Basic familiarity with Heroku is assumed.

Howdy!

I'm Mårten Gustafson

I’m Mårten

- My speaker bio says I work at *Plan 3*, but we're actually a part of Schibsted
- (*Plan 3 was an internal startup within Schibsted*)
- I serve as the tech lead for one of our internal software platforms

Why?

- Why did we migrate from virtualised servers to Heroku?

Servers

We saw no need

- Heroku seemed to be the easiest and most mature PaaS platform
- Our codebase was already split up into separate components (*but with weird interdependencies*)

Architecture

12 factor compliance

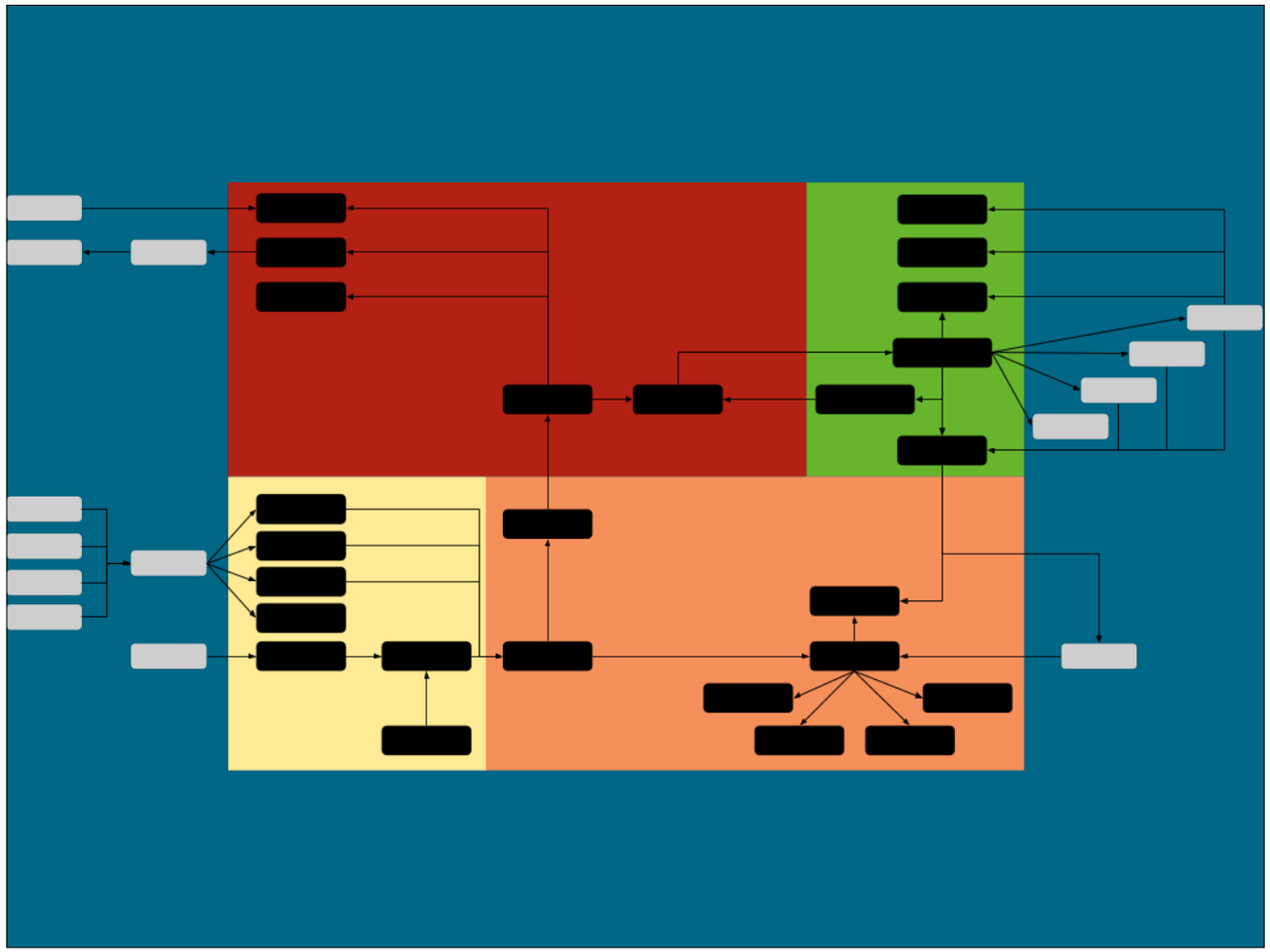
- The 12 factor methodology (12factor.net) on how to build services is derived from Heroku
- A system built in accordance with 12 factor has high portability
- Using Heroku mostly guarantees compliance with 12 factor (*without thinking about it*)
- Migrating to Heroku forced us to clean up our code base

DevOps

No ops/infra team

- The team does all ops and infra work
- Everyone participates in on-call rotation

