METHODIST UNIVERSITY

7 Skills To Jumpstart Your Technical Career

Dr. Sally Wahba
Principal Software Engineer
Splunk, a Cisco Company
@sallyky

Agenda

1. **Contributing to existing code**

METHODIST UNIVERSITY

Agenda

1. Contributing to existing code
2. **Software debugging**

METHODIST UNIVERSITY

Agenda

1. Contributing to existing code
2. Software debugging
3. **Scripting and software testing**

METHODIST UNIVERSITY

Agenda

1. Contributing to existing code
2. Software debugging
3. Scripting and software testing
4. **Version control**

METHODIST UNIVERSITY

Agenda

1. Contributing to existing code
2. Software debugging
3. Scripting and software testing
4. Version control
5. **Building code**

METHODIST UNIVERSITY

Agenda

1. Contributing to existing code
2. Software debugging
3. Scripting and software testing
4. Version control
5. Building code
6. **Continuous Integration/Continuous Delivery/Deployment**

Agenda

1. Contributing to existing code
2. Software debugging
3. Scripting and software testing
4. Version control
5. Building code
6. Continuous Integration/Continuous delivery
7. **Linux and Containers**

METHODIST UNIVERSITY

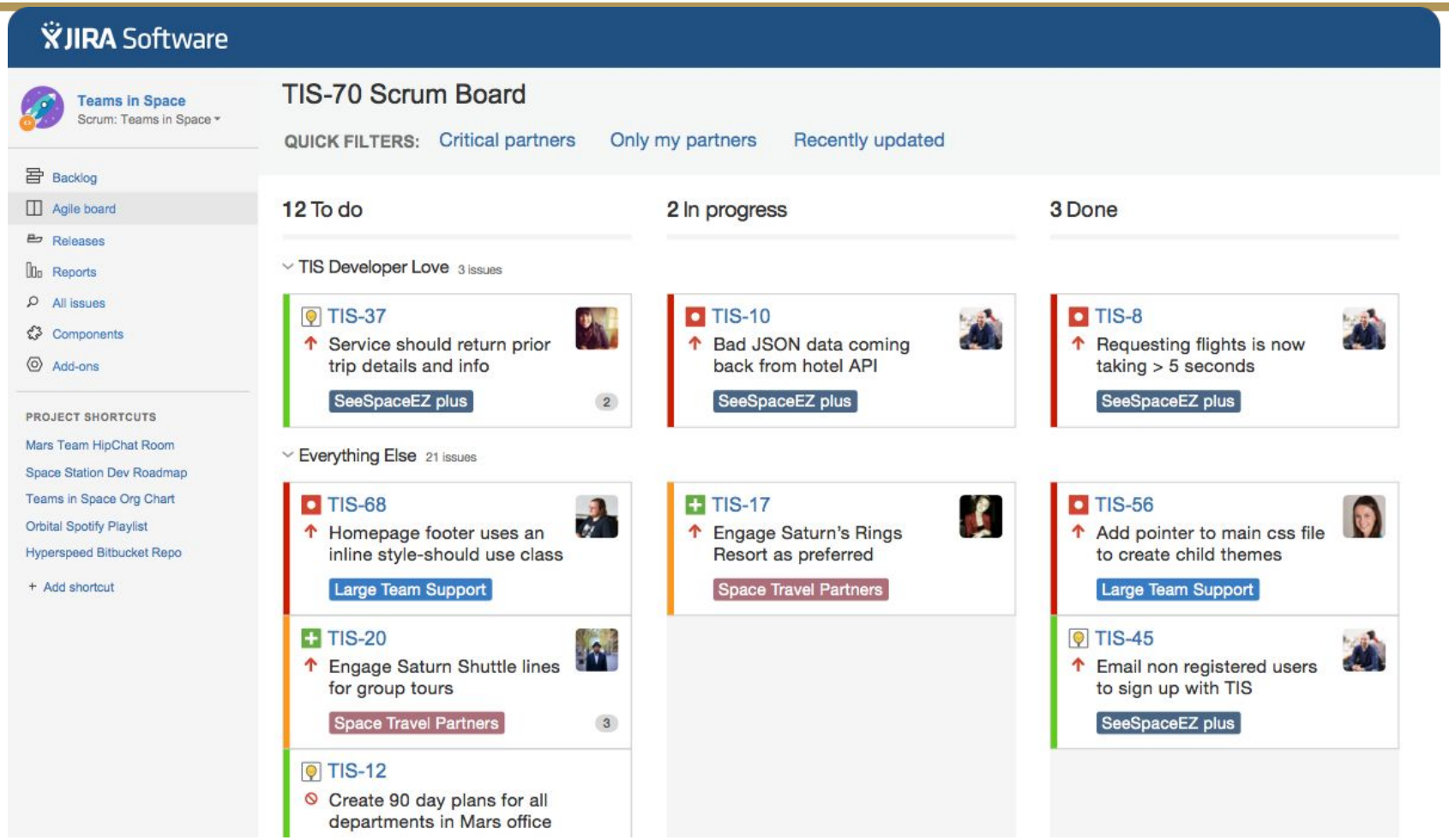
Story Time

METHODIST UNIVERSITY

Story Time

METHODIST UNIVERSITY

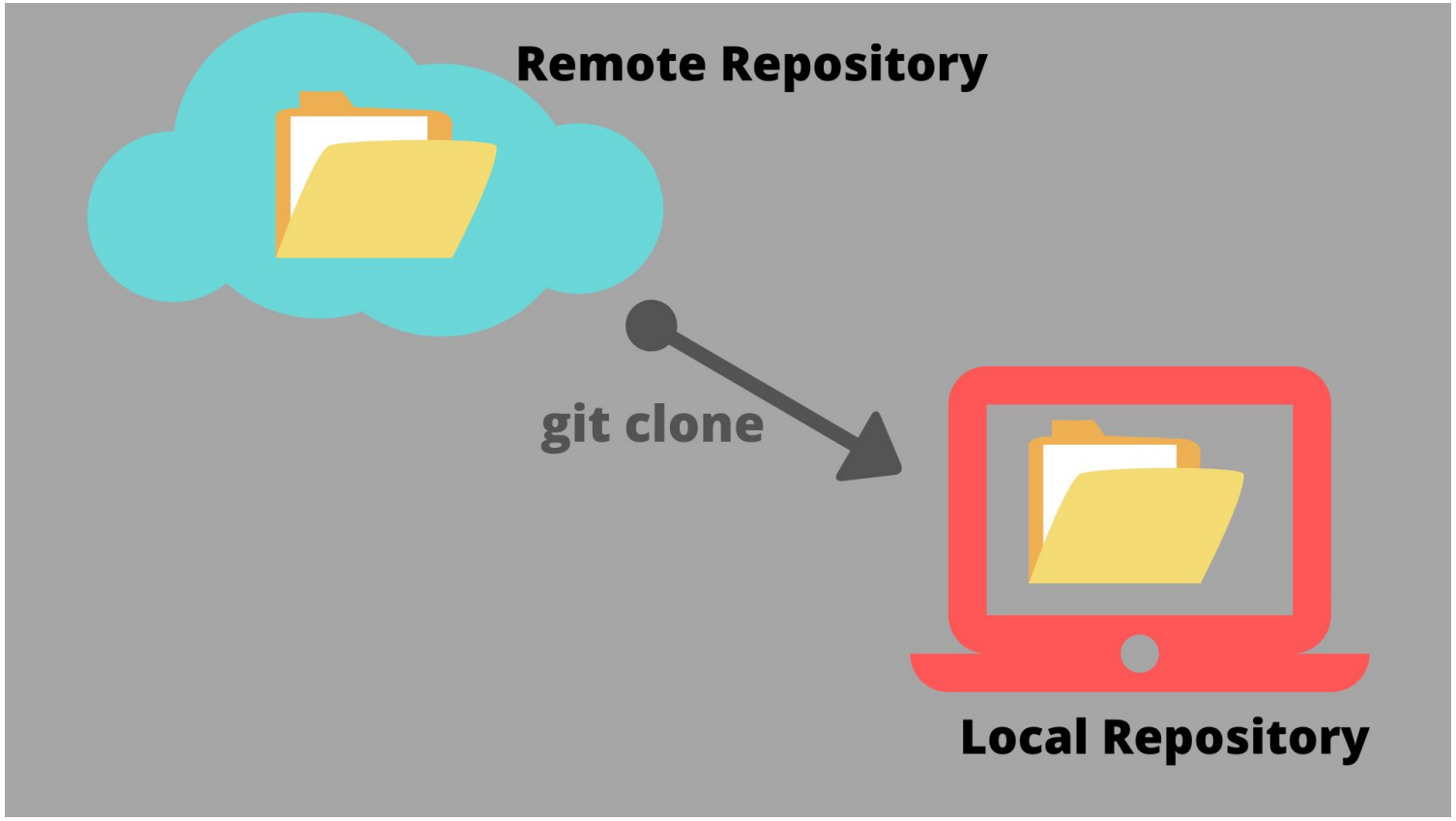
First Ticket



METHODIST UNIVERSITY

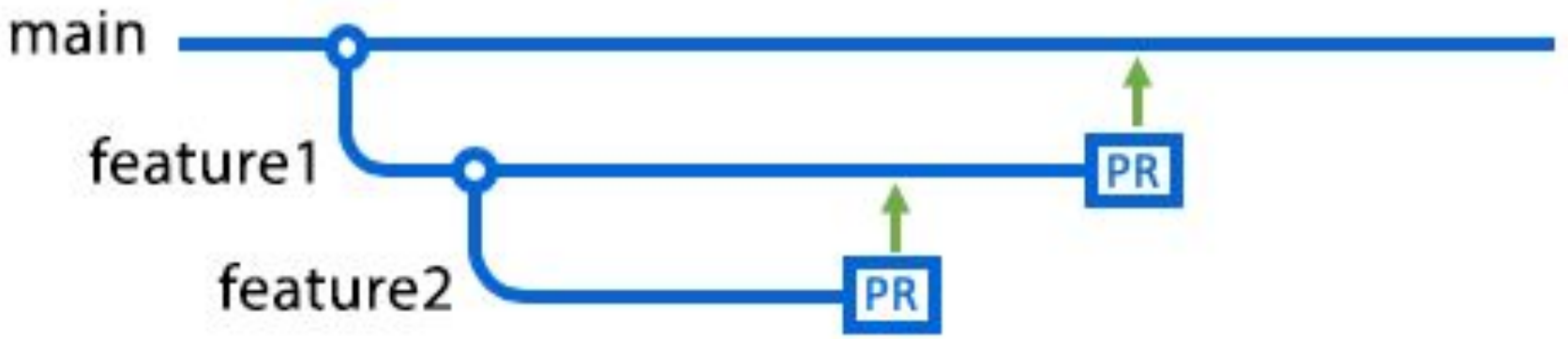
Version Control

Version Control



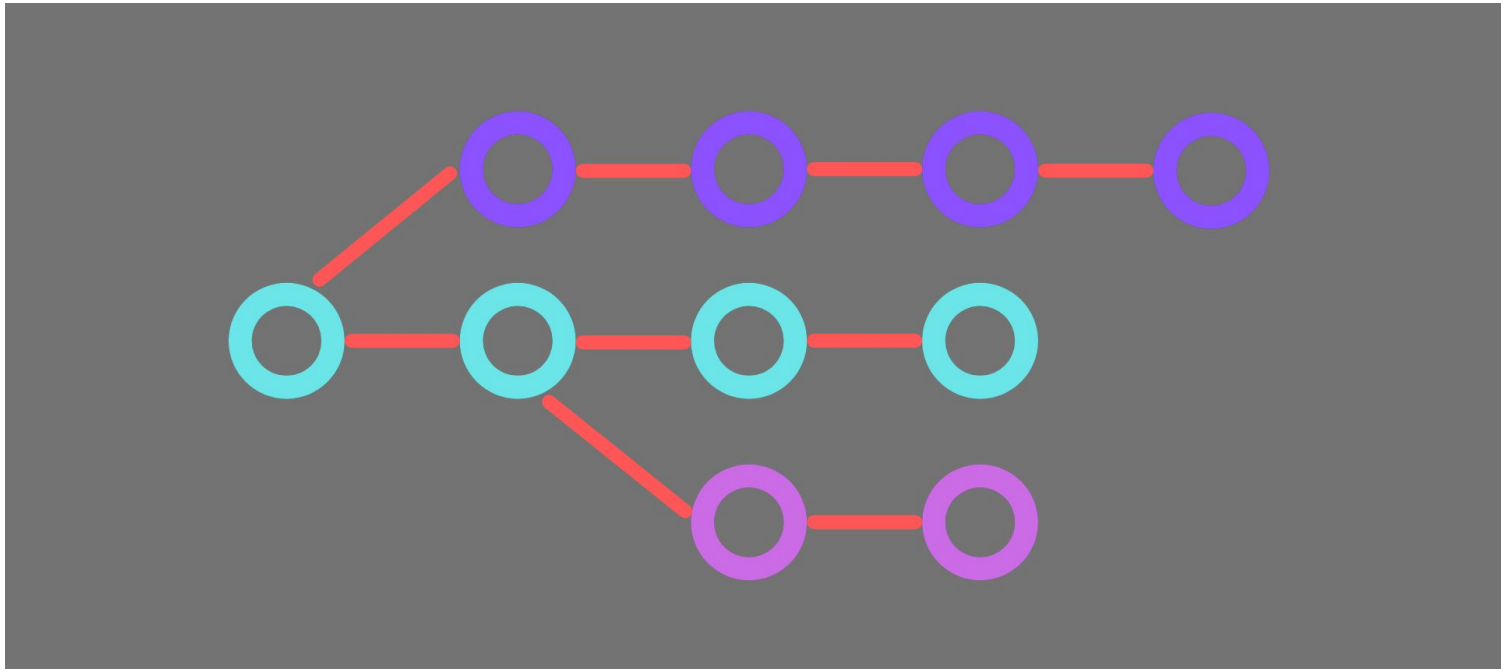
$ git clone git@github.com:myawesomecompany/awesomesource

