Deployment without Angst

MOTEL

Hi!

WHO YOU'RE DEALING WITH

Lead Engineer at Motel
@staylorwr

MOTEL

The Team

WHO YOU'RE DEALING WITH

Holy moly look at all of these great people

MOTEL

Motel ❤️ Elixir

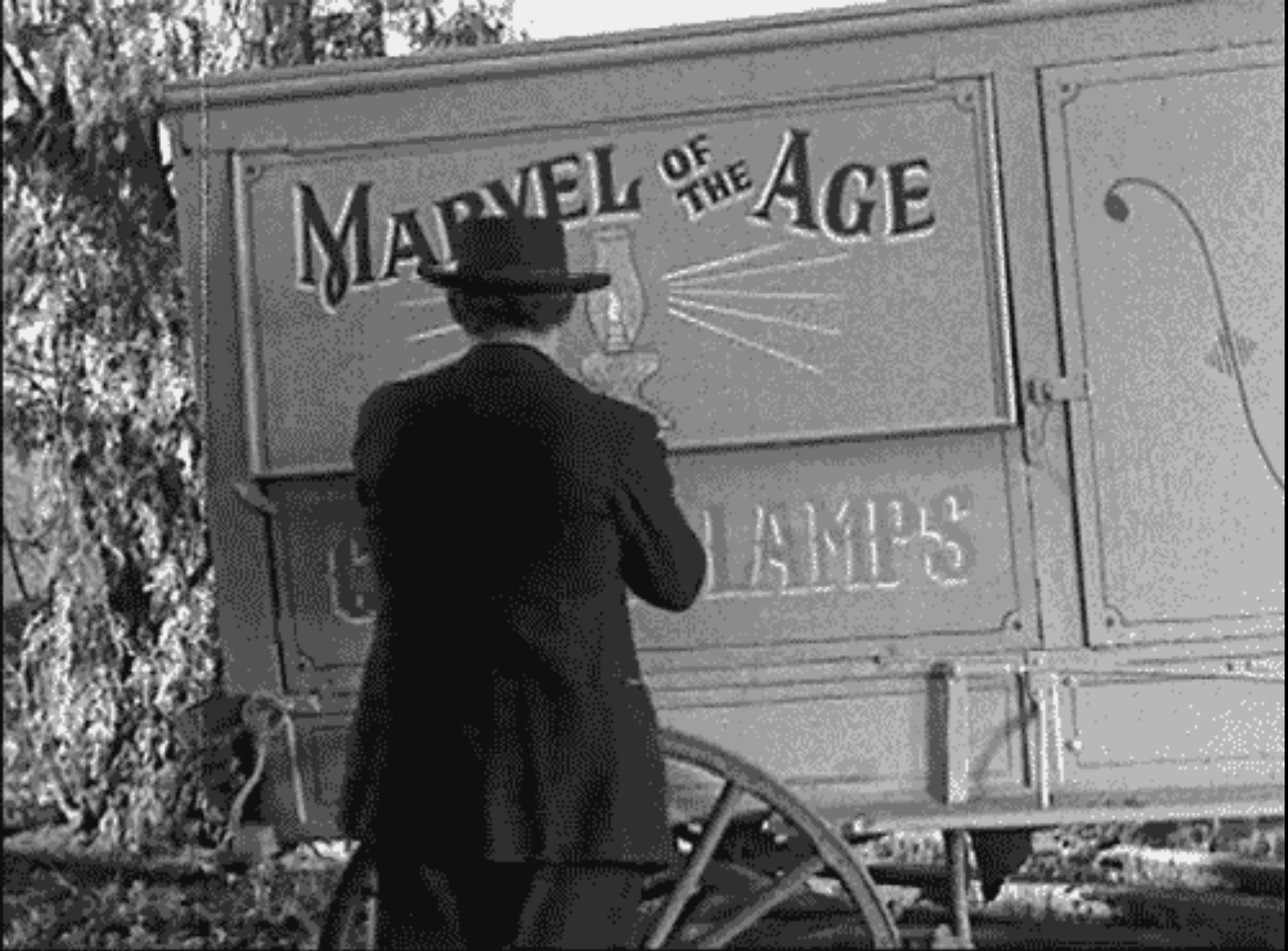
FOR OVER A YEAR

- Internal Investigation
- Internal Friday Project
- 80% of new dev projects now Elixir

So how do we…deploy it?

GOOD QUESTION.

I’ve been working on a solution that I think makes sense, but I’d love your feedback in making it better.


MARVEL OF THE AGE
LAMPS

So how do we deploy it?

GOOD QUESTION

- Plan
- Create
- Verify
- Package
- Release
- Configure
- Monitor

So how do we deploy it?