Our setup



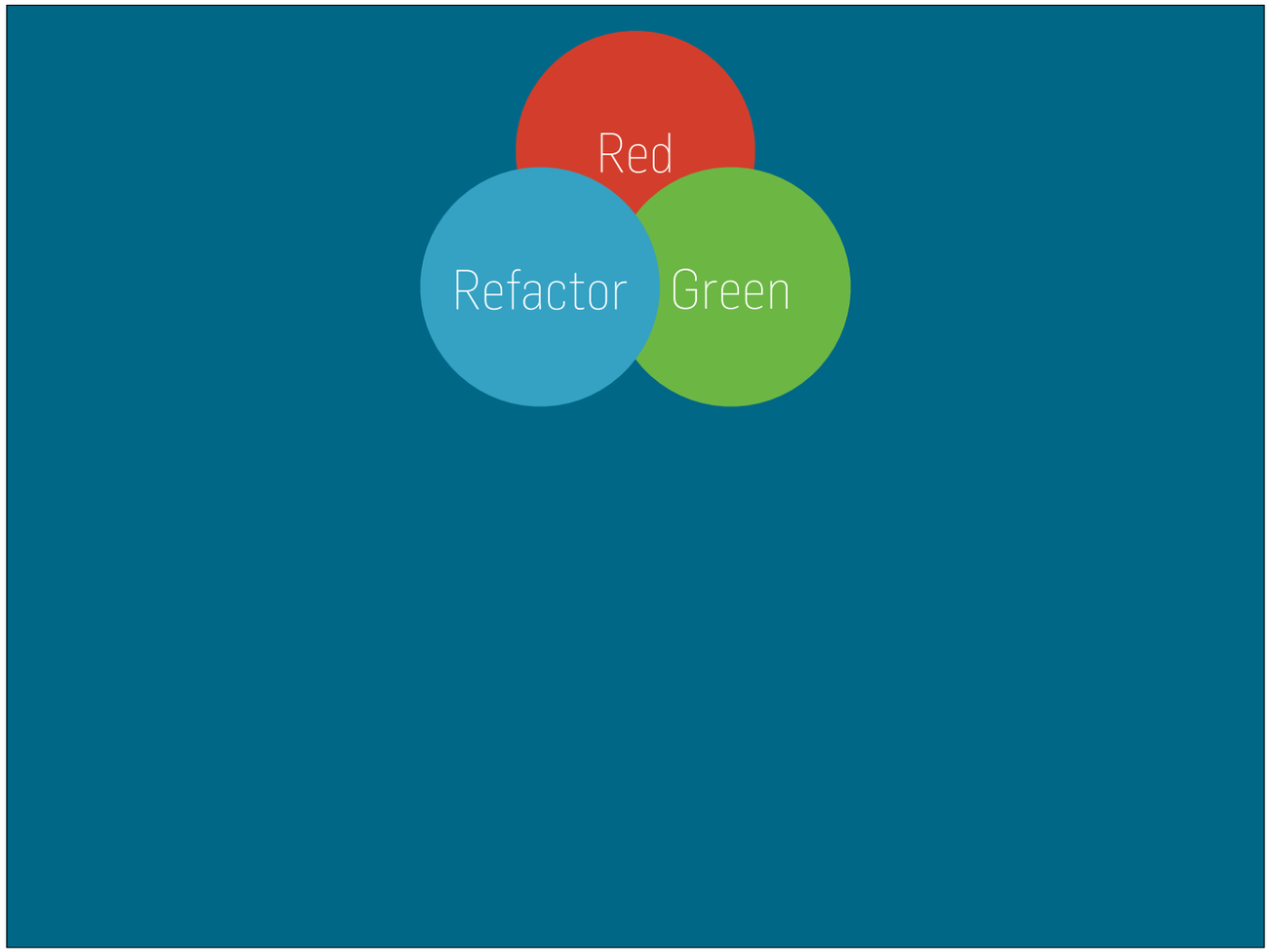
- The system I is made up of around 25 individual services (*1 staging & 1 prod = 50 apps*)
- This is a simplified (*and anonymised*) solution diagram
- Each black box is a service running on Heroku
- Coloured boxes represent bounded contexts (domain driven design)
- This is developed and operated by a team of 20 spread across Oslo, Kraków & Stockholm

