Features

Feature branches

Feature flags

Feature toggles

@compiledwrong

What is a feature?

A thing that your thing (application) does, that users use, that is a coherent chunk of thing-ness

What is a feature flag/toggle?

not A/B testing (which requires much monitoring & data analysis)

- Some features should not be in production YET
  - (so put them behind a (fancy) if statement)
- BUT we want to not have long lived feature branches
  - (because merging is terrible if someone refactors or makes cross-cutting changes)
- SO we have to do (something)

Taxonomy of flags/toggles

- **very temporary** (only while a new feature is being developed, to enable developer workflow) OR **long-lasting** (very tempting, more dangerous)
- **When** the state of the flag can be changed (runtime or deploy/restart -time)
- What level of **granularity** the flag controls: application-wide OR user-specific (maybe with groups of users!)

Where do you keep them?

config file (deploy-time)

OR

environmental variable sourced from external system (deploy-time)

OR

database/datastore or API calls to external config system (runtime enablement)

NOT

NOT! in-memory (resets on deploy)

Who toggles your toggles, and how?

- **Who** changes them? (Developers via commit? Ops via env variable + restart? Product via UI?)
- JMX (reduce chance of unauthorized UI/external access)
- Developers, in a commit, which is then deployed
- [mild danger] Admin-only console for your Product team (access protection)
- [DANGER] Someone runs a sql insert in prod (harder to get traceability/logging)

Log everything!

User-specific feature flags/toggles

- gradual rollout (trusted partners first)
- (accidental) load testing (forgot an index?)
- automatic enablement of server-side feature on iOS/Android app upgrade

Long-lasting feature toggles

1. increase developer pain :(
2. gradual rollout and frequent rollback
3. long-maintained iOS/Android client versions in the wild that can/not support different features
4. [DANGER] combinatorial explosion of testing complexity
5. [DANGER] did you accidentally write a permissions system?

[DANGER]

accidentally using feature flags as a user/admin permissions system

- charging $$ per feature
- features for certain users only (i.e. roles)
- A/B testing (additional effort required for meaningful metrics)
- log everything!

[DANGER]

(feature) enablement vs (user) setting

(the user configuring their usage of the feature)

Once a feature is enabled, a user can configure it. What do you do with the configuration if the feature is turned off again? Keep it? Delete it? What if the configuration options mean different things when you turn the feature back on again?

Really weird data bugs

Actual Code

Ruby

- github.com/FetLife/rollout
- github.com/jnunemaker/flipper

Java

- github.com/clun/ff4j
- github.com/togglz/togglz

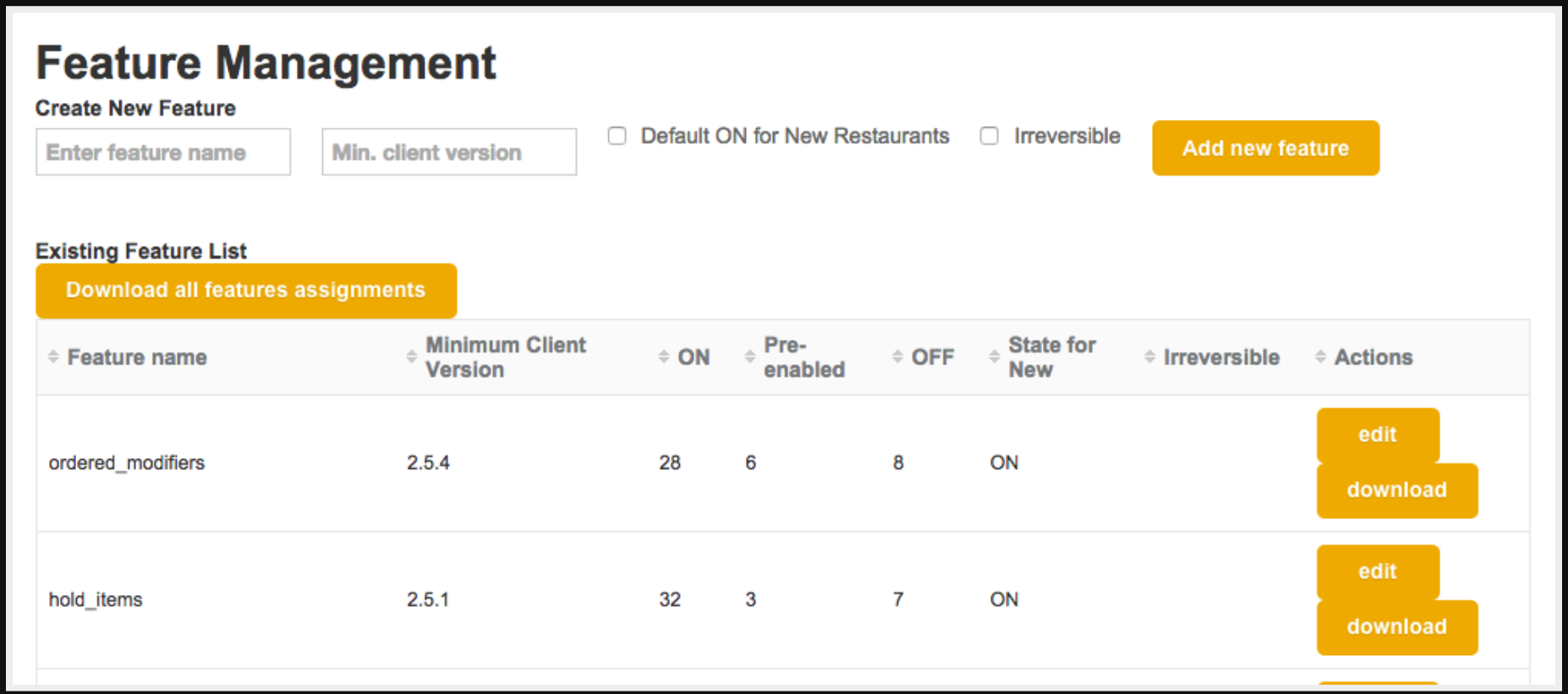
Above and beyond:

- github.com/github/scientist

Screenshots! (from UAT)

from BreadcrumbPro

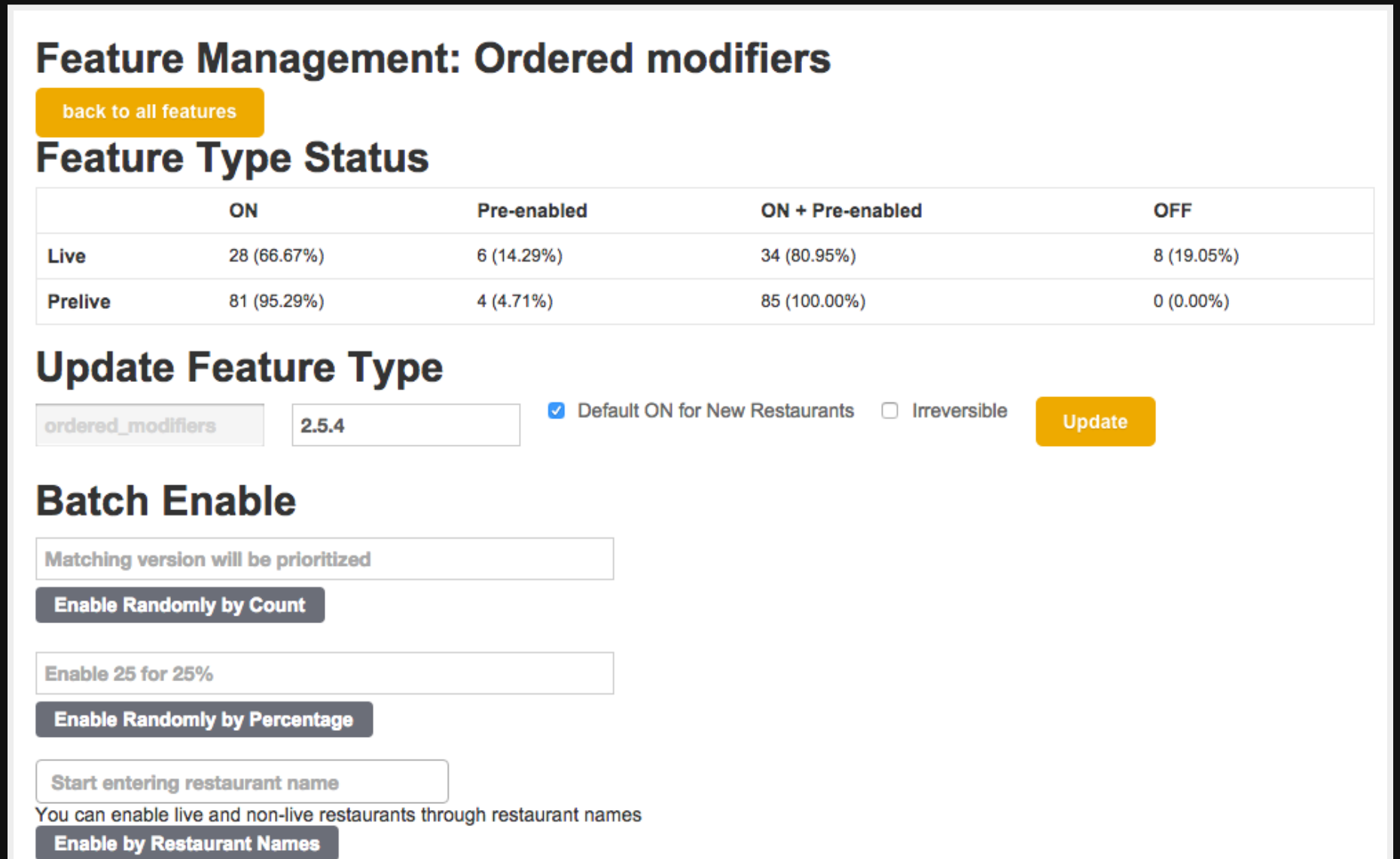
Product users love it. All of the power, none of the danger.




```
before_destroy :ensure_feature_is_reversible
def ensure_feature_is_reversible
  !feature_type.irreversible? # double negative
end
```