GOOD QUESTION

Package

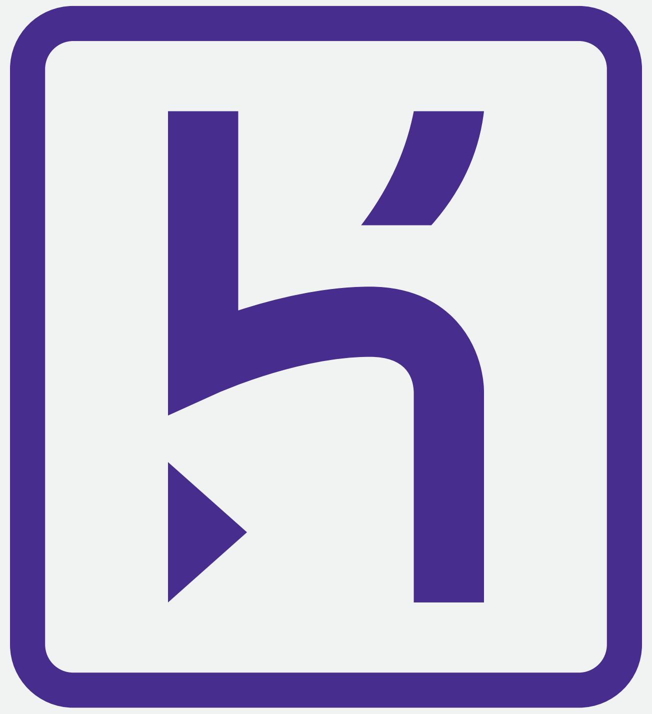
Release

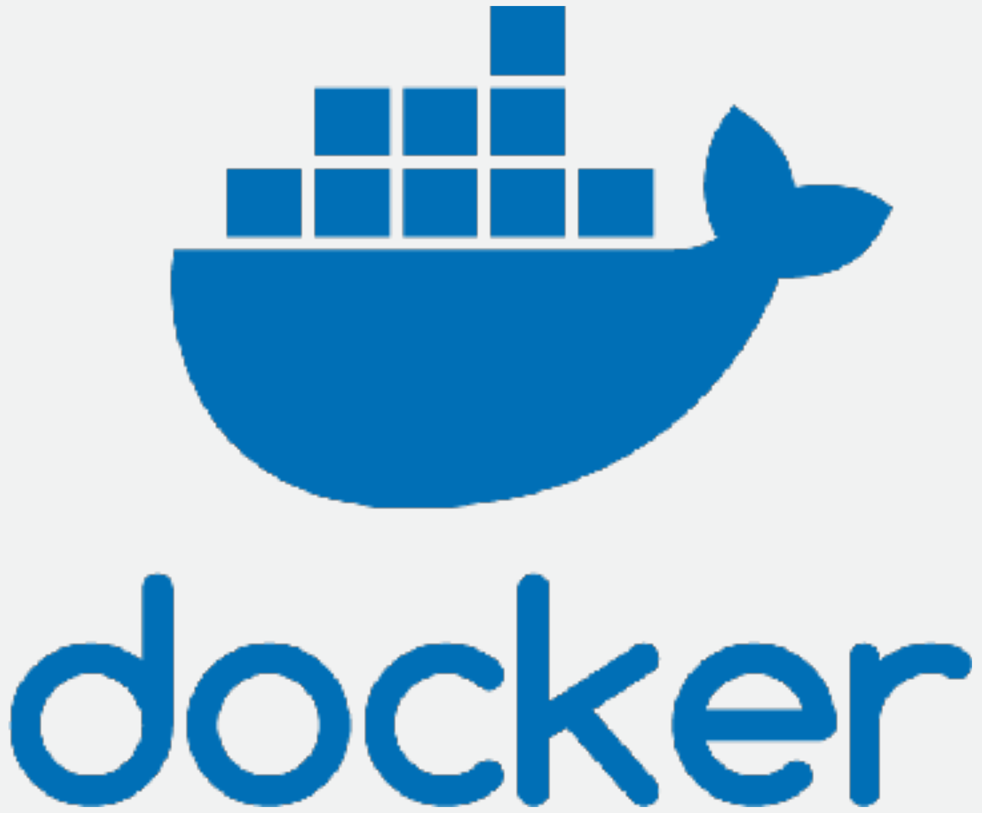
Configure

Monitor

So how do we deploy it?

GOOD QUESTION





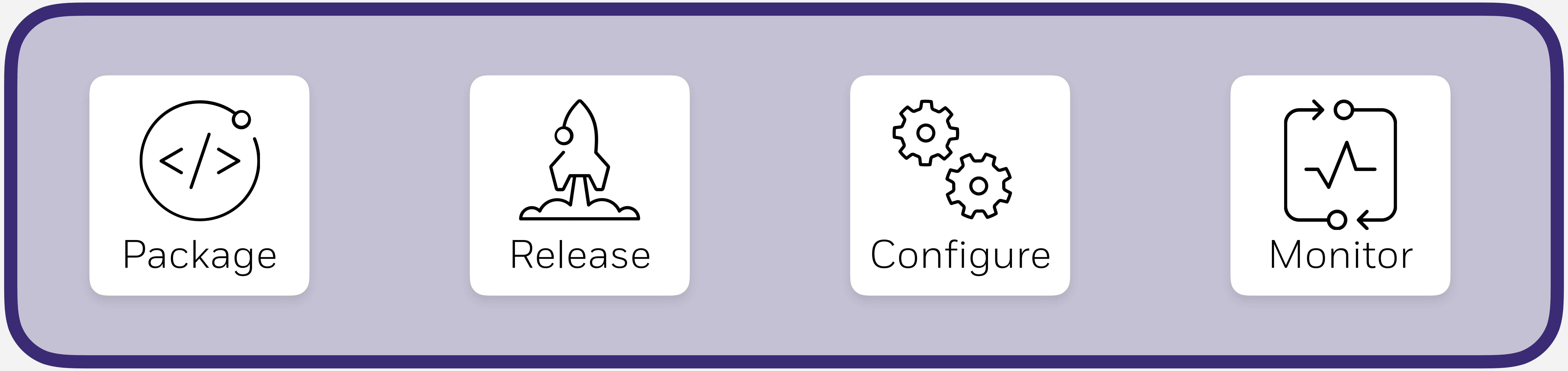
What about Heroku?

USE THE PLATFORM

- Package
- Release
- Configure
- Monitor

What about Heroku?

USE THE PLATFORM



Package

Release

Configure

Monitor

303

What about Heroku?

BUT…

- Heroku Dynos Restart: No Process Based Persistence
- Heroku Dynos don't connect: No Shared Channel State
- `mix phx.server` isn't performant

What about OTP?