Our history

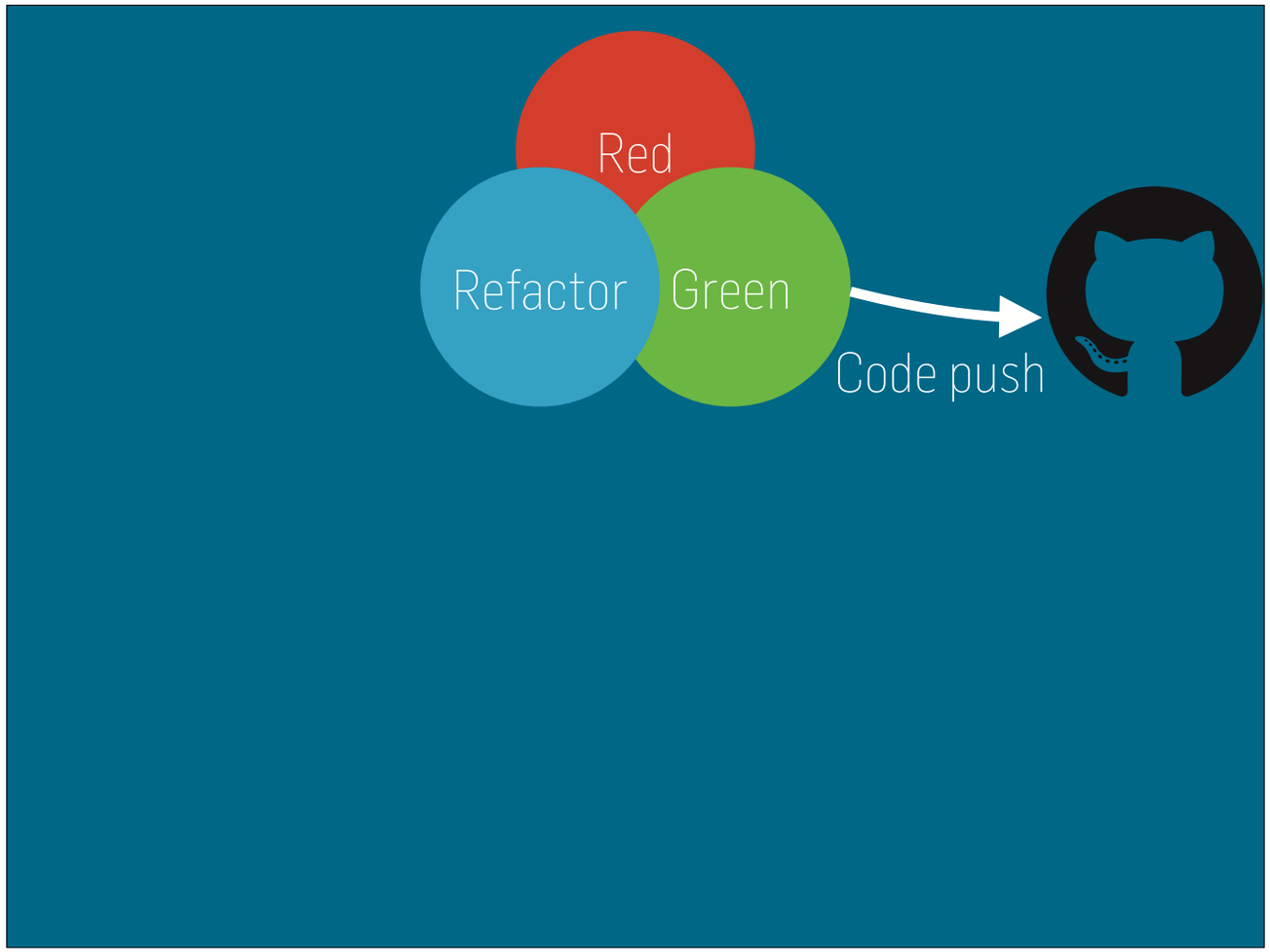
Glesys => UpCloud => Heroku

- Initially we ran everything on virtualised servers which we managed ourselves
- In two years we've been through three hosting environments for production

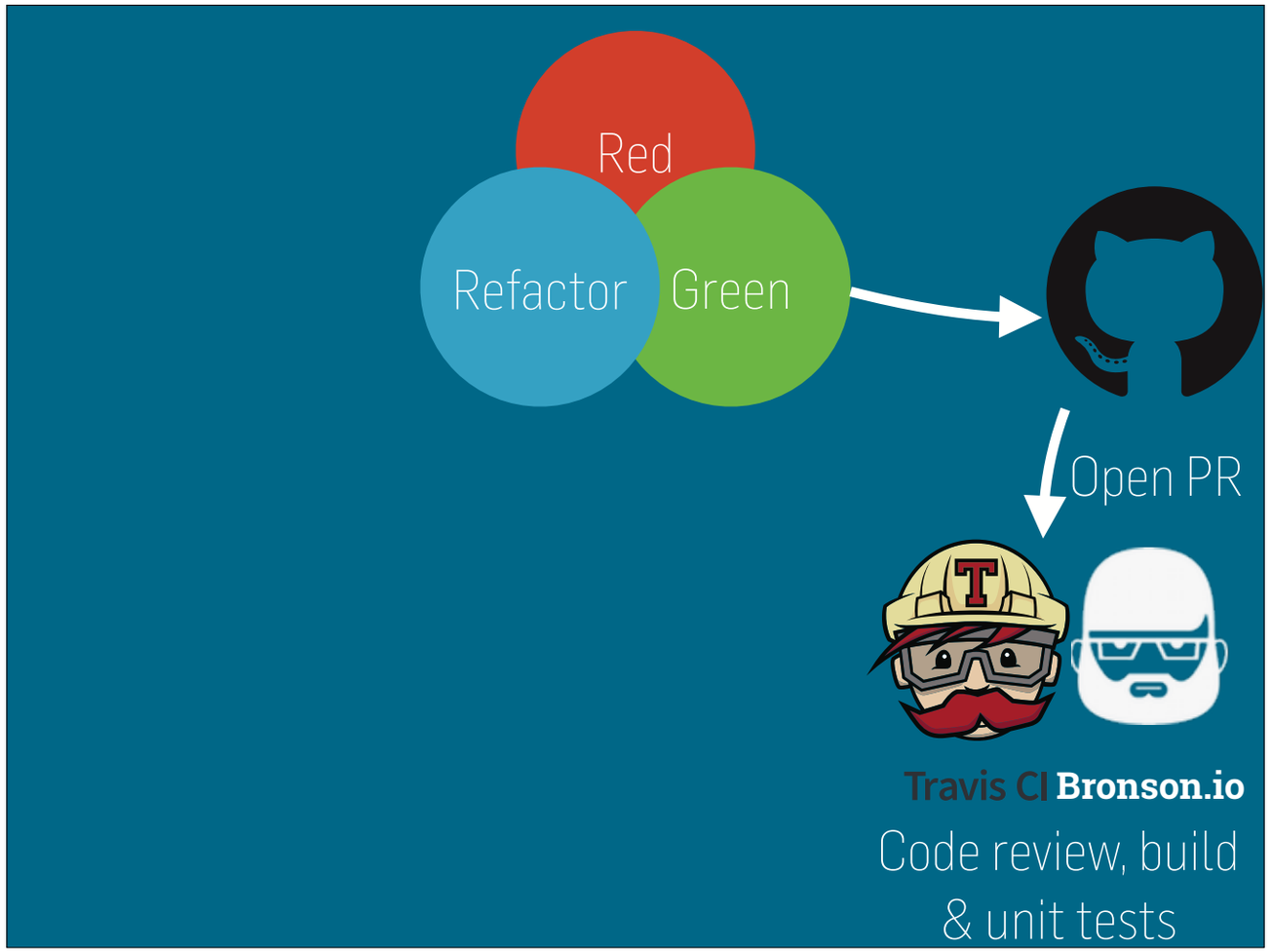
- First experiments in November 2013
- Then gradual migration and refactoring on a per service basis
- Finally terminated the old servers September 2014 after finally pushing through and migrating the most intertwined service (and refactoring it substantially)



