indium
Make Technology Work

Azure

Azure DevOps - GitHub Merge Automation

Digital Services

Success Story

About Client

The client is a leading mobile technology service provider helping taxi companies and ride-hailing service providers get instant access to cutting-edge technology solutions for the riders, drivers, dispatch coordinators, and fleet managers.

Project Overview

Indium developed a solution which helped the client to automate the merging of various code branches, through shell scripts. The utility auto-identifies the work items in Azure DevOps, based on the QA completion status and merges the code changes into the production branch in Git repositories.

Business Requirement

- Develop an automatic solution to merge all the code commits
- Utilities to retrieve the work items in Quality Assurance stage from Azure DevOps.
- Ability to merge the code associated with user stories/ bugs with the production code branch
- Visibility for QA Engineers to check functionality in the UAT environment

Solution

Designed and developed a solution to merge all the QA certified changes into UAT environment:

- Utility to retrieve the work items (User stories, bugs) in Quality Assurance stage from Azure DevOps, using Azure CLI
- Utility to identify all the code commits/ merged pull requests made by developers in Develop branch, using Git CLI
- Utility to merge all the commits in the pull requests into production branch, using Git CLI and GitHub CLI

Business Impact

- The solution helped to achieve near 100% accuracy in merging all Utility requests, without any human error
- 90% reduction in time and effort, compared to the previous manual process
- Developers and QA engineers got a clear visibility of the functionalities

Business
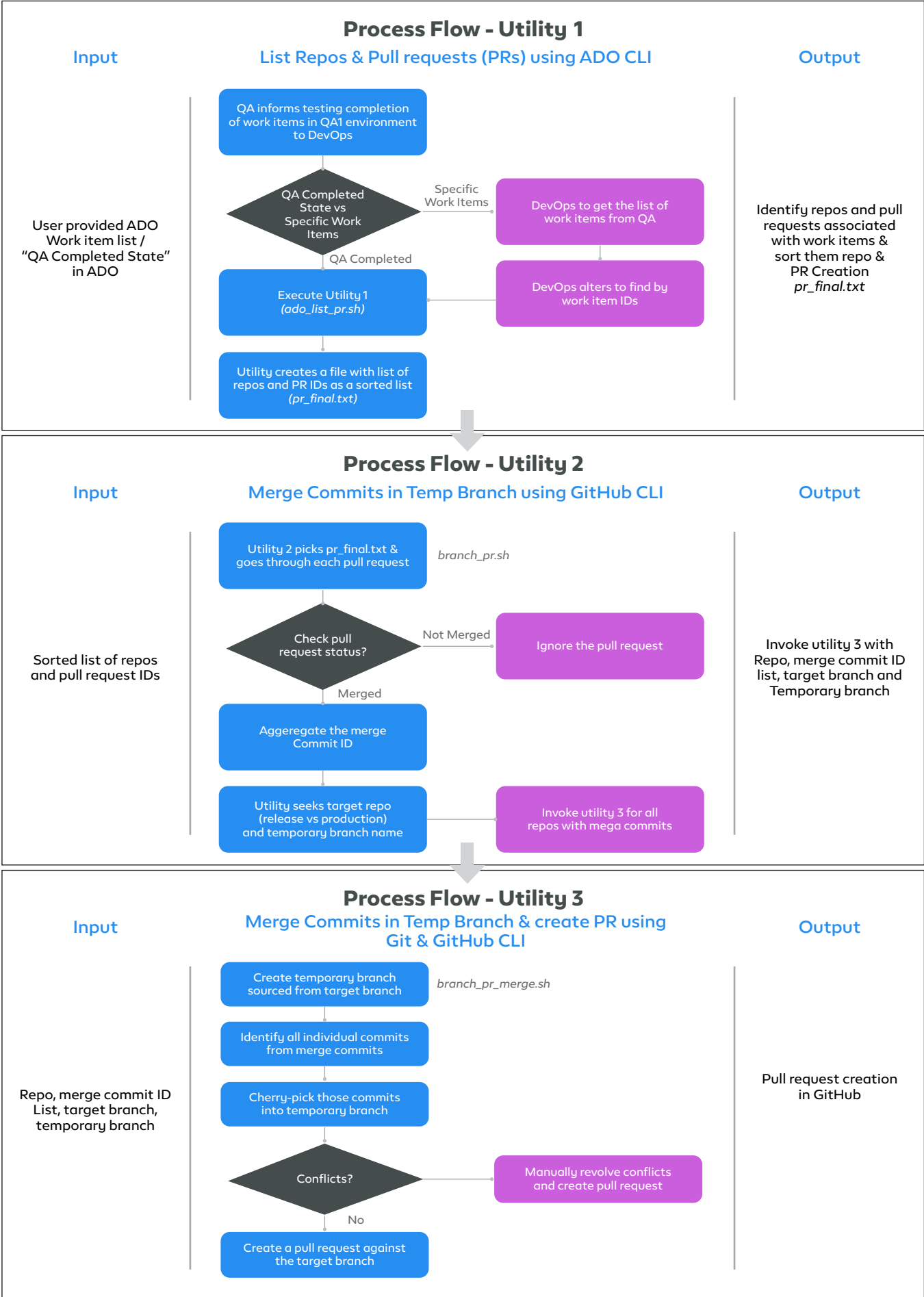Transport

Domain
DevOps

Tools
Azure CLI, Git CLI, GitHub CLI, Shell Script

Key Highlights

- Near 100% accuracy in merging all Utility requests
- 90% reduction in time and effort by automating the process

Architecture

Process Flow - Utility 1
List Repos & Pull requests (PRs) using ADO CLI

Input

User provided ADO
Work item list /
"QA Completed State"
in ADO

QA informs testing completion
of work items in QA1 environment
to DevOps

QA Completed
State vs
Specific Work
Items

Specific Work Items → DevOps to get the list of work items from QA

DevOps alters to find by work item IDs

QA Completed

Execute Utility 1
(ado_list_pr.sh)

Utility creates a file with list of
repos and PR IDs as a sorted list
(pr_final.txt)

Output

Identify repos and pull
requests associated
with work items &
sort them repo &
PR Creation
pr_final.txt

Process Flow - Utility 2
Merge Commits in Temp Branch using GitHub CLI

Input

Sorted list of repos
and pull request IDs

Utility 2 picks pr_final.txt &
goes through each pull request

branch_pr.sh

Check pull
request status?

Not Merged → Ignore the pull request

Merged

Aggeregate the merge
Commit ID

Utility seeks target repo
(release vs production)
and temporary branch name

Invoke utility 3 for all
repos with mega commits

Output

Invoke utility 3 with
Repo, merge commit ID
list, target branch and
Temporary branch

Process Flow - Utility 3
Merge Commits in Temp Branch & create PR using Git & GitHub CLI

Input

Repo, merge commit ID
List, target branch,
temporary branch

Create temporary branch
sourced from target branch

branch_pr_merge.sh

Identify all individual commits
from merge commits

Cherry-pick those commits
into temporary branch

Conflicts?

→ Manually revolve conflicts
and create pull request

No

Create a pull request against
the target branch

Output

Pull request creation
in GitHub

INDIA

Chennai | Bengaluru | Mumbai
Toll-free: 1800-123-1191

USA

Cupertino | Princeton
Toll-free: 1 888 207 5969

UK

London

SINGAPORE

+65 9630 7959

MALAYSIA

Kuala Lumpur
+60 (3) 2298 8465

General Inquiries
info@indiumsoftware.com

Sales Inquiries
sales@indiumsoftware.com