EDGE X FOUNDRY™

EdgeX Semver Explore

Thursday March 18, 2021

EdgeX Semver Explore: Background

History

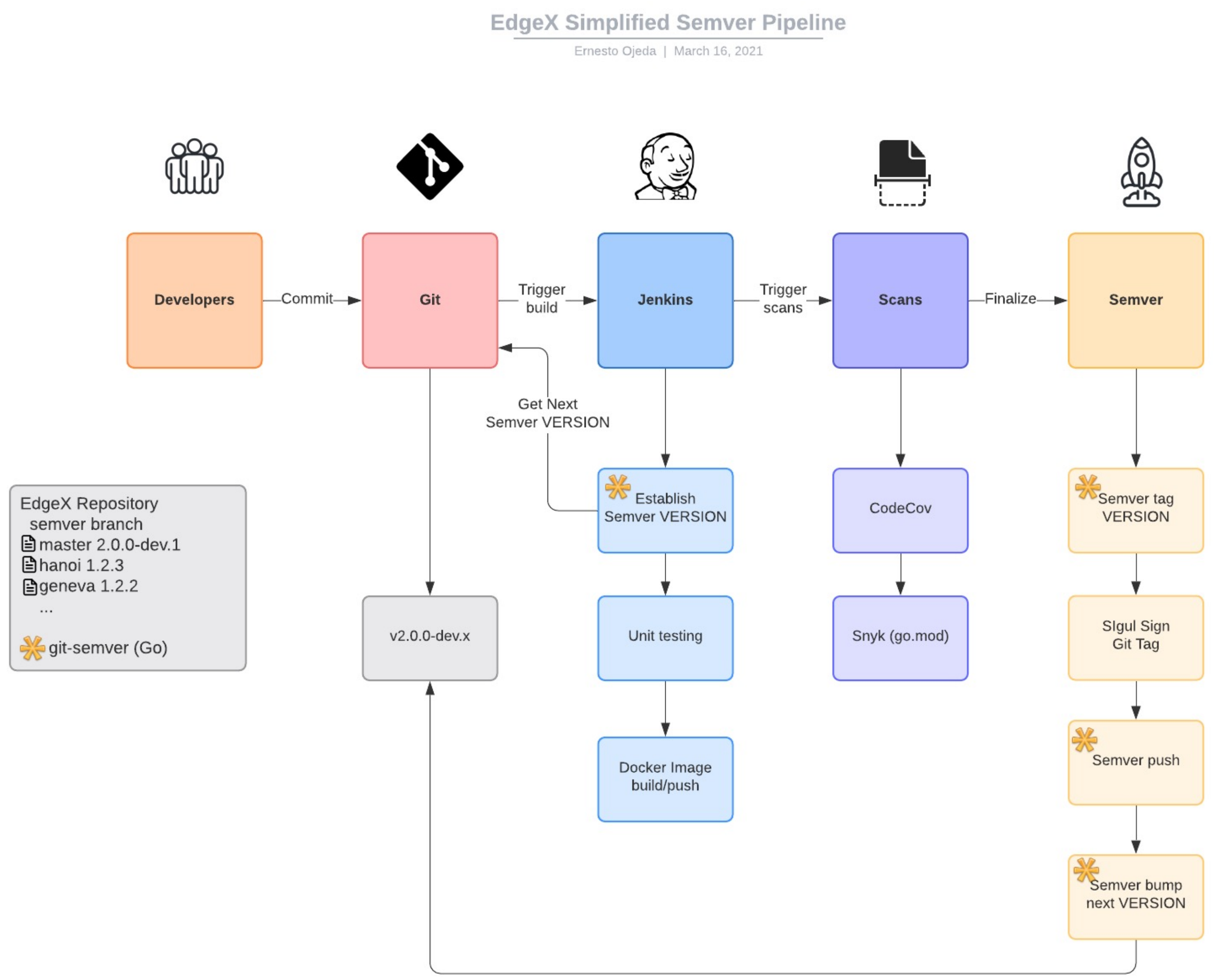
https://github.com/edgexfoundry/git-semver

- Established March 2019
- Written in Golang
- CLI tool that manages semver versions across multiple edgex repositories

Main motivation driving explore

- Tech debt
- Aligning to other more widely used “industry” tools
- Potential feature gap with LTS releases

High Level Semver Architecture



EdgeX Semver Explore: git-semver

Current Status

Used to manage semver version across 20+ repos

Semver version maintained in separate git branch

Gaps

No active development

Maintainability Issues

- Basic unit test code coverage, need refactor to get better code coverage
- No easy way to functionally test when fixing CVE
- Written in Go but the DevOps team not well versed in Go. Is this something the larger EdgeX community would want to maintain longer term?

EdgeX Semver Explore: Release Process

Current Status

New process created over the past three releases

Centrally managed in cd-management repo, YAML based

Release process Jenkins heavy, but well unit tested

Gaps

LTS release process may become more complex, we may require further process changes/features

Misc bugs with certain edge cases

Long term maintainability

EDGEXFOUNDRY™

EdgeX Semver Explore: Path Foward

Where do we go from here?

- Are there a more commonly used tools?
- Are we following the right paradigm with semver? i.e. separate branch holding semver version, pre-release bumping on every merge to master vs only bump when conventional commit matches specific noun
- What do other open-source projects do for releases?
- Can we decentralize releases down to the repo level?
- Ultimately can we remove people from the equation?

EdgeX Semver Explore: Path Foward

Semver/Release Automation Ecosystem

- Automatic semver versioning/bumping based on conventional commits
- Auto-generated CHANGELOG via conventional commits
- Auto-generated Release Notes with conventional commits
- Plugin based architecture allowing for more extensibility
- Decentralized, each repo is released on its own
- Pre-release version would still be supported

Is this the right approach?

- Would require a fairly heavy lift to switch
- Probably will require process changes with branching
- Aligning to well maintained industry tools will bring down our tech debt