Concourse CI

Mario Fernandez

Outline

Initially...

🔗 Basics

× CLI

🤔 Pipelines

✓ UI

Initially...

thousands of lines of a self written groovy DSL

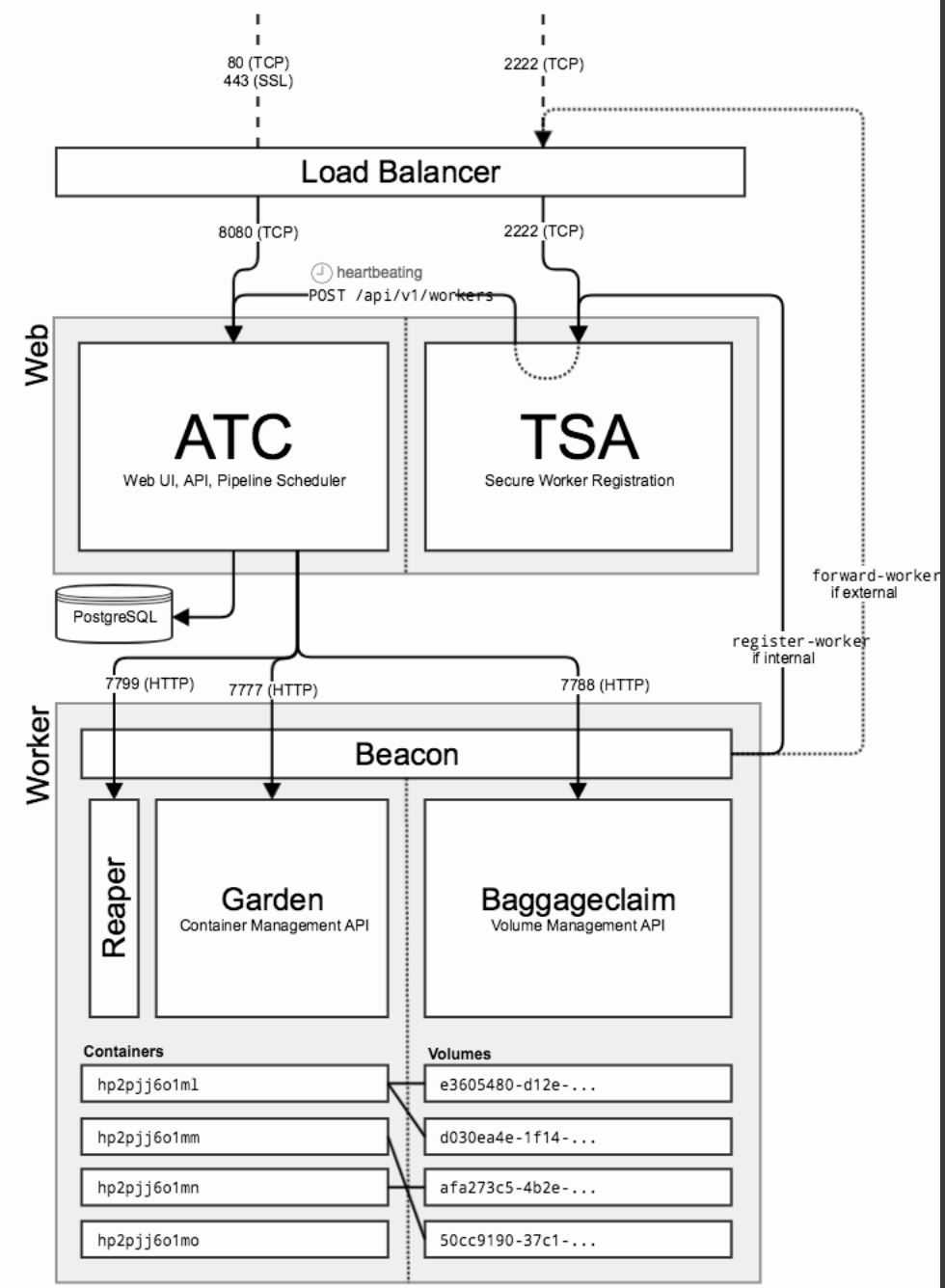
4 test systems

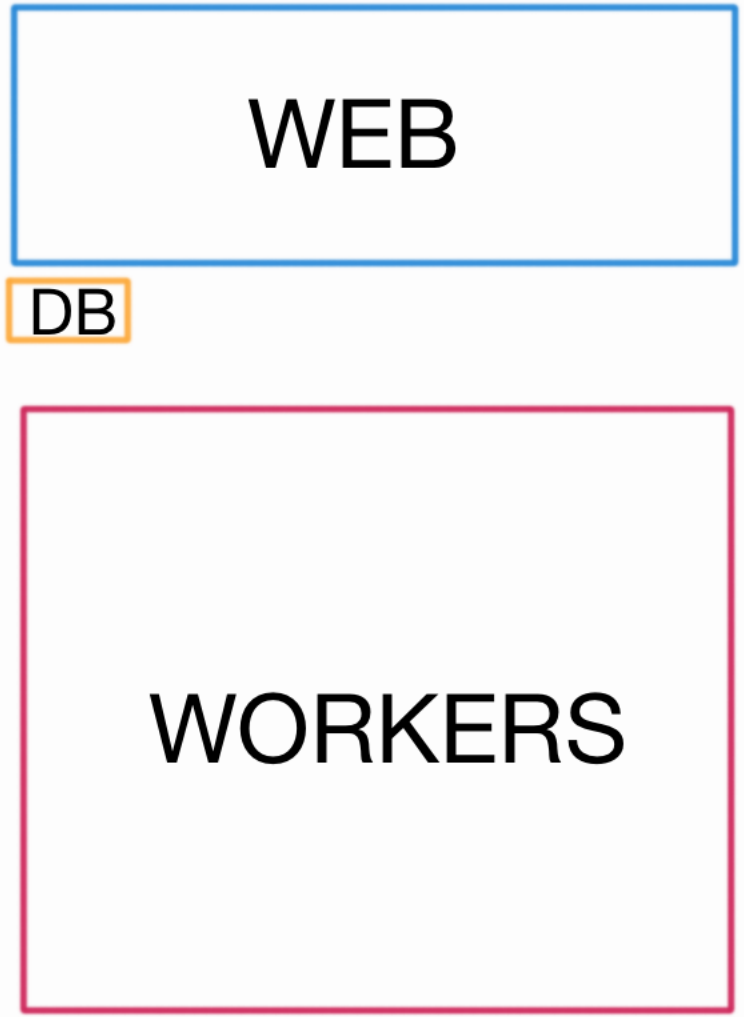
self written deployment framework based on Chef

decided to explore options, and chose Concourse

The basics

Written in Go


80 (TCP)
443 (SSL)
2222 (TCP)
Load Balancer
8080 (TCP)
2222 (TCP)
heartbeating
POST /api/v1/workers
Web
ATC
Web UI, API, Pipeline Scheduler
TSA
Secure Worker Registration
PostgreSQL
forward-worker
if external
register-worker
if internal
7799 (HTTP)
7777 (HTTP)
7788 (HTTP)
Worker
Beacon
Reaper
Garden
Container Management API
Baggageclaim
Volume Management API
Containers
Volumes
hp2pjj6o1ml
hp2pjj6o1mm
hp2pjj6o1mn
hp2pjj6o1mo
e3605480-d12e-...
d030ea4e-1f14-...
afa273c5-4b2e-...
50cc9190-37c1-...


WEB
DB
WORKERS

everything runs in containers

yaml based

take it or leave it approach

CLI

fly

Easily available

```
FROM alpine:3.8

ENV SHA1='7b0805adfba328d09cfa7ac377777cb6e270d66f' \
    VERSION='4.2.1' \
    URL='https://github.com/concourse/concourse/releases/download'

RUN apk update && \
    apk --no-cache add curl bash

SHELL ["/bin/bash", "-o", "pipefail", "-c"]

RUN curl -Lk "${URL}/v${VERSION}/fly_linux_amd64" -o /usr/bin/fly && \
    echo "${SHA1}  /usr/bin/fly" | sha1sum -c - && \
    chmod +x /usr/bin/fly
```

Update pipeline

```
version 4.2.1 already matches; skipping
jobs:

  job audit has been removed:
  - name: audit
  - serial: true
  - plan:
  - - aggregate:
  -   - get: git
  -   - get: shared-tasks
  - - task: audit
  -   file: shared-tasks/tasks/yarn/audit/task.yml
  -   image: dev-container

configuration updated
```

pipelines don't update automatically

Status

```
$ fly -t ci workers

name               containers platform tags team state   version
ip-1.aws.compute   14         linux    none none running 2.1
ip-2.aws.compute   10         linux    none none running 2.1
ip-3.aws.compute   18         linux    none none running 2.1


the following workers have not checked in recently:

name               containers platform tags team state   version
ip-4.aws.compute   1          linux    none none running 2.1

these stalled workers can be cleaned up by running:

    fly -t ci prune-worker -w (name)
```

