baur

A Monorepository Build Tool Written in Go

Fabian Holler,
simplesurance GmbH

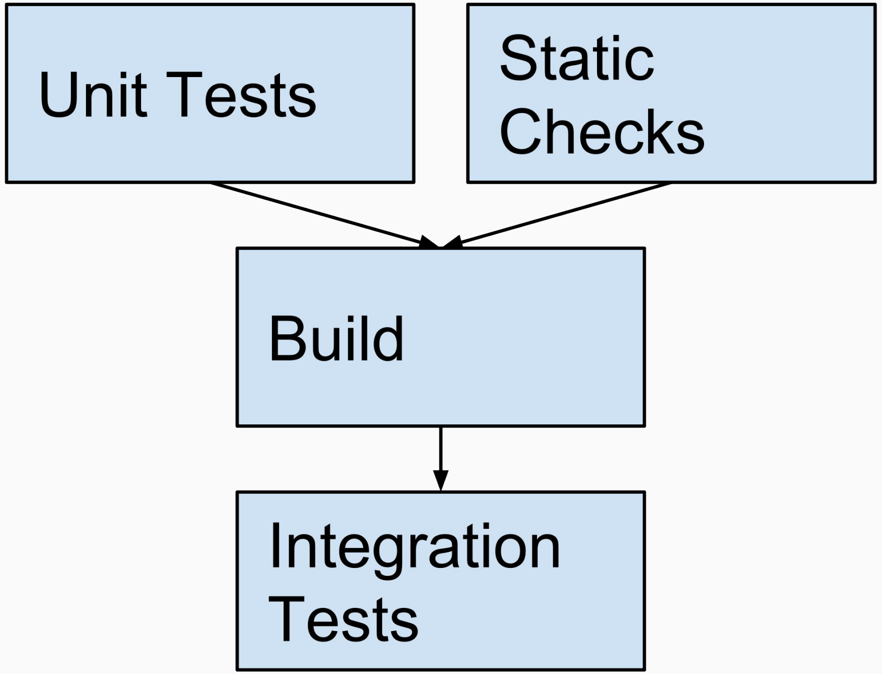
Content

- Problem to solve
- Concept
- Demo
- Future

The Past: The Big Refactoring

- Monolith => Microservices
- Monorepo

Challenge: Continuous Integration



simplesurance

Determine which applications changed and have not been built successfully in CI before

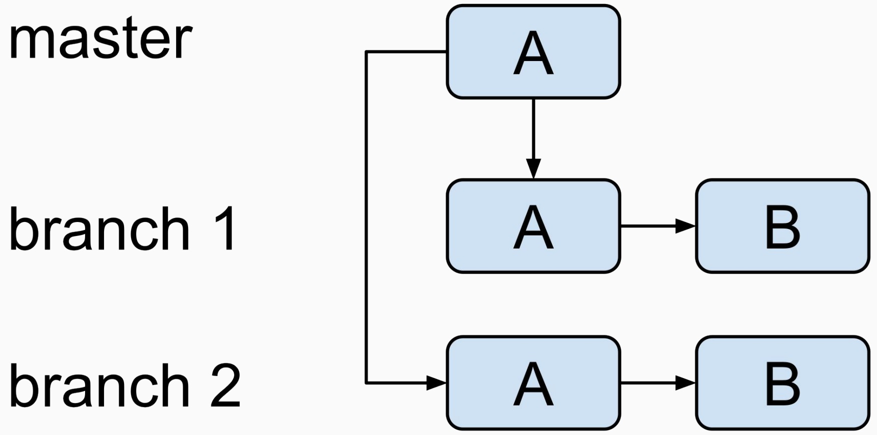
Idea #1: Use Git History (1/2)

```
$ git diff --name-only develop master

go/code/user/store/postgres/storage.go ✓
go/pb/userpb/user.pb.go ×
go/vendor/github.com/mozillazg/go-unidecode/table/x000.go ×
```

☹ Coarse Granularity

simplesurance

Idea #1: Use Git History (2/2)