Version Control



METHODIST UNIVERSITY

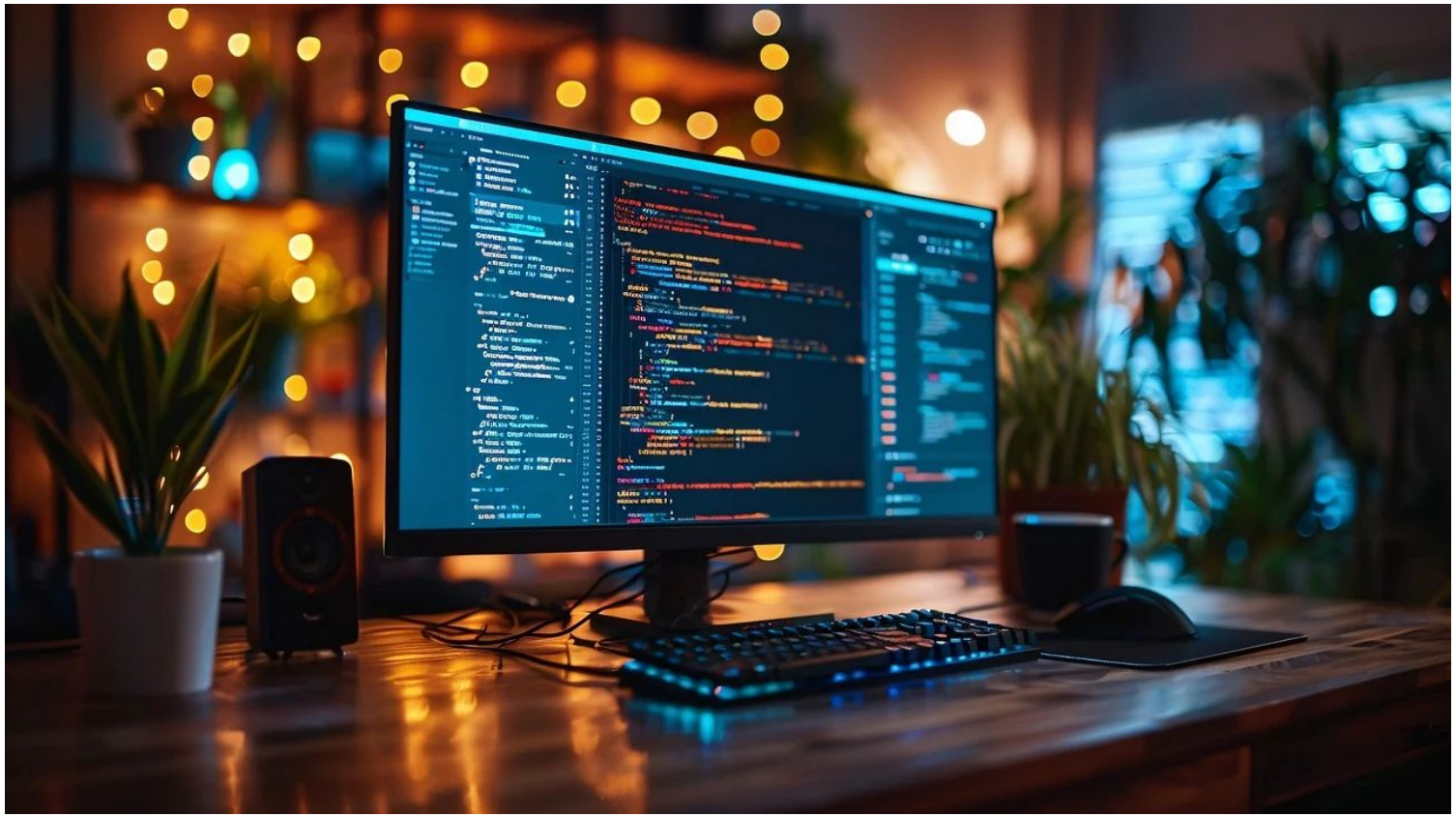
Version Control



$ git checkout -b myFirstTicket

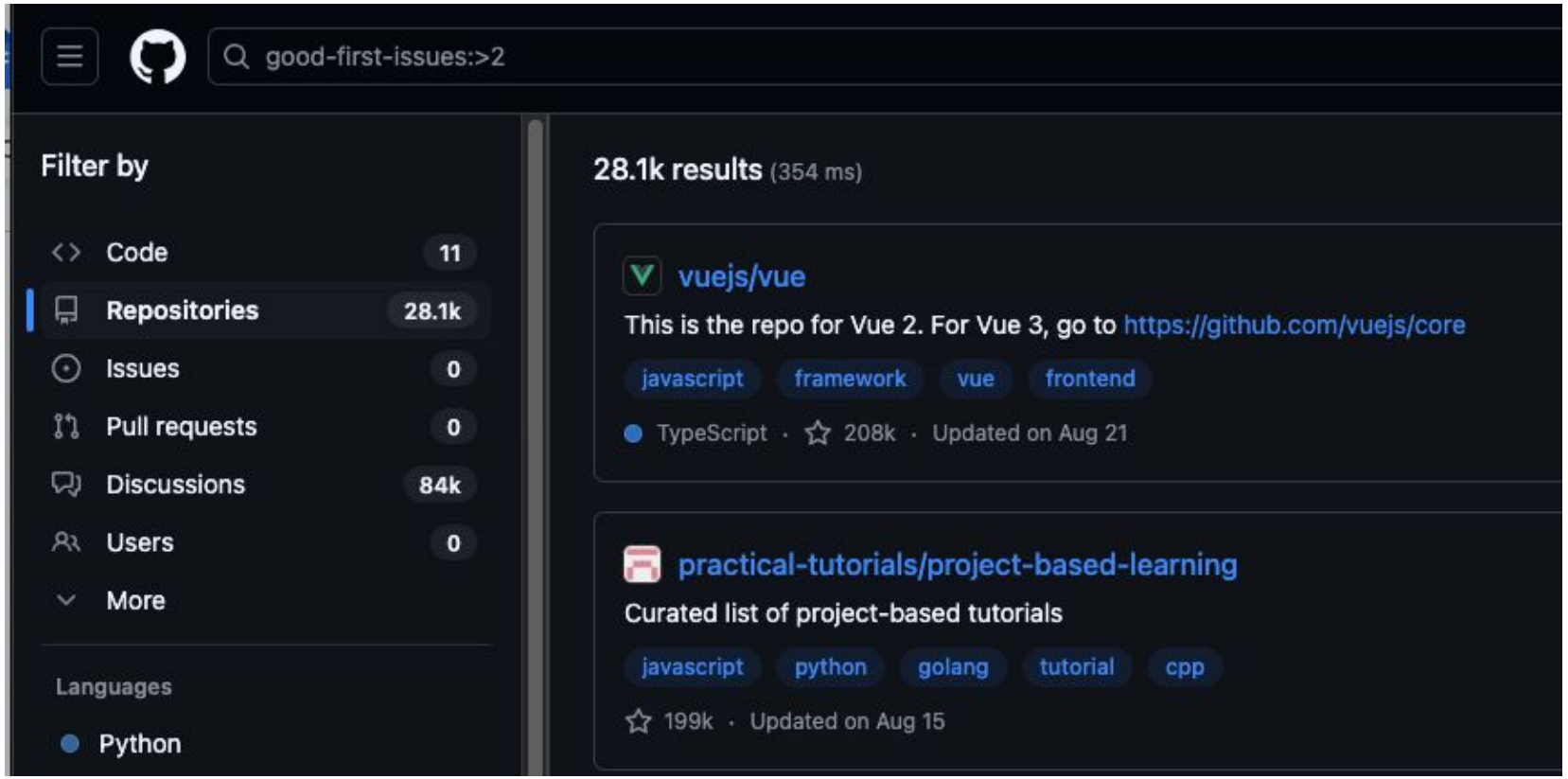
METHODIST UNIVERSITY

Contributing to Existing Code

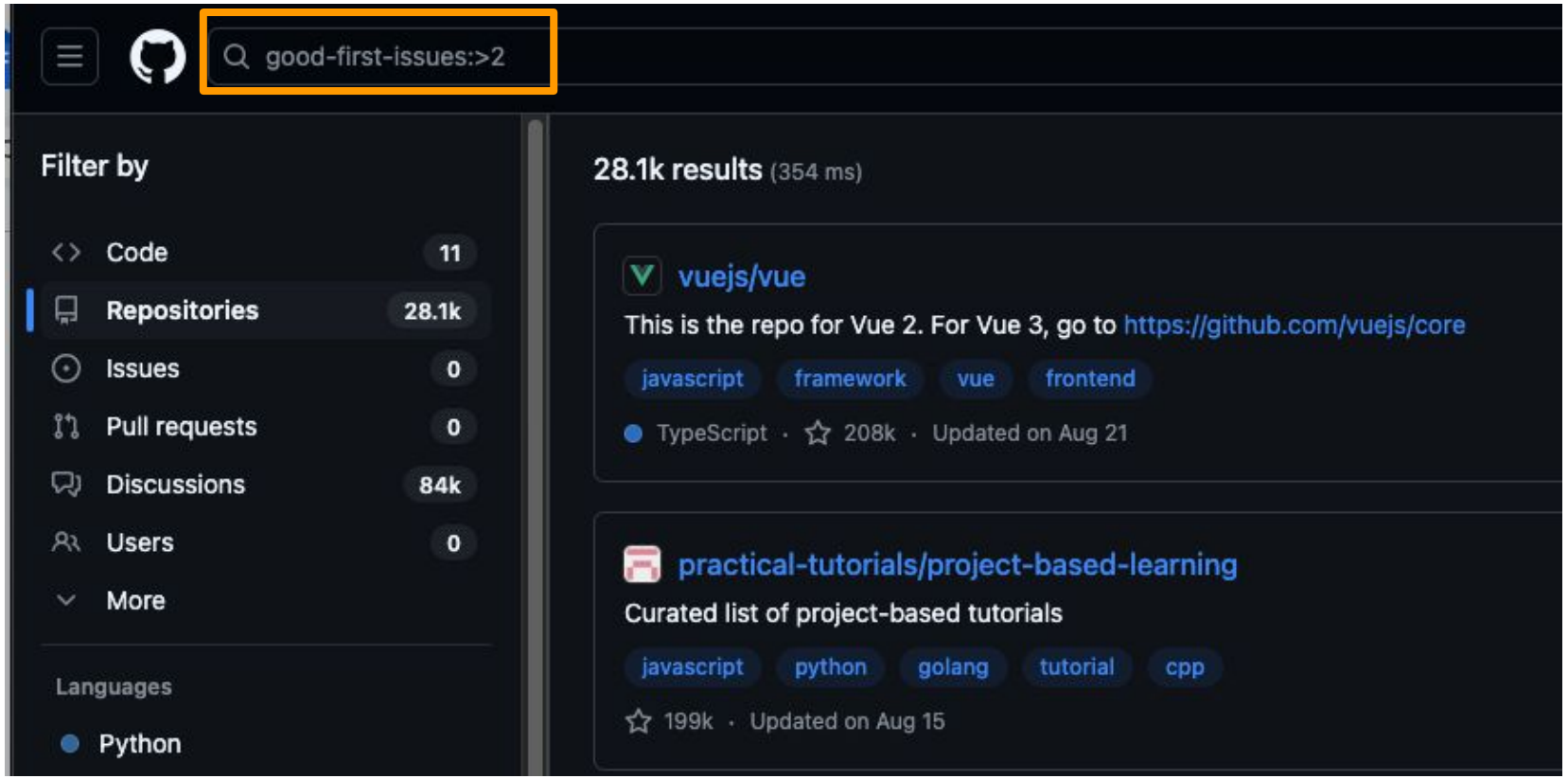


METHODIST UNIVERSITY

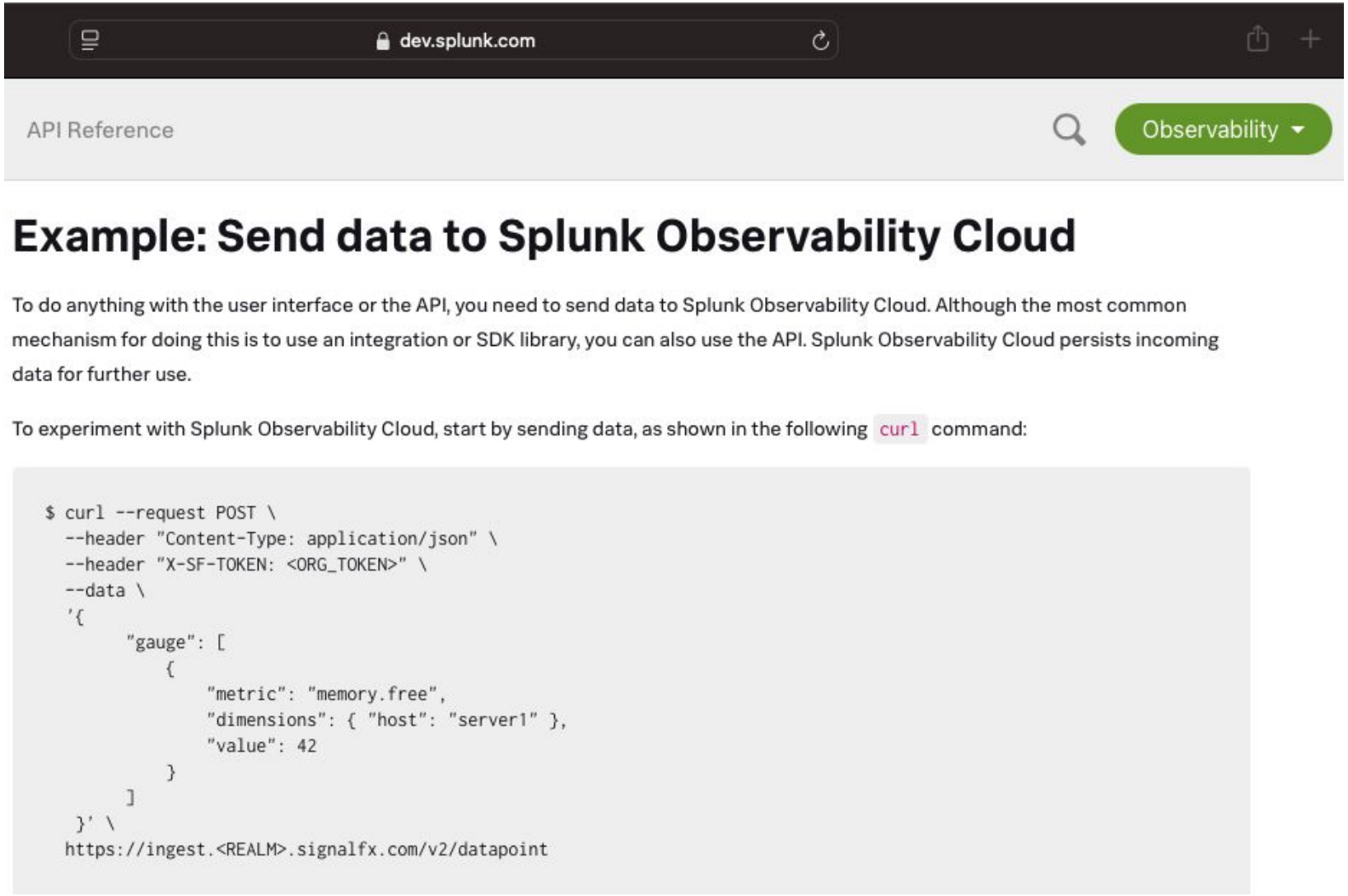
Contributing to Existing Code



Contributing to Existing Code

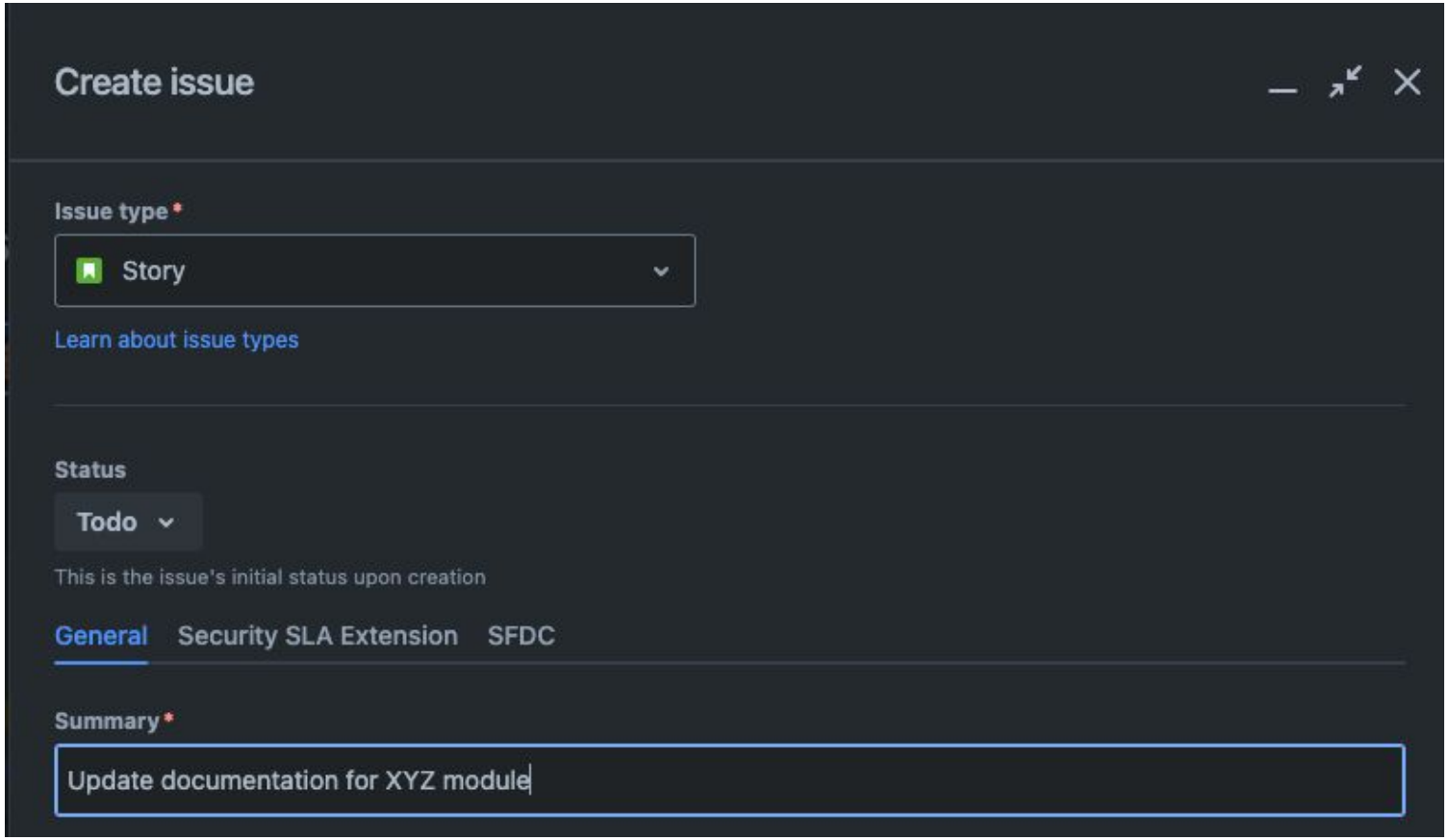


Contributing to Existing Code



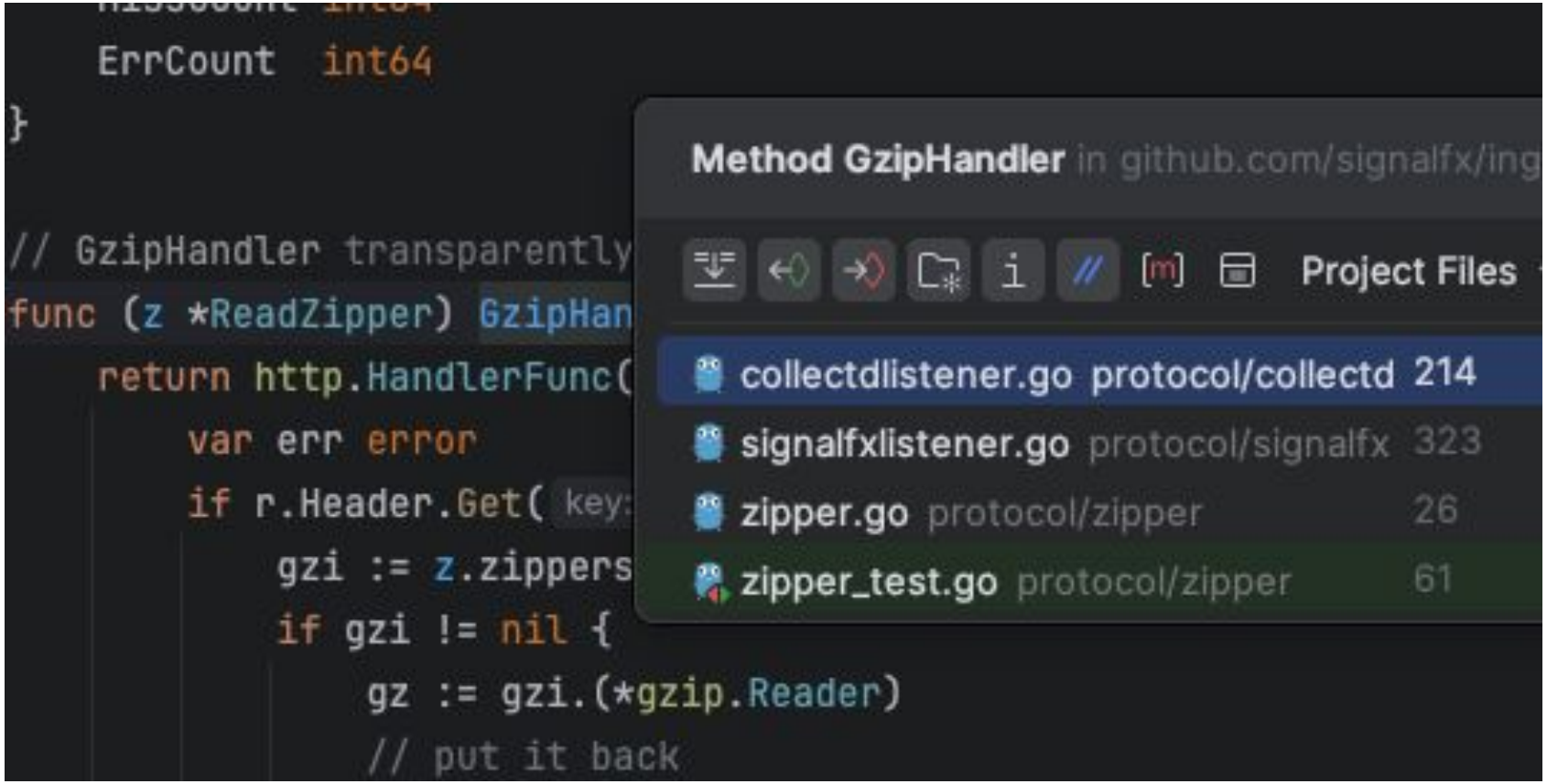
METHODIST UNIVERSITY

Contributing to Existing Code



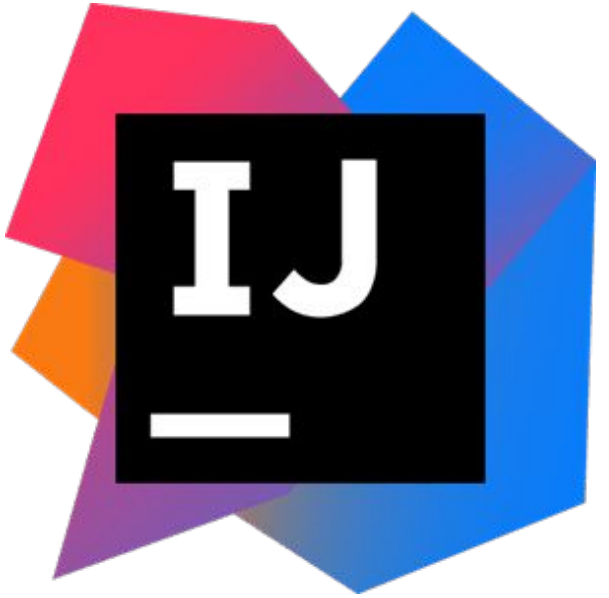
METHODIST UNIVERSITY