Hijack

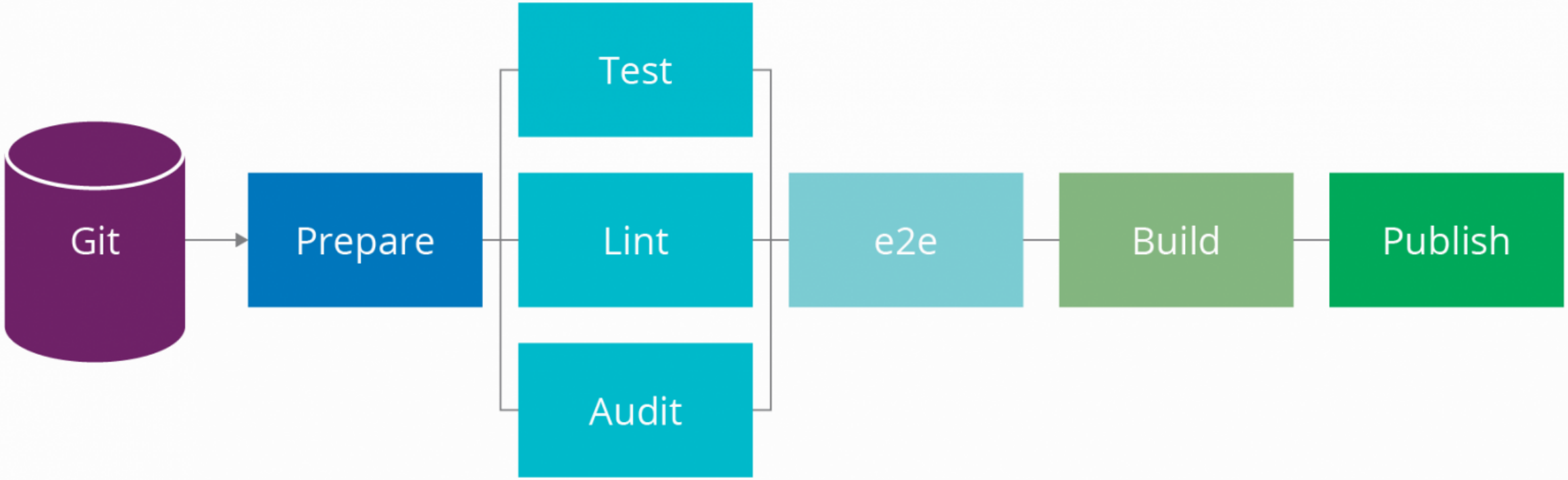
```
$ fly -t concourse hijack -j service-api/test
1: build #95, step: test-integration, type: task
2: build #95, step: test-unit, type: task
choose a container: 2
bash-4.3# pwd
/tmp/build/1944c6a2
bash-4.3# cd git
bash-4.3# ./go test-unit
```

Hijack won't work if the container does not have bash

And many more

```
Available commands:
  abort-build           Abort a build (aliases: ab)
  builds                List builds data (aliases: bs)
  check-resource        Check a resource (aliases: cr)
  check-resource-type   Check a resource-type (aliases: crt)
  checklist             Print a Checkfile of the given pipeline (aliases:
  clear-task-cache      Clears cache from a task container (aliases: ctc)
  containers            Print the active containers (aliases: cs)
  destroy-pipeline      Destroy a pipeline (aliases: dp)
  destroy-team          Destroy a team and delete all of its data (aliase
  execute               Execute a one-off build using local bits (aliases
  (...)
```

Defining pipelines


Git
Prepare
Test
Lint
Audit
e2e
Build
Publish

Building blocks

- Resources
- Jobs
- Tasks

Resources

artifacts used in the pipeline

fetched from and pushed to

Types

- git
- docker
- s3
- ...

