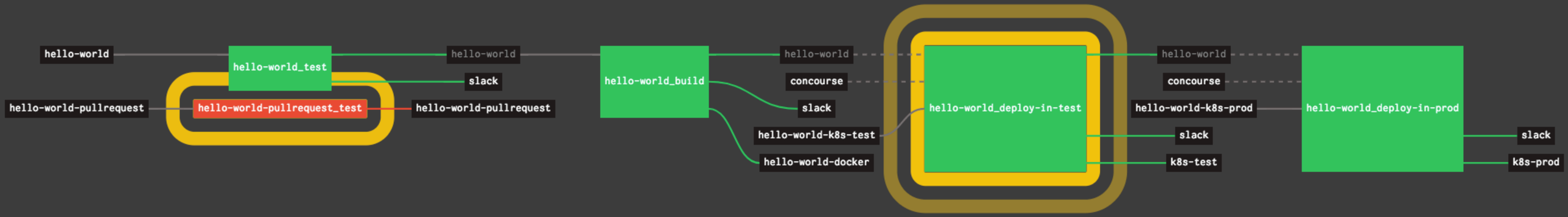
Concourse
CI с кубиками и на чистом YAML



Devino Telecom

Максим Залысин

succeeded
errored
aborted
paused
failed
pending
started
. dependency
_ dependency (trigger)

cli:
version: v4.2.1

codefresh

integrity

circleci

Strider CD
Open Source Continuous Integration & Deployment Server

CODESHIP

semaphore

Jenkins

GitLab

Bamboo

TC

NEVERCODE

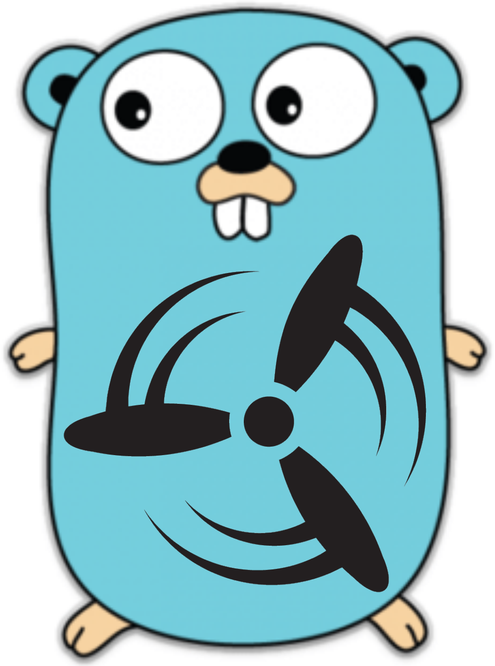
Travis CI

go

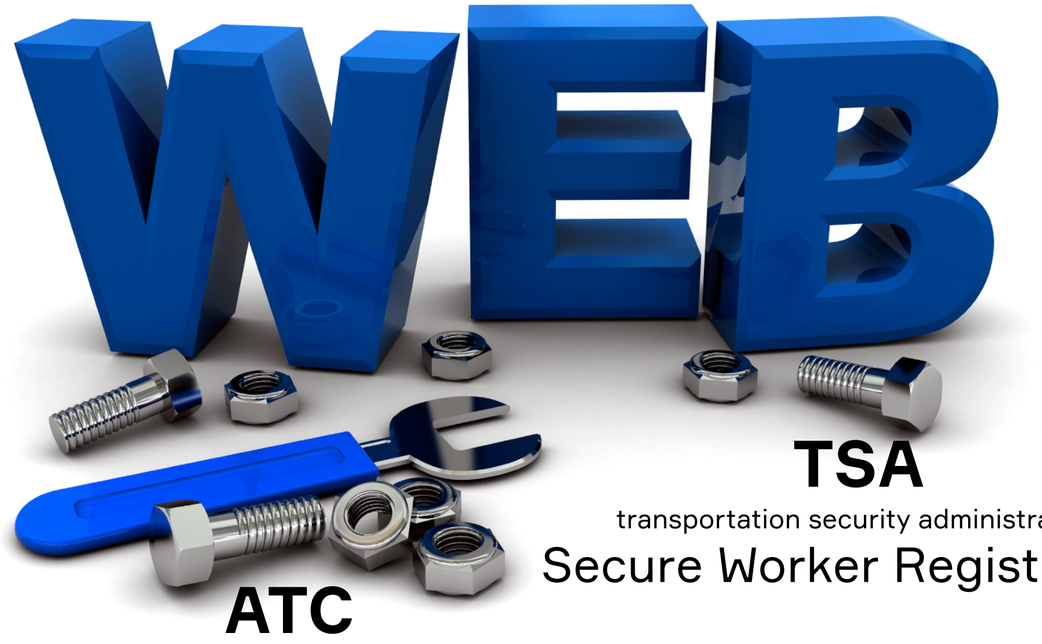

codefresh
integrity
circleci
Strider CD
Open Source Continuous Integration & Deployment Server
CODESHIP
Jenkins
Bamboo
TC
NEVERCODE
Travis CI
go

О чем доклад?

- Как устроен Concourse CI
- Для чего CLI-утилита fly?
- Что делает Concourse CI в Devino Telecom?
- Что такое pipeline в Concourse?
- Для чего resource?
- Что происходит в job`е?



Go



TSA

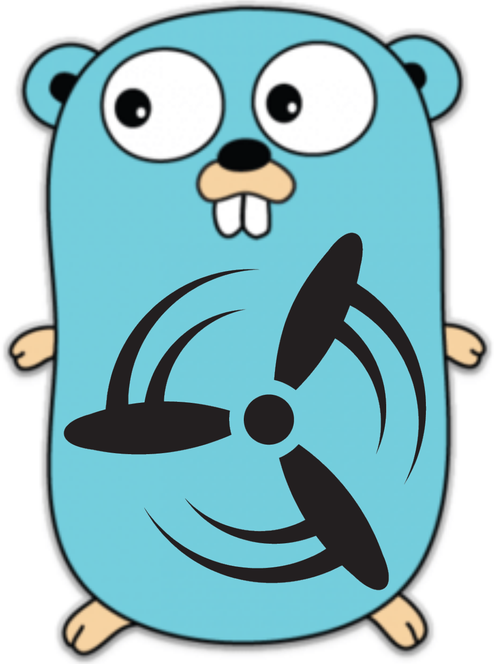
transportation security administration

Secure Worker Registration

ATC

air traffic control

Web UI, API, Pipeline Scheduler



Go

TSA

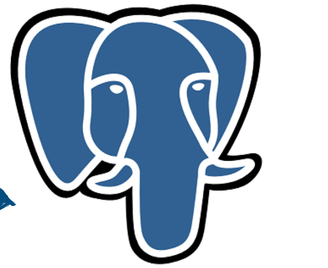
transportation security administration

Secure Worker Registration

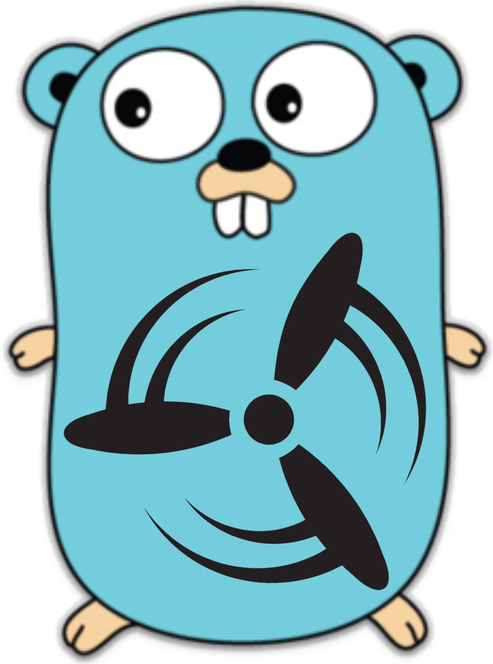
ATC

air traffic control

Web UI, API, Pipeline Scheduler



PostgreSQL



Go

TSA

transportation security administration

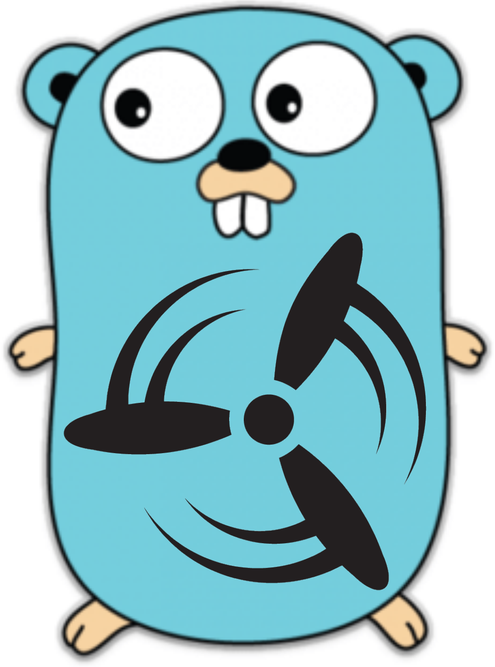
Secure Worker Registration

ATC

air traffic control

Web UI, API, Pipeline Scheduler

PostgreSQL



Go

TSA

transportation security administration

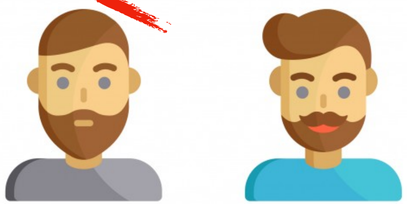
Secure Worker Registration

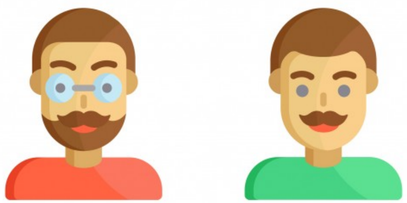
ATC

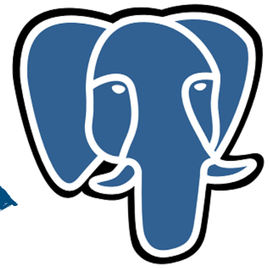
air traffic control

Web UI, API, Pipeline Scheduler

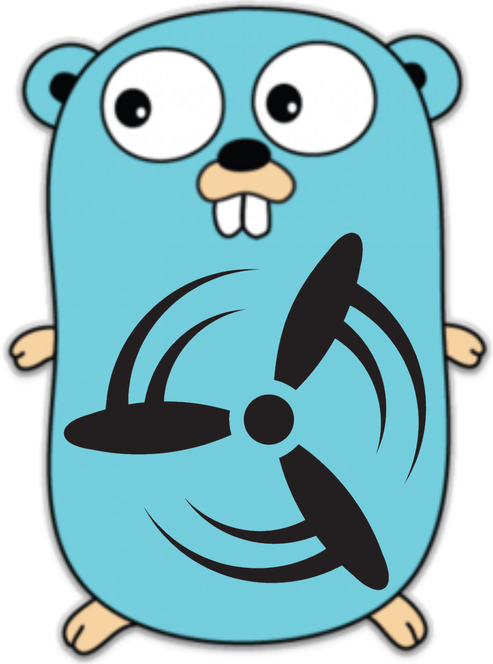
fly







PostgreSQL



Go

TSA

transportation security administration

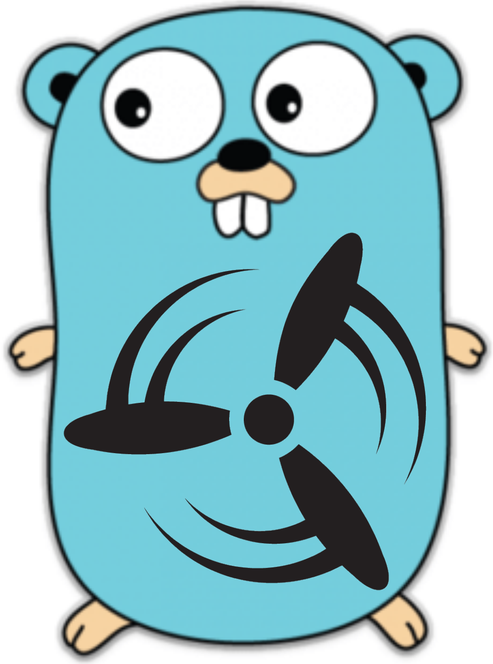
Secure Worker Registration

ATC

air traffic control

Web UI, API, Pipeline Scheduler

fly



Go

Для чего CLI-утилита fly?