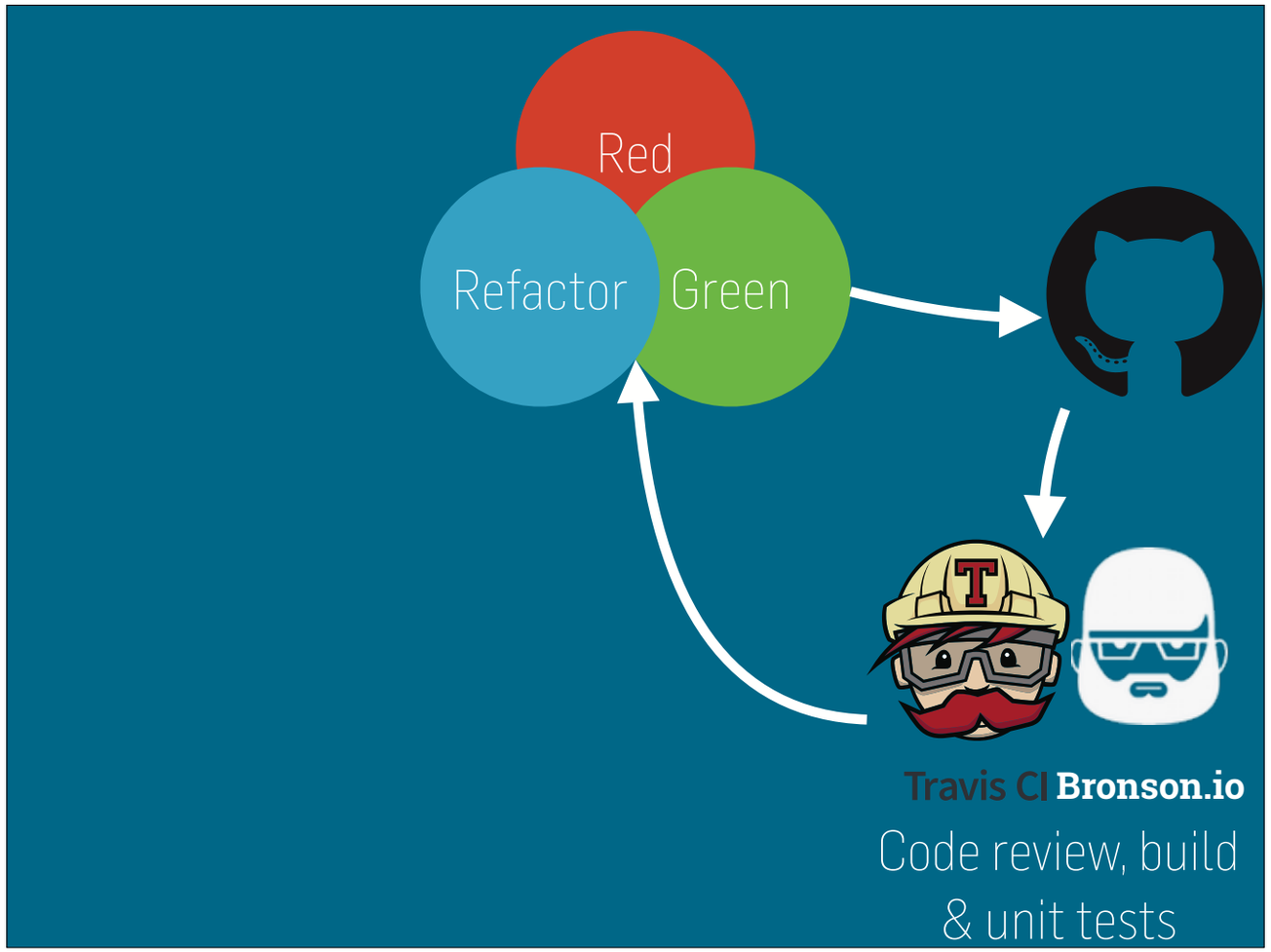
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