https://concourse-ci.org/included-resources.html

```
resources:
- name: git
  type: git
  source:
    uri: "git@github.com:sirech/example-concourse-pipeline.git"
    branch: master

- name: dev-container
  type: docker-image
  source:
    repository: "AWS_ACCOUNT.aws.com/dev-container"
```

extensibility through new resource types

it is the only way to store artifacts in concourse

Jobs

Jobs determine the actions of your pipeline

```yaml
- name: lint
  serial: true
  plan:
  - aggregate:
    - get: git
      passed: [prepare]
      trigger: true
    - get: dev-container
      passed: [prepare]
  - aggregate:
    - task: lint-sh
      image: dev-container
      params:
        <<: *common-params
        TARGET: sh
      file: "git/pipeline/tasks/linter/task.yml"
```

you cannot pass anything between jobs

full pipeline cannot be retriggered

Tasks

```
- task: lint-js
  image: dev-container
  params:
    NPM_TOKEN: ((npm_auth_token))
    TARGET: js
  file: "git/pipeline/tasks/linter/task.yml"
```

```
platform: linux
inputs:
  - name: git
caches:
  - path: "git/node_modules"
params:
  TARGET:
run:
  path: "pipeline/tasks/linter/task.sh"
  dir: git
```

WORKDIR, ENTRYPOINT, USER, CMD are ignored

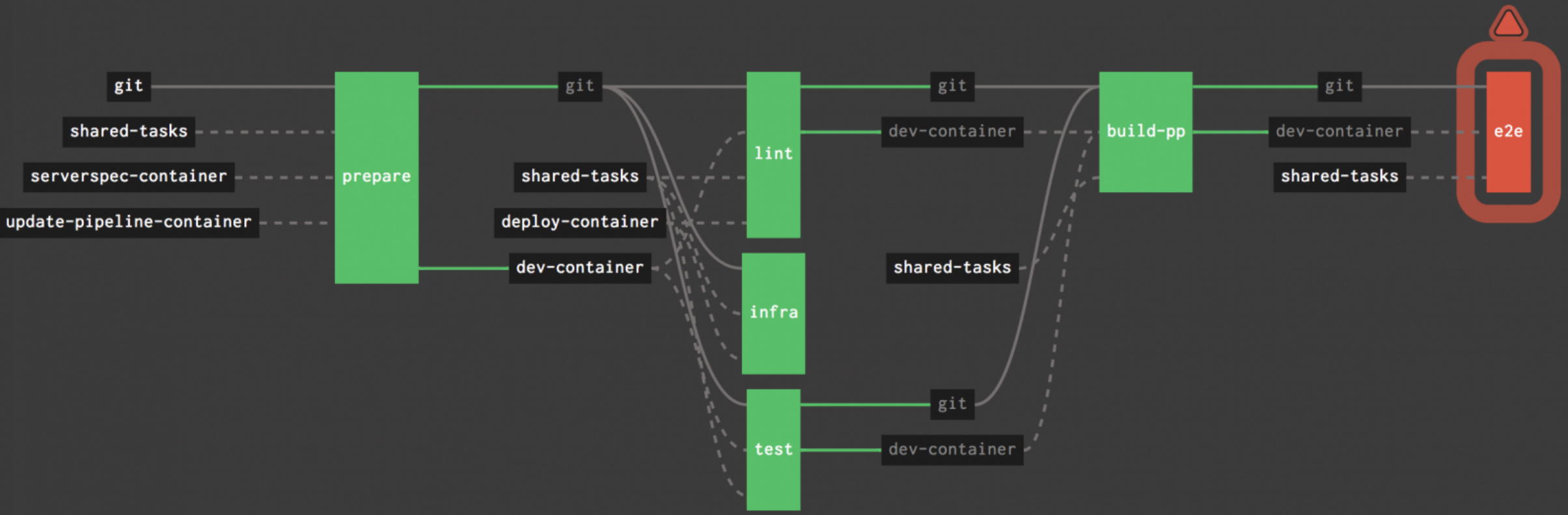
```
#!/bin/sh

set -e

yarn
./go "linter-${TARGET}"
```

reuse through shared tasks

shared containers

no arbitrary code

The UI


git
shared-tasks
serverspec-container
update-pipeline-container
prepare
git
shared-tasks
deploy-container
dev-container
lint
infra
test
git
dev-container
shared-tasks
git
dev-container
build-pp
git
dev-container
shared-tasks
e2e

search

main MEMBER

infrastructure-base

4d 18h

containers

28d 17h

traefik

4d 15h

slice1-ui

2d 19h

slice2-api

23h 36m

slice2-ui

22h 44m

slice2-api

15h 30m

slice3-api

21h 37m

code driven

isolation

convenience (paralellization, caching)

limiting (in a good way)

cannot save artifacts

does not update pipelines on push

ignores some Docker directives

sparse documentation

operational burden

https://www.thoughtworks.com/insights/blog/modernizing-your-build-pipelines

https://github.com/sirech/example-concourse-pipeline