- Управление team`ами

```
$ fly -t main set-team --team-name=develop --ldap-group=…
```

- Управление pipeline`ами

```
$ fly -t develop --pipeline=HELLO-WORLD --config=pipelines/hello-world.yml
```

- Тестирование task`ов

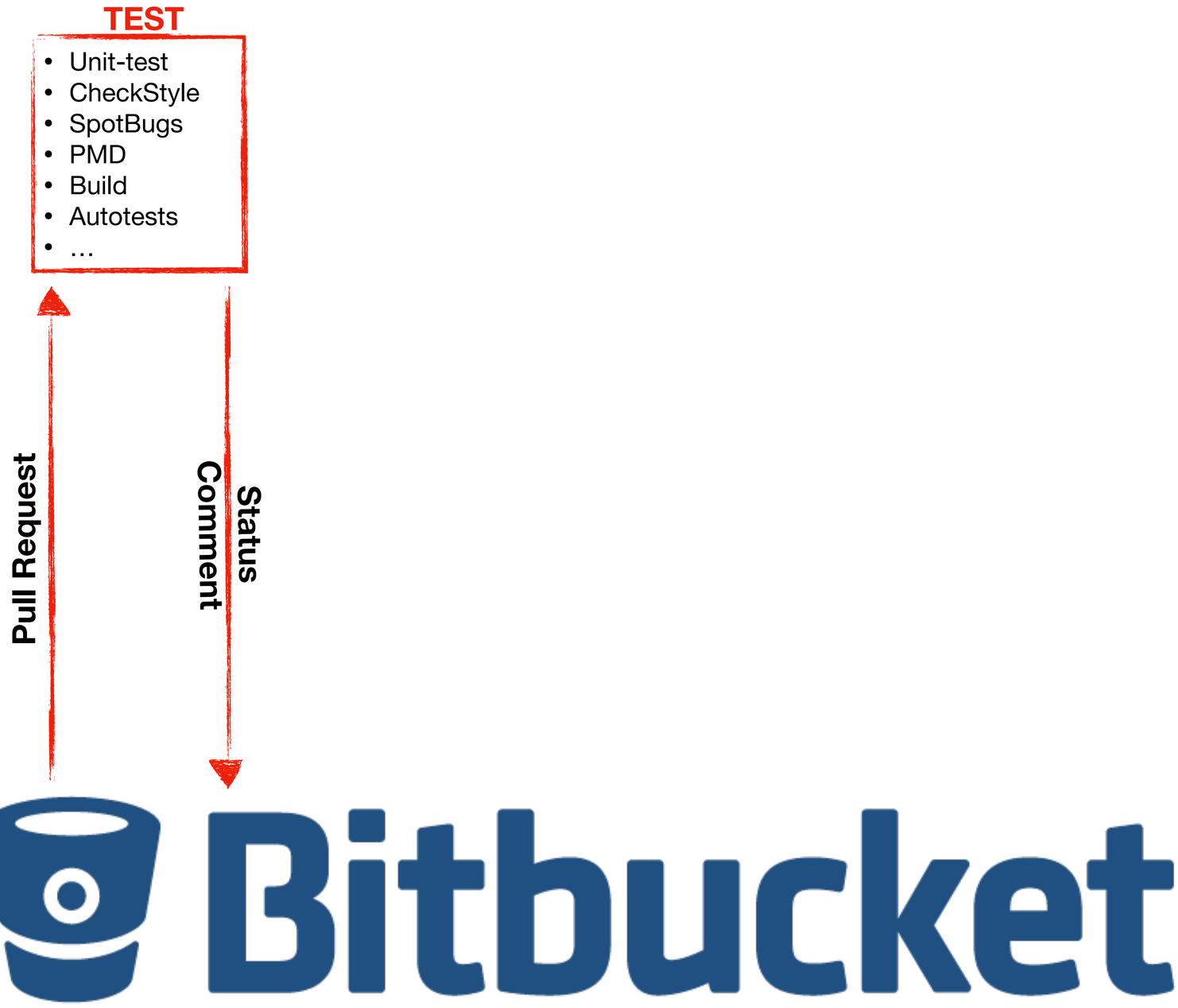
```
$ fly -t develop execute --config=ci/build.yml --input=source=./
```

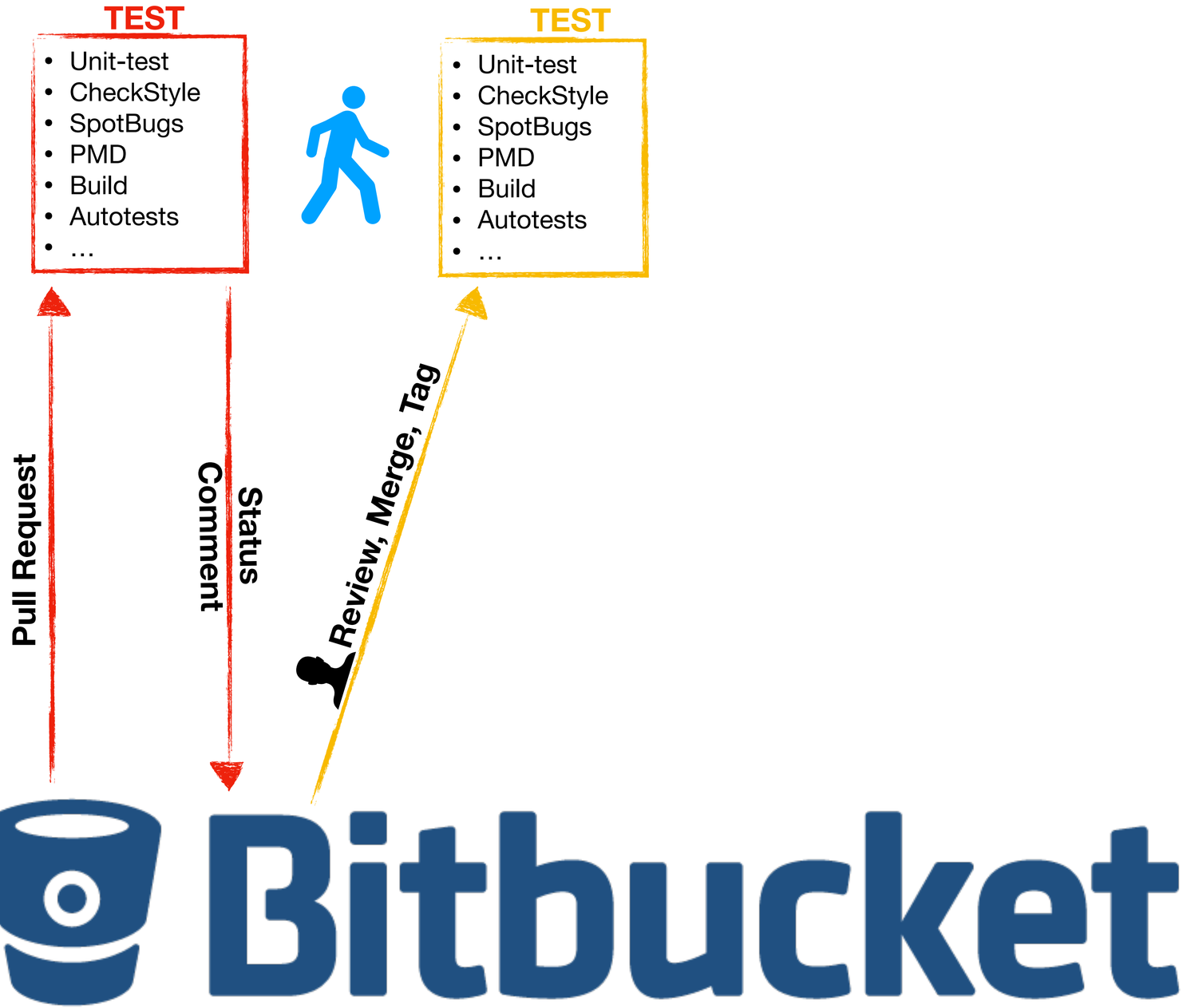
- Отладка работающего task`а

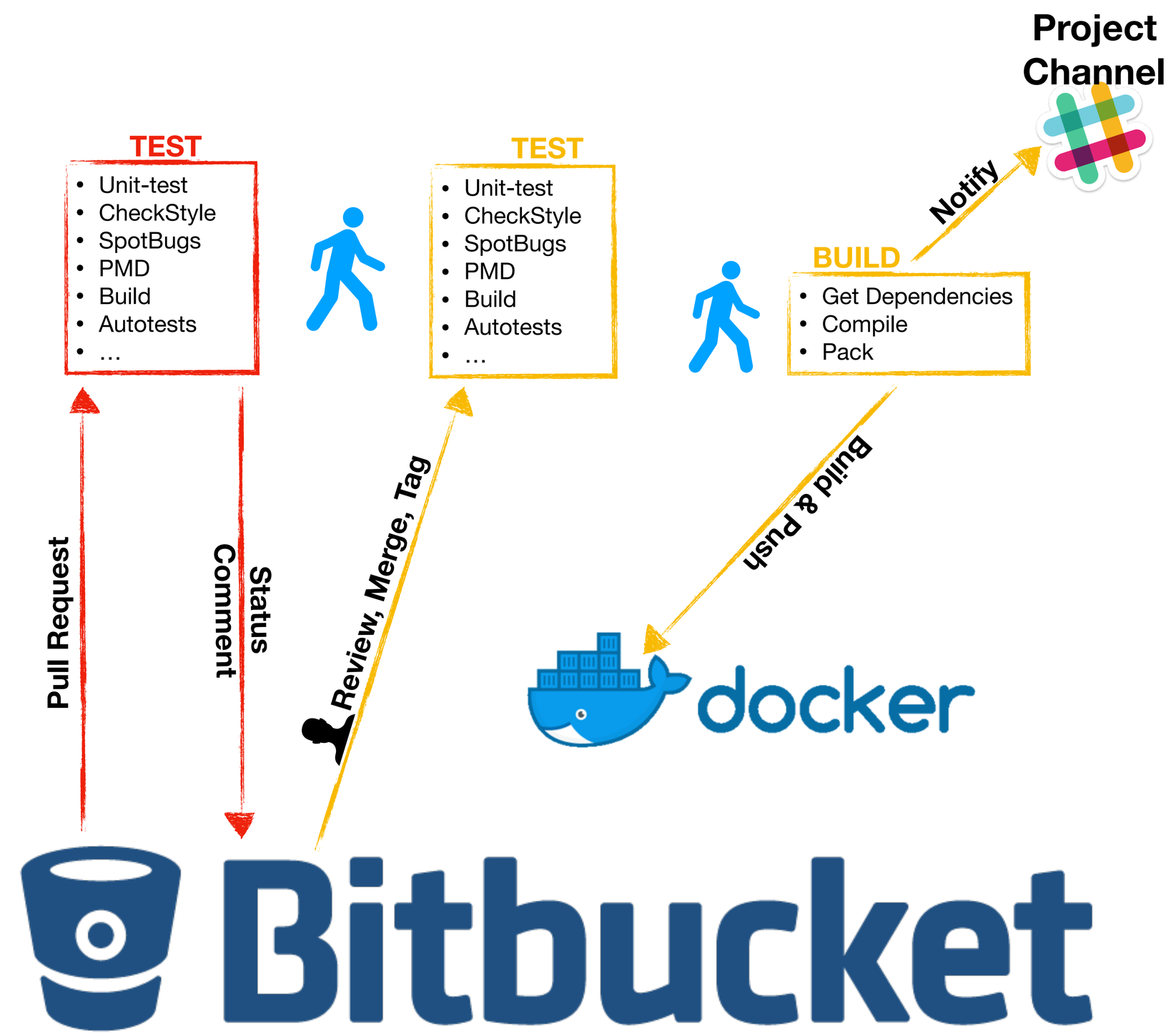
```
$ fly -t develop hijack --job=HELLO-WORLD/hello-world_build
```

