Semantic Versioning:

- What is Semantic Versioning?

- A versioning scheme: MAJOR.MINOR.PATCH[-QUALIFIER]

- Enables traceability, clarity, and automation in CI/CD pipelines

- Why is it important?

- Uniquely identifies each build/artifact

- Prevents conflicts and confusion in artifact repositories

Best Practice Approach with branching strategy:

|  |  |  |
| --- | --- | --- |
| **Branch** | **Pattern** | **Example** |
| main/master | X.Y.Z | 1.2.3 |
| develop | X.Y.Z-SNAPSHOT | 1.2.3-SNAPSHOT |
| feature/\* | X.Y.Z-feature-branch-BUILDID | 1.2.3-feature-ABC-42 |
| hotfix/\* | X.Y.Z-hotfix-branch-BUILDID | 1.2.4-hotfix-bug-17 |
| PR builds | X.Y.Z-pr-PRID-SHA | 1.2.3-pr-7-abcd123 |

Pipeline Logic:

- Compute version dynamically in pipeline using Bash:

- Detect branch type

- Compose version string accordingly

a. Set Variables:

yaml

variables:

baseVersion: '1.2.3'

b. Bash Script to Compute Version:

yaml

- script: |

echo "##[group]Calculate Build Version"

branchName=$(echo "$(Build.SourceBranchName)" | tr '/' '-' | tr '\_' '-')

shortSha=$(echo "$(Build.SourceVersion)" | cut -c1-7)

version="$(baseVersion)"

if [[ "$(Build.SourceBranch)" == "refs/heads/main" || "$(Build.SourceBranch)" == "refs/heads/master" ]]; then

buildVersion="$version"

elif [[ "$(Build.SourceBranch)" == "refs/heads/develop" ]]; then

buildVersion="$version-SNAPSHOT"

elif [[ "$(Build.SourceBranch)" == refs/heads/feature/\* ]]; then

buildVersion="$version-feature-$branchName-$(Build.BuildId)"

elif [[ "$(Build.SourceBranch)" == refs/heads/hotfix/\* ]]; then

buildVersion="$version-hotfix-$branchName-$(Build.BuildId)"

elif [[ "$(Build.SourceBranch)" == refs/pull/\* ]]; then

buildVersion="$version-pr-$(System.PullRequest.PullRequestId)-$shortSha"

else

buildVersion="$version-unstable-$branchName-$(Build.BuildId)"

fi

echo "Computed Version: $buildVersion"

echo "##vso[task.setvariable variable=BuildVersion;isOutput=true]$buildVersion"

echo "##[endgroup]"

name: SetVersion

displayName: 'Calculate Semantic Version'

POM Setup & Sample:

- POM uses `${revision}` property:

xml

<version>${revision}</version>

<properties>

<revision>1.2.3</revision>

</properties>

===========================================================================================================

Current Implementation

- Release & Snapshot versioning based on pipeline parameter

- Strategy:

- main + BuildType=release → X.Y.Z (e.g., 1.3.1)

- any + BuildType=snapshot → X.Y.Z-SNAPSHOT (e.g., 1.3.1-SNAPSHOT)

- User selects BuildType at queue time

trigger:

branches:

include:

- main

parameters:

- name: BuildType

type: string

displayName: "Select Build Type (snapshot or release)"

default: 'release'

values:

- snapshot

- release

pool:

vmImage: ubuntu-latest

stages:

- stage: BuildDataPlatform

displayName: 'Build Data Platform'

jobs:

- job: Build

displayName: 'Build'

steps:

- checkout: self

# Extract base version from pom.xml <revision>

- script: |

echo "##[group]Extract and Set Semantic Version"

BASE\_VERSION=$(xmllint --xpath "/\*[local-name()='project']/\*[local-name()='properties']/\*[local-name()='revision']/text()" pom.xml)

if [[ -z "$BASE\_VERSION" ]]; then

echo "ERROR: Could not extract <revision> from pom.xml"

exit 1

fi

if [[ '${{ parameters.BuildType }}' == 'release' ]]; then

VERSION="$BASE\_VERSION"

else

VERSION="$BASE\_VERSION-SNAPSHOT"

fi

echo "##vso[task.setvariable variable=revision]$VERSION"

echo "Build version set to $VERSION"

echo "##[endgroup]"

displayName: 'Compute Semantic Version'