MAYBE… BUT BLEH

Package	Release	Configure	Monitor
Distillery	edeliver	danger	BYO

DISTILLERY

- Add **distillery** as a dependency

```elixir
# Specifies your project dependencies.
defp deps do
  [
    {:phoenix, "~> 1.3.0"},
    {:phoenix_pubsub, "~> 1.0"},
    {:phoenix_ecto, "~> 3.2"},
    {:postgrex, ">= 0.0.0"},
    {:phoenix_html, "~> 2.10"},
    {:gettext, "~> 0.11"},
    {:cowboy, "~> 1.0"},
    {:credo, "~> 0.8"},
    {:excoveralls, "~> 0.7", only: :test},
    {:distillery, "~> 1.5"}
  ]
end
```

DISTILLERY

- Add `distillery` as a dependency
- Run `mix release.init`

```
use Mix.Releases.Config,
  default_release: :foo,
  default_environment: Mix.env

environment :dev do
  set dev_mode: true
  set include_erts: false
  set include_system_libs: false
  set cookie: :dev
end

environment :prod do
  set include_erts: true
  set include_system_libs: true
  set cookie: :prod
end

release :foo do
  set version: current_version(:foo)
end
```

DISTILLERY

- Add `distillery` as a dependency
- Run `mix release.init`
- Compile your assets

```
# Brunch build production
./node_modules/brunch/bin/brunch b -p

# Phoenix Digest
MIX_ENV=prod mix phoenix.digest
```

DISTILLERY

- Add `distillery` as a dependency
- Run `mix release.init`
- Compile your assets
- Build a release

```
# Brunch build production
./node_modules/brunch/bin/brunch b -p

# Phoenix Digest
MIX_ENV=prod mix phoenix.digest

# Build a release
MIX_ENV=prod mix release
```

DISTILLERY

- Add `distillery` as a dependency
- Run `mix release.init`
- Compile your assets
- Build a release
- Ship it to your server(s)

```
# EDeliver
mix edeliver build release
mix edeliver release to production
```

DISTILLERY

- Add `distillery` as a dependency
- Run `mix release.init`
- Compile your assets
- Build a release
- Ship it to your server(s)
- ....
- Profit?

DISTILLERY

- Lots of configuration, variables defined on compile time
- Snowflake servers
- Infrastructure is your responsibility
- No clear path towards review apps
- BUT
  - Hot reloads
  - Great performance
  - The Elixir Way

I have no idea what I'm doing
gifbin.com

What about Docker?

YEAH… NO

Package

Distillery

Release

Kubernetes

Configure

Helm

Monitor

BYO

DOCKER + DISTILLERY

- Fetch Elixir Dependencies

```
FROM elixir:1.5.2-alpine as asset-builder-mix-getter

ENV HOME=/opt/app

RUN mix do local.hex --force, local.rebar --force

# Cache elixir deps
COPY config/ $HOME/config/
COPY mix.exs mix.lock $HOME/

WORKDIR $HOME/

RUN mix deps.get

########################################################################

FROM node:6 as asset-builder

ENV HOME=/opt/app
WORKDIR $HOME

COPY --from=asset-builder-mix-getter \
  $HOME/deps $HOME/deps

WORKDIR $HOME/
```

DOCKER + DISTILLERY

- Fetch Elixir Dependencies
- Fetch & Build Assets

```
####################################################

FROM node:6 as asset-builder

ENV HOME=/opt/app
WORKDIR $HOME

COPY --from=asset-builder-mix-getter \
  $HOME/deps $HOME/deps

WORKDIR $HOME/assets
COPY assets/ ./
RUN yarn install
RUN ./node_modules/.bin/brunch build --production

####################################################

FROM bitwalker/alpine-elixir:1.5.2 as releaser

ENV HOME=/opt/app

# dependencies for comeonin
RUN apk add --no-cache build-base cmake
```

DOCKER + DISTILLERY

- Fetch Elixir Dependencies
- Fetch & Build Assets
- Compile & Digest

```
FROM bitwalker/alpine-elixir:1.5.2 as releaser

ENV HOME=/opt/app

# dependencies for comeonin
RUN apk add --no-cache build-base cmake

# Install Hex + Rebar
RUN mix do local.hex --force, local.rebar --force

ARG ERLANG_COOKIE
ENV $ERLANG_COOKIE $ERLANG_COOKIE

# Cache elixir deps
COPY config/ $HOME/config/
COPY mix.exs mix.lock $HOME/

ENV MIX_ENV=prod
RUN mix do deps.get --only $MIX_ENV, deps.compile

COPY . $HOME/

# Digest precompiled assets
COPY --from=asset-builder $HOME/priv/static/ $HOME/
priv/static/
```

DOCKER + DISTILLERY

- Fetch Elixir Dependencies
- Fetch & Build Assets
- Compile & Digest

```
####################################################

FROM bitwalker/alpine-elixir:1.5.2 as releaser

ENV HOME=/opt/app

# dependencies for comeonin
RUN apk add --no-cache build-base cmake

# Install Hex + Rebar
RUN mix do local.hex --force, local.rebar --force

ARG ERLANG_COOKIE
ENV $ERLANG_COOKIE $ERLANG_COOKIE

# Cache elixir deps
COPY config/ $HOME/config/
COPY mix.exs mix.lock $HOME/

ENV MIX_ENV=prod
RUN mix do deps.get --only $MIX_ENV, deps.compile

COPY . $HOME/

# Digest precompiled assets
```

DOCKER + DISTILLERY

- Fetch Elixir Dependencies
- Fetch & Build Assets
- Compile & Digest
- Release

```
####################################################

FROM alpine:3.6

ENV LANG=en_US.UTF-8 \
    HOME=/opt/app/ \
    TERM=xterm

ARG ERLANG_COOKIE
ENV ERLANG_COOKIE $ERLANG_COOKIE

ENV MYPROJECT_VERSION=0.0.2

RUN apk add --no-cache ncurses-libs openssl bash

EXPOSE 5000
ENV PORT=5000 \
    MIX_ENV=prod \
    REPLACE_OS_VARS=true \
    SHELL=/bin/sh

COPY --from=releaser \
  $HOME/_build/prod/rel/phoenix_test/releases/
$MYPROJECT_VERSION/phoenix_test.tar.gz $HOME \
  WORKDIR $HOME
```

