Zephyr® Project
Developer Summit

Zephyr® Project
Developer Summit

Full Developer Workflow with West

Al Semjonovs, *Google*

Email: asemjonovs@google.com
Discord: asemjonovs#8232
Github: asemjonovs

#EMBEDDEDOSSUMMIT

Agenda

- What is West?
- Streamline and simplify the developer workflow
  - Workspace Setup
  - Dependency manager
  - Twister integration
  - Manage PR dependencies

What is West?

West is Zephyr's swiss army knife command line tool for syncing, building, and flashing your builds.

Most used built-in commands

- `init`
- `update`

Most used Zephyr project extensions

- `build`
- `flash`
- `debug`

Streamline and simplify the developer workflow

1. Setup
2. Toolchain Dependency Manager
3. Twister integration
4. PR Dependency Manager

Setup

Problem

- Multiple steps are needed to setup a Zephyr based project. See the Zephyr Getting Started Guide

Goal

- Subcommand: `west setup`
- Setup your developer environment with a single command.
  - Creates python virtual environment
  - Installs Zephyr’s python dependencies
  - Downloads and installs latest Zephyr SDK
  - Allow this command to be overridable for project specific setup.

Zephyr® Project
Developer Summit

Toolchain Dependency Manager

Problem

- West setup will installs tools, how do we keep these up to date?

Goal:

- Override `west update` to handle toolchains in addition to the code.
- Store current version of tools and dependencies in the west manifest.
- Installs needed changes.
- Similar to current handling of git repositories but with tools.

Twister Integration

Problem

- Invoking twister from another project can lead to inaccurate test results due to different versions of twister.

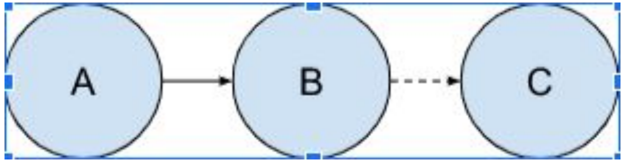
Goal

- Subcommand: `west twister`
- Avoid mismatch of twister version with project currently being worked on.
- Use west to identify path to twister of current project.

Manage PR Dependencies

Problem

Working with multiple branches that have dependencies on each other can be difficult to manage.



$$\begin{tikzcd} A \arrow[r] & B \arrow[r, dashed] & C \end{tikzcd}$$

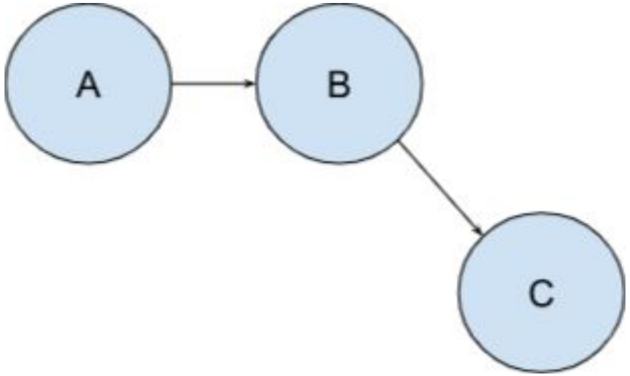
Adding commit C forces a new review process to start for a PR

Manage PR Dependencies

if C should be separate from the existing PR, the developer must create a branch.

This has two issues

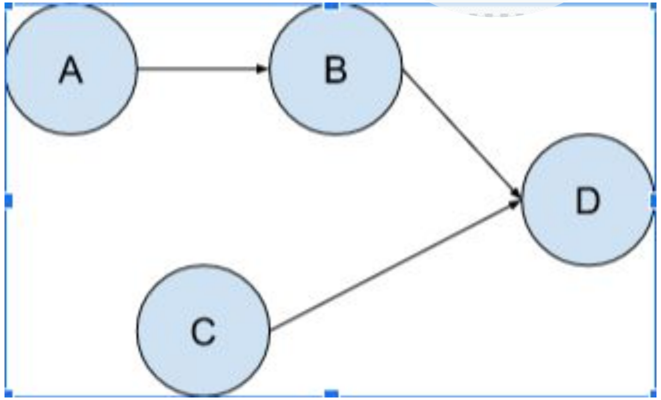
1. C can’t be posted for review without A and B being bundled into the same PR
2. Every new feature built on C will also require a new branch



Manage PR Dependencies

This would be impossible to do

- A and B create a PR and are reviewed together
- C creates a second PR and is reviewed parallel to A/B
- D has a dependency on A/B/C and cannot be reasonably reviewed until they merge.



Zephyr® Project
Developer Summit

West push

Goal: Improve Zephyr upstream workflow and PR management.

Solution

- All changes exist in a single local branch
- Changes are grouped with keyword matching `^topic#(\w+)$` in their commit message.
- Matching capture groups are grouped together, cherry-picked to a new branch, and submitted as a new PR (or updated if already exists)

Zephyr® Project
Developer Summit

Thanks!

Zephyr Discord: @asemjonovs