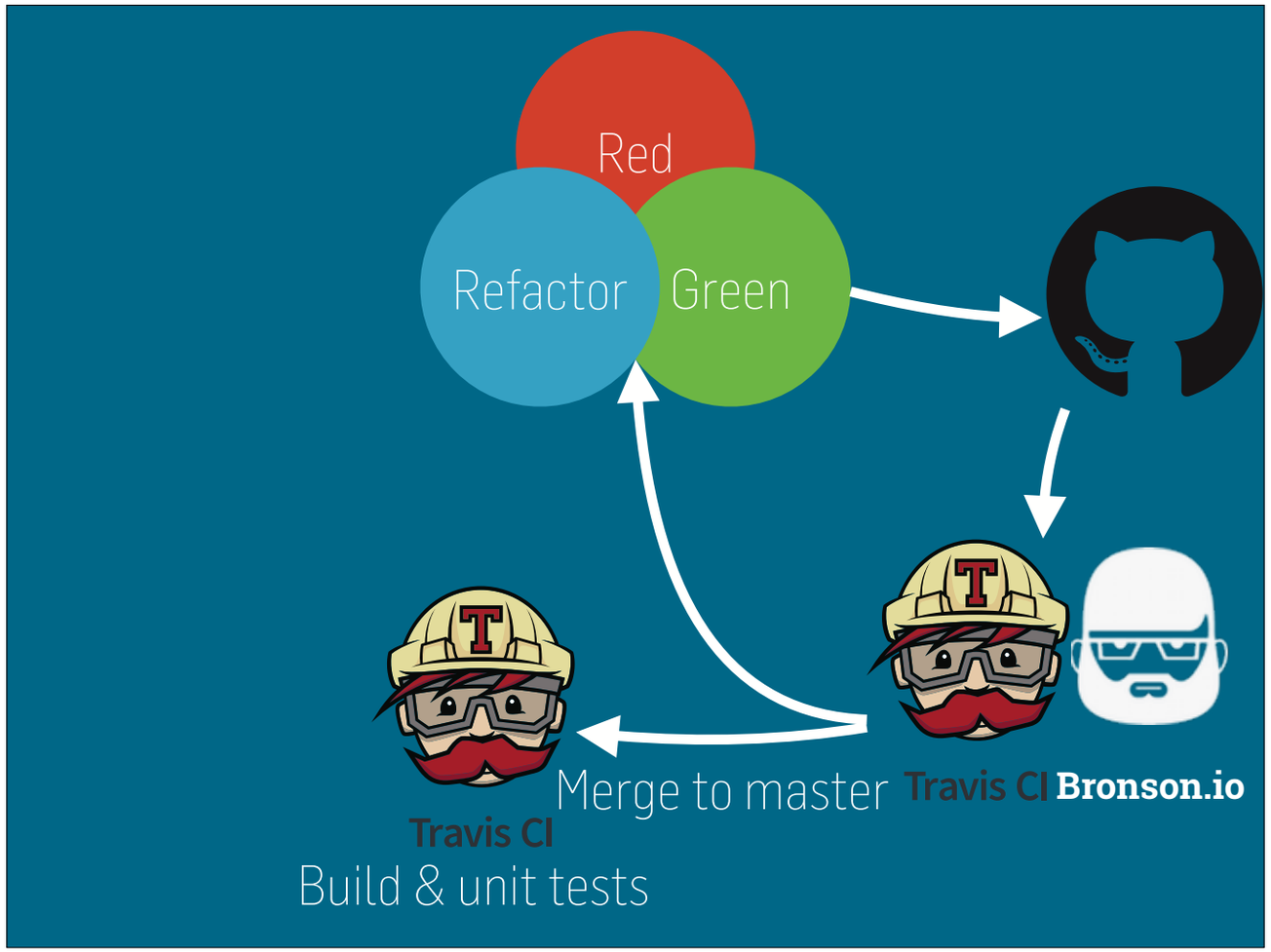
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



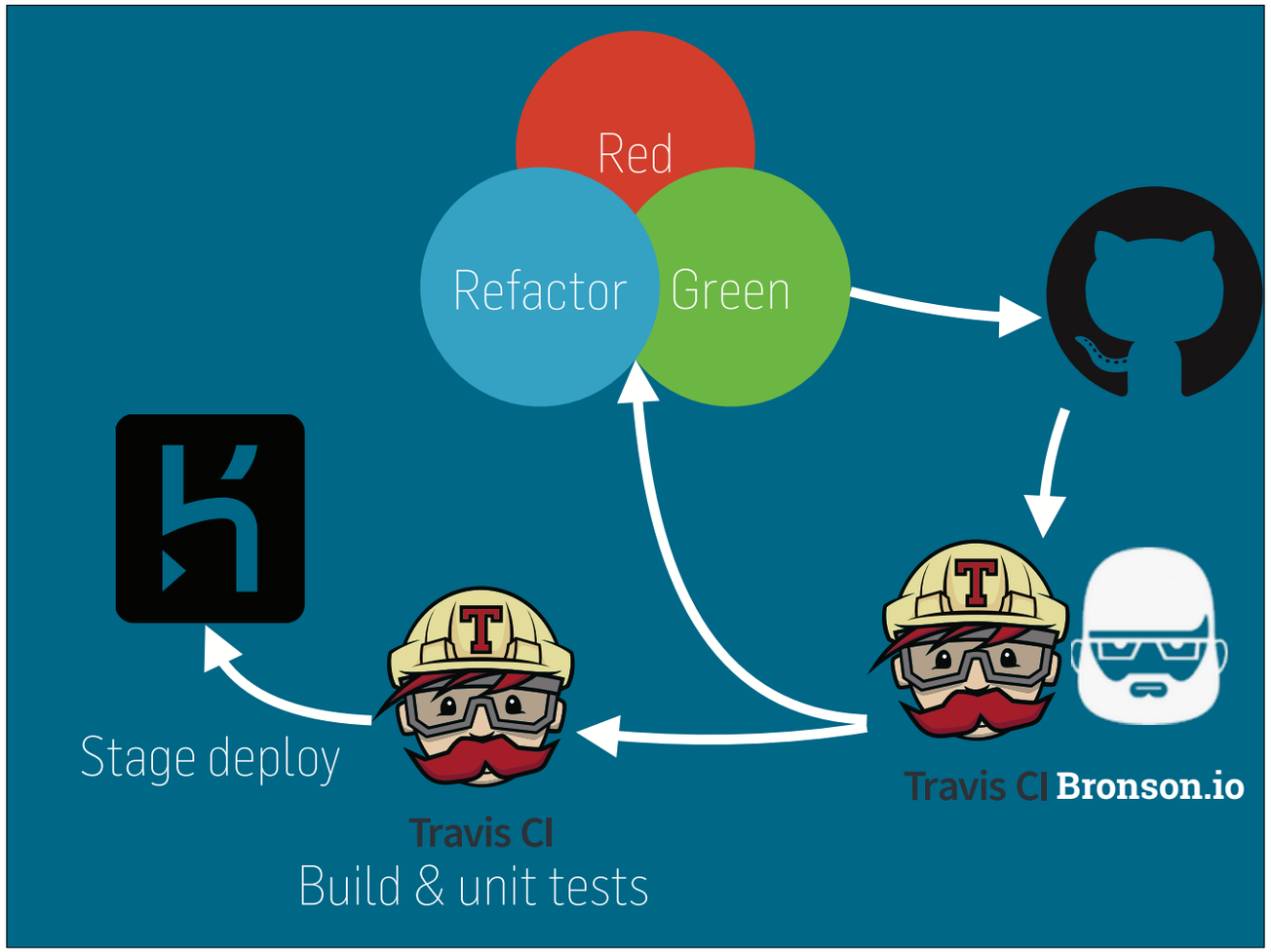
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