DOCKER + DISTILLERY

- Fetch Elixir Dependencies
- Fetch & Build Assets
- Compile & Digest
- Release

```
    HOME=/opt/app/ \
    TERM=xterm

ARG ERLANG_COOKIE
ENV ERLANG_COOKIE $ERLANG_COOKIE

ENV MYPROJECT_VERSION=0.0.2

RUN apk add --no-cache ncurses-libs openssl bash

EXPOSE 5000
ENV PORT=5000 \
    MIX_ENV=prod \
    REPLACE_OS_VARS=true \
    SHELL=/bin/sh

COPY --from=releaser \
  $HOME/_build/prod/rel/phoenix_test/releases/
$MYPROJECT_VERSION/phoenix_test.tar.gz $HOME \
  WORKDIR $HOME

RUN tar -xzf phoenix_test.tar.gz

ENTRYPOINT ["/opt/app/bin/phoenix_test"]
CMD ["foreground"]
```

DOCKER + DISTILLERY

- We're not done yet
- We need other tools
  - Kubernetes
  - Helm
- All the dev ops
- Docker containers are firewalled
- No Hot Upgrades

HACK

What is this Gitlab?

OHHHHH

Package

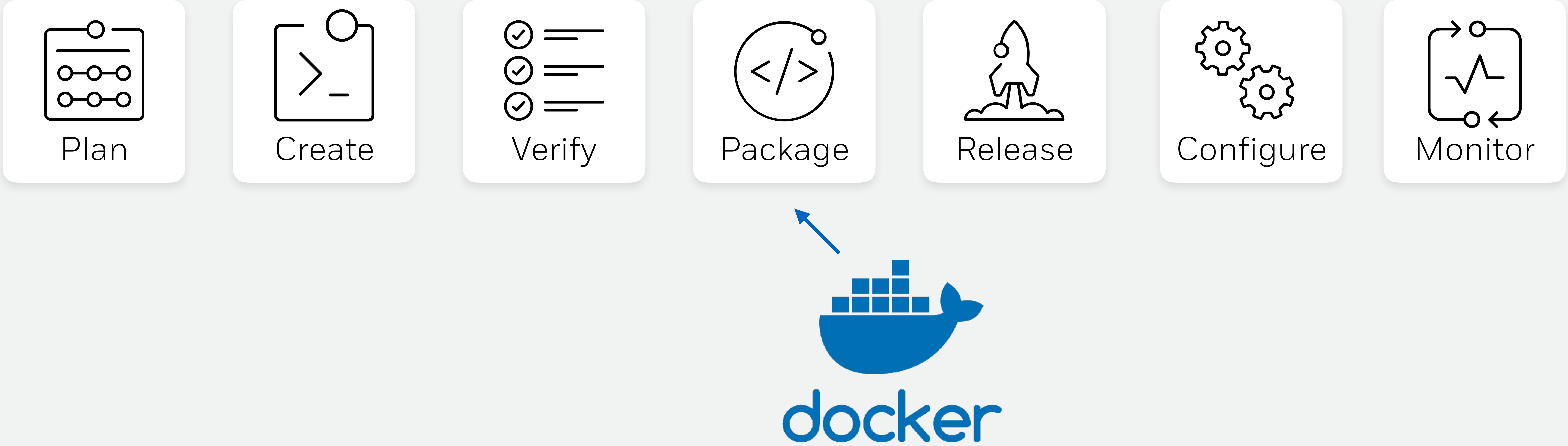
Release

Configure

Monitor

What is this Gitlab?