Contributing to Existing Code



METHODIST UNIVERSITY

Building Code



METHODIST UNIVERSITY

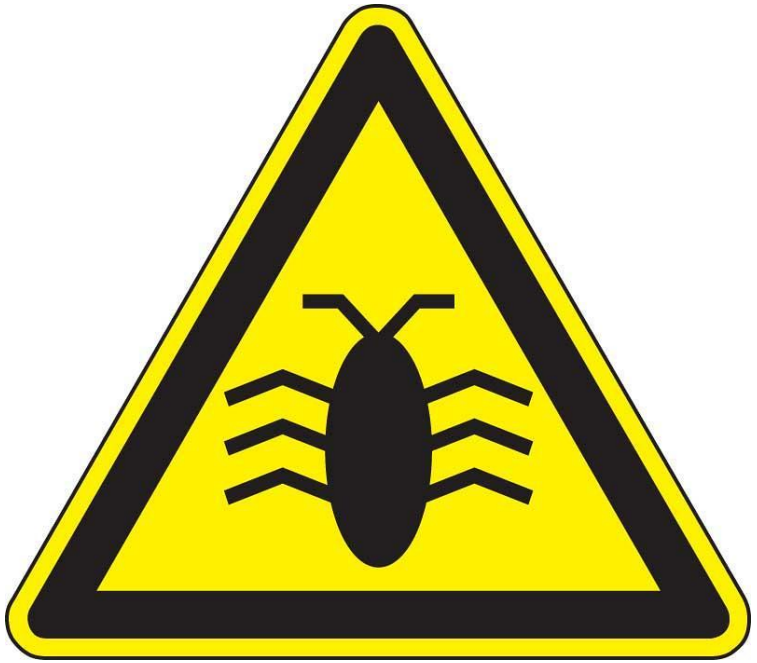
Building Code

Makefile

```
1   SHELL                = /bin/bash
2   BASE                 = $(CURDIR)
3   CP_RF                = cp -rf
4   GOLANGCI_LINT_VERSION = 1.49.0
5
6   # enable module support across all go commands.
7   export GO111MODULE = on
8
9   .SILENT: ;               # no need for @
10  .ONESHELL: ;             # recipes execute in same shell
11  .NOTPARALLEL: ;          # wait for this target to finish
12
13  # default is verification of sfxinternalgo which includes all services