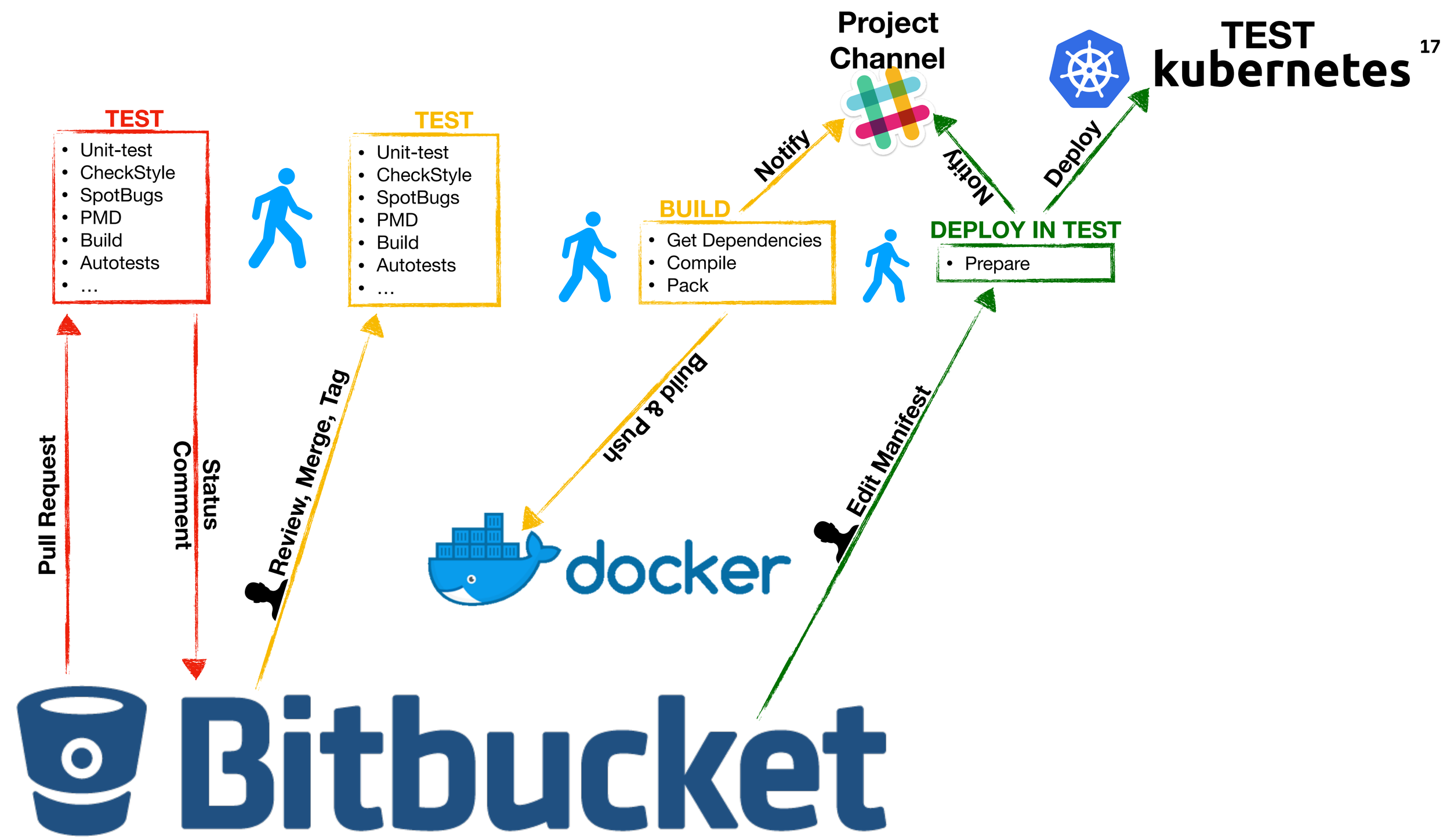
- …

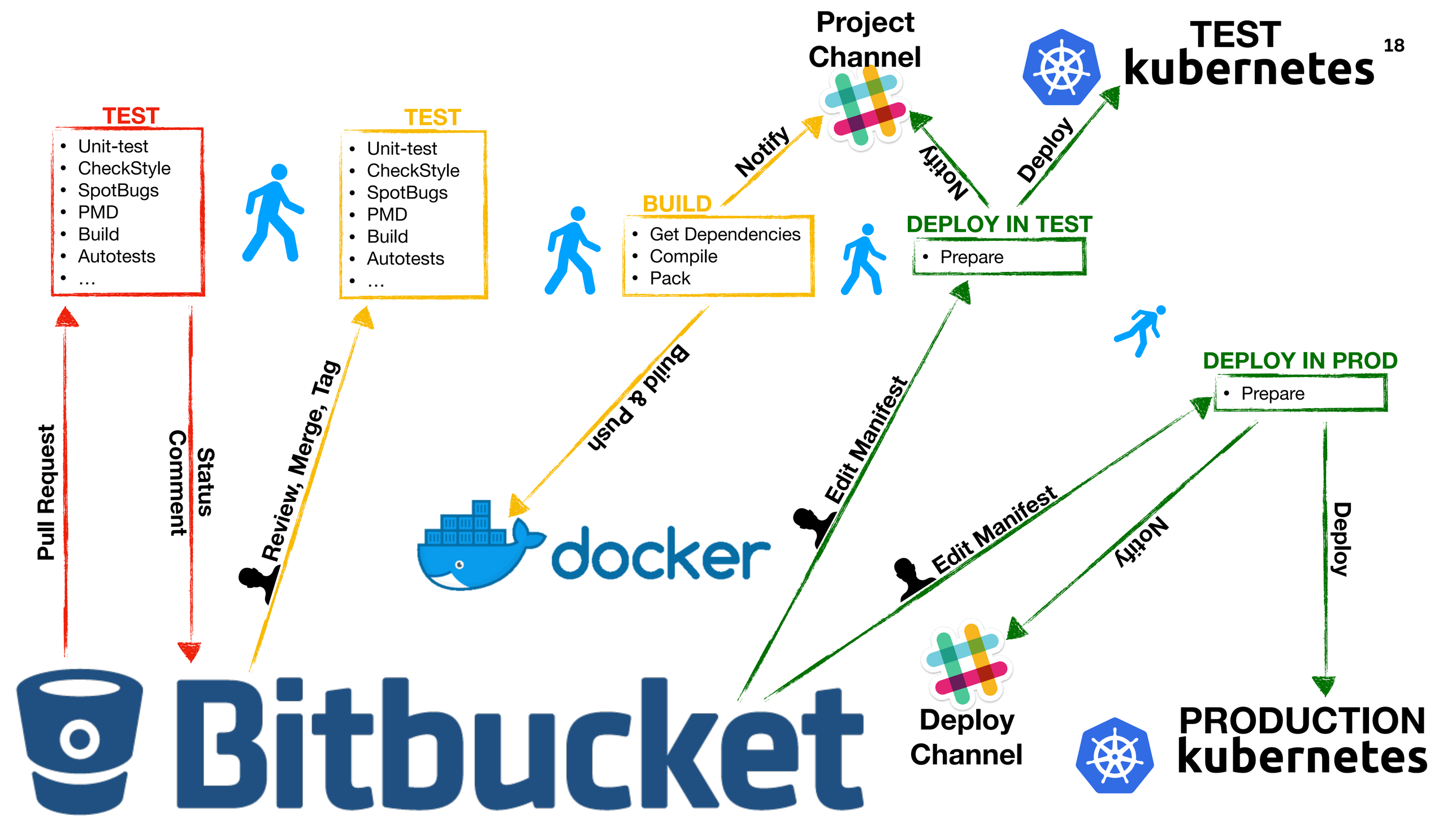
Что делает Concourse CI в Devino Telecom?

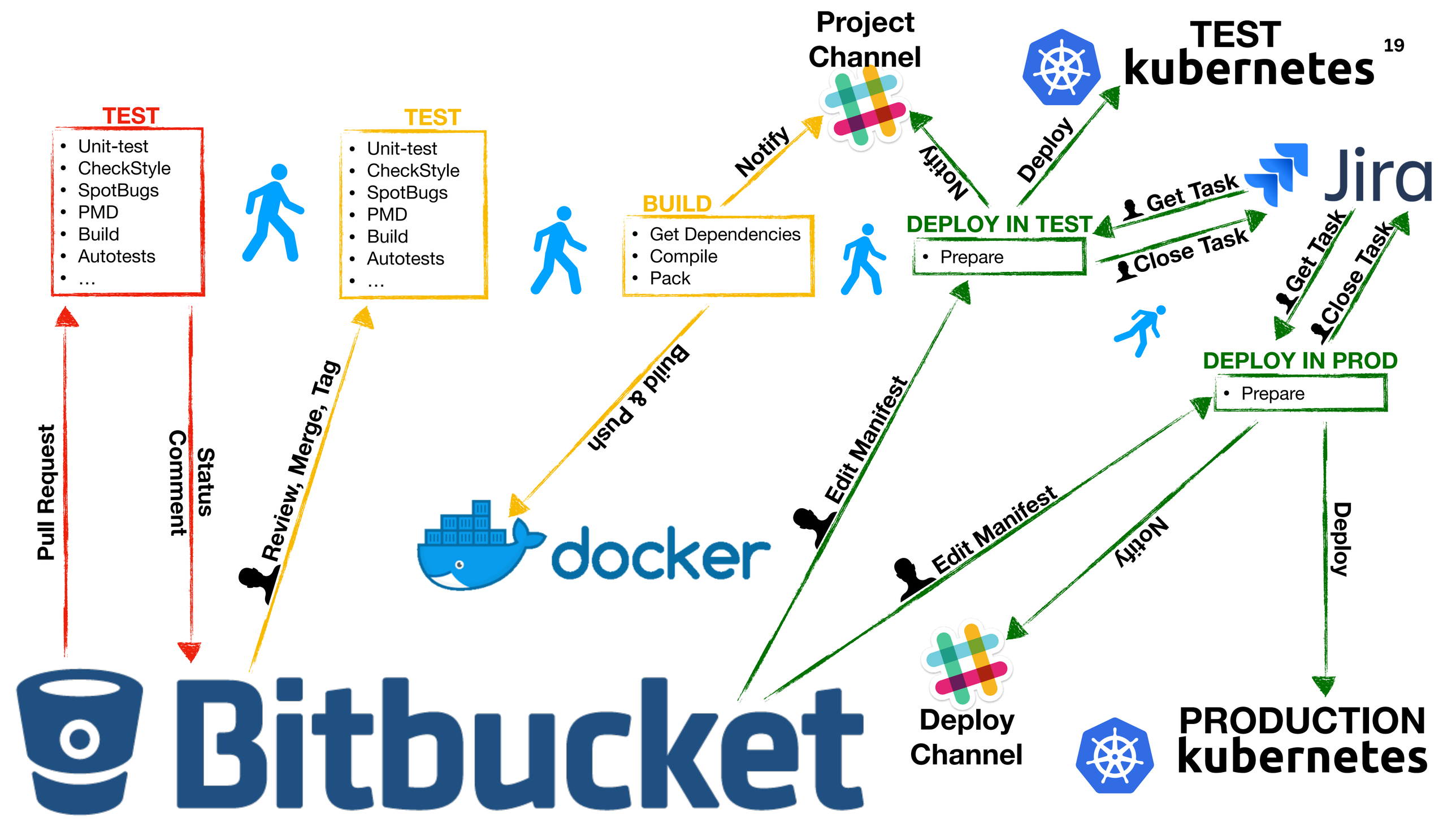

TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
Pull Request
Status
Comment
Bitbucket


TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
Pull Request
Comment
Status
Review, Merge, Tag
Bitbucket


Project Channel
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
BUILD
Get Dependencies
Compile
Pack
Notify
Pull Request
Comment
Status
Review, Merge, Tag
Build & Push
docker
Bitbucket

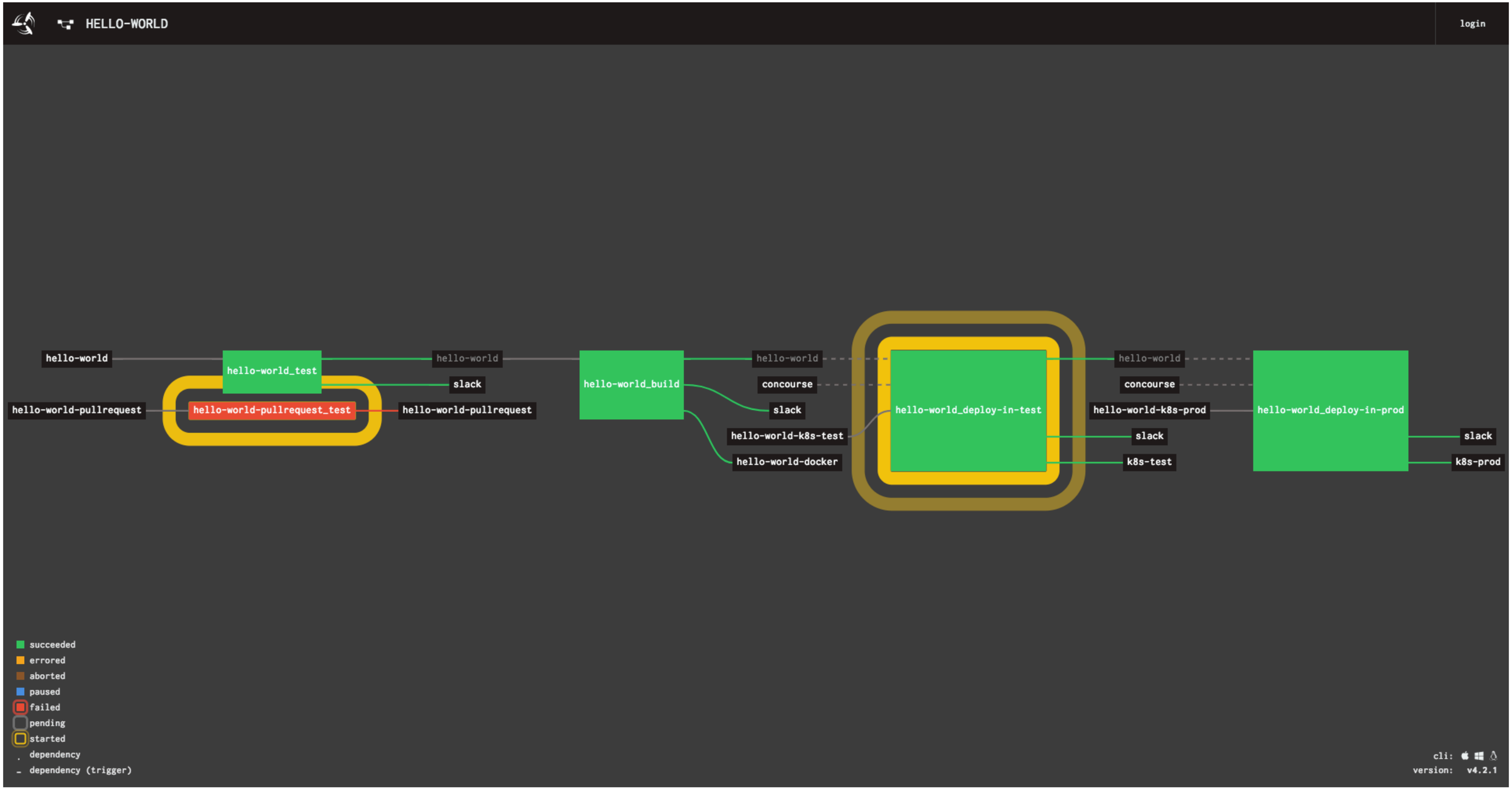

Project Channel
TEST kubernetes
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
BUILD
Get Dependencies
Compile
Pack
DEPLOY IN TEST
Prepare
Notify
Notify
Deploy
Pull Request
Comment
Status
Review, Merge, Tag
Build & Push
Edit Manifest
docker
Bitbucket


Project Channel
TEST kubernetes
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
Notify
Notify
Deploy
BUILD
Get Dependencies
Compile
Pack
DEPLOY IN TEST
Prepare
Pull Request
Comment
Status
Review, Merge, Tag
Build & Push
docker
Edit Manifest
Edit Manifest
DEPLOY IN PROD
Prepare
Notify
Deploy
Bitbucket
Deploy Channel
PRODUCTION kubernetes

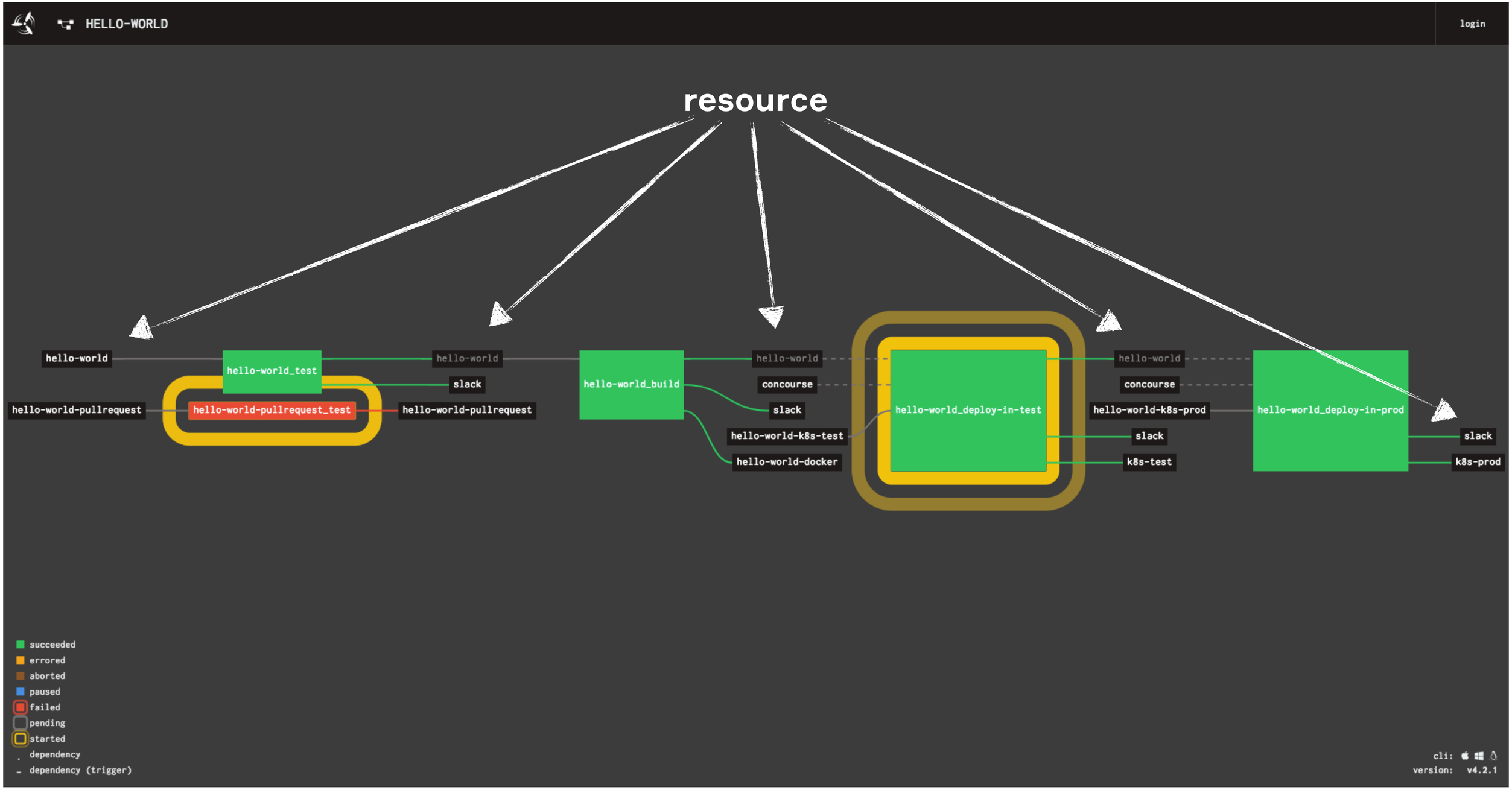

Project Channel
TEST kubernetes
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
TEST
Unit-test
CheckStyle
SpotBugs
PMD
Build
Autotests
…
Notify
BUILD
Get Dependencies
Compile
Pack
Notify
Deploy
DEPLOY IN TEST
Prepare
Get Task
Close Task
Jira
Get Task
Close Task
DEPLOY IN PROD
Prepare
Pull Request
Comment
Status
Review, Merge, Tag
Build & Push
docker
Edit Manifest
Edit Manifest
Notify
Deploy
Deploy Channel
PRODUCTION kubernetes
Bitbucket

Что такое pipeline в Concourse?