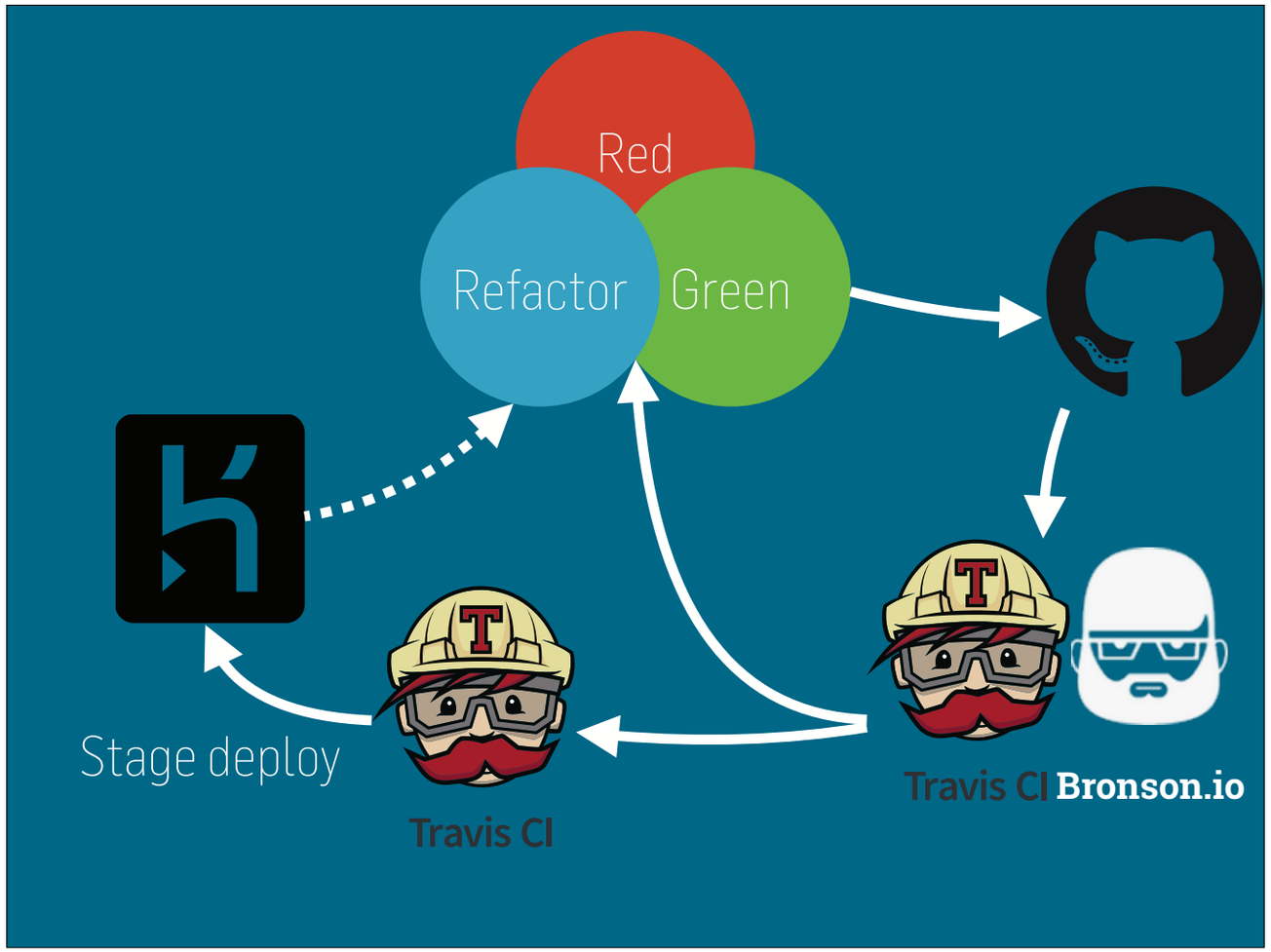
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