14  .PHONY: verify
15  verify: install-tools lint test
```

METHODIST UNIVERSITY

Software Debugging



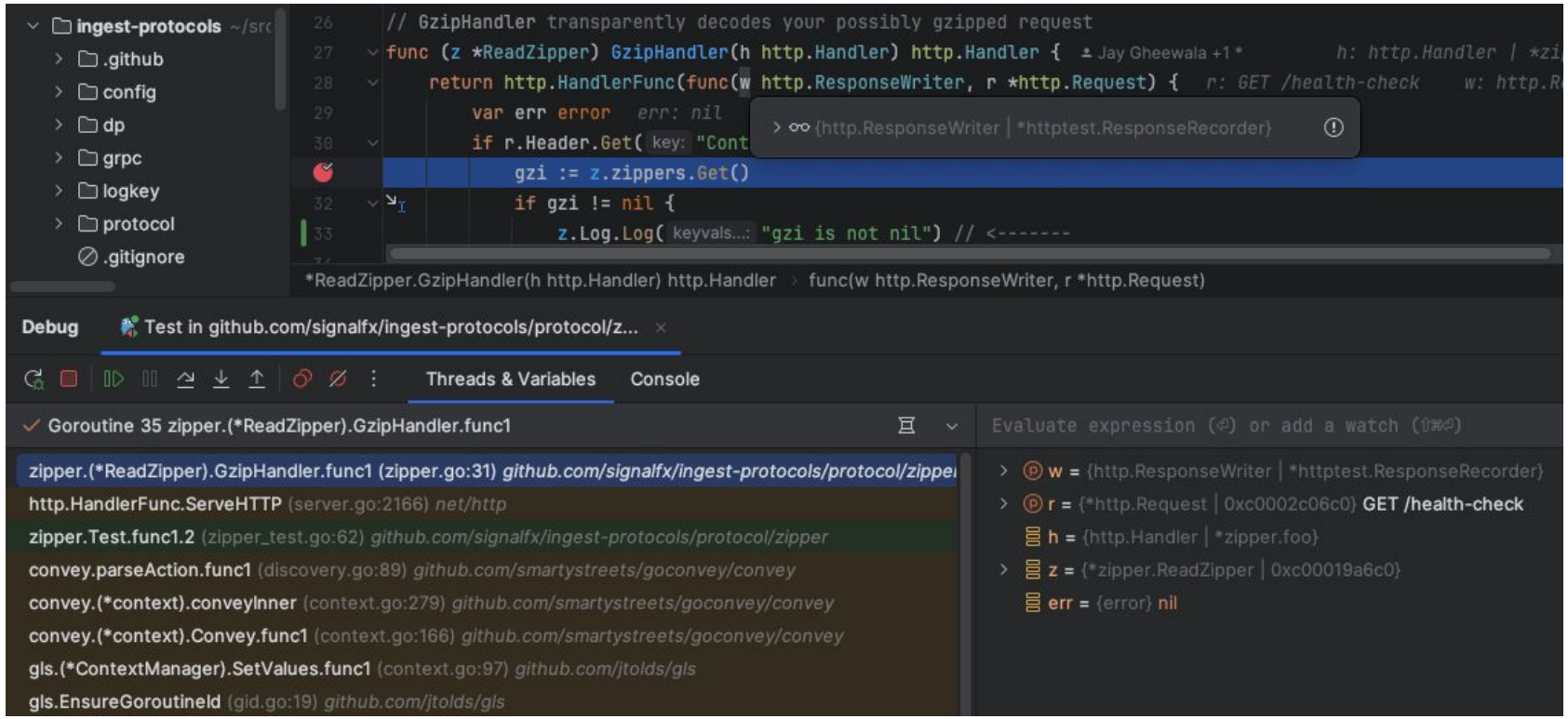
METHODIST UNIVERSITY

Software Debugging

```
if gzi != nil {
    z.Log.Log( keyvals...: "gzi is not nil") // <--------
    gz := gzi.(*gzip.Reader)
    // put it back
    defer z.zippers.Put(gz)
    err = gz.Reset(r.Body)
    if err == nil {
        defer log.IfErr(z.Log, gz.Close())
        // nasty? could construct another object but seems expensive
        r.Body = gz
        z.Log.Log( keyvals...: "Incrementing hit count") // <--------
        atomic.AddInt64(&z.HitCount, delta: 1)
```