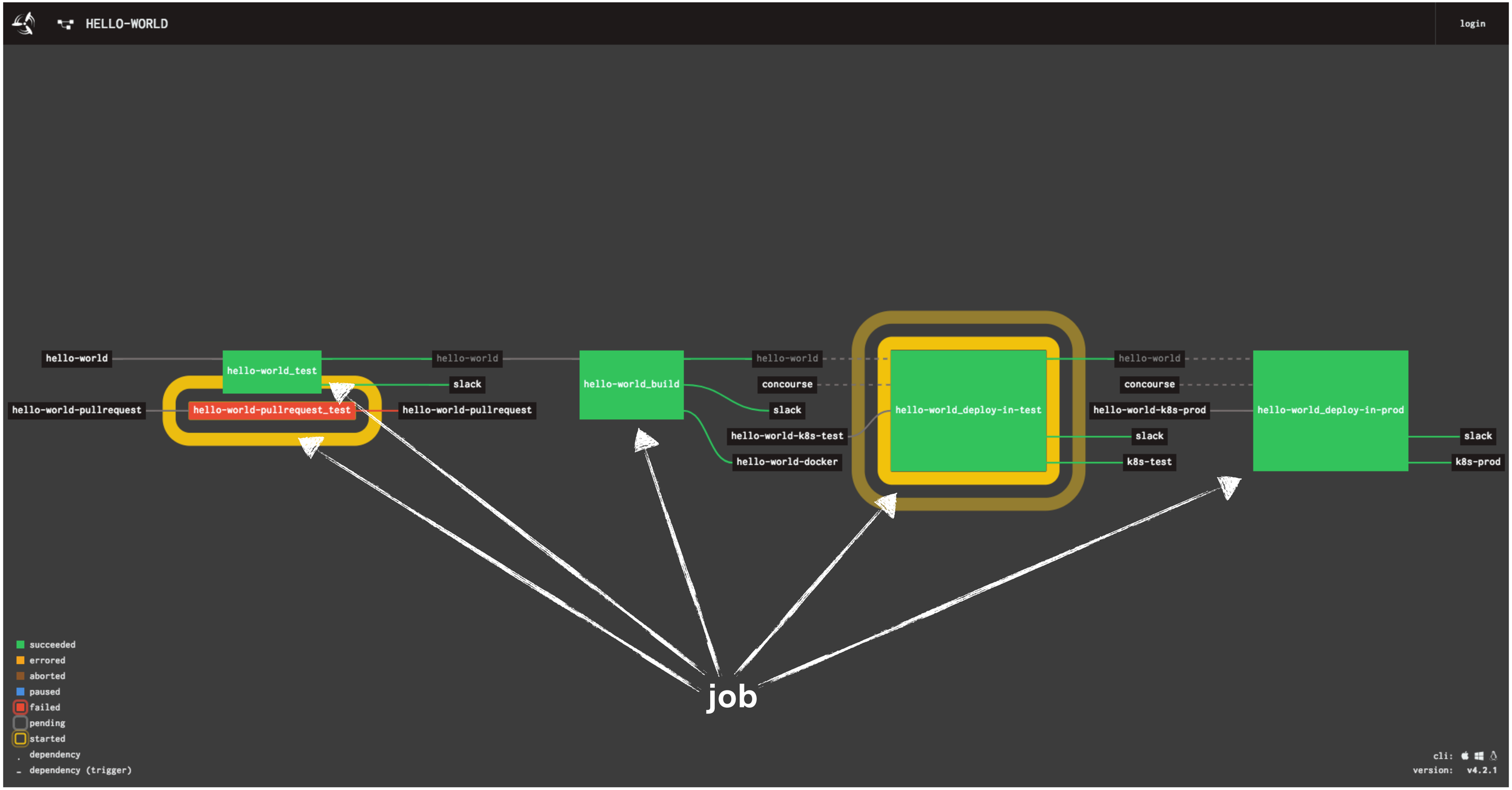
Concourse CI v4.x.x - Pipeline



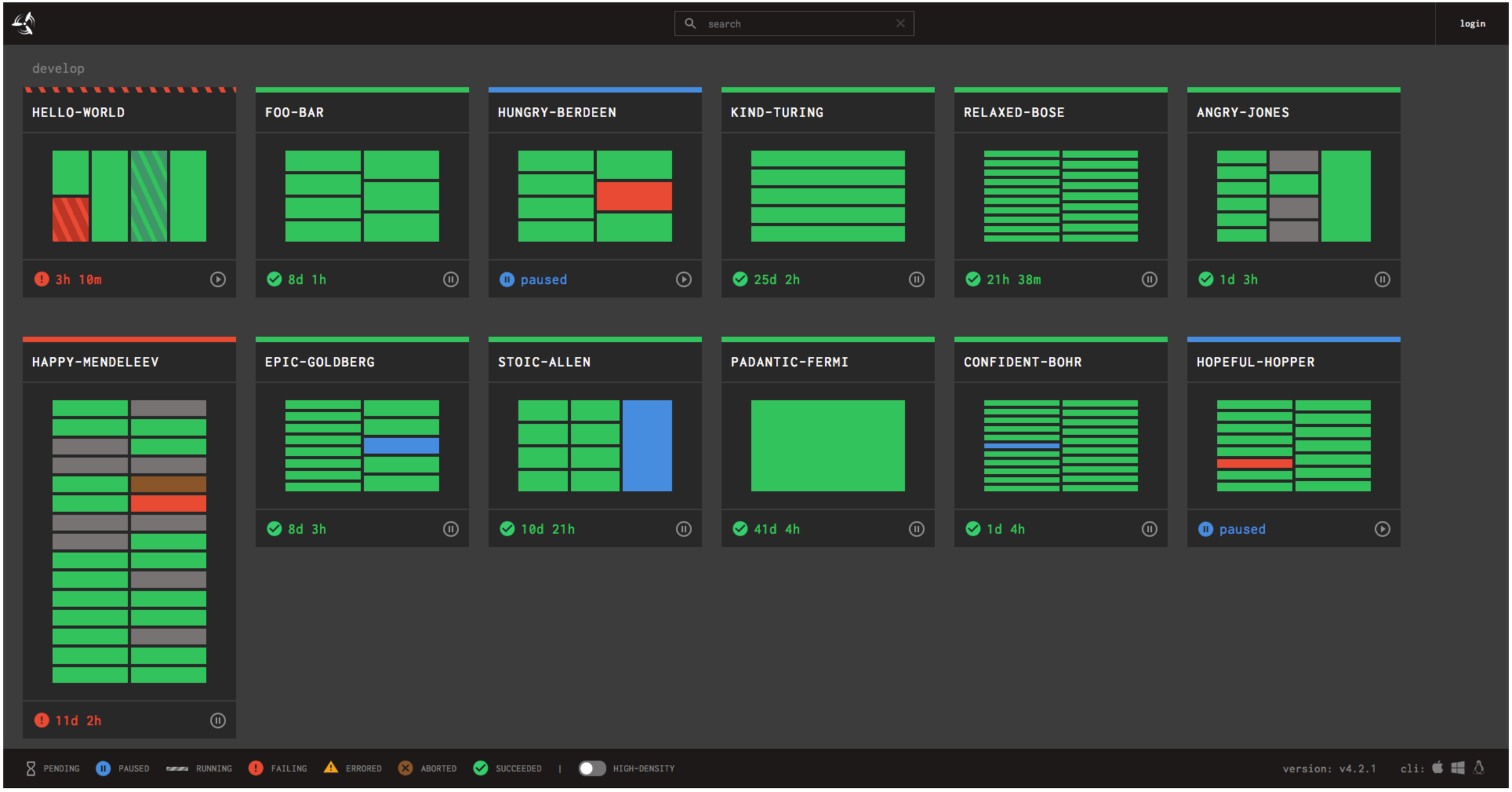
Concourse CI v4.x.x - Pipeline



Concourse CI v4.x.x - Pipeline

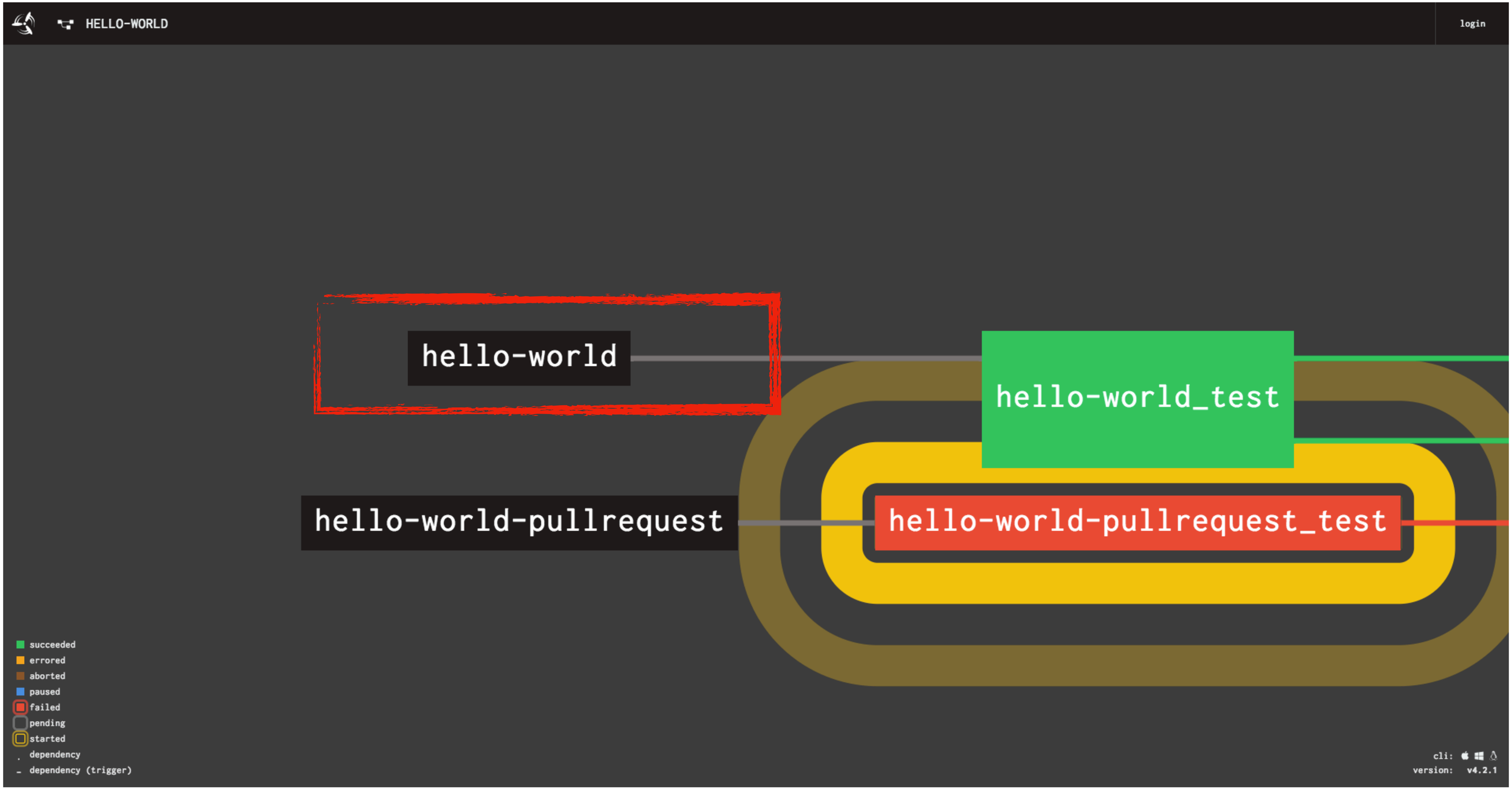


Concourse CI v4.x.x - Main Dashboard



Для чего resource?

Concourse CI v4.x.x - Resource



Concourse CI v4.x.x - Resource


```yaml
resources:
- name: hello-world
  type: git
  source:
    uri: ssh://git@git.example.com/ex/hello-world.git
    private_key: {{bitbucket-private-key}}
    branch: master
    tag_filter: v*
```

Concourse CI v4.x.x - Resource


```yaml
resources:
- name: hello-world
  type: git
  source:
    uri: ssh://git@git.example.com/ex/hello-world.git
    private_key: {{bitbucket-private-key}}
    branch: master
    tag_filter: v*
```

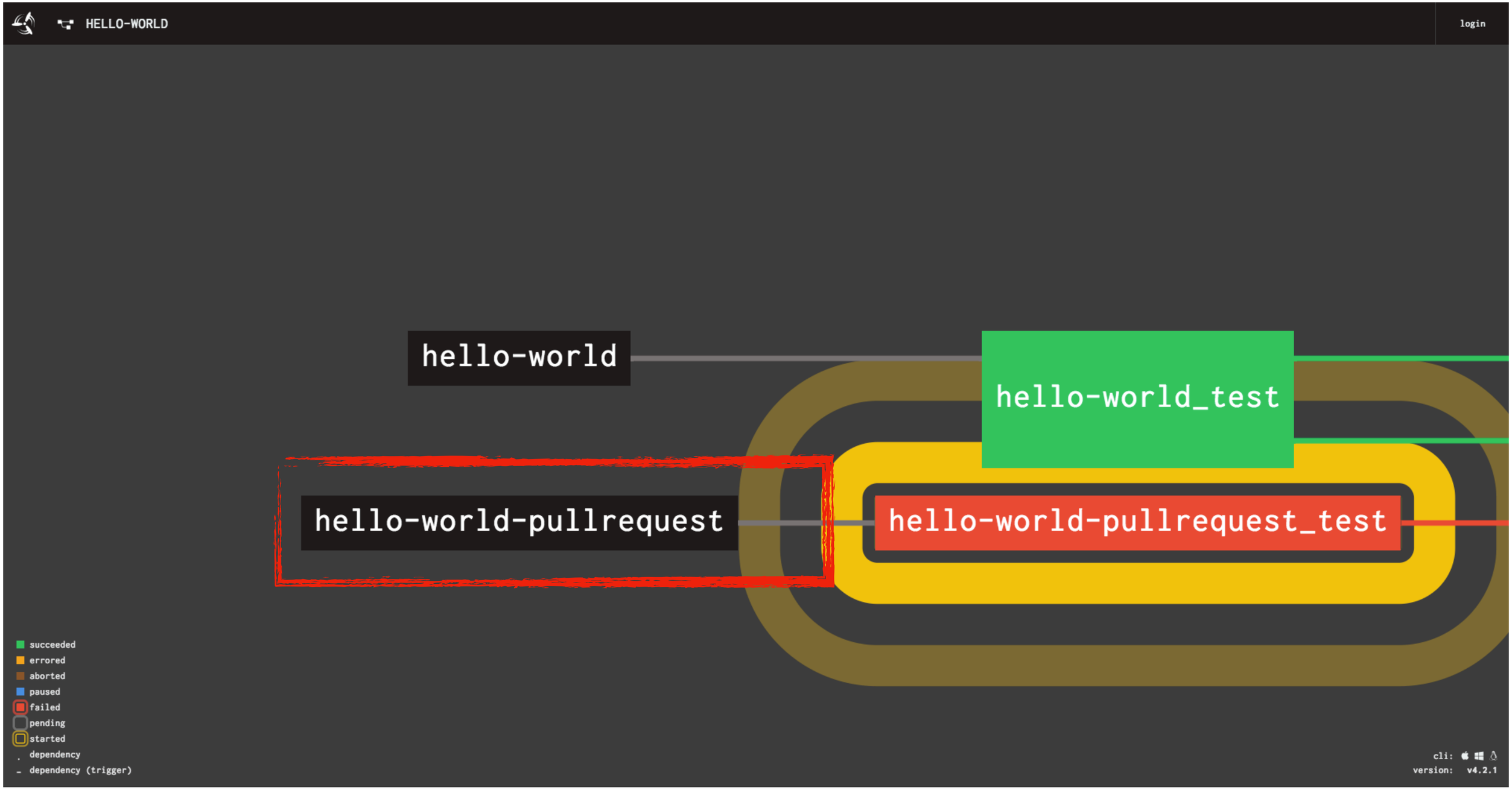
Concourse CI v4.x.x - Resource


```yaml
resources:
- name: hello-world
  type: git
  source:
    uri: ssh://git@git.example.com/ex/hello-world.git
    private_key: {{bitbucket-private-key}}
    branch: master
    tag_filter: v*
```