OHHHHH



Plan

Create

Verify

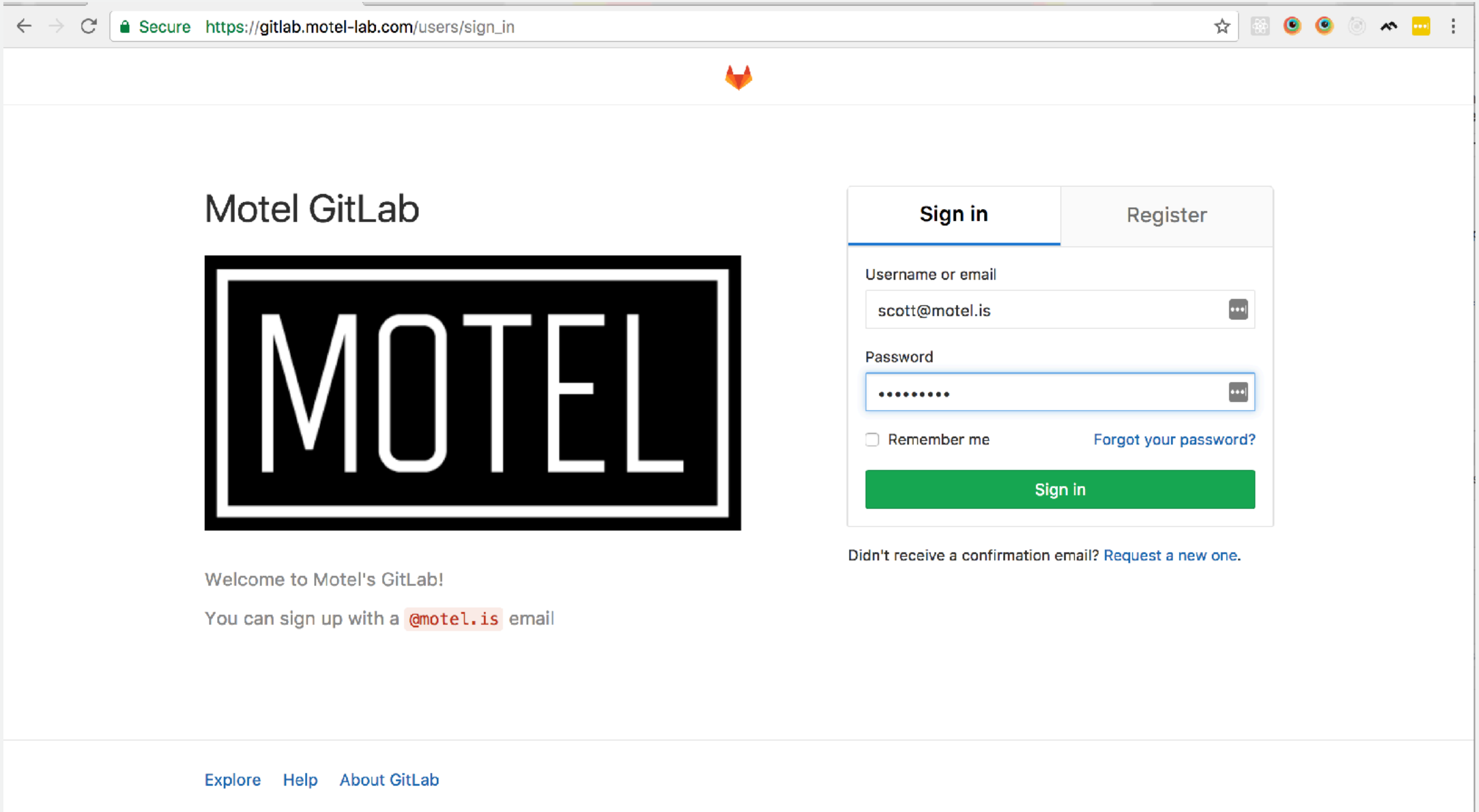
Package

Release

Configure

Monitor

GITLAB CI



Secure https://gitlab.motel-lab.com/users/sign_in

Motel GitLab

Welcome to Motel's GitLab!

You can sign up with a `@motel.is` email

Sign in | Register

Username or email: scott@motel.is

Password: ••••••••••

□ Remember me

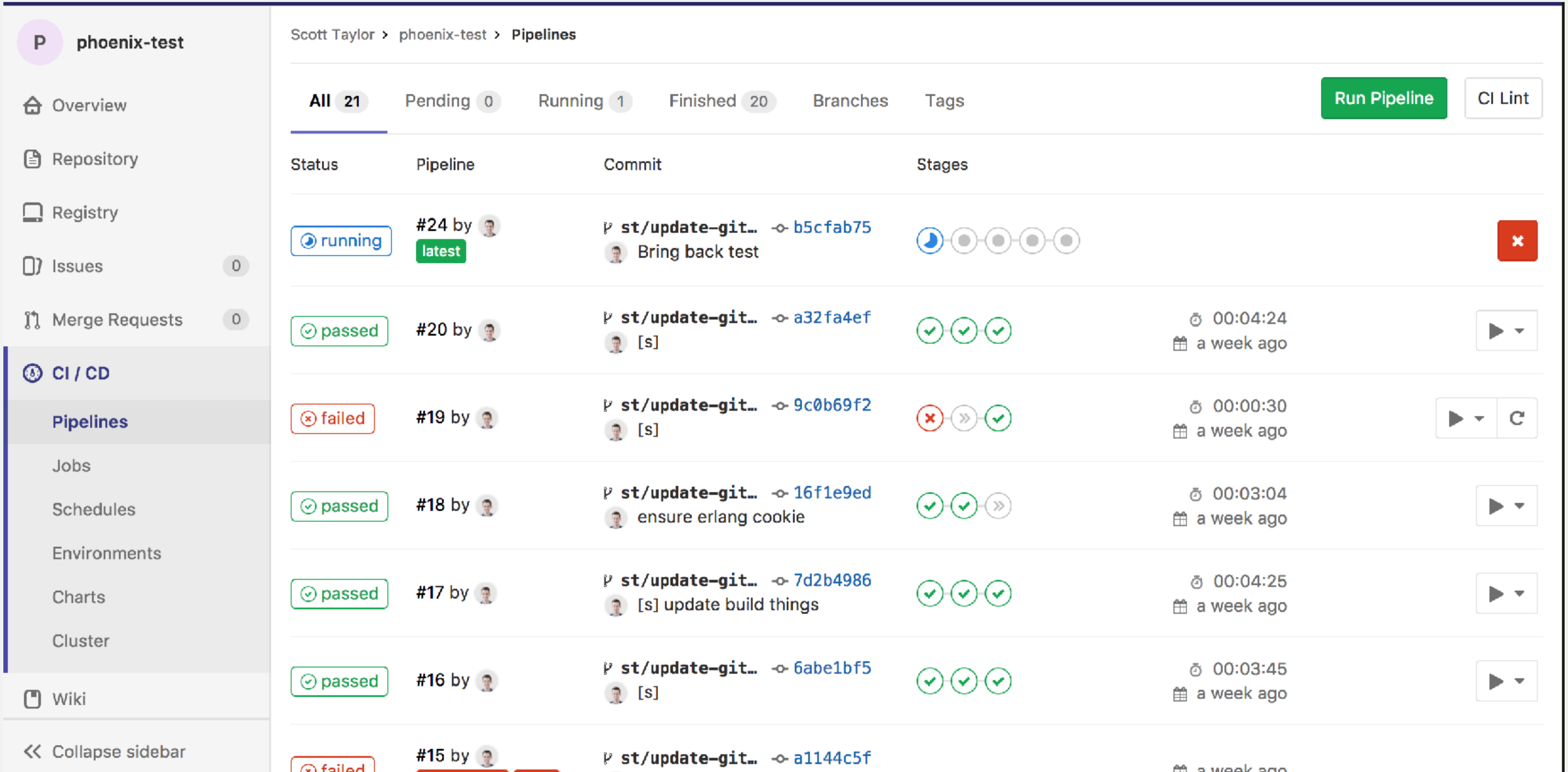
Forgot your password?

Sign in

Didn't receive a confirmation email? Request a new one.

