Everything you always wanted to know about CI/CD pipelines

Elena Kubantseva & Pablo Carle

October 31st, 2023

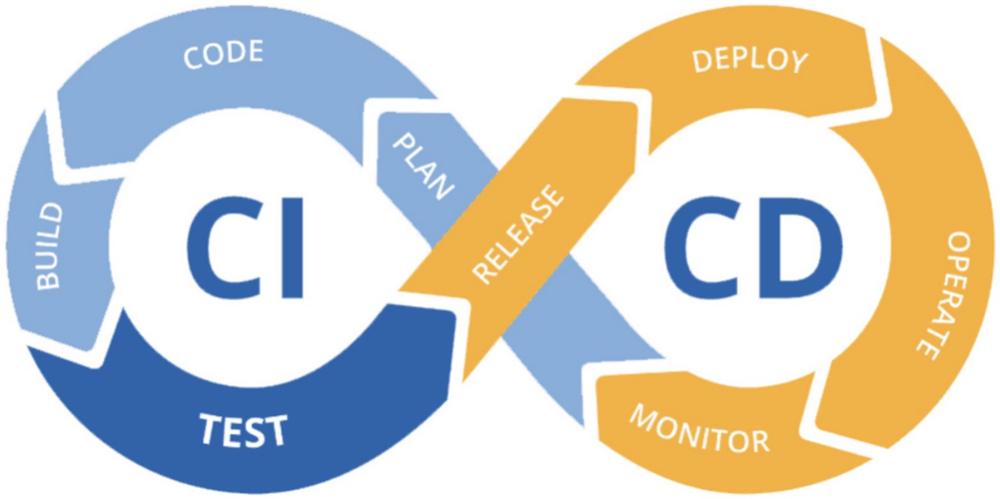
Agenda

- What is CI/CD?
- Zowe CLI: a modern tool to work with z/OS
- Challenges with mainframe application development
- Design a pipeline
- Best practices
- Resources

BROADCOM® SOFTWARE

What is CI/CD?

- Continuous Integration: integrate your work early and often
- Continuous Delivery: have your software ready for delivery at any time



Zowe CLI

Zowe CLI

Enables remote access to mainframe products & services*

Capabilities:
- Interact with mainframe files
- Submit jobs
- Issue TSO and z/OS console commands
- Integrate z/OS actions into scripts
- Produce responses as JSON

*Majority of the cases requires z/OSMF

Plugins:
- Endevor
- Sysview
- OPS/MVS
- Other plugins

Installation guide for Zowe CLI

```
root@SSH:~/gh/zowe/zowe-cli# 
```

Zowe CLI

Profiles:

- z/OSMF:

```
zowe profiles create zosmf-profile PROFILE_NAME \
   --host HOST_NAME \
   --port PORT \
   --ow \
   --reject-unauthorized false
```

- SSH:

```
zowe profiles create ssh-profile PROFILE_NAME \
   --host HOST_NAME \
   --port PORT \
   --overwrite
```

- Endevor:

```
zowe profiles create endevor PROFILE_NAME \
   --host HOST_NAME \
   --port PORT \
   --prot https \
   --base-path EndevorService/api/v2 \
   --reject-unauthorized false

zowe profiles create endevor-location-profile PROFILE_NAME \
   --env ENVIRONMENT \
   --sys SYSTEM \
   --sub SUB_SYSTEM \
   --sn 1 \
   --com 'sample comment' \
   --cci 'CCID'
```

Secure usage of the credentials:

```
export ZOWE_OPT_USER=USER
export ZOWE_OPT_PASSWORD=PASSWORD
```

Challenges with Mainframe application development

Differences with typical workflows

- Mainframe shared resources
  - ports, zFS, proclib
- Usually a long process to deploy and test the application.
- Installation-specific testing
  - ESM-specific behaviour

Design a pipeline

Design a pipeline

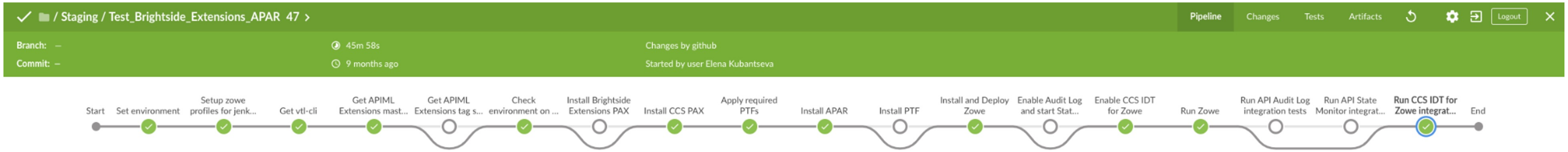
Evaluating automation tools, Jenkins, Ansible, GitHub actions

Some of the currently available tooling include:

- Jenkins
  - Imperative / Declarative
- Github Actions
  - Reusable modules
  - Github specific
- Ansible
  - Playbooks

Design a pipeline

An example of a Jenkins Pipeline


✓ / Staging / Test_Brightside_Extensions_APAR 47 >

Branch: –
Commit: –
45m 58s
9 months ago
Changes by github
Started by user Elena Kubantseva