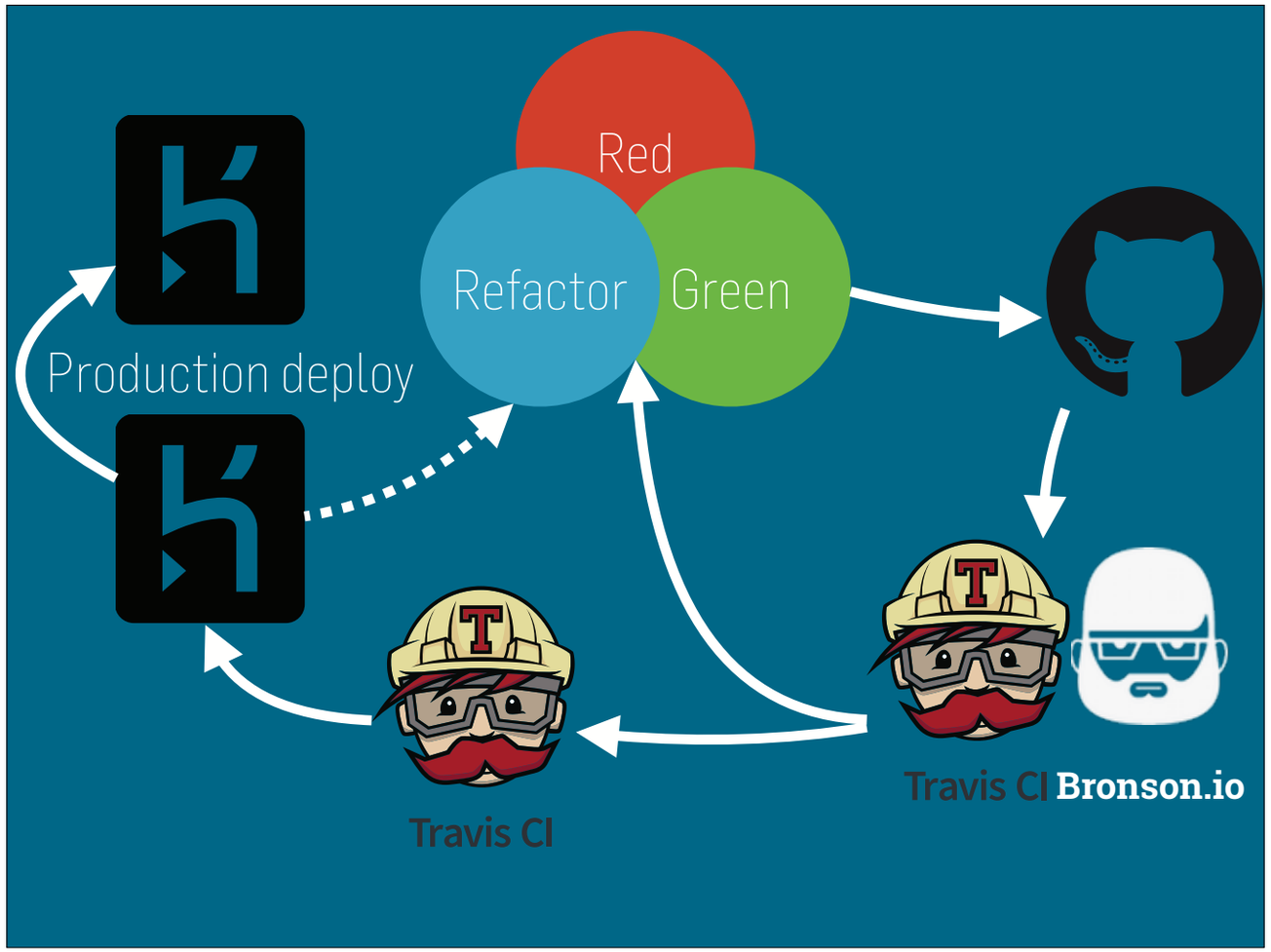
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



