Good Commitizenship

RVASDUG

Photo Credit:
Baskervill

Who is Kinsale?

- A Technology Driven Excess and Surplus (E&S) Insurance Carrier
- Founded 2009
- KNSL, 2016 (NYSE)
- Headquartered in RVA

We're Hiring! 😀

Welcome RVASDUG

- Agenda
  - Introductions
  - Funny Because It’s True
  - How Do We Fix This?
    - Git Hooks
    - Some Obvious, but Important Notes
  - Tooling
    - Husky
    - Conventional Commit Specification
      - The Standard and its variants
    - Commitlint (Commitizen, ...)
  - Release and Version Automation
  - Demo

Introductions

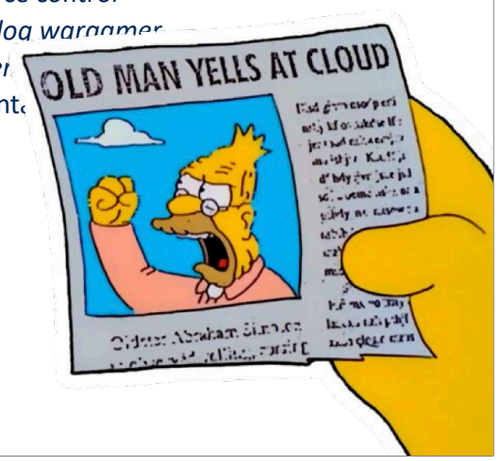
Photo Credit: Baskervill

Introductions

- Timothy Stone
  - Some call me... *Tim*
  - Senior Solution Architect, KNSL
  - *...a father, blogger, OSS committer, source control nerd, Java certified web developer, analog wargamer, home brewer, and lifelong TTRPG gamer.*
  - Find me on LinkedIn, GitLab, GitHub, Untappd

Introductions

- Timothy Stone
  - Some call me... *Tim*
  - Senior Solution Architect, KNSL
  - *...a father, blogger, OSS committer, source control nerd, Java certified web developer, analog wargamer, home brewer, and lifelong TTRPG gamer*
  - Find me on LinkedIn, GitLab, GitHub, Unt
  - I'm opinionated



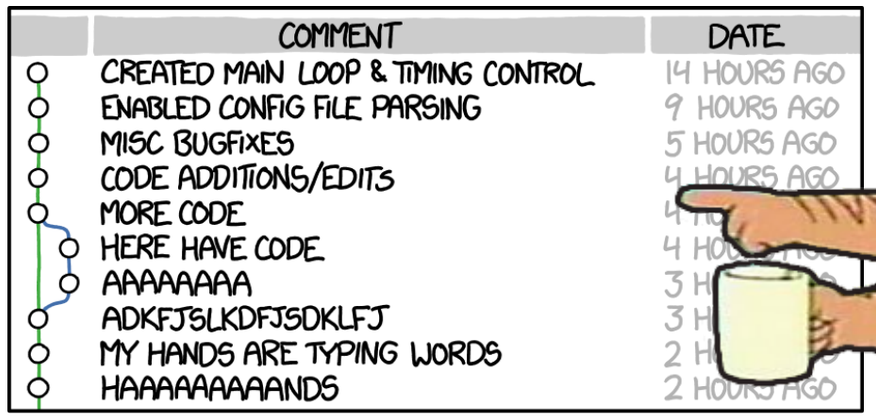
Introductions

- Timothy Stone
  - Some call me... *Tim*
  - Senior Solution Architect, KNSL
  - *...a father, blogger, OSS committer, source control nerd, Java certified web developer, analog wargamer, home brewer, and lifelong TTRPG gamer.*
  - Find me on LinkedIn, GitLab, GitHub, Untappd
  - I’m opinionated
  - I will be using the command line, a lot.
    - Z-Shell + Oh My ZSH + git plugin
    - I’ll try to vocalize actions and expand aliases
  - All code available will be available on *GitLab*

It’s Funny Because It’s True

https://xkcd.com/1296

It’s Funny Because It’s True



https://xkcd.com/1296

How Do We Fix This?

