Cider-CI

Multi-service integration tests

Traditional CI:

Continous *build*s

→ single shell script

→ hooks *around* that script ("before", "after", …)

Problems

→ hard to make faster/parallelize

→ hard to set up manage complex setups

Cider-CI approach

→ complex, but explicit configuration

→ very little assumptions about your workflow

→ pro: hackability

→ con: simples cases are relatively verbose

Cider-CI overview

→ **Project** has *1* **(git) Repo**

→ **Repo** contains configuration for *1*+ **Jobs**

→ each **Job** runs *1*+ **Tasks** *in parallel*

→ each **Task** runs *1*+ (shell) **Scripts** *in order*

→ *Jobs* can be **triggered** from branches and **depend** on each other

→ *Tasks* can be re-tried

→ *Scripts* can **depend** on each other

Details, quick walkthrough

(Excerpt from much longer Talk about Cider-CI)

→ HTML: **http://drtom.ch/talks/2015/CL/**

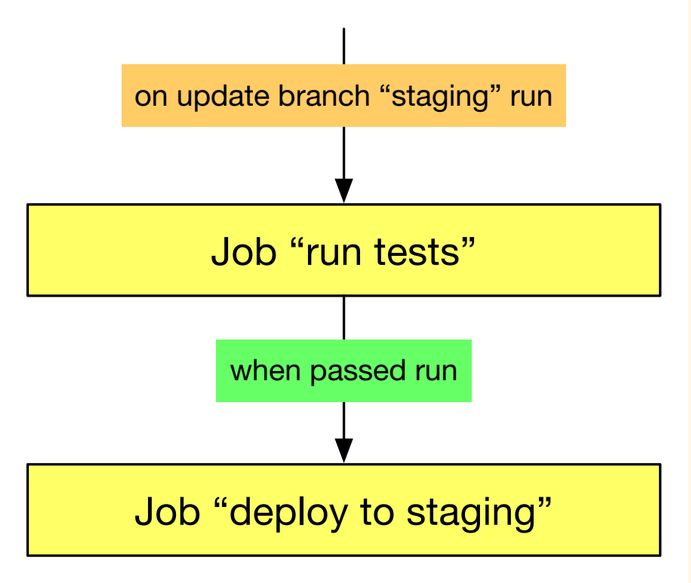
→ PDF: **http://drtom.ch/talks/2015/CL/slides.pdf**

JOBS

EXAMPLES

- run test-suite
- perform static code checks
- build
- deploy

jobs can be **triggered** and can **depend on each other**



PROJECT CONFIGURATION

`cider-ci.yml` file in the project

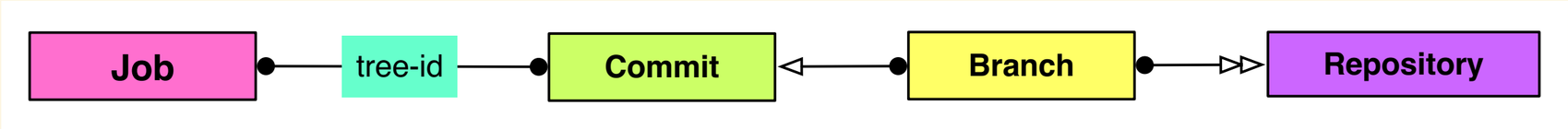
```
jobs:
  deploy_test:
    name: Deploy to test

    depends-on:
    - type: job
      job: integration-tests
      states: [passed]

    run-on:
    - type: branch
      include-match: ^master$

    # specify tasks etc
```

The source is the truth.

configuration: reproducible, reviews, audits ???