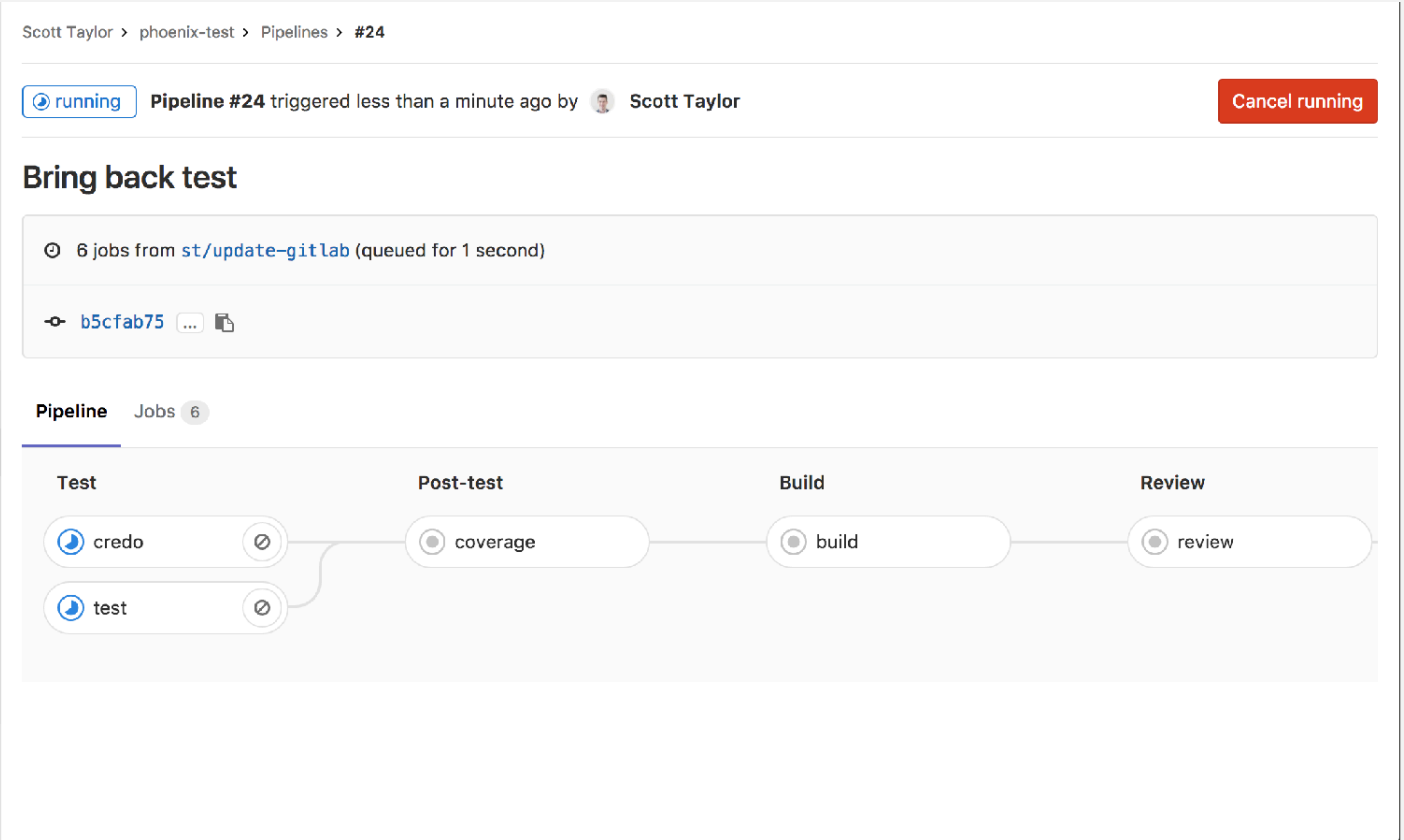
Explore Help About GitLab

GITLAB CI



Status	Pipeline	Commit	Stages	
running	#24 by latest	st/update-git… b5cfab75 Bring back test		
passed	#20 by	st/update-git… a32fa4ef [s]	✓ ✓ ✓	00:04:24 a week ago
failed	#19 by	st/update-git… 9c0b69f2 [s]	✗ » ✓	00:00:30 a week ago
passed	#18 by	st/update-git… 16f1e9ed ensure erlang cookie	✓ ✓ »	00:03:04 a week ago
passed	#17 by	st/update-git… 7d2b4986 [s] update build things	✓ ✓ ✓	00:04:25 a week ago
passed	#16 by	st/update-git… 6abe1bf5 [s]	✓ ✓ ✓	00:03:45 a week ago
failed	#15 by	st/update-git… a1144c5f		a week ago

GITLAB CI



Scott Taylor › phoenix-test › Pipelines › **#24**

running **Pipeline #24** triggered less than a minute ago by **Scott Taylor**

Cancel running

Bring back test

6 jobs from `st/update-gitlab` (queued for 1 second)

`b5cfab75` ...

Pipeline Jobs 6

Test	Post-test	Build	Review
credo	coverage	build	review
test			

GITLAB-CI.YML

- We define the stages for the devops cycle

```
image: alpine:latest

variables:
  MIX_ENV: "test"

stages:
  - test
  - post-test
  - build
  - review
  - production
  - cleanup

.test_job_template: &test_job_definition
  image: elixir:1.5.2
  before_script:
    - apt-get update
    - apt-get install -y postgresql-client
    - mix local.hex --force
    - mix local.rebar --force
    - mix deps.get --only test

build:
  stage: build
  image: docker:git
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests

```
test:
  <<: *test_job_definition
  stage: test
  services:
    - postgres:9.6
  script:
    - "mix ecto.reset"
    - "mix coveralls.html | tee cov.out"
  artifacts:
    paths:
      - cov.out
      - cover/

credo:
  <<: *test_job_definition
  stage: test
  script:
    - mix credo --strict

coverage:
  image: alpine
  stage: post-test
  script:
    - cat cov.out
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel

```
credo:
  <<: *test_job_definition
  stage: test
  script:
    - mix credo --strict

coverage:
  image: alpine
  stage: post-test
  script:
    - cat cov.out
  coverage: '/\[TOTAL\]\s+(\d+\.\d+%)$/'

review:
  stage: review
  variables:
    ERLANG_COOKIE: $ERLANG_COOKIE
  script:
    - check_kube_domain
    - install_dependencies
    - download_chart
    - ensure_namespace
    - install_tiller
    - create_secret
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel
- Collect Code Coverage results