- Git Hooks

  "Like many other Version Control Systems, Git has a way to fire off custom scripts when certain important actions occur...

- Possibly obvious, but an important note
  - Client-side hooks are not shared
  - `.git/hooks` exists only on the developer's clone...
  - `.git` *ignored in the binary and* cannot be committed
    - `git check-ignore -v .git`

Altering the default behaviors of tooling should only be undertaken when that tooling provides for the expectation and more importantly documents the behavior. Changing the default exclusion of .git/hooks is not expected by most.

Intentional tooling is more visible thus approachable.

Tooling

Photo Credit:
Baskervill

Introducing Husky

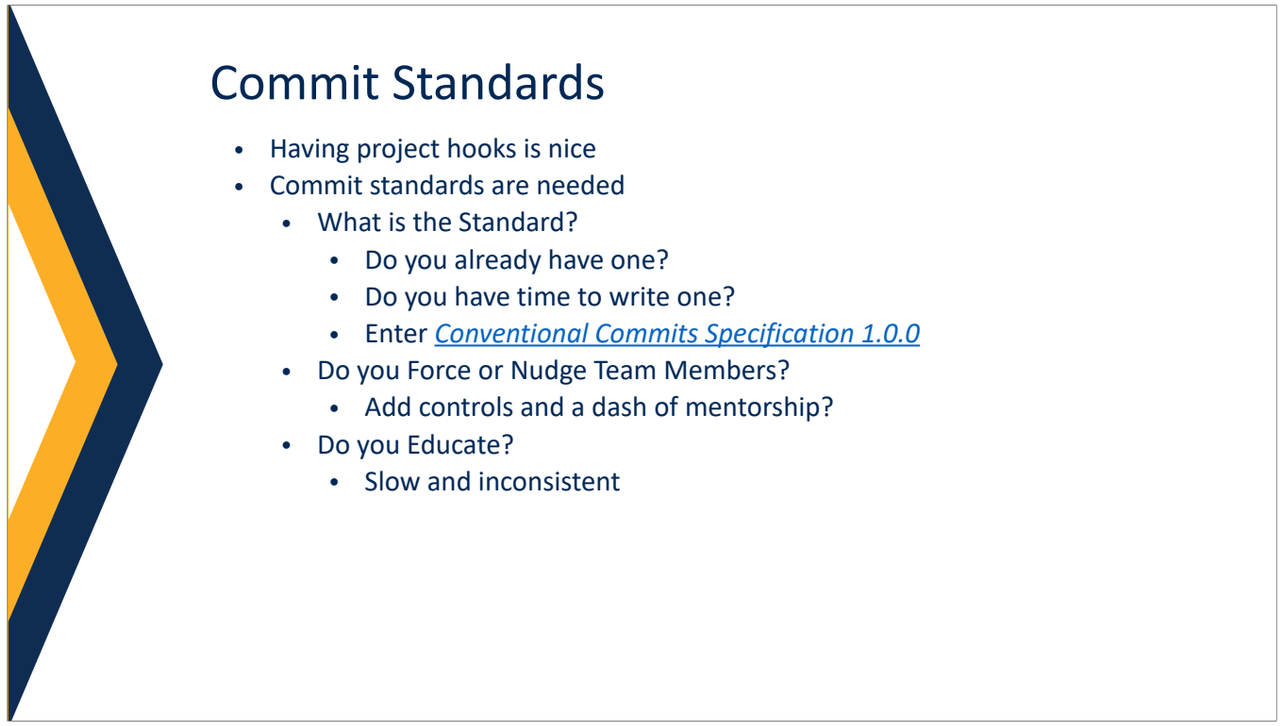
- Modern, native git hooks made easy
  - “Native git hooks?”
  - Sets `core.hookspath` to `.husky`

```
npm install –-save-dev husky
npx husky init
```

The `init` command simplifies setting up husky in a project. It creates a `pre-commit` script in `.husky/` and updates the `prepare` script in `package.json`.

```
git config --get core.hookspath
git add --all
git commit --verbose --all
```

- BAM! Project portable commit controls



Standards?
Many Open Source Projects and large organizations adopt a standard, e.g., Angular
Write a standard?