METHODIST UNIVERSITY

Software Debugging



METHODIST UNIVERSITY

Scripting and Software Testing

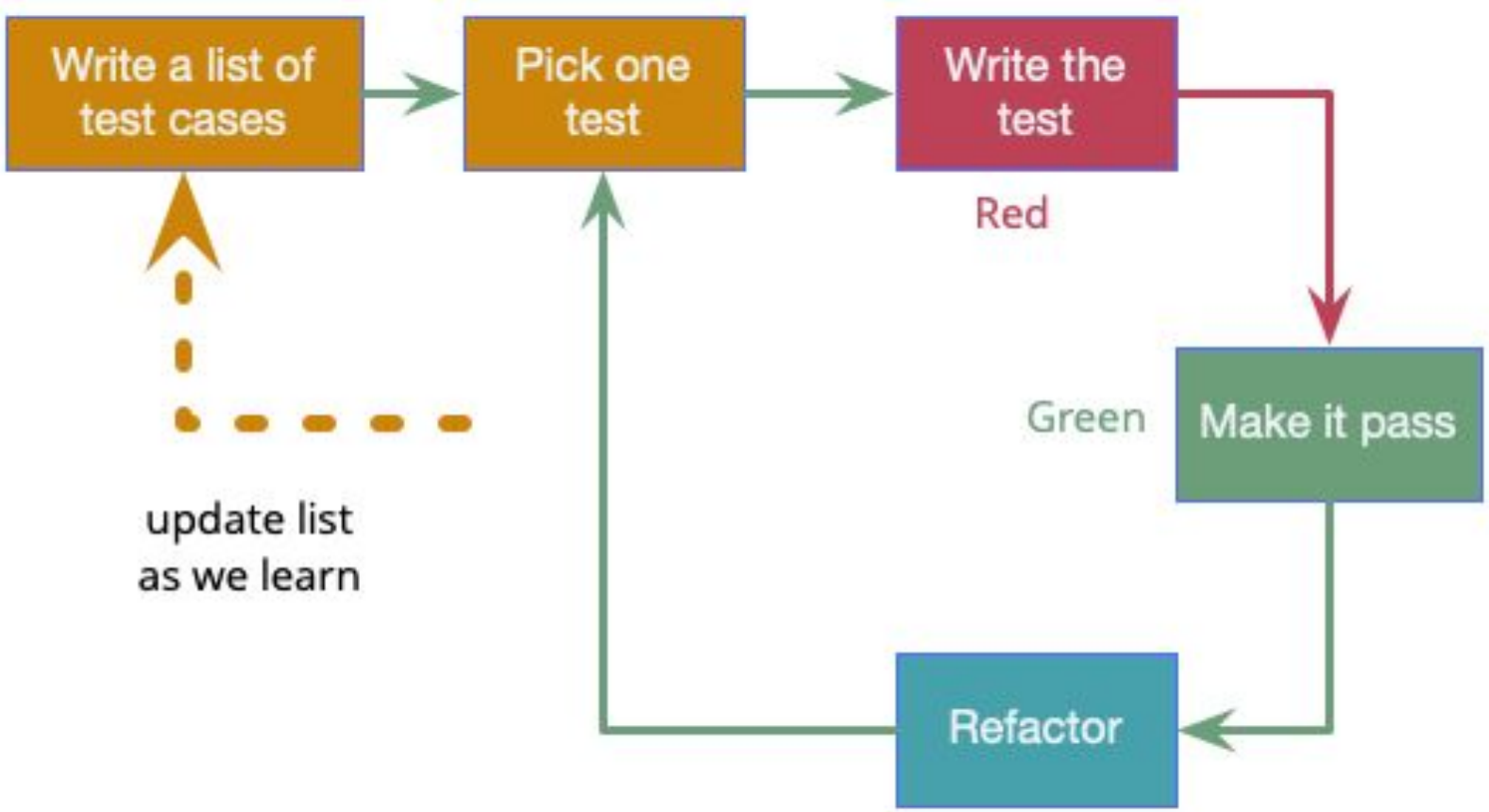
METHODIST UNIVERSITY

Scripting and Software Testing

```
func Test(t *testing.T) {
    Convey( items...: "Test zipper", t, func() {
        zippers := NewZipper()
        badZippers := newZipper(func(r io.Reader) (*gzip.Reader, error) {
            return new(gzip.Reader), errors.New( msg: "nope")
        })
        f := new(foo)
        zipped := new(bytes.Buffer)
        w := gzip.NewWriter(zipped)
        _, err := w.Write([]byte("OK"))
        So(err, ShouldBeNil)
        So(w.Close(), ShouldBeNil)
        tests := []struct {...}{
            { zipper: zippers,  name: "test non gzipped",  data: []byte("OK"),
```