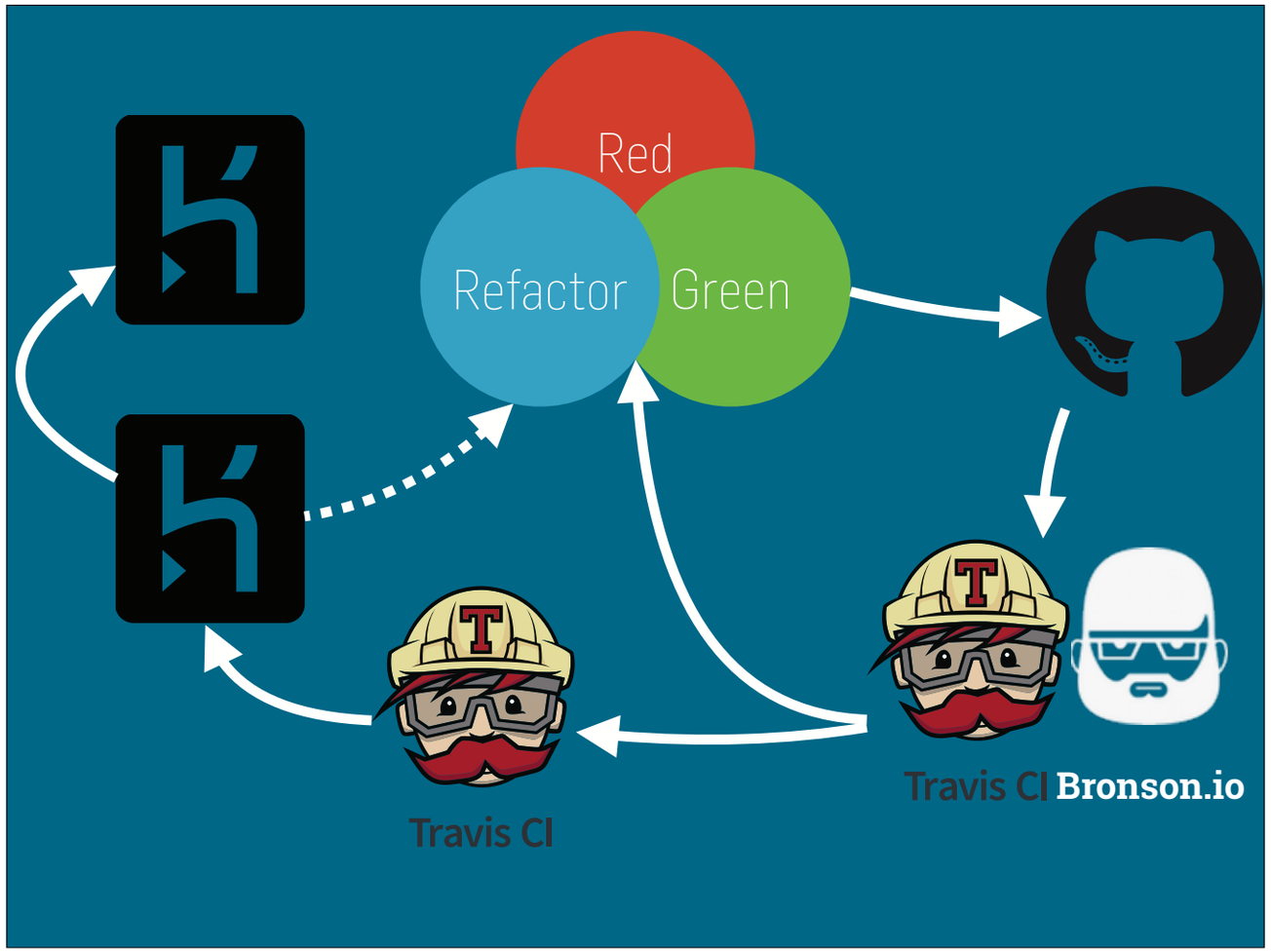
- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku



- Github for source code
- Bronson.io code reviews our pull requests (*and comments inline on Github*)
- Travis for continuous integration
- Every successful build on *master* is deployed by Travis to the staging app on Heroku

Learnings

Organizations

Organisation == Environment

- This allows us to separate Heroku organisation admins between environments
- Every service has 2 Heroku applications, (*1 in stage org, 1 in prod org*)
- Organizations also solves billing (*allows individuals to experiment with paid add-ons etc*)

Preboot

Zero downtime deploy

- Use at least 2 web dynos combined with Heroku Preboot to get zero downtime deployments

Add-ons

per-app per-env control

- Might be more expensive than central storage etc
- Storage and tooling isolation and freedom
- Self administration
- Per app per environment scaling (*and thus pricing per app per org*)
- Performance testing shows cost impact immediately

Delegate

Access & responsibility

- Heroku fits perfectly with our autonomous and decentralised organisation (*enforce 2FA*)
- Everyone can create new apps, tweak existing ones
- Allows rapid prototyping and experimentation (*in a production like setting*)
- Proof of concepts are trivial to “promote” to production (*+1 app & tweak add-ons*) or toss (*delete app*)

Pipelines

Don't push, promote

- Use the Heroku pipeline feature
- We never push code to production, instead it's always promoted from staging
- This forces us to have externalised all environment specific configuration from the app itself
  (as per the 12 factor manifesto)

Auto-deploy

At least to staging

- You can always circumvent this with manual push'es (*branch deployment*)

Transparency

Audit trail & chat log

- Heroku provides audit trail (*deploys and configuration changes*)
- Heroku provides basic runtime metrics
- Simple integration with chat for visibility into deploys (*both staging & prod*)

Challenges

Service discovery

We use DNS

- We have one domain for stage and one for production
- Services are named the same in both domains (*simple and self-explanatory*)

Auto-scaling

API vs add-ons

- The hard part is deciding on what criteria/metric to scale
- We don’t currently use any auto-scaling

Provisioning

terraform.io

- Currently we do this manually and with shell scripts (*Heroku command client*)
- Terraform would allow automatic provisioning across Heroku & other services (*AWS Route 53*)

SSL certificates

manual vs add-ons

- Not fun manually managing and deploying wild card certificates across 50 services
- We're evaluating an add-on (*Expedited SSL*)
- Pros: Simple provisioning & automatic renewal
- Cons: Much more expensive than buying 1 wild card cert and reusing that

Questions?

marten@plan3.se

- Thanks, questions?

Idea:

PaaS as self-service dev env?

- I have this idea that most organisations could use a PaaS such as Heroku as their development environment as it would give…
  - …developers free reign to a self-service environment (*fosters experimentation*)
  - …cost/budget transparency (*fosters responsibility*)
  - …enforces good solution design and portability (*using carrot rather than stick*)