Start → Set environment → Setup zowe profiles for jenk... → Get vtl-cli → Get APIML Extensions mast... → Get APIML Extensions tag s... → Check environment on ... → Install Brightside Extensions PAX → Install CCS PAX → Apply required PTFs → Install APAR → Install PTF → Install and Deploy Zowe → Enable Audit Log and start Stat... → Enable CCS IDT for Zowe → Run Zowe → Run API Audit Log integration tests → Run API State Monitor integrat... → Run CCS IDT for Zowe integrat... → End

Design a pipeline

An example pipeline in Jenkins

- Pipeline configuration and setup

```
pipeline {
  agent {
    docker {
      . . .
    }
  }

  environment { . . . }

  options { . . . }

  parameters { . . . }
```

- Stages

```
stages {
  stage("Install dependencies and initial configuration") {
    . . .
  }

  stage("Build application") { . . . }
  stage("Deploy") { . . . }
  stage("Run tests") { . . . }

  post { . . . }
```

BROADCOM® SOFTWARE

Design a pipeline

Build the application

- Platform-independent
- Uploading the files to z/OS
- Running a command on z/OS
  - i.e. make
- Download build artifacts

Upload

```
zowe zos-files upload dir-to-uss "local_dir"
"/a/ibmuser/my_dir"
```

Run a command

```
zowe zos-ssh issue command "make" --cwd
/a/ibmuser/my_dir
```

Download artifacts

```
zowe zos-files download uss-file
"/a/ibmuser/MyJava.class" -b -f "java/MyJava.class"
```

BROADCOM® SOFTWARE

Design a pipeline

Deploying the application to z/OS

- Make sure you have enough space for the installation
- Upload the artifacts
- Make sure to have all permissions
- Unzip/unpax artifacts (optional)
- Prepare the configuration files
- Upload datasets for running the application (such as JOB / STC definitions)

Design a pipeline

Deploying the application to z/OS (Example with SMP/E installation)

1. Create ZFS with a size of 2.5xPAX minimum:

```
zowe zos-files create zos-file-system <fileSystemName>
zowe zos-files mount file-system <fileSystemName> <mountPoint>
```

1. Upload PAX into the previously created ZFS:

```
zowe zos-files upload file-to-uss <inputfile> <USSFileName>
```

1. Unpax and then unzip (GIMUNZIP) your PAX and it’s content:

```
zowe zos-jobs submit local-file <localFile>
```

1. Create duplicates of APPLY and ACCEPT jobs, but without CHECK.
2. Replace parameterized values in the macro:

```
ISREDIT CHANGE ALL GLOBALHLQ     ${smpe.hlq}
```

1. Upload all JCLs to the dataset:

```
zowe zos-files upload dir-to-pds <inputdir> <dataSetName>
```

1. Submit JCL, which will run REXX script to apply macro.
2. Submit all jobs for PAX installation in proper order.

   in cycle “for”: `zowe zos-jobs submit data-set <dataset>`

Design a pipeline

Running your application on z/OS

- Submit JOB, start STC or run application in USS?
  - `zowe zos-jobs submit data-set <dataset>`
  - `zowe zos-console issue command "S <startedTask>"`
  - `zowe uss issue ssh <command>`
- Run integration tests
- Stop application
  - `zowe zos-jobs cancel job "<jobId>"`
  - `zowe zos-console issue command "P <startedTask>"`
- Clean up
  - `zowe zos-files unmount file-system <fileSystemName>`
  - `zowe zos-files delete zos-file-system <fileSystemName>`
  - `zowe zos-files delete data-set <dataSetName>`
- Troubleshoot problems
  - Zowe Explorer is the best tool to quickly check what is wrong with your JOB/STC/configuration and fix it

Design a pipeline

Package and distribute

- Package (with Endevor):
  - Check sandbox doesn't exist
  - Create sandbox or use existing one
  - Add / update / delete / move elements
  - Create / reset package
- Publish
  - Your z/OS distribution system
  - Artifactory

Find sandbox
```
zowe endevor list subsystems <sandboxName> …
```

Create sandbox
```
zowe endevor add element <sandboxName> -g …
```

Add/update element
```
zowe endevor update element <elementName> …
```

Delete element
```
zowe endevor delete element <elementName> …
```

Move element
```
zowe endevor move element <elementName> …
```

Create package
```
zowe endevor create package <packageName> …
```

Reset package
```
zowe endevor reset package <packageName> …
```

Design a pipeline

Metrics, reports and notifications

- Notify if the pipeline fails

```
  i.e. emailext plugin / slack notifications
```

- Save test reports

```
archiveArtifacts
  artifacts: '**/build/test-results/**',
  allowEmptyArchive: true,
  onlyIfSuccessful: true
```

- Save logs that can be lost

```
zowe jobs download output <Job ID>
```

Best Practices

- Single steps for multiple parts
- Scripts in the steps
- Logging
- Containerization (e.g. Docker, IBM zCX and zD&T)
- Parallel execution of steps
    - Place the lock on resources
- Clean up as a post action
- Use “replay” to troubleshoot pipeline

Resources

- Zowe Docs: https://docs.zowe.org/
- Zowe Install Packaging: https://github.com/zowe/zowe-install-packaging
  - Zowe CI/CD
  - Github actions / Ansible examples
- Broadcom Ansible Playbooks: https://broadcommfd.github.io/broadcom-ansible-collections/

Q&A

BROADCOM SOFTWARE

Thank you

BROADCOM®
SOFTWARE