METHODIST UNIVERSITY

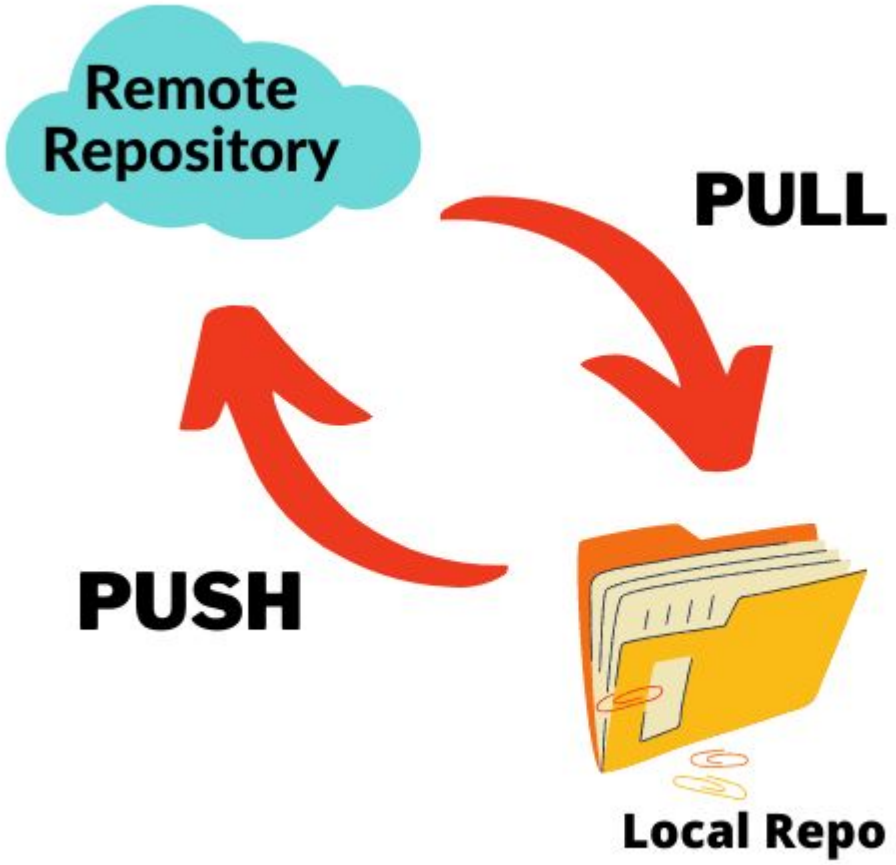
Scripting and Software Testing



Scripting and Software Testing

METHODIST UNIVERSITY

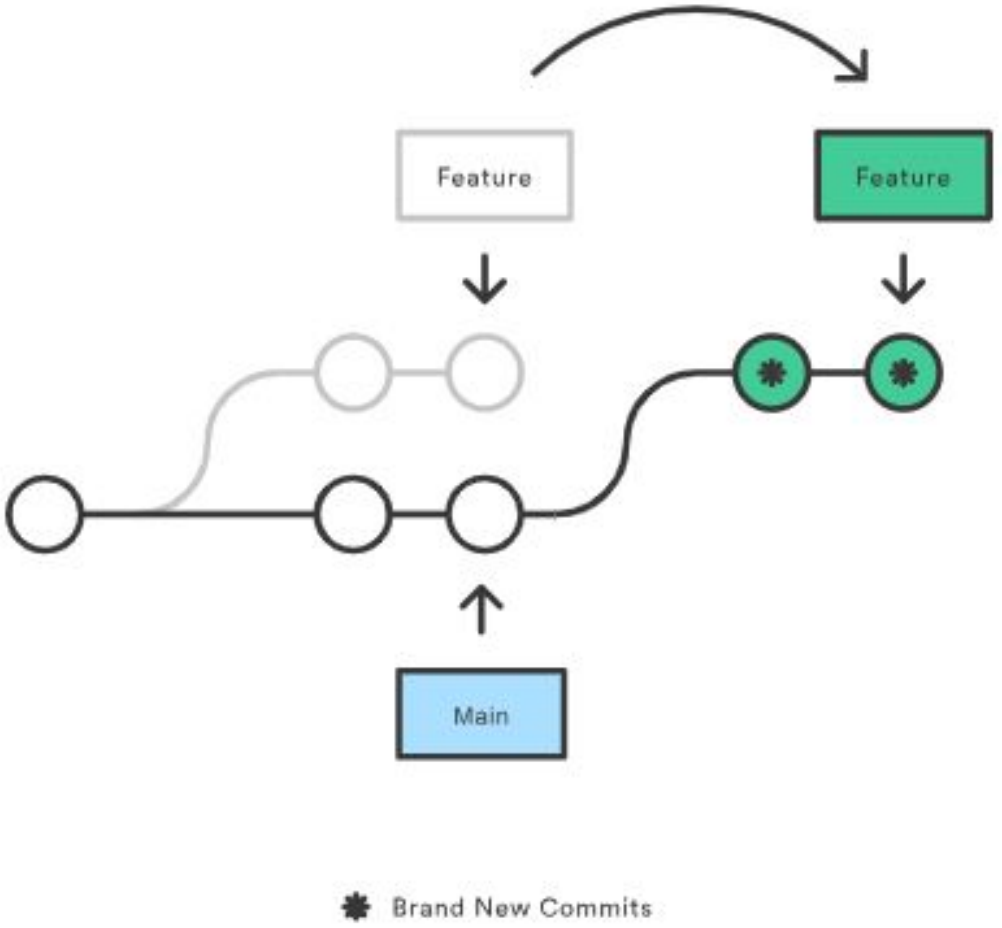
Version Control



$ git push –set-upstream origin myFirstTicket

METHODIST UNIVERSITY

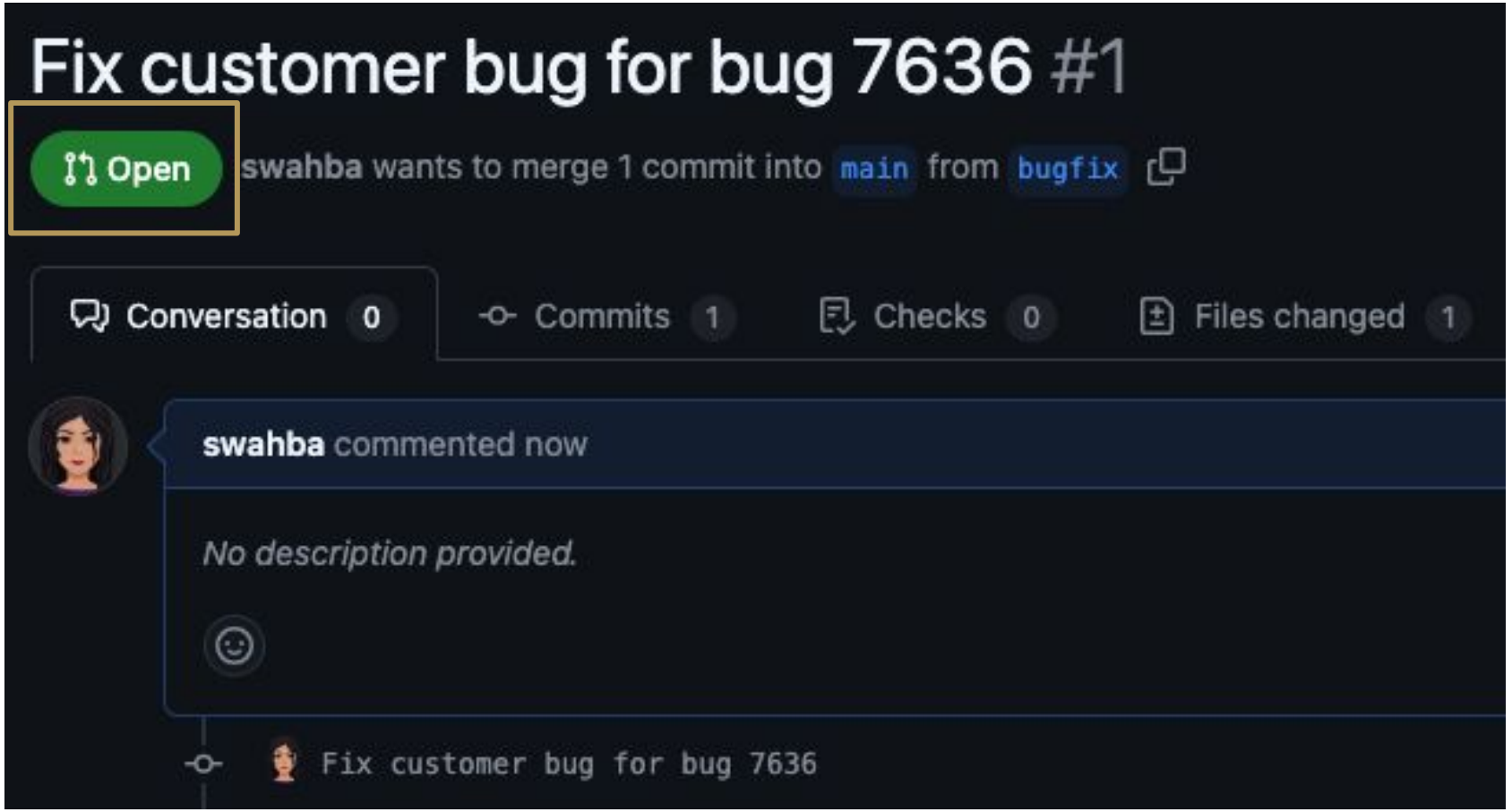
Version Control



$ git rebase main

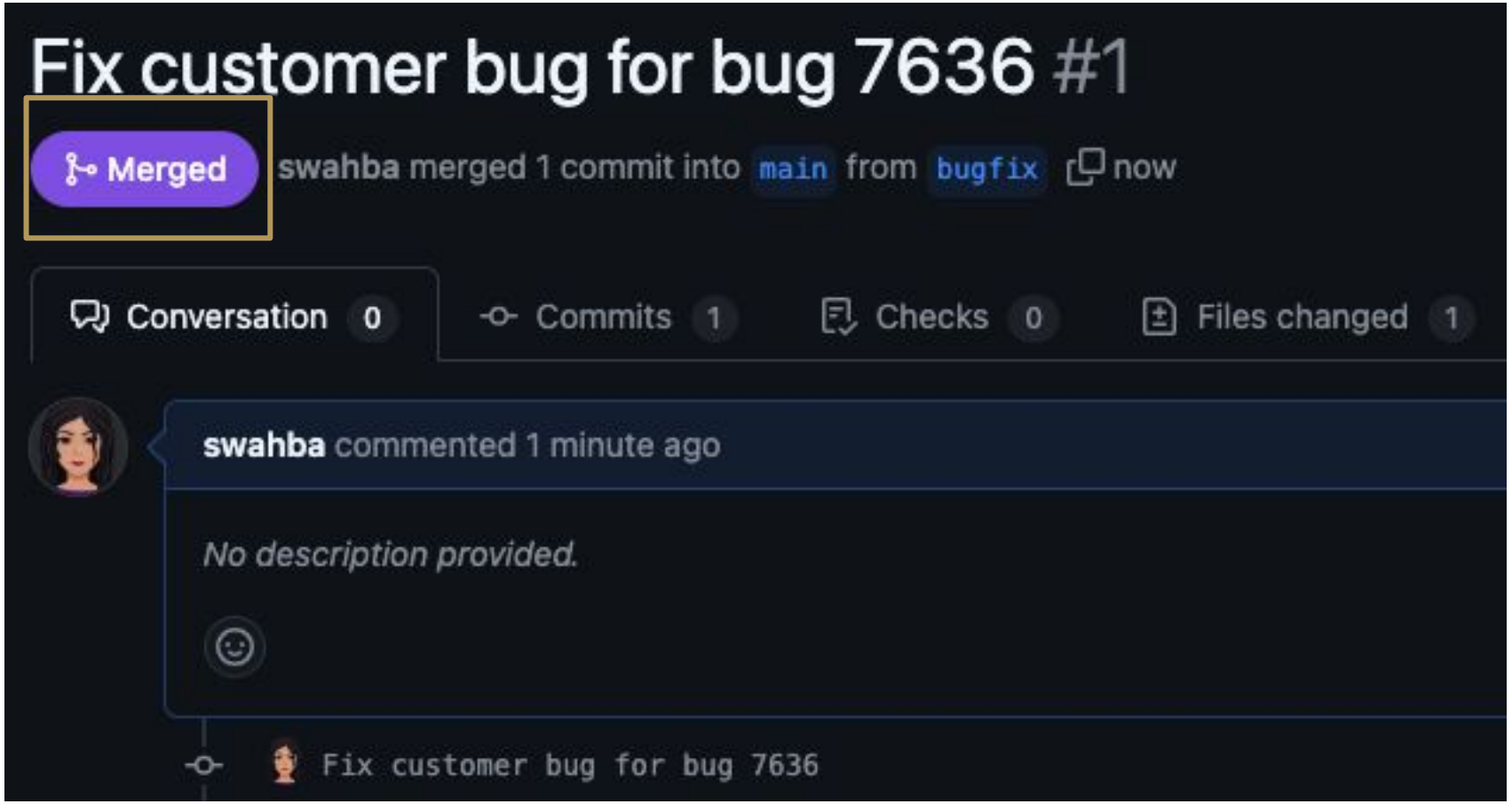
METHODIST UNIVERSITY

Version Control



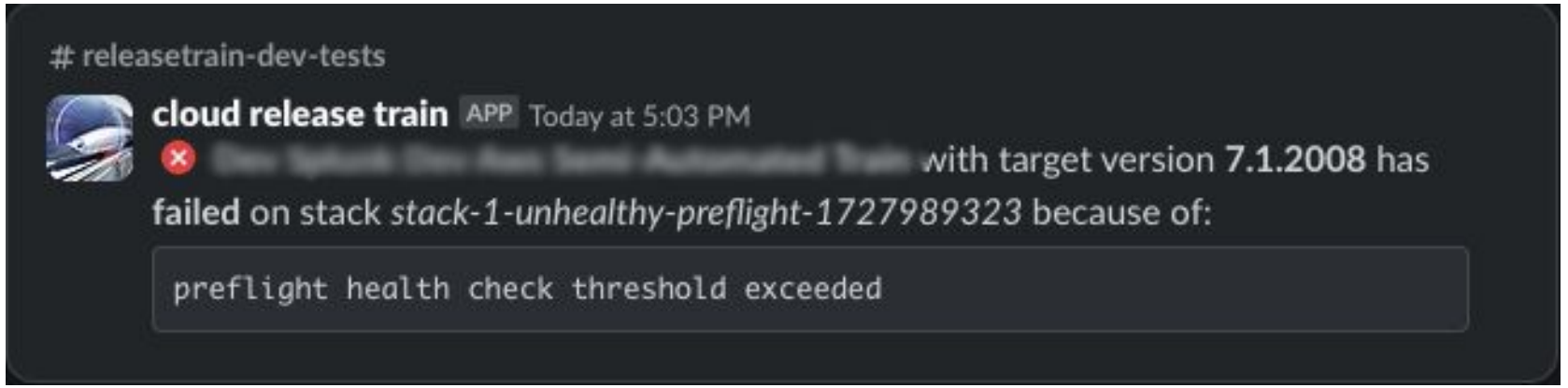
METHODIST UNIVERSITY

Version Control



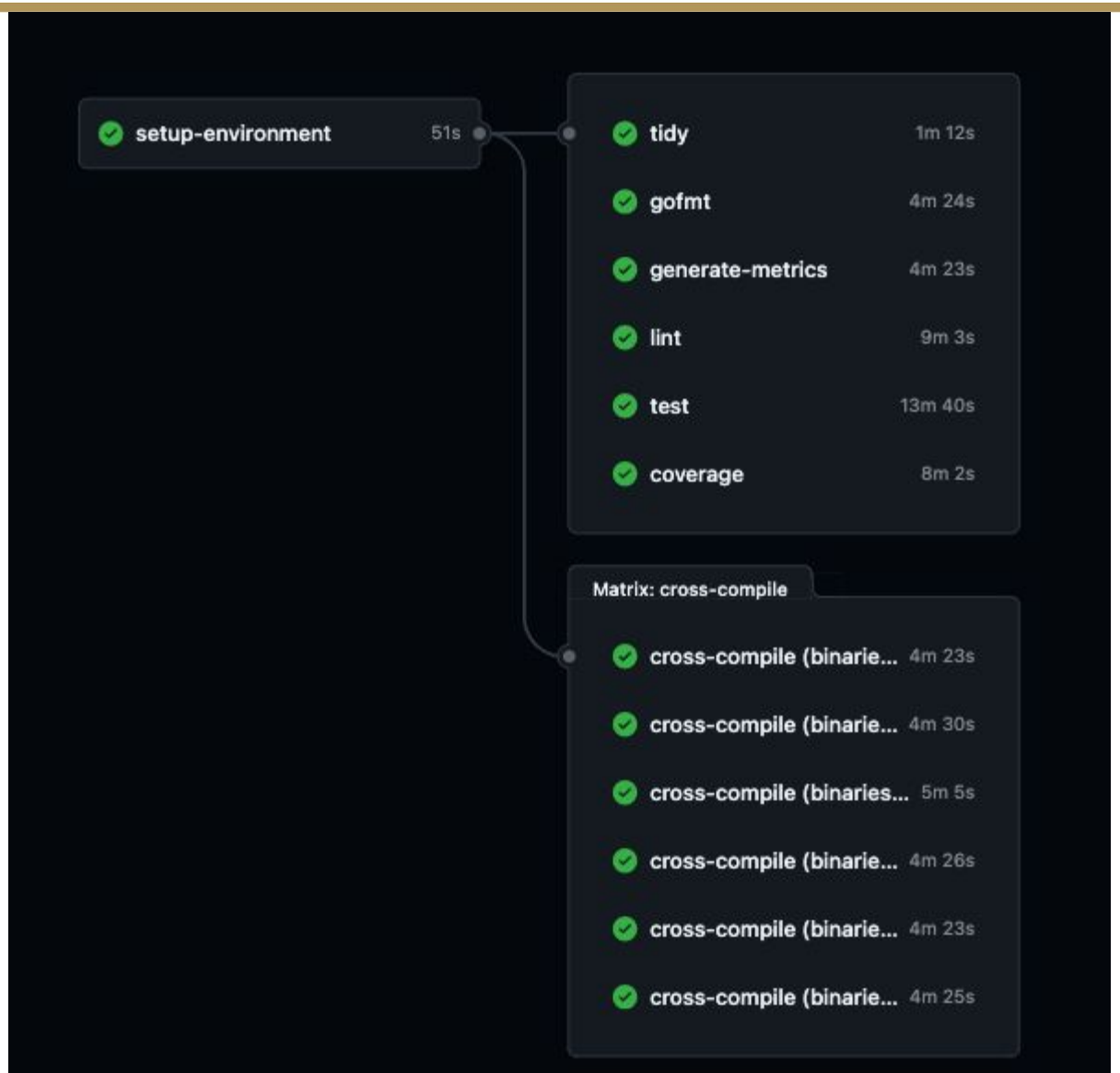