Concourse CI v4.x.x - Resource


```yaml
resources:
- name: hello-world
  type: git
  source:
    uri: ssh://git@git.example.com/ex/hello-world.git
    private_key: |
      -----BEGIN EC PRIVATE KEY-----
      ...
      -----END EC PRIVATE KEY-----
    branch: master
    tag_filter: v*
```

Concourse CI v4.x.x - Resource (community)



Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: {{bitbucket-username}}
    password: {{bitbucket-password}}
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: {{bitbucket-username}}
    password: {{bitbucket-password}}
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: {{bitbucket-username}}
    password: {{bitbucket-password}}
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: {{bitbucket-username}}
    password: {{bitbucket-password}}
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: {{bitbucket-username}}
    password: {{bitbucket-password}}
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource (community)


```yaml
resource_types:
- name: bitbucket-pullrequest
  type: docker-image
  source: {repository: laurentverbruggen/concourse-bitbucket-pullrequest-resource}

resources:
- name: hello-world-pullrequest
  type: bitbucket-pullrequest
  source:
    uri: https://git.example.com/scm/ex/hello-world.git
    username: concourse
    password: #sEcrEt_pAss-&1
    only_for_branch: master
    only_without_conflicts: true
    rebuild_when_target_changed: true
```

Concourse CI v4.x.x - Resource Types

Provided With Concourse

- The `git` resource can pull and push to git repositories.
- The `hg` resource can pull and push to Mercurial repositories.
- The `time` resource can start jobs on a schedule or timestamp outputs.
- The `s3` resource can fetch from and upload to S3 buckets.
- The `archive` resource can fetch and extract .tar.gz archives.
- The `semver` resource can set or bump version numbers.
- The `github-release` resource can fetch and publish versioned GitHub resources.
- The `docker-image` resource can fetch, build, and push Docker images
- The `tracker` resource can deliver stories and bugs on Pivotal Tracker
- The `pool` resource allows you to configure how to serialize use of an external system. This lets you prevent test interference or overwork on shared systems.
- The `cf` resource can deploy an application to Cloud Foundry.
- The `bosh-io-release` resource can track and fetch new BOSH releases from bosh.io.
- The `bosh-io-stemcell` resource can track and fetch new BOSH stemcells from bosh.io.

Concourse CI v4.x.x - Resource Types

Provided With Concourse

- The `git` resource can pull and push to git repositories.
- The `hg` resource can pull and push to Mercurial repositories.
- The `time` resource can start jobs on a schedule or timestamp outputs.
- The `s3` resource can fetch from and upload to S3 buckets.
- The `archive` resource can fetch and extract .tar.gz archives.
- The `semver` resource can set or bump version numbers.
- The `github-release` resource can fetch and publish versioned GitHub resources.
- The `docker-image` resource can fetch, build, and push Docker images
- The `tracker` resource can deliver stories and bugs on Pivotal Tracker
- The `pool` resource allows you to configure how to serialize use of an external system. This lets you prevent test interference or overwork on shared systems.
- The `cf` resource can deploy an application to Cloud Foundry.
- The `bosh-io-release` resource can track and fetch new BOSH releases from bosh.io.
- The `bosh-io-stemcell` resource can track and fetch new BOSH stemcells from bosh.io.

Community

Resource Name	Maintained By
Slack Reading and Posting	by @jleben
Slack notifications	by @cloudfoundry-community
Github Pull Requests	by @telia-oss
GitLab Merge Requests	by @swisscom
OpenStack Swift	by @sapcc
Key Value resource	by @swce
Flowdock notifications	by @starkandwayne
Email	by @pivotal-cf
Formatted Email	by @santoshjpawar
Email with integrated MTA	by @mdomke
Bintray	by @jamiemonserrate
Perforce	by @olhtbr
BOSH Errands	by @starkandwayne
BOSH Config: cloud-config and runtime-config	by @EMC-Dojo
BOSH Release	by @dpb587
Pool Trigger	by @sfmobile
Pivotal Network	by @pivotal-cf-experimental
FTP	by @aequitas
lftp, access resources via ftp, http, sftp and fish	by @openSUSE
Cloudformation	by @pivotal-cf-experimental
Generic HTTP API	by @aequitas
Hockey App	by @seadowg
Concourse Pipelines	by @robdimsdale

Concourse CI v4.x.x - Resource Types

Provided With Concourse

- The `git` resource can pull and push to git repositories.
- The `hg` resource can pull and push to Mercurial repositories.
- The `time` resource can start jobs on a schedule or timestamp outputs.
- The `s3` resource can fetch from and upload to S3 buckets.
- The `archive` resource can fetch and extract .tar.gz archives.
- The `semver` resource can set or bump version numbers.
- The `github-release` resource can fetch and publish versioned GitHub resources.
- The `docker-image` resource can fetch, build, and push Docker images
- The `tracker` resource can deliver stories and bugs on Pivotal Tracker
- The `pool` resource allows you to configure how to serialize use of an external system. This lets you prevent test interference or overwork on shared systems.
- The `cf` resource can deploy an application to Cloud Foundry.
- The `bosh-io-release` resource can track and fetch new BOSH releases from bosh.io.
- The `bosh-io-stemcell` resource can track and fetch new BOSH stemcells from bosh.io.

Community

Resource Name	Maintained By
Slack Reading and Posting	by @jleben
Slack notifications	by @cloudfoundry-community
Github Pull Requests	by @telia-oss
GitLab Merge Requests	by @swisscom
OpenStack Swift	by @sapcc
Key Value resource	by @swce
Flowdock notifications	by @starkandwayne
Email	by @pivotal-cf
Formatted Email	by @santoshjpawar
Perforce	by @olhtbr
BOSH Errands	by @starkandwayne
BOSH Config: cloud-config and runtime-config	by @EMC-Dojo
BOSH Release	by @dpb587
Pool Trigger	by @sfmobile
Pivotal Network	by @pivotal-cf-experimental
FTP	by @aequitas
lftp, access resources via ftp, http, sftp and fish	by @openSUSE
Cloudformation	by @pivotal-cf-experimental
Generic HTTP API	by @aequitas
Hockey App	by @seadowg
Concourse Pipelines	by @robdimsdale

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

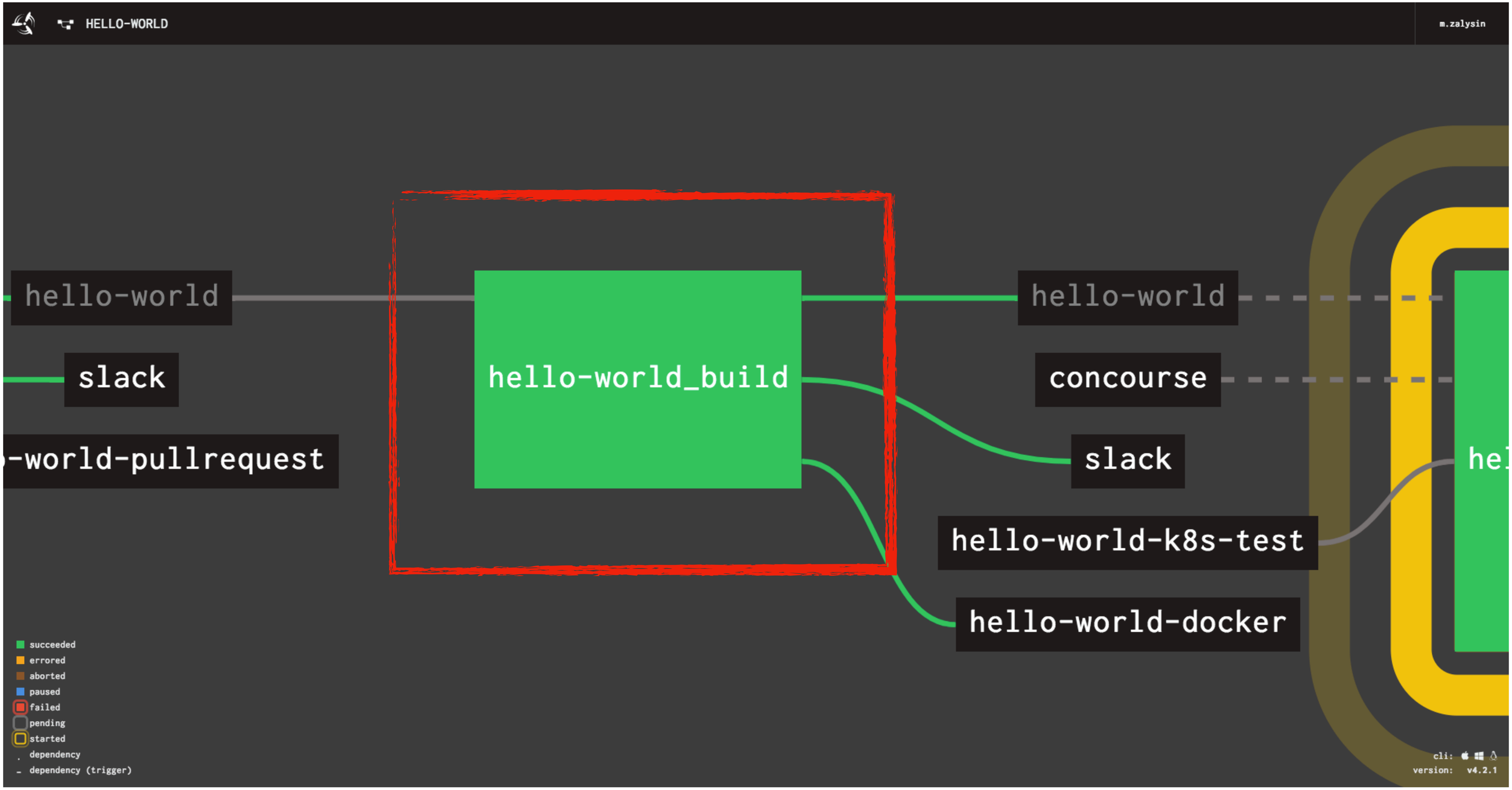
Concourse CI v4.x.x - Resource (community)


```
$ docker run --rm --entrypoint ls devinotelecom/concourse-slack-alert-resource -l /opt/resource
total 19340
-rwxr-xr-x    1 root     root       6451874 Nov  1 09:57 check
-rwxr-xr-x    1 root     root       6452080 Nov  1 09:57 in
-rwxr-xr-x    1 root     root       6879238 Nov  1 09:57 out
```