Screenshots! (from UAT)

from BreadcrumbPro



Assigned Restaurants

1 2 3 4 5 … Next › Last »

Restaurant Name	Required Version	Status
(automation) Megatron	2.3.0	Pre-enabled
(benm) Benm-redux	2.6.7	On
(dwalin) Testin' Time	2.6.2	On

DB schemae

modified, based on BreadcrumbPro github.com/compwron/feature_feature

Features have attributes, which differ per restaurant

```
create_table :restaurants do |t|
  t.string :client_version, null: false
  # other important stuff, like name
end

create_table :features do |t|
  t.timestamps
  t.string   :name, null: false
  t.string   :minimum_client_version, null: false
  t.boolean  :on_for_new_restaurants, null: false, default: false
  t.boolean  :reversible, null: false, default: false
end

create_table :restaurant_features do |t|
  t.uuid     :restaurant_id, null: false
  t.uuid     :feature_id, null: false
end
add_index :restaurant_features, [:feature_id]
add_index :restaurant_features, [:restaurant_id]
```

FeatureToggle.rb

modified, based on BreadcrumbPro github.com/compwron/feature_feature

```
module Concerns::Restaurant::FeatureToggle
  extend ActiveSupport::Concern

  included do
    after_update :update_features, if: :client_version_changed?
    on_create :initiate_features
  end

  def update_features
    Feature.find_each do |feature_type|
      feature = self.features.where(feature_type_id: feature_type.id).first_or_initiali
      feature.update_status
      feature.save! # in the real world, there is error handling here
    end
  end

  def initiate_features
    Feature.where(new_restaurants_on: true).find_each do |feature|
      rfeature = restaurant_features.where(feature_id: feature.id).new
      rfeature.initialize_enable
    end
```

Feature.rb

modified, based on BreadcrumbPro github.com/compwron/feature_feature

```
class Feature < ActiveRecord::Base
  UNSAFE_FEATURES = ['place_holder', Features::BAD_FEATURE_2]
  scope :safe_to_enable, -> { where "name not in (?)", UNSAFE_FEATURES }

  has_many :restaurant_features, dependent: :destroy # important!!
  after_update :update_feature_list, if: :client_version_changed?

  def update_status
    if enabled? && !has_sufficient_version?
      pre_enable
    elsif pre_enabled? && has_sufficient_version?
      enable
    end
  end

  def update_feature_list
    features.includes(:feature_owner).find_each do |feature|
      update_feature(feature)
    end
  end
end
```

Lessons learned

(Over and over and over)

Requirements evolve! Use evolutionary design. The first thing is rarely the right thing.

Lessons Learned

How do you migrate features between feature toggle systems? How do you test the migration? Especially if their states are being modified even during the deploy (for business reasons)?

Lessons Learned

Deploy strategy: how do you feature toggle the feature toggle feature?

We first encapsulated every toggle in a class, then backfilled and traded out the contents of that one class in one closely-monitored deploy.

Thanks

Thanks, BreadcrumbPro team!!

AMA => twitter.com/compiledwrong

compwron.github.io/presentations/feature_feature/index.html

compwron.github.io/2015/12/10/features-of-features-with-rails

github.com/compwron/feature_feature

References

martinfowler.com/bliki/FeatureToggle.html

martinfowler.com/articles/feature-toggles.html

devchat.tv/ruby-rogues/252-rr-feature-toggles-with-pete-hodgson