METHODIST UNIVERSITY

CI/CD



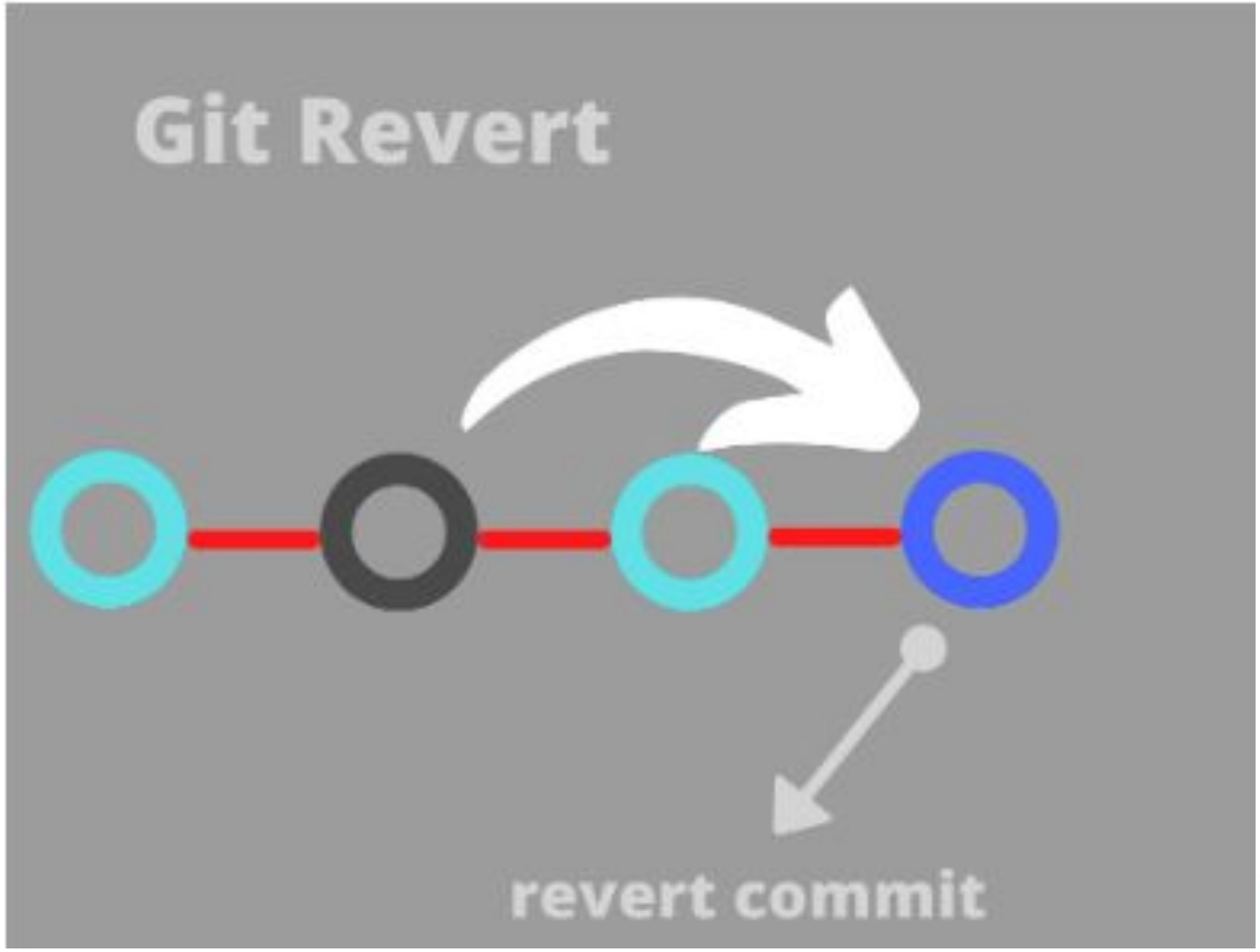
METHODIST UNIVERSITY

CI/CD



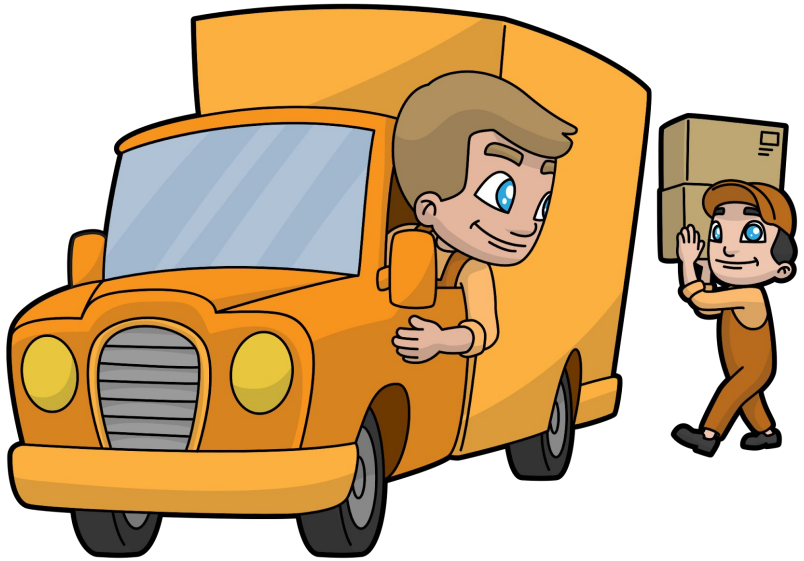
METHODIST UNIVERSITY

Version Control



METHODIST UNIVERSITY

CI/CD



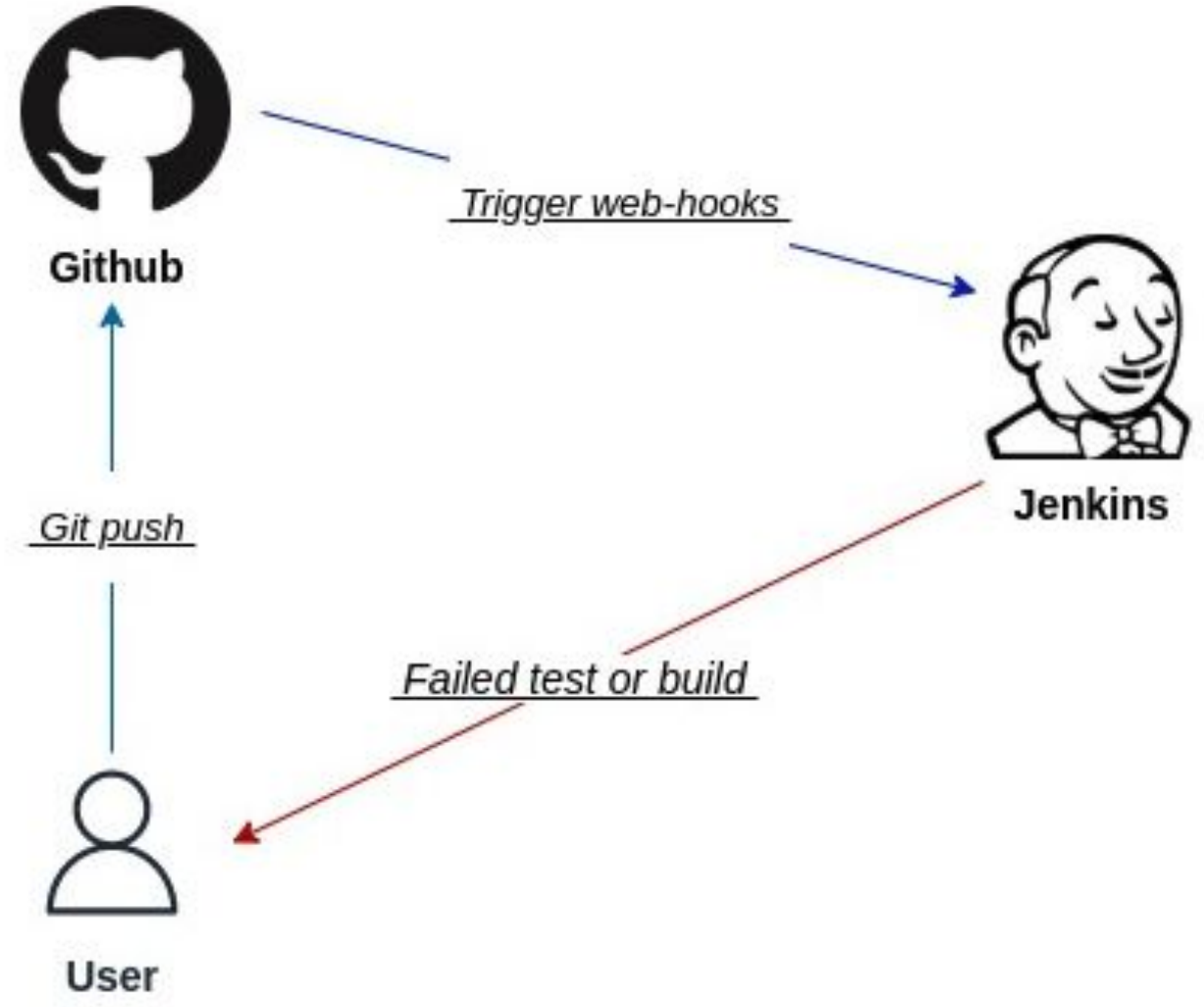
CI/CD

- Continuous integration is the practice of automatically and frequently integrating code changes to a shared source code repository.
- Continuous delivery refers to the integration, testing, and delivery of code changes.
- Continuous deployment automatically releases the updates into a production environment.