- Watch for ”fingerprinting” and “bikeshedding”
- Checkpoint: would it stand up to open source scrutiny?

Force or Nudge?
I prefer nudge using controls and add mentorship. A control provides light friction which nudges. Nudges lead to adoption of the control and the friction is removed. Mentor when developers struggle to adopt.

No control? Careful, absent a control or friction, behaviors will not adapt, and goals will be lost.

Commit Controls

- You've adopted standards...
- Enter Commitlint[1]
  - "Helps your team adhere to a commit convention"

```
% npm install --save-dev @commitlint/cli
% npm install --save-dev @commitlint/config-conventional
% echo "npx --no -- commitlint --edit \$1" > .husky/commit-msg
% chmod +x .husky/commit-msg[2]
% vi .commitlintrc.js
... Supports YAML, Typescript (.ts, .cts), JSON, CommonJS (.cjs),
and ECMA Modules (.mjs) too!
% npx commitlint --print-config
% git add --all
% git commit --verbose --all
```

1 There are others, e.g., commitizen. YMMV

2 In Husky v9, the executable bit is optional. The *nix executable bit is a committable change; ignored by native Windows; on Windows, but pipeline is *nix? `git --update-index --chmod=+x .husky/commit-msg`

```
module.exports = {
  extends: [
    '@commitlint/config-conventional'
  ],
  rules: {
    'body-empty': [2, 'never'],
    'body-min-length': [2, 'always', 12]
  }
}
```

Update-index is the plumbling API of git. After you commit the change, you can test it with git ls-files .husky/commit-msg.

Commit Communist!

- Push back happens.

```
git add --all
git commit --verbose --all --no-verify
```

  - `--no-verify` bypasses hooks by ignoring non-zero exit codes
- Do all my contributors need to use the Conventional Commits specification?
  - No
- Nudge and mentor, then... Squash
  - Merge leads can fixup commits
  - This becomes especially necessary when coupling commit messages to...
    - Pipelines and automated Semantic Versioning

FAQs

Photo Credit: Baskervill

My Project Isn’t JavaScript...

- Git Hooks are the Honeybadger
  - They “Don’t Care”
- `node` and `npm` (`yarn`, yada yada) should be available
  - For developers
  - In the pipeline
- Java, Python, Rust, Go...
  - Add
    - `.commitlintrc.js`
    - `package.json`[1]
    - `package-lock.json`
  - Husky and Git Hooks take care of the rest

```
1 npm install # runs prepare script, sets core.hookspath
```

My CICD Build Targets...

- YMMV
- GitLab uses Job runners with artifacts and dotenv support
- Pseudo-pipeline descriptor…