CIDER-CI AND THE SOURCE CODE

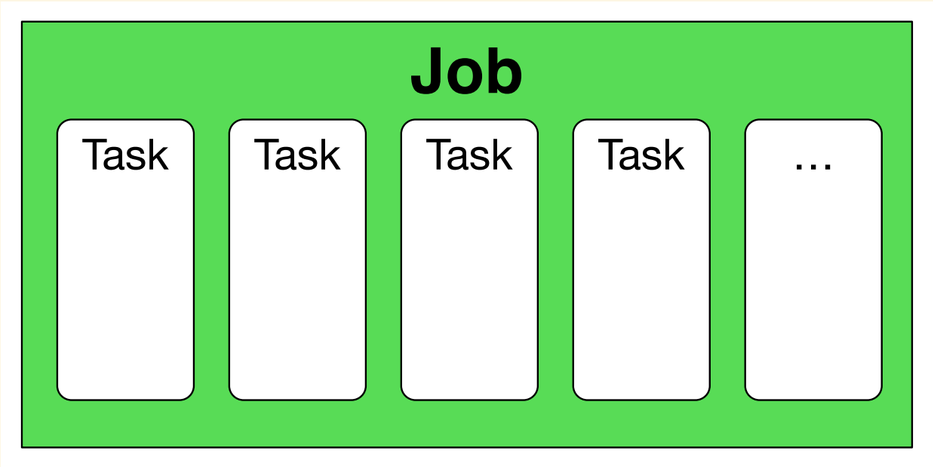
Cider-CI "knows" about commits, branches, submodules,…



`tree-id`: fingerprint of your source code

- **reproducibility**
- jobs can be **run at any time** (later)
- **binary search** for "bad" commits
- commit amends, squashing: **existing job remains valid**