☹ Missing build tracking => unnecessary rebuilds

simplesurance

Idea #2: Use next-gen Build Tools

- Bazel, Buck, Pants, Plz
  (Make on Steroid + DistCC + Ccache)

- [ ] Don’t track past builds
- [ ] Lacking support for Languages or Features

simplesurance

baur

Idea #3: baur (1/2)

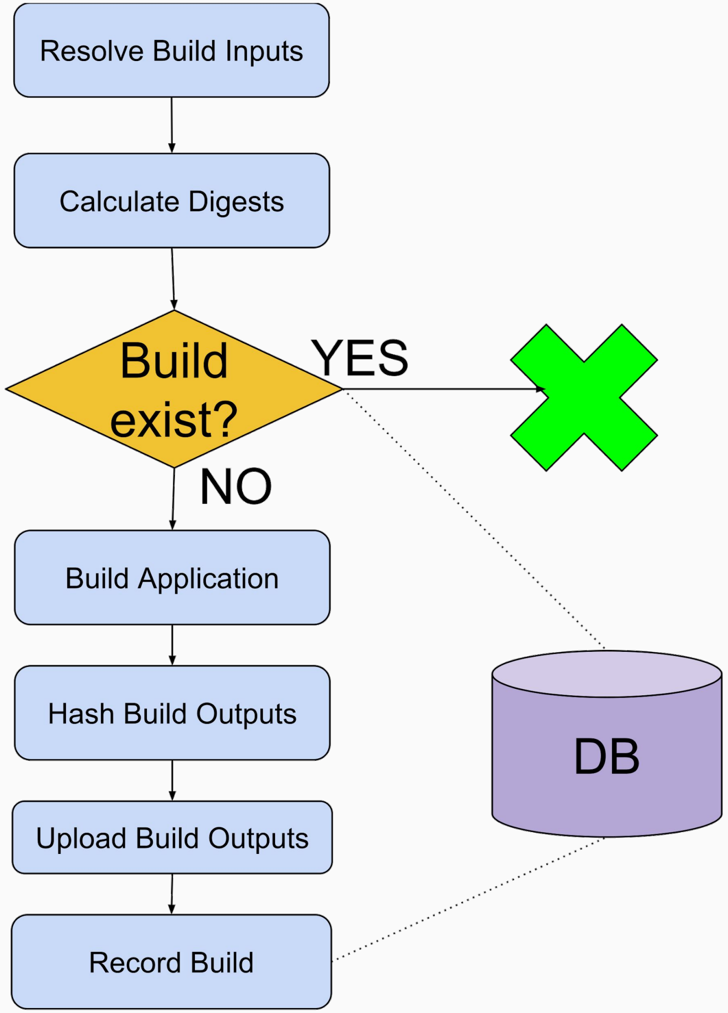
1. Discover build inputs per application
2. Uniquely identify Build Input State
3. Lookup which application in which state was build in the past

baur

Idea #3: baur (2/2)