# Common Build step (pass revision)

- template: MasterTemplates/java/Java\_Build.yml@DevopsRepo

parameters:

pomfilepath: 'pom.xml'

javaversion: '17'

mavengoals: 'clean package -Drevision=$(revision)'

JfrogTargetResolveSnapshotRepo: 'mstar-office-maven-virtual'

JfrogTargetResolveReleaseRepo: 'mstar-office-maven-virtual'

# Conditional Publish step for SNAPSHOT

- ${{ if eq(parameters.BuildType, 'snapshot') }}:

- template: MasterTemplates/JFrogUpload/java-package-publish.yml@DevopsRepo

parameters:

pomfilepath: 'pom.xml'

javaversion: '17'

mavengoals: 'deploy -Drevision=$(revision)'

JfrogTargetResolveSnapshotRepo: 'mstar-office-maven-virtual'

JfrogTargetDeploySnapshotRepo: 'mstar-office-maven-libs-snapshots-local'

# Conditional Publish step for RELEASE

- ${{ if eq(parameters.BuildType, 'release') }}:

- template: MasterTemplates/JFrogUpload/java-package-publish.yml@DevopsRepo

parameters:

pomfilepath: 'pom.xml'

javaversion: '17'

mavengoals: 'deploy -Drevision=$(revision)'

JfrogTargetResolveReleaseRepo: 'mstar-office-maven-virtual'

JfrogTargetDeployReleaseRepo: 'mstar-office-maven-libs-releases-local'

Sample Pom.xml file:

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>com.cat.catos</groupId>

<artifactId>catos-schema-parent</artifactId>

<version>1.23.0-REL.1</version>

</parent>

<groupId>com.cat.minestar</groupId>

<artifactId>minestar-schema-data-platform</artifactId>

<packaging>jar</packaging>

<version>${revision}</version>

<name>MineStar Schema Data Platform</name>

<description>

Schemas and generated Java classes for initial integration of legacy MineStar with the CatOS Data Platform.

</description>

<properties>

<revision>1.3.1</revision>

<catos-schema-common.version>1.6.0-REL.1</catos-schema-common.version>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>minestar.platform.protobuf</groupId>

<artifactId>minestar-core-protobuf</artifactId>

</dependency>

<dependency>

<groupId>com.cat.catos</groupId>

<artifactId>catos-library-common</artifactId>

</dependency>

<dependency>

<groupId>com.cat.catos</groupId>

<artifactId>catos-schema-common</artifactId>

<version>${catos-schema-common.version}</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>minestar.platform.protobuf</groupId>

<artifactId>minestar-core-protobuf-codegenerator</artifactId>

<dependencies>

<dependency>

<groupId>com.cat.catos</groupId>

<artifactId>catos-schema-common</artifactId>

<version>${catos-schema-common.version}</version>

</dependency>

</dependencies>

<executions>

<execution>

<id>generate-sources</id>

<phase>generate-sources</phase>

<configuration>

<addSwaggerAnnotations>true</addSwaggerAnnotations>

<protoFiles>

<protoFile>

${project.basedir}/src/main/resources/com/cat/minestar/dataplatform/telemetry/model/telemetry.proto

</protoFile>

<protoFile>

${project.basedir}/src/main/resources/com/cat/minestar/dataplatform/machine/model/machine.proto

</protoFile>

<protoFile>

${project.basedir}/src/main/resources/com/cat/minestar/dataplatform/activity/model/event.proto

</protoFile>

</protoFiles>

<generatedSourceDirectory>${project.build.directory}/generated-sources

</generatedSourceDirectory>

<generatedTestSourceDirectory>${project.build.directory}/generated-test-sources

</generatedTestSourceDirectory>

</configuration>

<goals>

<goal>generate-sources</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

</project>

Current Implementation - Artifact Table

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | **Branch** | **BuildType** | **Artifact Version** | **Artifactory Repo** |
| Release | main | release | 1.3.1 | releases-local |
| Snapshot | any | snapshot | 1.3.1-SNAPSHOT | snapshots-local |