```
  script:
    - mix credo --strict

coverage:
  image: alpine
  stage: post-test
  script:
    - cat cov.out
  coverage: '/\[TOTAL\]\s+(\d+\.\d+%)$/'

review:
  stage: review
  variables:
    ERLANG_COOKIE: $ERLANG_COOKIE
  script:
    - check_kube_domain
    - install_dependencies
    - download_chart
    - ensure_namespace
    - install_tiller
    - create_secret
    - deploy
  environment:
    name: review/$CI_COMMIT_REF_NAME
    url: http://$CI_PROJECT_PATH_SLUG-
```

GITLAB-CI.YML

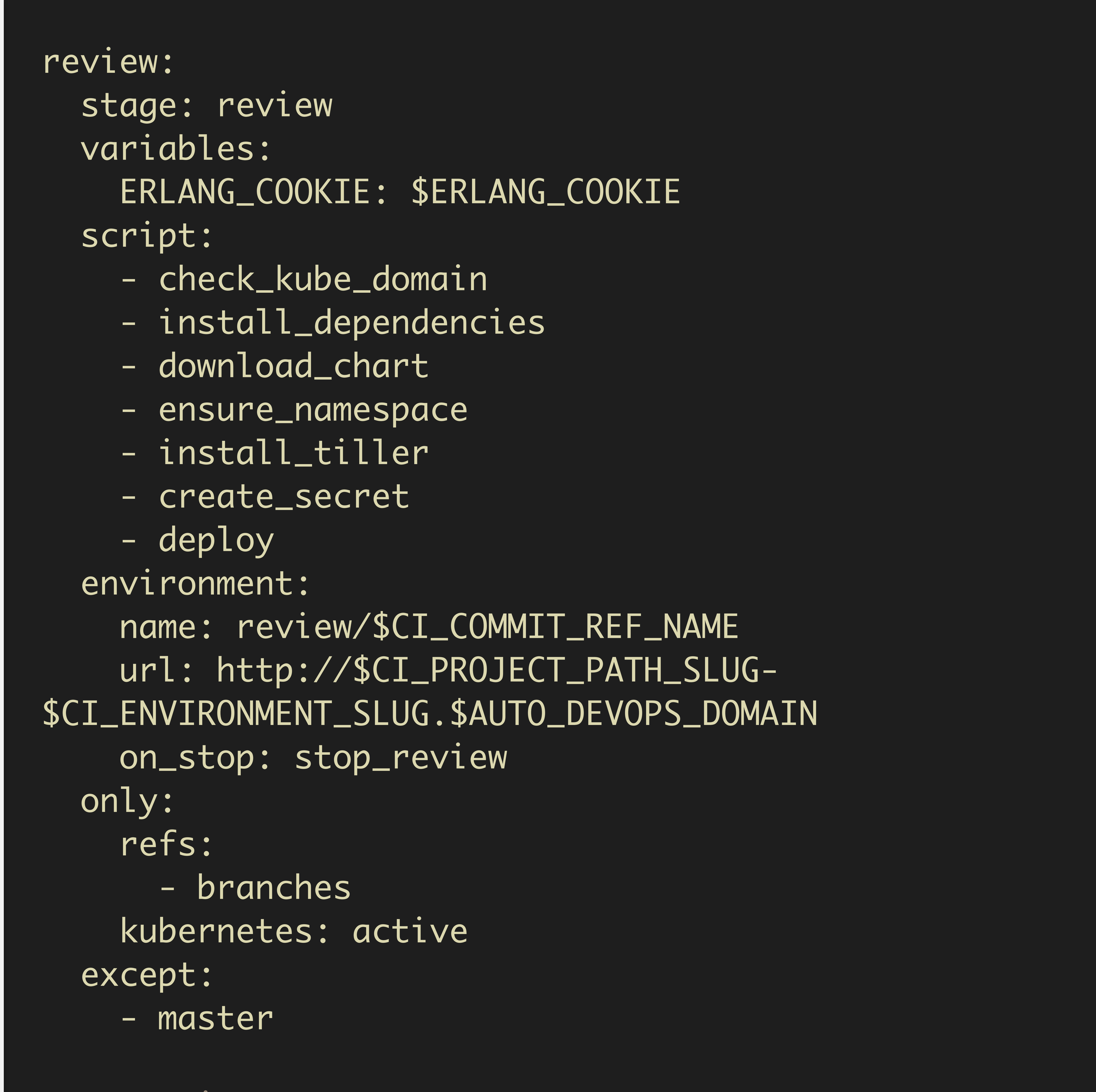
- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel
- Collect Code Coverage results
- Build a distillery release with docker

```yaml
build:
  stage: build
  image: docker:git
  services:
  - docker:dind
  variables:
    DOCKER_DRIVER: overlay2
    ERLANG_COOKIE: "foobar"
  script:
    - check_erl_cookie
    - setup_docker
    - build
  only:
    - branches

test:
  <<: *test_job_definition
  stage: test
  services:
    - postgres:9.6
  script:
    - "mix ecto.reset"
    - "mix coveralls.html | tee cov.out"
  artifacts:
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel
- Collect Code Coverage results
- Build a distillery release with docker
- Use Kubernetes to fire up a review app



```
review:
  stage: review
  variables:
    ERLANG_COOKIE: $ERLANG_COOKIE
  script:
    - check_kube_domain
    - install_dependencies
    - download_chart
    - ensure_namespace
    - install_tiller
    - create_secret
    - deploy
  environment:
    name: review/$CI_COMMIT_REF_NAME
    url: http://$CI_PROJECT_PATH_SLUG-
$CI_ENVIRONMENT_SLUG.$AUTO_DEVOPS_DOMAIN
    on_stop: stop_review
  only:
    refs:
      - branches
    kubernetes: active
  except:
    - master
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel
- Collect Code Coverage results
- Build a distillery release with docker
- Use Kubernetes to fire up a review app
- Once merged, stop the review app