- CLI:
  - List application states
  - Find matching build artifacts

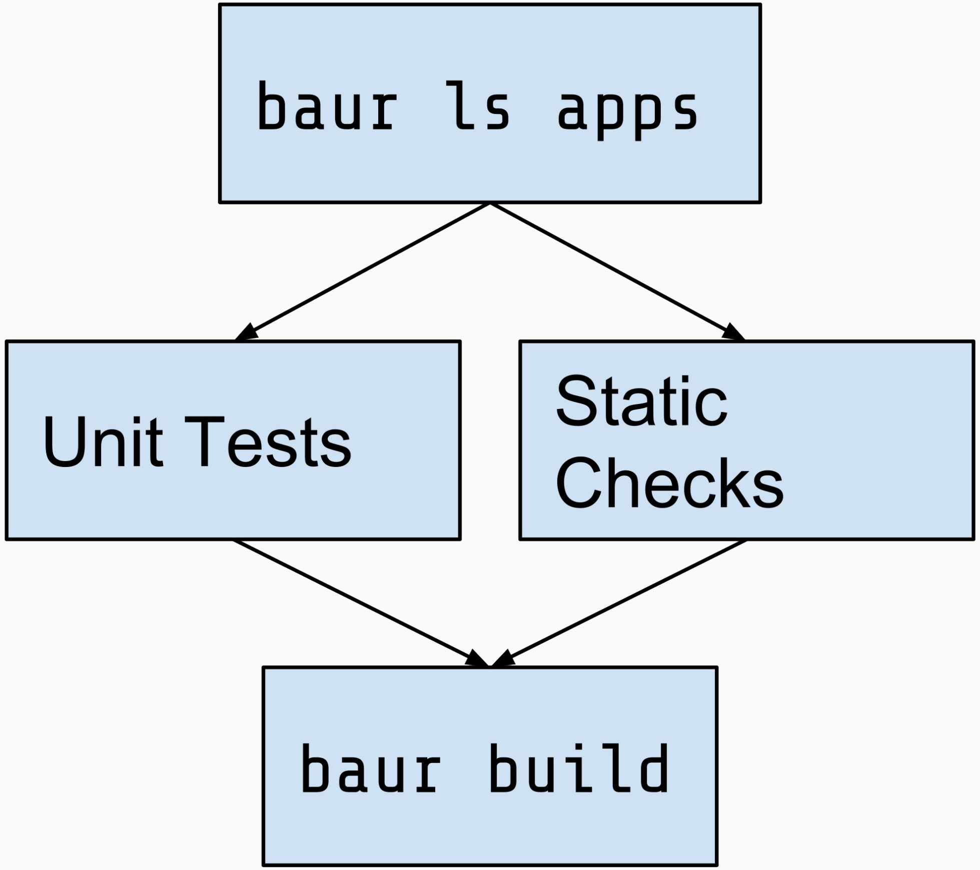
simplesurance



baur

simplesurance

Continuous integration flow with baur



baur: Repository Layout

```
.
├── .baur.toml
├── app1
│   └── .app.toml
└── app2
    └── .app.toml
```

`.baur.toml` Configuration File

```
# Version of baur configuration format
config_version = 1

[Database]
  postgresql_url = "postgres://postgres@localhost:5433/baur?sslmode=disable&connect_timeout=5"

[Discover]
  application_dirs = ["a-team/","b-team"]
  search_depth = 7
```

.app.toml Configuration File (1/2)

```
name = "user-service"

[Build]
  command = "make dist"

  [Build.Input]
      [Build.Input.Files]
      # Valid variables: $ROOT
      paths = [".app.toml"]

      [Build.Input.GitFiles]
      # Valid variables: $ROOT
      paths = ["Makefile"]

      [Build.Input.GolangSources]
      # Valid variables: $ROOT
      environment = ["GOFLAGS=-mod=vendor","GO111MODULE=on"]
      paths = ["."]
```