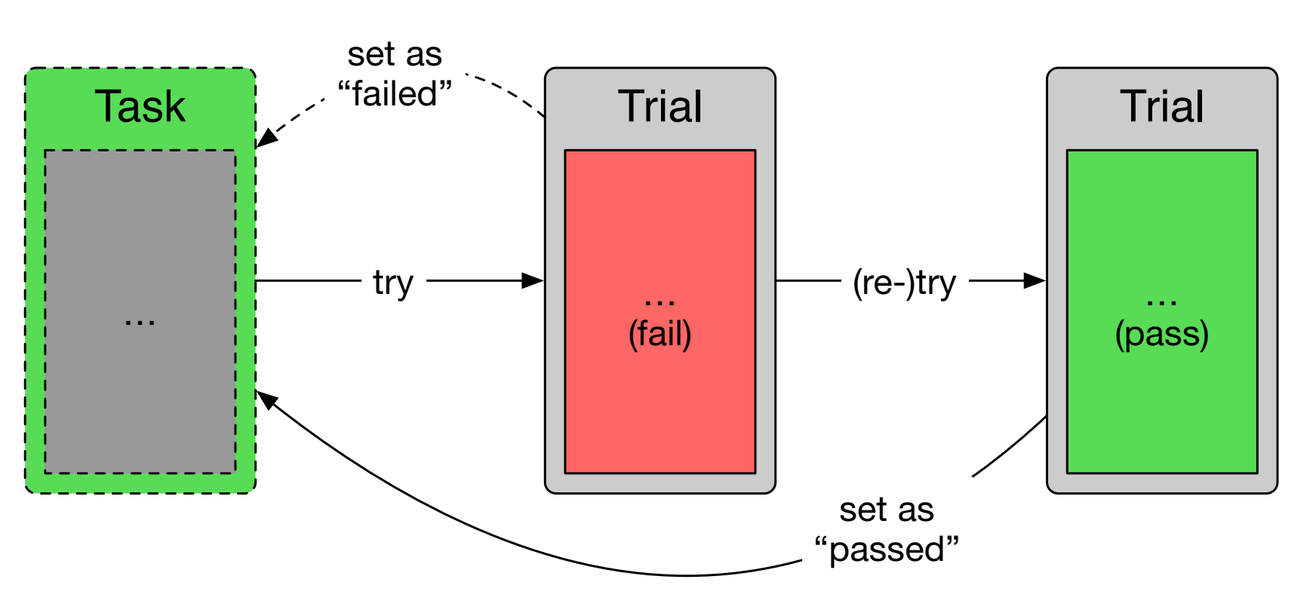
JOBS & TASKS



job: container and state aggregate for tasks

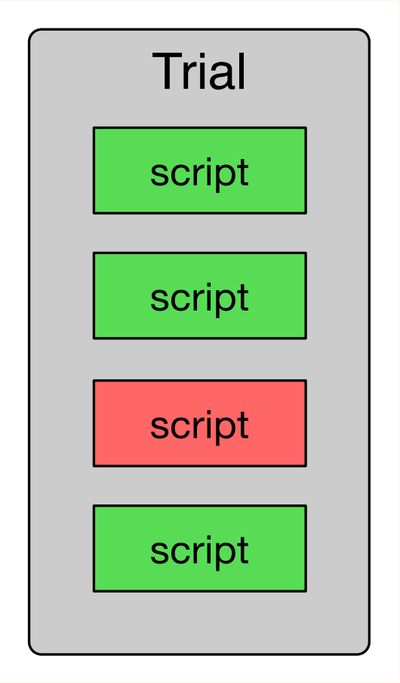
→ **parallelization**

TASKS & TRIALS



- **blueprint**
- container and state aggregate for trials

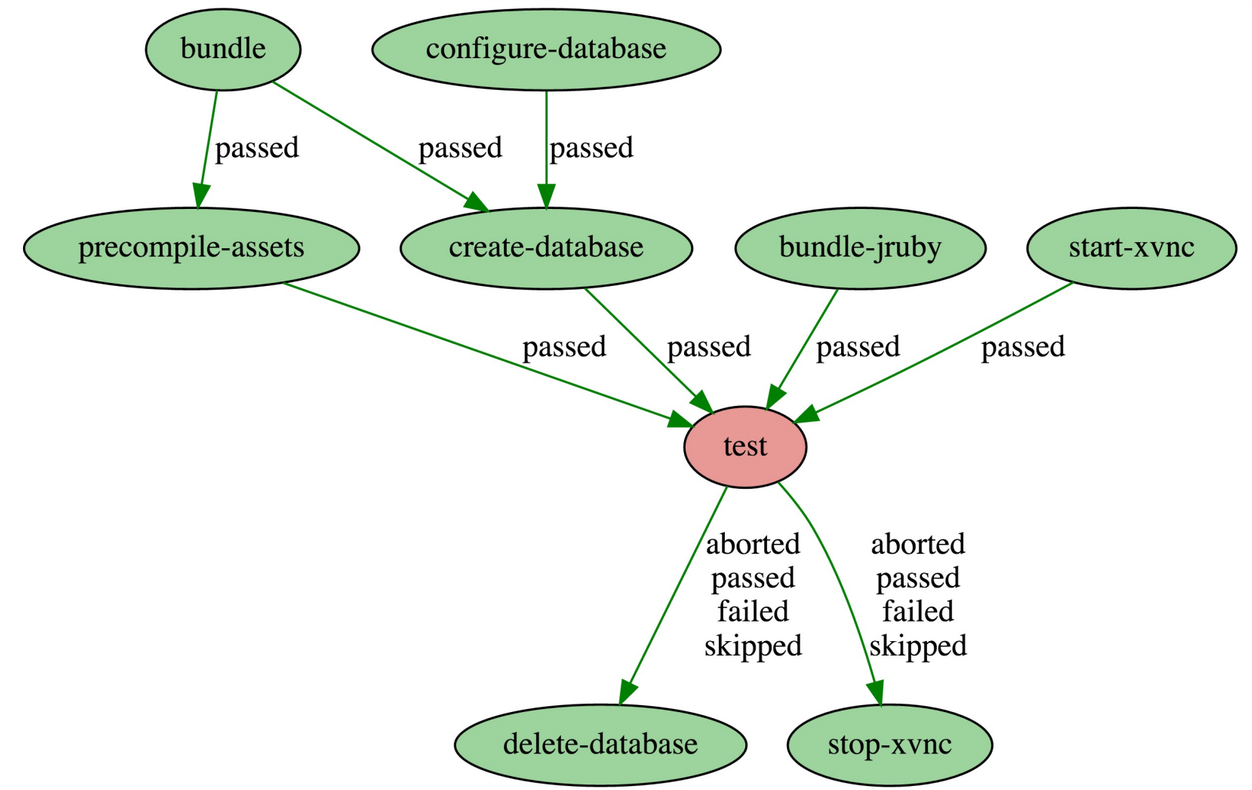
→ resilience



TRIAL & SCRIPTS

- actual unit of execution
- executed in the **same context**
- **depend** on each other

SCRIPT DEPENDENCIES



- traditional CI: one "build" ⇔ one script
- more modern: one main script + before and after "hooks"
- Cider-CI: **scripts with dependencies**

What is it good at?

→ **fast**: run lots of tasks in parallel & *retry* them

- *instead of* `sh tests/*`

→ **declarative** dependencies for tasks

- *instead of, or like very flexible "hooks"*

→ continous delivery: trigger and run different kinds of jobs

- *instead of* `make test && make build && …`
- Job "Test", triggers Job "Release", triggers "Deploy", …
- Job: **"Good To Merge"**
  - *depends on* "Lint", "Unit Test", "Feature Tests"

What does it not do?