```yaml
stop_review:
  stage: cleanup
  variables:
    GIT_STRATEGY: none
  script:
    - install_dependencies
    - delete
  environment:
    name: review/$CI_COMMIT_REF_NAME
    action: stop
  when: manual
  allow_failure: true
  only:
    refs:
      - branches
    kubernetes: active
  except:
    - master

production:
  stage: production
  script:
    - check_kube_domain
    - install_dependencies
```

GITLAB-CI.YML

- We define the stages for the devops cycle
- Run ExUnit tests
- Run Credo in parallel
- Collect Code Coverage results
- Build a distillery release with docker
- Use Kubernetes to fire up a review app
- Once merged, stop the review app
- Use deliver to hot upgrade production

```
      master

production:
  stage: production
  script:
    - check_kube_domain
    - install_dependencies
    - download_chart
    - ensure_namespace
    - install_tiller
    - create_secret
    - deliver
 environment:
    name: production
    url: http://$CI_PROJECT_PATH_SLUG.
$AUTO_DEVOPS_DOMAIN
#  when: manual
  only:
    refs:
      - master
    kubernetes: active

#
-----------------------------------------------------
------
```

What is this Gitlab?

OHHHHH

Plan

Create

Verify

Package

Release

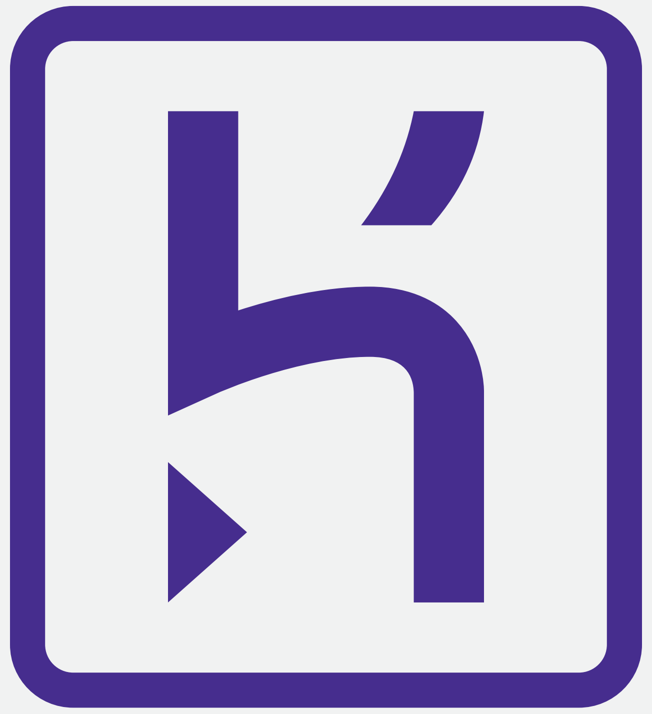
Configure

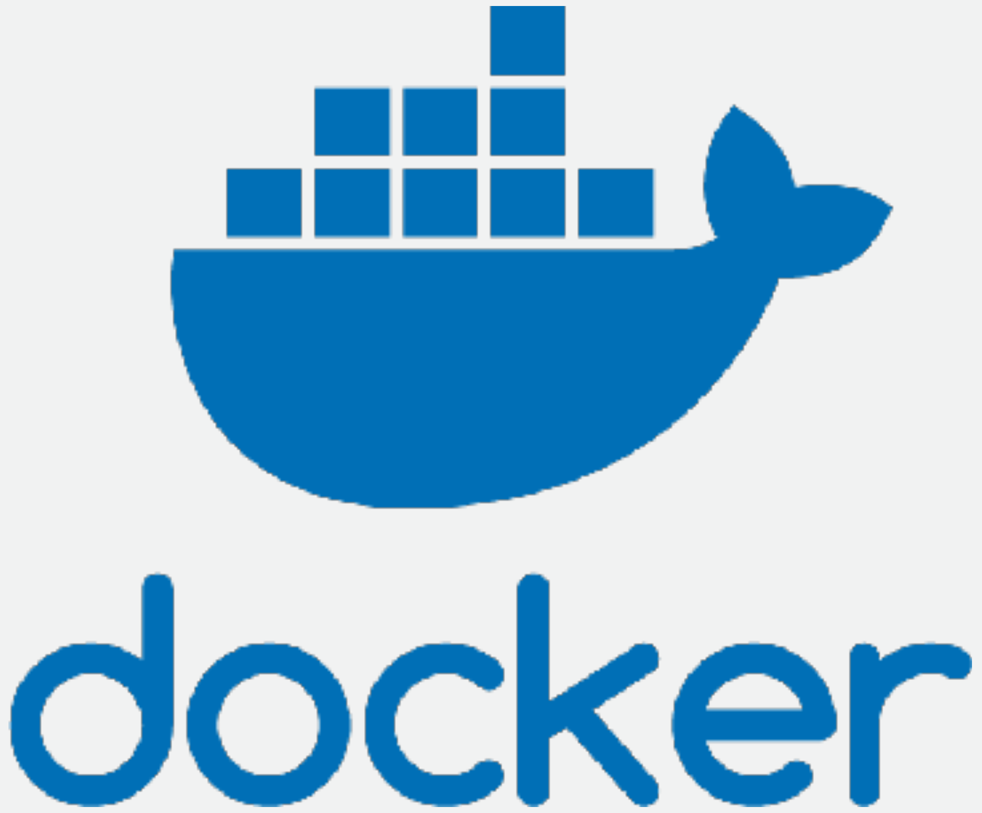
Monitor

That is so smooth!

So what have we learned?

NEARLY THERE





Thanks!

QUESTIONS?

Lead Engineer at Motel
@staylorwr

MOTEL