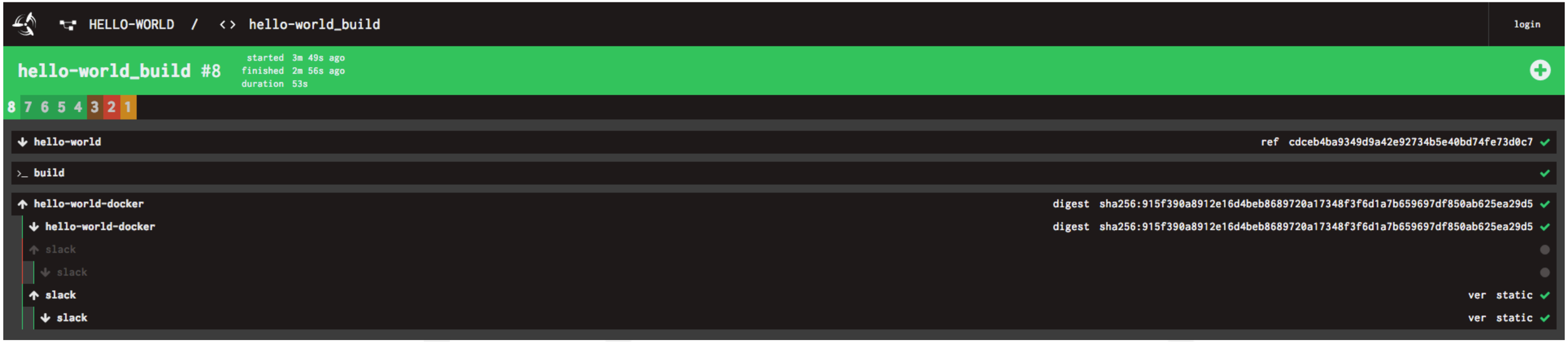
https://concourse-ci.org/implementing-resources.html

Что происходит в job`e?

Concourse CI v4.x.x - Job "*_build"

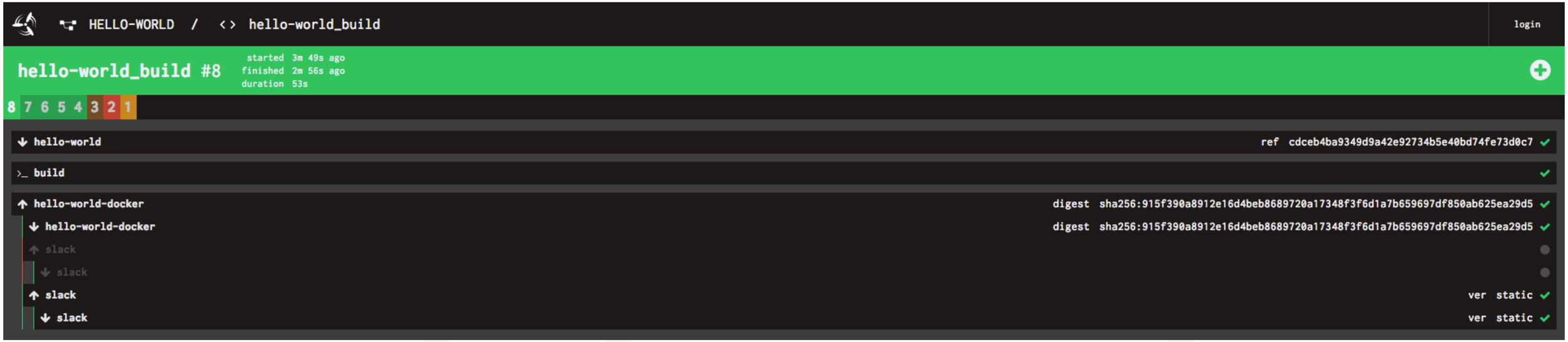


Concourse CI v4.x.x - Job "*_build"



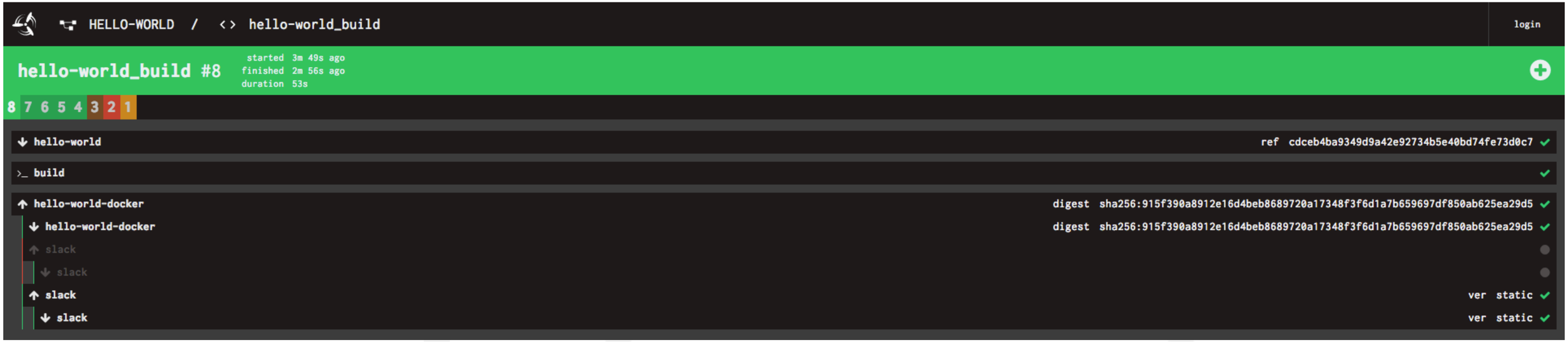

```yaml
jobs:
- name: hello-world_build
  plan:
  - get: hello-world
    passed: ["hello-world_test"]
    trigger: true
  - task: build
    file: hello-world/ci/build.yml
    input_mapping: {source: hello-world}
  - put: hello-world-docker
    params: {build: docker, tag: docker/version, tag_as_latest: true}
  on_failure:
    put: slack
    params: {alert_type: failed, message: "Build FAILED"}
  on_success:
    put: slack
    params: {alert_type: success, message: "Build SUCCESS"}
```

Concourse CI v4.x.x - Job "*_build"



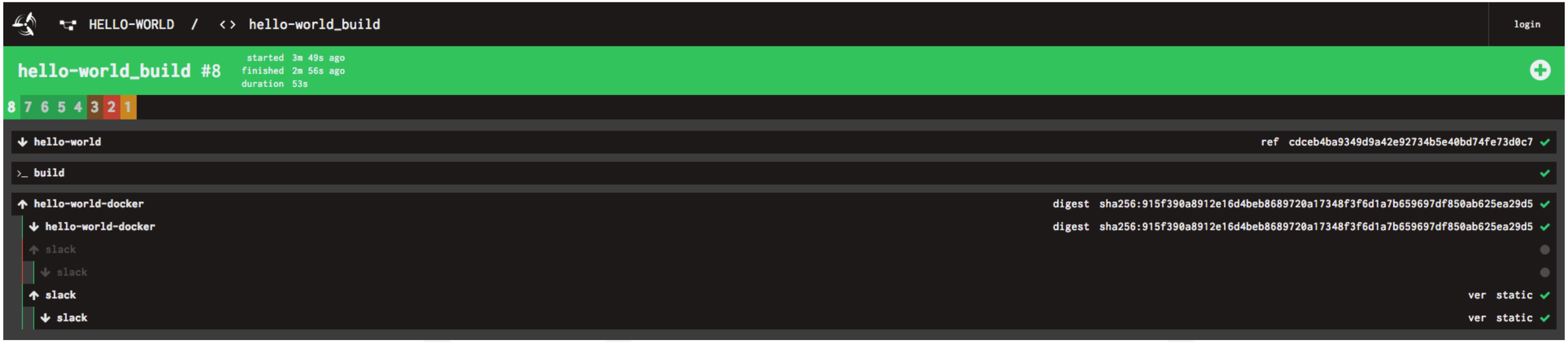

```yaml
jobs:
- name: hello-world_build
  plan:
  - get: hello-world
    passed: ["hello-world_test"]
    trigger: true
  - task: build
    file: hello-world/ci/build.yml
    input_mapping: {source: hello-world}
  - put: hello-world-docker
    params: {build: docker, tag: docker/version, tag_as_latest: true}
  on_failure:
    put: slack
    params: {alert_type: failed, message: "Build FAILED"}
  on_success:
    put: slack
    params: {alert_type: success, message: "Build SUCCESS"}
```

Concourse CI v4.x.x - Job "*_build"




```yaml
jobs:
- name: hello-world_build
  plan:
  - get: hello-world
    passed: ["hello-world_test"]
    trigger: true
  - task: build
    file: hello-world/ci/build.yml
    input_mapping: {source: hello-world}
  - put: hello-world-docker
    params: {build: docker, tag: docker/version, tag_as_latest: true}
  on_failure:
    put: slack
    params: {alert_type: failed, message: "Build FAILED"}
  on_success:
    put: slack
    params: {alert_type: success, message: "Build SUCCESS"}
```

Concourse CI v4.x.x - Job "*_build"

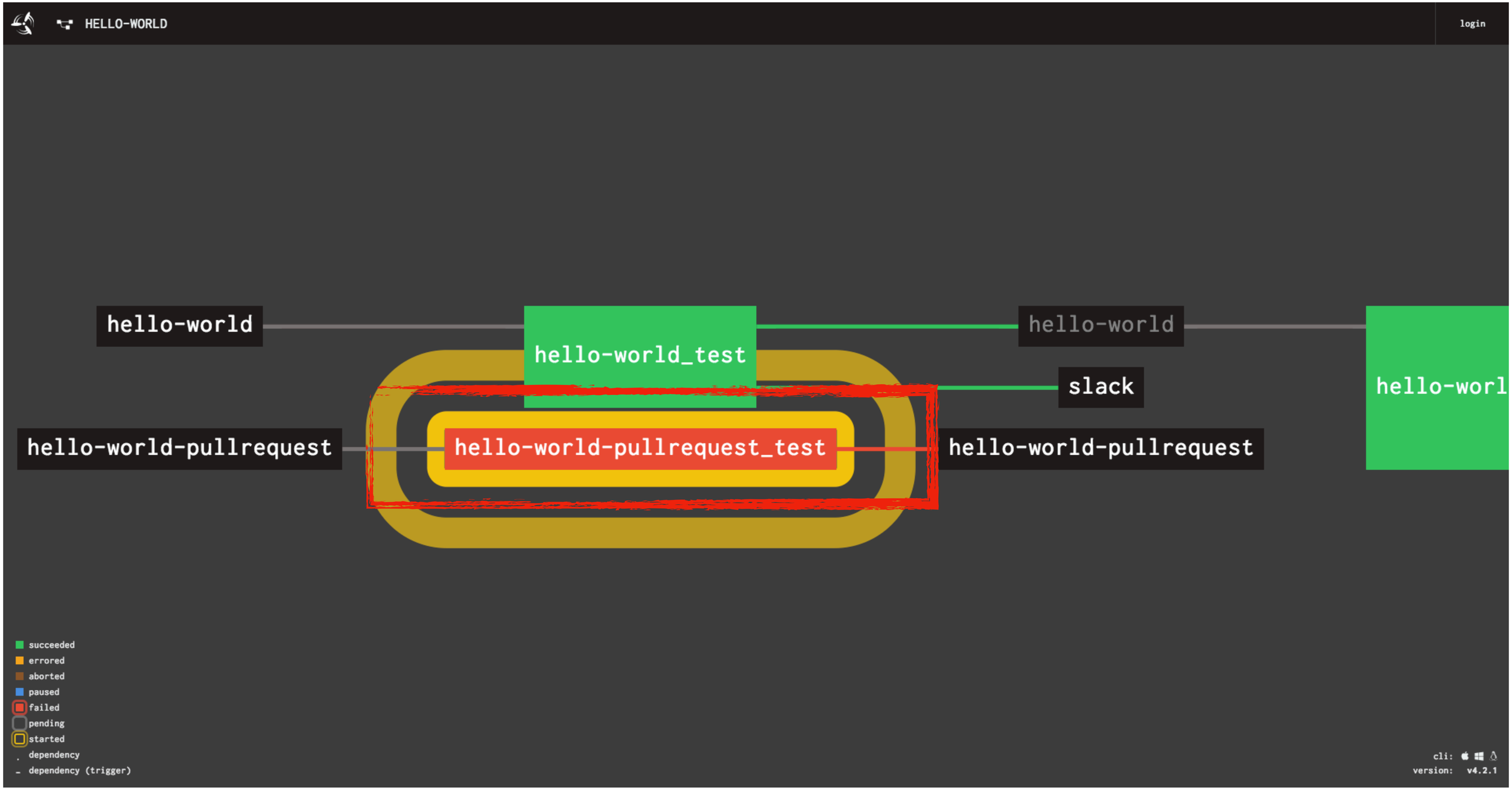



```yaml
jobs:
- name: hello-world_build
  plan:
  - get: hello-world
    passed: ["hello-world_test"]
    trigger: true
  - task: build
    file: hello-world/ci/build.yml
    input_mapping: {source: hello-world}
  - put: hello-world-docker
    params: {build: docker, tag: docker/version, tag_as_latest: true}
  on_failure:
    put: slack
    params: {alert_type: failed, message: "Build FAILED"}
  on_success:
    put: slack
    params: {alert_type: success, message: "Build SUCCESS"}
```