→ **Access management**

- always trusts the repository, control (push) access there

→ **Secrets management**

- set up your own infrastructure and/or excutors

→ *but:* executors can be told to only accept code from "blessed" repos

Complex example (CI-ception)

Trial for the task *spec/features/attachments_spec.rb* in job *Integration-Tests* of *Cider-CI*

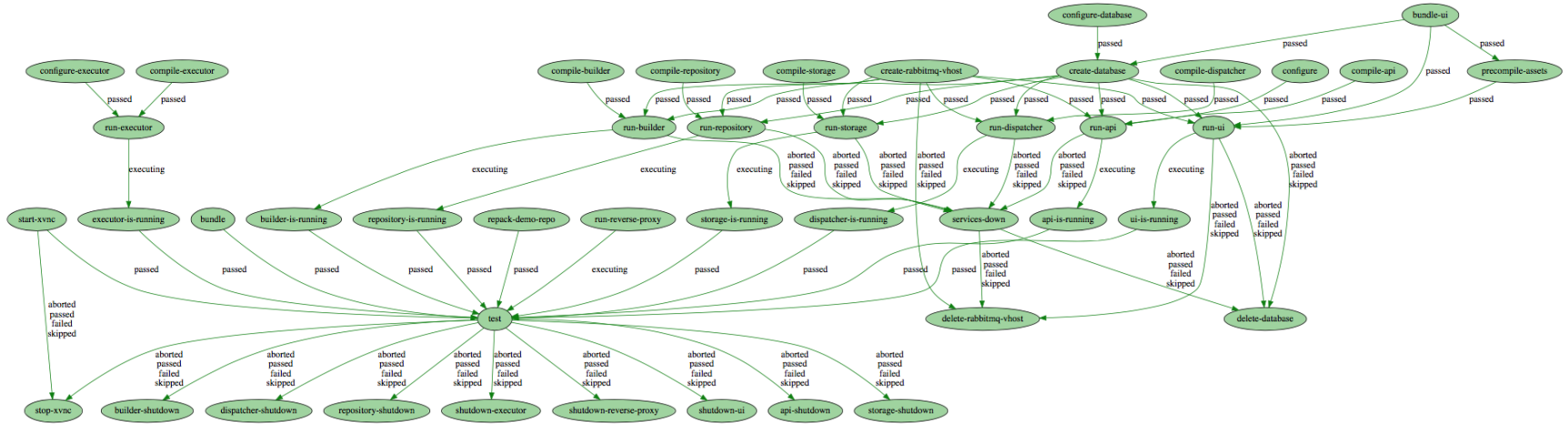
Retry

passed created **14 days ago on Sunday, 29th November** started **14 days ago on Sunday, 29th November** executed in **3 minutes** (167.83 seconds) on **cislave5**

Scripts Overview

configure | run-reverse-proxy | configure-database | start-xvnc | create-rabbitmq-vhost | configure-executor | repack-demo-repo | compile-storage | compile-builder | compile-api | compile-dispatcher | compile-executor | compile-repository | run-executor | executor-is-running | bundle | bundle-ui | create-database | precompile-assets | run-ui | run-api | run-storage | ui-is-running | api-is-running | run-repository | run-dispatcher | run-builder | storage-is-running | builder-is-running | repository-is-running | dispatcher-is-running | test | shutdown-ui | builder-shutdown | shutdown-reverse-proxy | dispatcher-shutdown | stop-xvnc | storage-shutdown | shutdown-executor | api-shutdown | repository-shutdown | services-down | delete-database | delete-rabbitmq-vhost

Start-Dependencies



Terminate-Dependencies

Cider-CI tests itself

API

REST-ful API to implement any workflow you want

→ "nightly" builds and deploys

→ integrate with external services

Try it out

Try Cider-CI, open source, installs with two commands:

→ **http://docs.cider-ci.info/introduction/quick-start/**

→ or read the sources: **https://github.com/cider-ci/cider-ci**

Run either a single instance for your organization or one per project.

(Re-) use your existing infrastructure or run on-demand on AWS.

Or come talk to me on Day 2 and 3.
See the wiki, but most likely I will be at the freifunk assembly.

THX!