Build Inputs

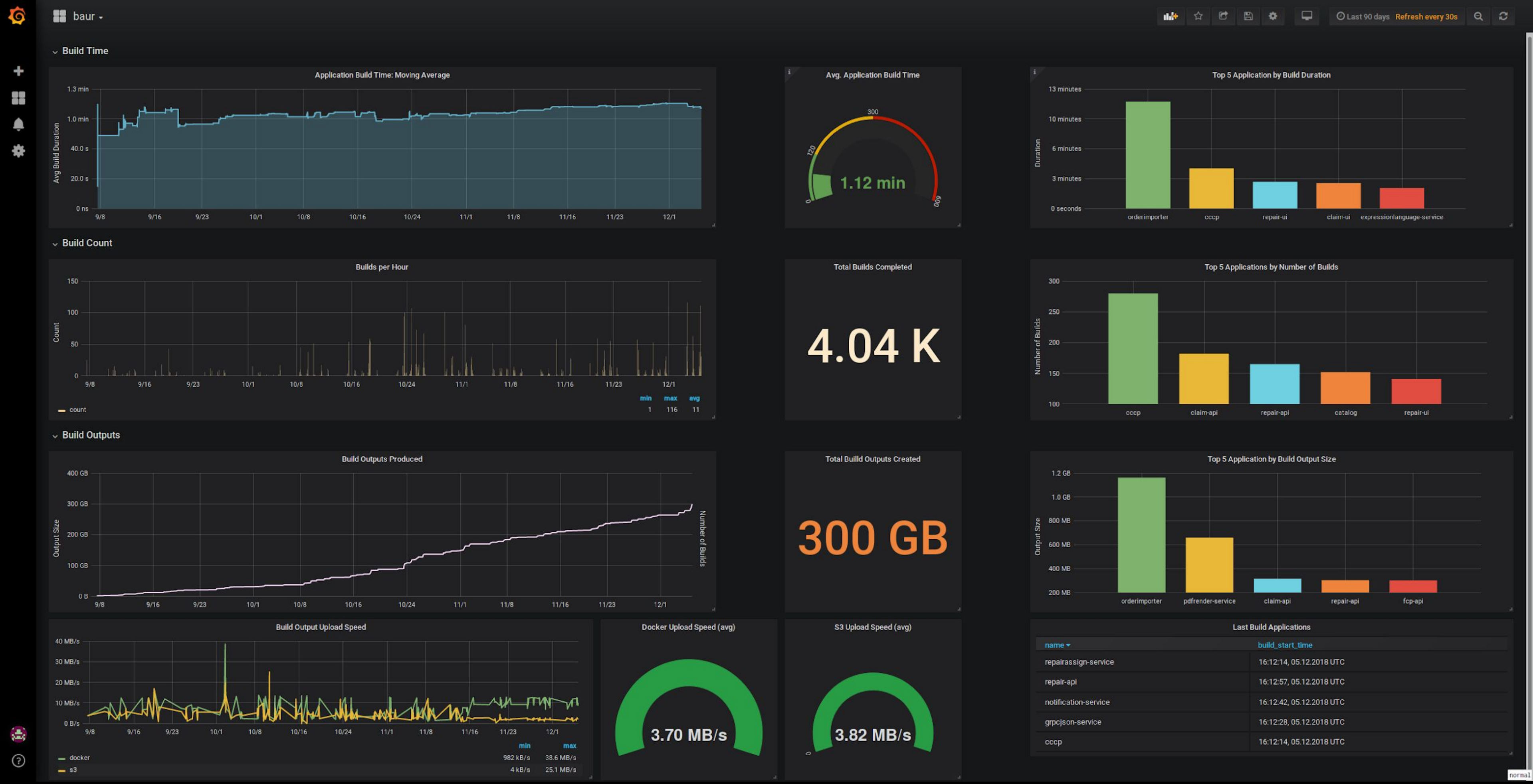
- **Crucial** to ensure correct functionality
- Build Inputs:
  - Build environment
    - Containers
  - Source files
  - Build flags
  - `.app.toml`

simplesurance

.app.toml Configuration File (2/2)

```
[Build.Output]
    [[Build.Output.DockerImage]]
    idfile = "t-container.id"

    [Build.Output.DockerImage.RegistryUpload]
    repository = "simplesurance/user-service"

    # Tag that is applied to the image, valid variables: $APPNAME, $UUID, $GITCOMMIT
    tag = "$GITCOMMIT"

    [[Build.Output.File]]
    path = "dist/t.tar.xz"

    [Build.Output.File.FileCopy]
    path = "/mnt/fileserver/build_artifacts/$APPNAME-$GITCOMMIT.tar.xz"

    [Build.Output.File.S3Upload]
    bucket = "go-artifacts/"
    dest_file = "$APPNAME-$GITCOMMIT.tar.xz"
```

baur demo



Last Build Applications

name ▾	build_start_time
repairassign-service	16:12:14, 05.12.2018 UTC
repair-api	16:12:57, 05.12.2018 UTC
notification-service	16:12:42, 05.12.2018 UTC
grpcjson-service	16:12:28, 05.12.2018 UTC
cccp	16:12:14, 05.12.2018 UTC

simplesurance

Future

- Build Inputs
  - Input Resolver for more Language
  - Restrict and monitor access to Build Input Files for `build.command`
- Release Tracking + Changelogs
- Run and Track test and check results(?)

- https://github.com/simplesurance/baur
- https://github.com/simplesurance/baur-example