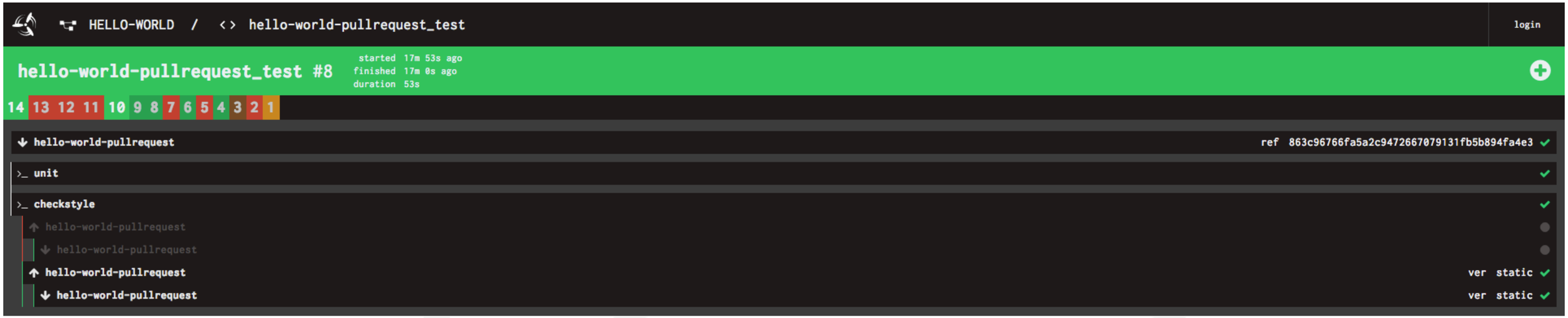
Concourse CI v4.x.x - Task ci/build.yml


```yaml
platform: linux
image_resource:
  type: docker-image
  source: {repository: gradle, tag: 4.10.2-jdk8-alpine}
inputs:
- name: source
outputs:
- name: docker
caches:
- path: source/.gradle/caches
run:
  dir: source
  user: root
  path: sh
  args:
  - -exc
  - |
    gradle --gradle-user-home=.gradle build

    cp build/libs/source.jar ../docker/application.jar
    cp ci/Dockerfile > ../docker/Dockerfile

    apk add --no-cache --no-progress git
    git describe --tags | sed 's/^v//' > ../docker/version
```

Concourse CI v4.x.x - Job "*-pullrequest_test"

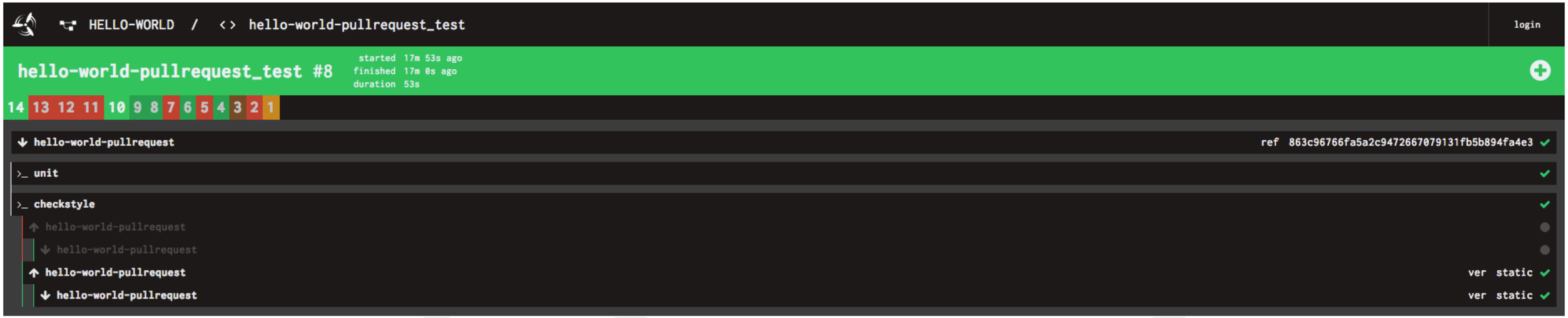


Concourse CI v4.x.x - Job "*-pullrequest_test"



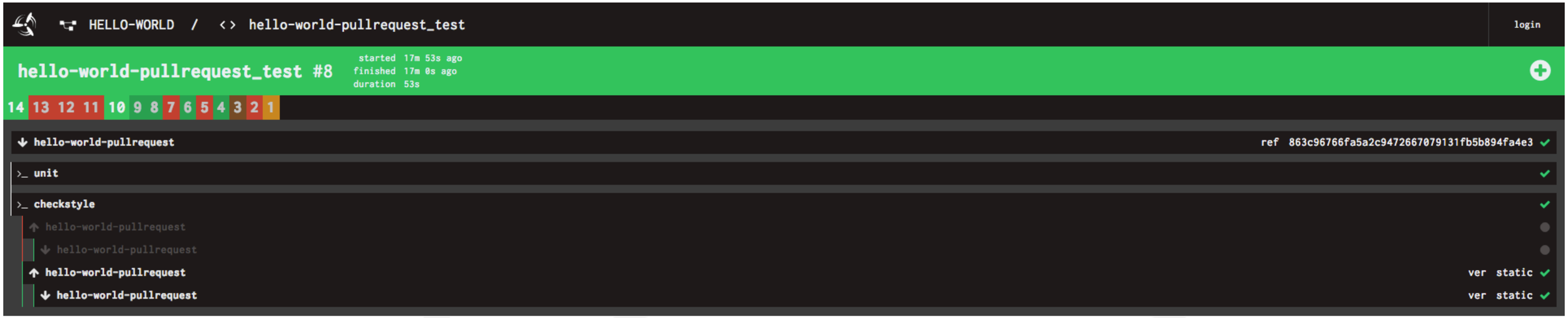

```yaml
jobs:
- name: hello-world-pullrequest_test
  plan:
  - get: hello-world-pullrequest
    trigger: true
  - aggregate:
    - task: unit
      file: hello-world-pullrequest/ci/unit.yml
      input_mapping: {source: hello-world-pullrequest}
    - task: checkstyle
      file: hello-world-pullrequest/ci/checkstyle.yml
      input_mapping: {source: hello-world-pullrequest}
  on_failure:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequestt, status: failure}
  on_success:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequest, status: success}
```

Concourse CI v4.x.x - Job "*-pullrequest_test"



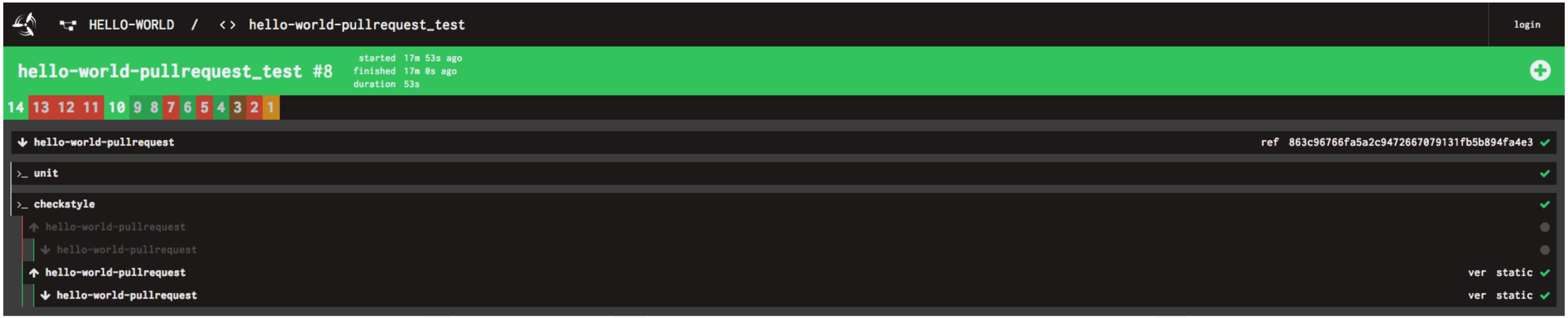

```yaml
jobs:
- name: hello-world-pullrequest_test
  plan:
  - get: hello-world-pullrequest
    trigger: true
  - aggregate:
    - task: unit
      file: hello-world-pullrequest/ci/unit.yml
      input_mapping: {source: hello-world-pullrequest}
    - task: checkstyle
      file: hello-world-pullrequest/ci/checkstyle.yml
      input_mapping: {source: hello-world-pullrequest}
  on_failure:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequestt, status: failure}
  on_success:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequest, status: success}
```

Concourse CI v4.x.x - Job "*-pullrequest_test"




```yaml
jobs:
- name: hello-world-pullrequest_test
  plan:
  - get: hello-world-pullrequest
    trigger: true
  - aggregate:
    - task: unit
      file: hello-world-pullrequest/ci/unit.yml
      input_mapping: {source: hello-world-pullrequest}
    - task: checkstyle
      file: hello-world-pullrequest/ci/checkstyle.yml
      input_mapping: {source: hello-world-pullrequest}
  on_failure:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequestt, status: failure}
  on_success:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequest, status: success}
```

Concourse CI v4.x.x - Job "*-pullrequest_test"




```yaml
jobs:
- name: hello-world-pullrequest_test
  plan:
  - get: hello-world-pullrequest
    trigger: true
  - aggregate:
    - task: unit
      file: hello-world-pullrequest/ci/unit.yml
      input_mapping: {source: hello-world-pullrequest}
    - task: checkstyle
      file: hello-world-pullrequest/ci/checkstyle.yml
      input_mapping: {source: hello-world-pullrequest}
  on_failure:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequestt, status: failure}
  on_success:
    put: hello-world-pullrequest
    params: {path: hello-world-pullrequest, status: success}
```

Преимущества!

- YAML - удобно, структурировано
- CLI для управления CI`кой
- Интеграция с чем угодно через resource
- Просто написать свой resource
- Job одновременно выполняет множество task`ов в связке с разными resource`ами
- Переиспользуемые файлы task`ов

Недостатки?

- Требовательность к ресурсам
- Бывает, worker залипает
- Сложно повторно запустить job на устаревшем ресурсе
- Нет разделения доступа по ролям в одной team`e

Вопросы?

Ссылки:

- https://concourse-ci.org
- https://github.com/devinotelecom
- https://hub.docker.com/u/devinotelecom
- https://t.me/magnaz