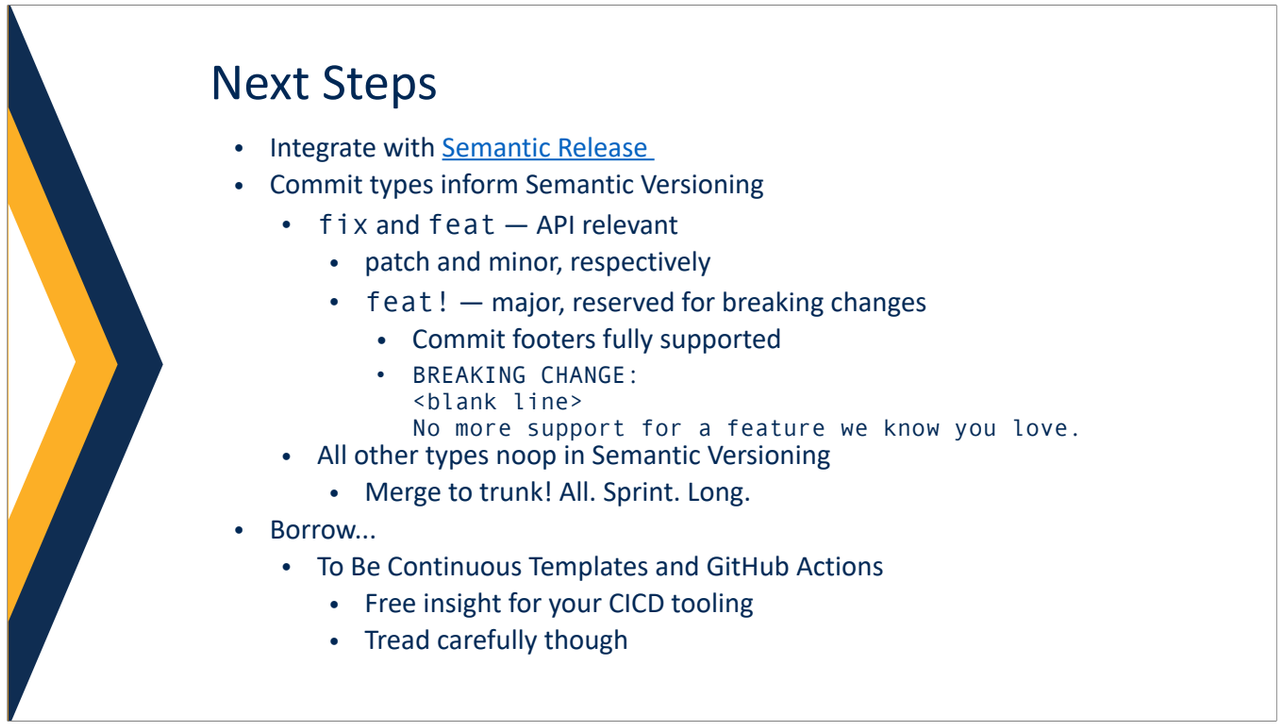
```
semantic-release-info:      # job
   stage: .pre
   image: node:latest       # job runner
   script:
     - ...
     - semantic-release --dry-run  # writes dotenv to pass to release job
mvn-build:
  image: maven:3.8-eclipse-temurin-17
  script:
    ...
mvn-release:
  image: maven:3.8-eclipse-temurin-17
  script:
    - mvn release:prepare -DreleaseVersion=${SEMREL_INFO_NEXT_VERSION}"...
    - ...
```

My CICD Pipeline Makes Commits

- Common concern
- Pipelines may fail
- Options:
  - Configure CICD tooling, e.g.,

    `-DscmDevelopmentCommitComment="ci(foo): prepare..."`
    - Can be tedious and involve a lot of testing
  - Test executing context
    - `is-ci` to the rescue!
      - Supports 40+ CI platforms

```
npm install --save-dev is-ci
npm pkg set scripts.prepare="is-ci || husky"
```



Borrowing should be done when you can fully understand the intent of the contribution and the project.

Case: TBC recently achieved full GitLab CI/CD catalog availability with support for components, think GitHub Actions. The new audience did not have any of the supporting context of the years of work by CI/CD professionals, and often did not RTM before posing questions.

TBC has detailed Usage and Architecture documentation. GitHub Actions often lacks this foundational work. Know before adoption. Prepare to ask questions of the contributors and communities.

Demo

- A walk through of adding commitlint to a project.

```
npm i -D husky
npx husky init
git config --get core.hookspath
git add --all
git commit --verbose —all
npm i -D @commitlint/cli @commitlint/config-conventional
echo "npx --no -- commitlint --edit \$1" > .husky/commit-msg
chmod +x .husky/commit-msg # optional in Husky v9
cat << EOF >   .commitlintrc.js  # Supports YAML, Typescript (.ts, .cts), JSON, CommonJS (.cjs), and ECMA Modules (.mjs) too!
module.exports = {
  extends: [
    '@commitlint/config-conventional'
  ],
  rules: {
    'body-empty': [2, 'never'],
    'body-min-length': [2, 'always', 12]
  }
}
EOF
npx commitlint --print-config
git add --all
git commit --verbose —all
npm i -D  is-ci
npm pkg set scripts.prepare="is-ci || husky"
```


QUESTIONS?
Photo Credit: Baskervill