to a single script execution.
2. **Error Prevention**: Validates repository cleanliness, branch availability, and tag uniqueness, preventing broken builds or deployments.
3. **Consistency**: Ensures all repositories are on **main** or **master** and up-to-date before tagging, maintaining release integrity.
4. **Scalability**: Handles any number of repositories via a JSON file, perfect for large microservices projects.
5. **CI/CD Integration**: Seamlessly integrates with Google Cloud Build by pushing tags to trigger production deployments.

Time and Effort Savings

- **Manual Process:**
  - **Tasks**: Verify **tags.json**, navigate to 12 repositories, check cleanliness (***git status)***, checkout main/master (***git checkout***), pull updates (***git pull***), create tags (***git tag***), push tags (***git push***).
  - **Commands**: ~50–60 commands for 12 services.
  - **Time**: 10–20 minutes per release, plus 5–10 minutes per error (e.g., dirty repository, duplicate tag).
  - **Total**: 10–60 minutes per release cycle.
- **Script Process:**
  - **Command**: **./tag_releases.sh -p -f tags.json**
  - **Time**: ~10–30 seconds for 12 repositories, including validation, tagging, and pushing.
  - **Savings**: ~90–95% time reduction, saving 10–60 minutes per release, plus hours of error recovery.

Prerequisites

- **Bash 3.2+**: Compatible with macOS (tested on macOS with Bash 3.2).
- **Git**: Installed and configured with access to repositories.
- **jq**: Required to parse the JSON file (brew install jq on macOS).
- **Repositories**:
  - Stored in a local project folder (default: $HOME/projects, e.g., ~/projects/repo-1).
  - Must have a main branch (or master as fallback).
  - Must be initialized as Git repositories with remote access.
- **JSON File**: A file (e.g., **tags.json**) with repository-tag mappings, e.g.:

```
{
  "repo-1": "v1.0.1",
  "repo-2": "v2.0.0",
  "repo-3": "v3.1.0"
}
```

- **Google Cloud Build:** Configured with tag-based triggers (e.g., v1.3.1) for production deployments.

Full Script

You can view the script in the below link: (Inside Post#2 folder)

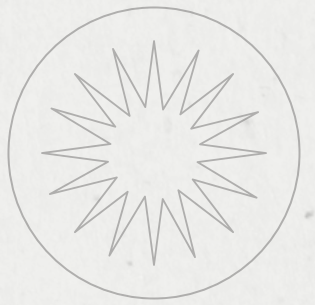
https://github.com/Mohamed-Azarudeen-M/my-linkedin-post.git

Script Explanation

The script is organized into logical functions, executed in a **main** function for clarity:

1. **parse_args:** Parses command-line options (**-p|--push** for pushing tags, **-f|--file** for the JSON file).
2. **validate_input**: Checks if the JSON file exists and **jq** is installed.
3. **is_repo_valid_and_clean**: Verifies the repository is a valid Git repo with no uncommitted changes.
4. **prepare_repo_branch**: Checks out **main** or **master** and pulls the latest changes.
5. **apply_and_push_tag**: Creates a tag and pushes it (if **-p** is used), checking for existing tags to avoid duplicates.
6. **main**: Orchestrates the workflow, processing each repository-tag pair from the JSON file.

How the Script Works



1. **Parse Arguments**: Reads **-p|--push** (to push tags) and **-f|--file <tag_file>** (e.g., **tags.json**).
2. **Validate Input**: Ensures the JSON file exists and **jq** is available.
3. **Process JSON File**:
    - Reads **tags.json** using jq to extract repository-tag pairs (e.g., **repo-1=v1.0.1**).
    - Skips invalid pairs (e.g., empty tags).
4. **For Each Repository**:
    - Validates the repository is clean and valid.
    - Checks out **main** or **master** and pulls updates.
    - Applies the tag (e.g., **v1.0.1**) and pushes it to trigger Google Cloud Build (if -p is used).
5. **Error Handling**: Skips problematic repositories (e.g., dirty, missing branch) with clear warnings, ensuring partial success.

Example Usage

JSON File (tags.json):

```
{
  "repo-1": "v1.0.1",
  "repo-2": "v2.0.0",
  "repo-3": "v3.1.0"
}
```

Command:

```
./tag_releases.sh -p -f tags.json
```

Output (all repositories clean, on main):

```
Push mode enabled, will push tags to remote repositories.
Processing repository: /Users/helix/projects/repo-1 with tag 'v1.0.1'
Checking out main branch for '/Users/helix/projects/repo-1'
Tagging 'v1.0.1' for '/Users/helix/projects/repo-1'
Pushing tag 'v1.0.1' for '/Users/helix/projects/repo-1'
Processing repository: /Users/helix/projects/repo-2 with tag 'v2.0.0'
Checking out main branch for '/Users/helix/projects/repo-2'
Tagging 'v2.0.0' for '/Users/helix/projects/repo-2'
Pushing tag 'v2.0.0' for '/Users/helix/projects/repo-2'
Processing repository: /Users/helix/projects/repo-3 with tag 'v3.1.0'
Checking out main branch for '/Users/helix/projects/repo-3'
Tagging 'v3.1.0' for '/Users/helix/projects/repo-3'
Pushing tag 'v3.1.0' for '/Users/helix/projects/repo-3'
```

Example Usage

Error Case (e.g., repo-2 is dirty):

```
Push mode enabled, will push tags to remote repositories.
Processing repository: /Users/helix/projects/repo-1 with tag 'v1.0.1'
Checking out main branch for '/Users/helix/projects/repo-1'
Tagging 'v1.0.1' for '/Users/helix/projects/repo-1'
Pushing tag 'v1.0.1' for '/Users/helix/projects/repo-1'
Processing repository: /Users/helix/projects/repo-2 with tag 'v2.0.0'
Error: '/Users/helix/projects/repo-2' is dirty, please clean and try
again
Skipping '/Users/helix/projects/repo-2' due to invalid or dirty state
Processing repository: /Users/helix/projects/repo-3 with tag 'v3.1.0'
Checking out main branch for '/Users/helix/projects/repo-3'
Tagging 'v3.1.0' for '/Users/helix/projects/repo-3'
Pushing tag 'v3.1.0' for '/Users/helix/projects/repo-3'
```

Recommendations for Use

- **Validate JSON Syntax:**

  Add to validate_input:

```
if ! jq . "$TAG_FILE" >/dev/null 2>&1; then
  echo "Error: Invalid JSON syntax in '$TAG_FILE'"
  exit 1
fi
```

- **Enforce Semantic Versioning:**

  Add to the loop:

```
if ! echo "$tag" | grep -qE '^[vV][0-9]+\.[0-9]+\.[0-9]+$';
then
  echo "Warning: Tag '$tag' for '$repo' does not match semantic
versioning, skipping"
  continue
fi
```

- **Log Output:**

  Save output for auditing:

```
./tag_releases.sh -p -f tags.json | tee release-20250712-
2016.log
```

- **Custom Branches:**

  Add a -b|--branch option to support branches other than main/master.

THANK YOU