CI/CD

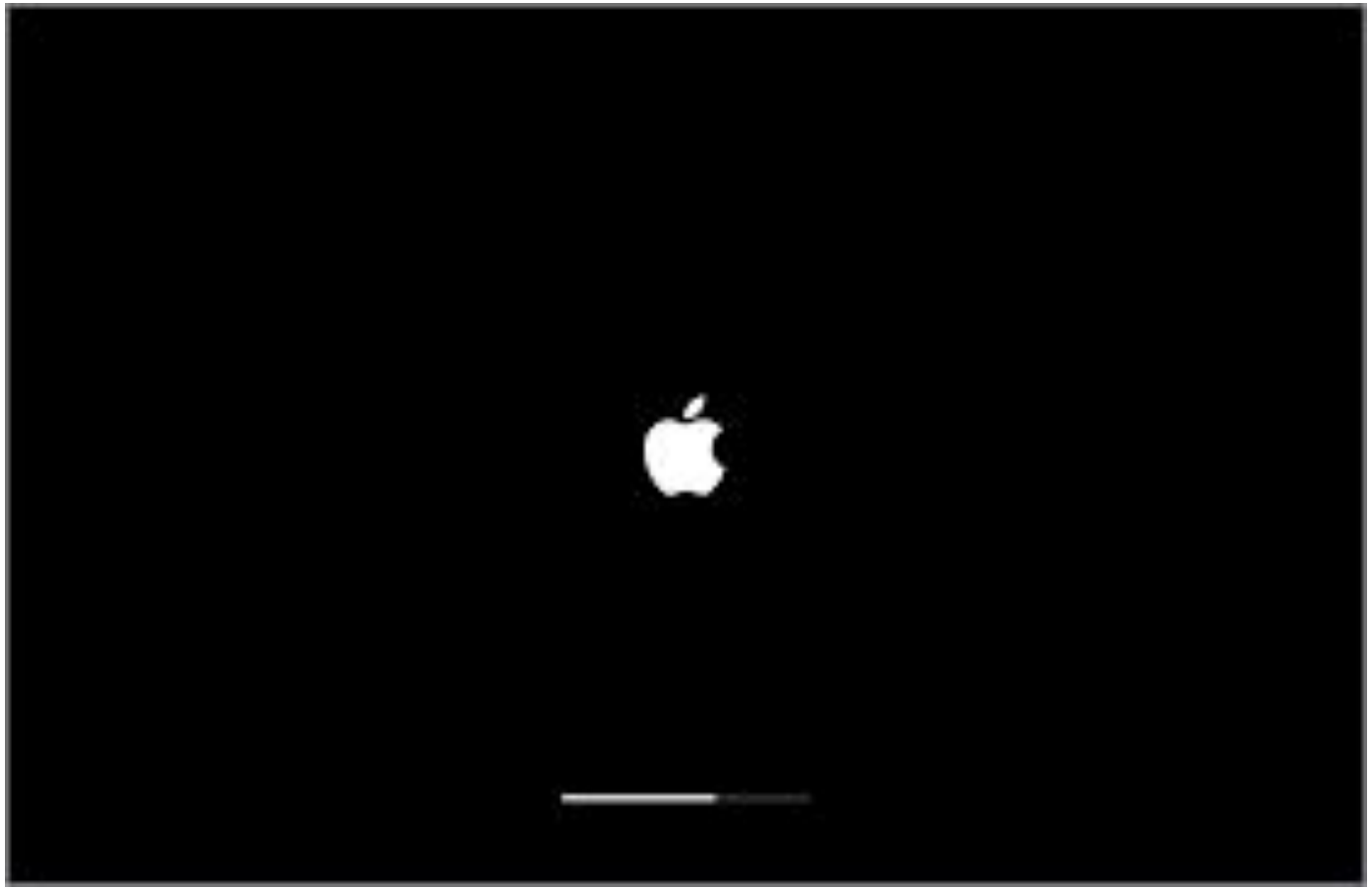
METHODIST UNIVERSITY

CI/CD



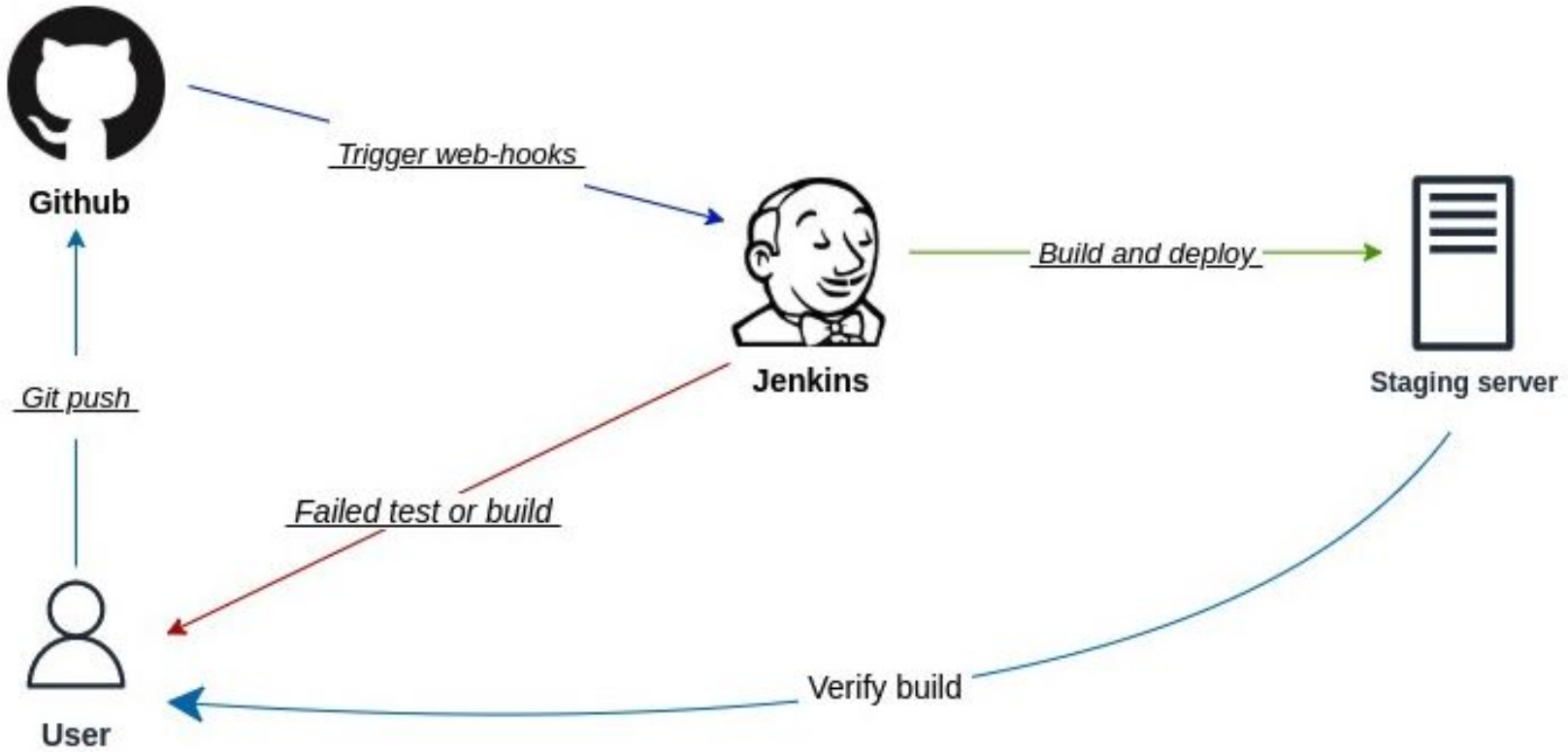
METHODIST UNIVERSITY

CI/CD



METHODIST UNIVERSITY

CI/CD

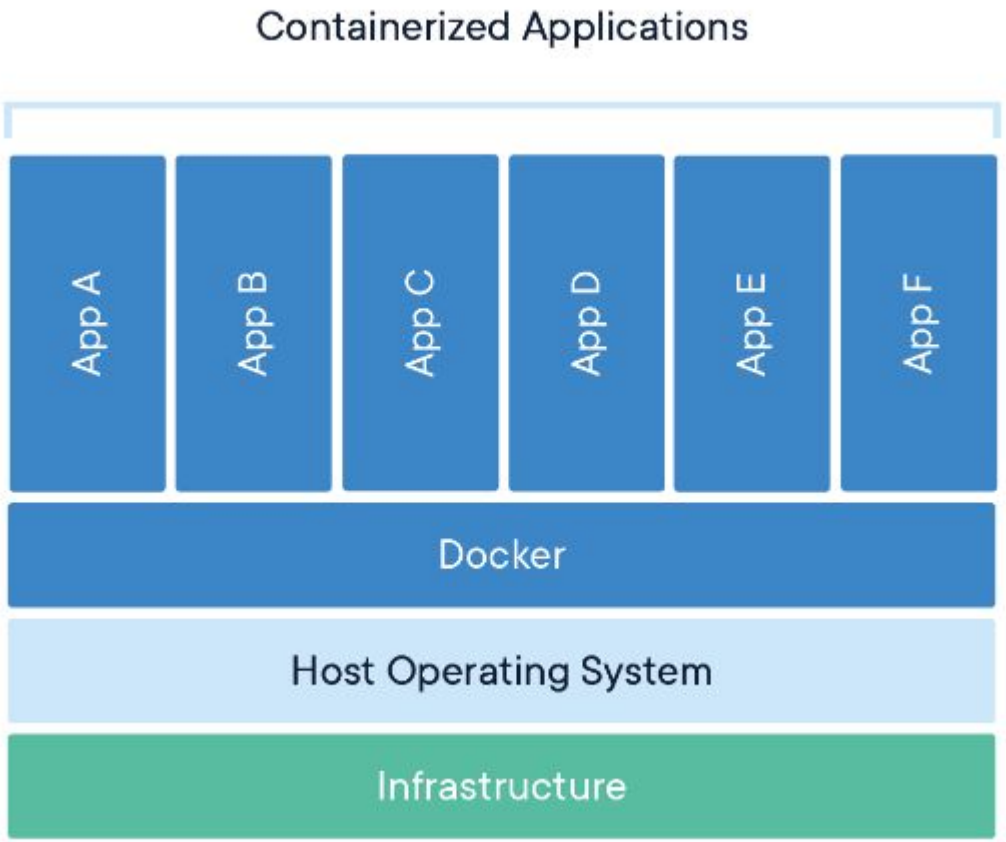


METHODIST UNIVERSITY

Linux & Containers

METHODIST UNIVERSITY

Linux & Containers

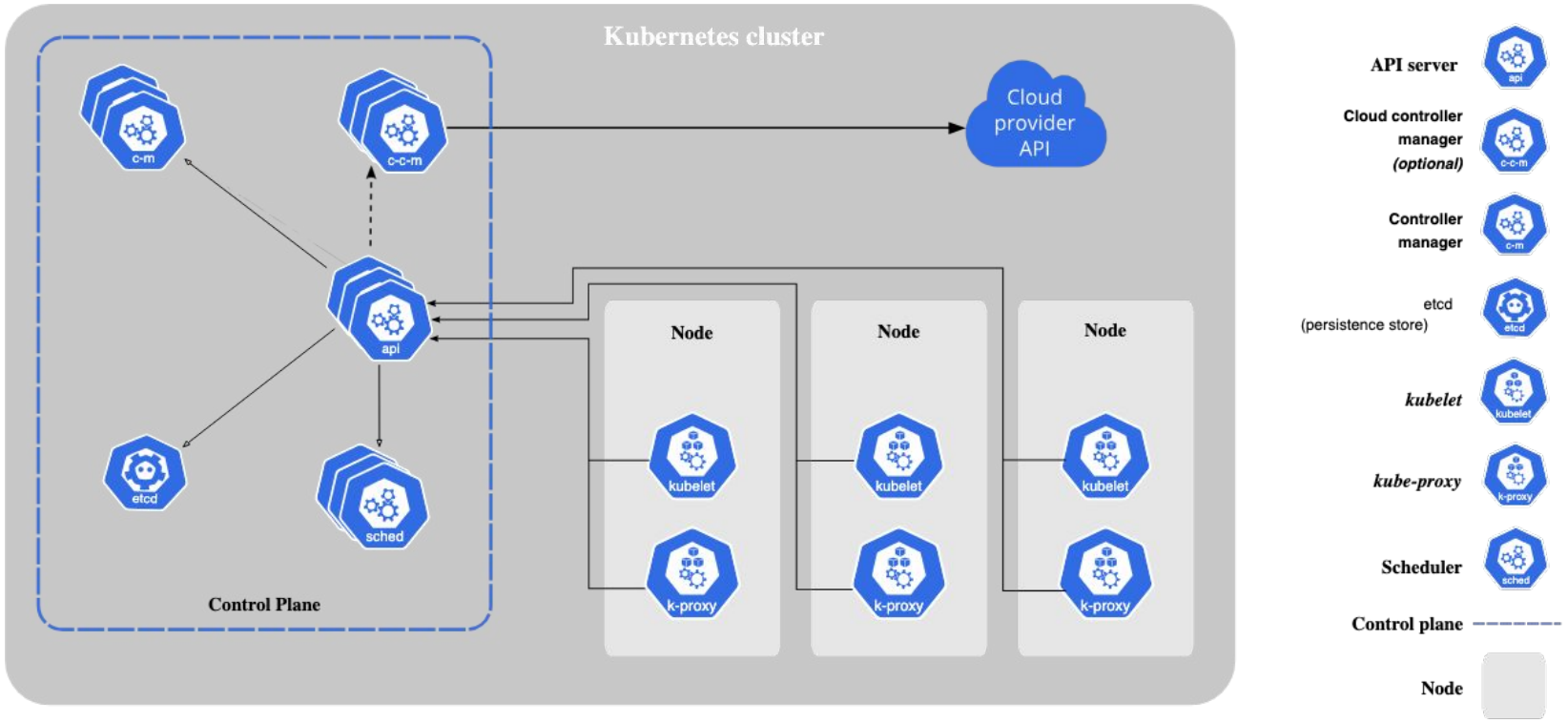


METHODIST UNIVERSITY

Linux & Containers

METHODIST UNIVERSITY

Linux & Containers



METHODIST UNIVERSITY

METHODIST UNIVERSITY

Summary

1. Contributing to existing code
2. Software debugging
3. Scripting and software testing
4. Version control
5. Building code
6. Continuous Integration/Continuous delivery
7. Linux and Containers

Slides

The slides are available at:

https://github.com/swahba/conferencetalks

METHODIST UNIVERSITY

Acknowledgements

Special thanks to my colleagues:

- Amneet Kaur
- Gail Carmichael
- Jessi Jullie

METHODIST UNIVERSITY

Questions?

METHODIST UNIVERSITY

Thank you!

@sallyky
linkedin.com/in/sallywahba/

METHODIST UNIVERSITY

Image sources referenced in speaker notes.

METHODIST UNIVERSITY