No dedicated QA role, no staging servers, faster deploys and better ownership

Testing directly in production

Bibek Shrestha, Nov 8, 2022

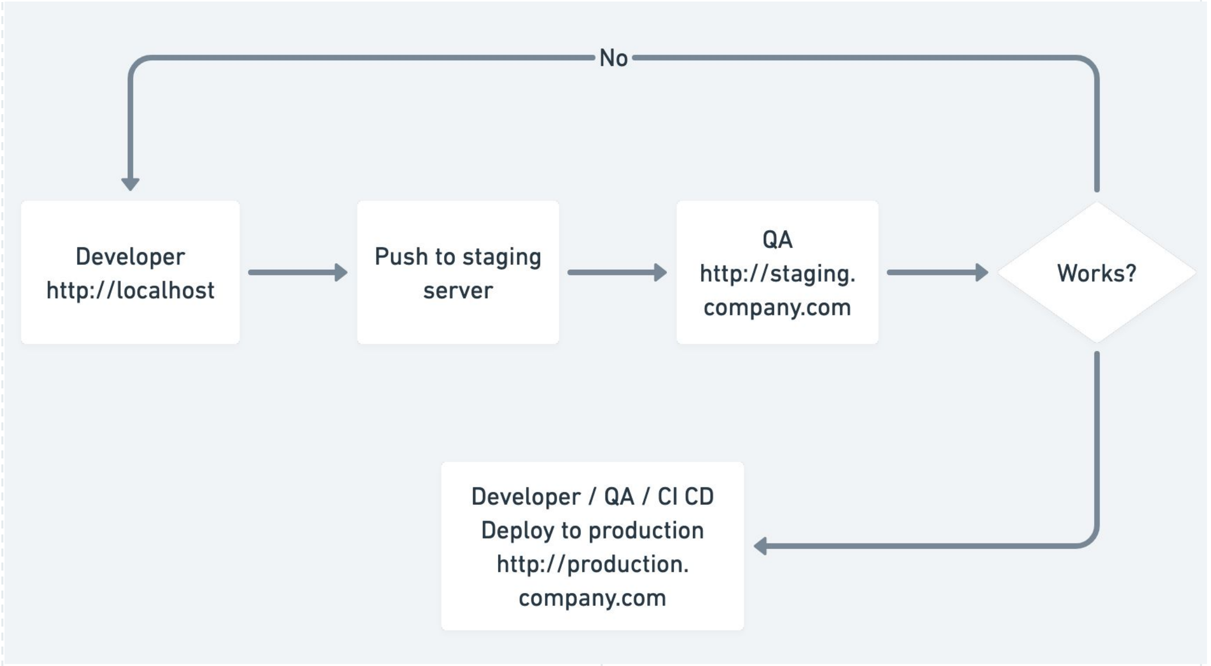
A typical workflow in a company

Exists in many companies.

Clear role distinction between developers and QA.

Two environments: staging vs production.

Deploy is manual: for both staging and for production


No
Developer http://localhost
Push to staging server
QA http://staging.company.com
Works?
Developer / QA / CI CD Deploy to production http://production.company.com

Issues

Developer

Is the person with the most context of the feature being shipped. And where bugs can occur. And all the edgecases that goes into releasing the feature.

However, in previously described setup …

The responsibility usually ends in **“It works in localhost”.**

Is not involved in further steps of release cycle.

Gets notified of bugs by QA.

Quality Assurance

Is usually the most stressed out person. Needs to know about each and every feature being released, by all the developers in the team.

QA role is a very repetitive role. Have to repeatedly test many functionalities day in day out.

QA might not be aware of edge cases.

Easy to be burnt out in this role.

Deployment process

Many manual steps.

Resource contention for staging server. What if multiple developers want to QA to test their features? Who deploys first?

Developer waits until QA finishes. QA then notifies developer of the bugs. And the cycle continues until QA stamps their approval. That’s a lot of back and forth.

Do you still need to test in production?

Staging != Production

Many subtle differences.

Some services might be running, others might not be. Network setup might be different, caching strategies might be different, etc.

Testing in staging is still not enough.

Testing in production

Developer is responsible for the complete cycle until feature is successfully released.

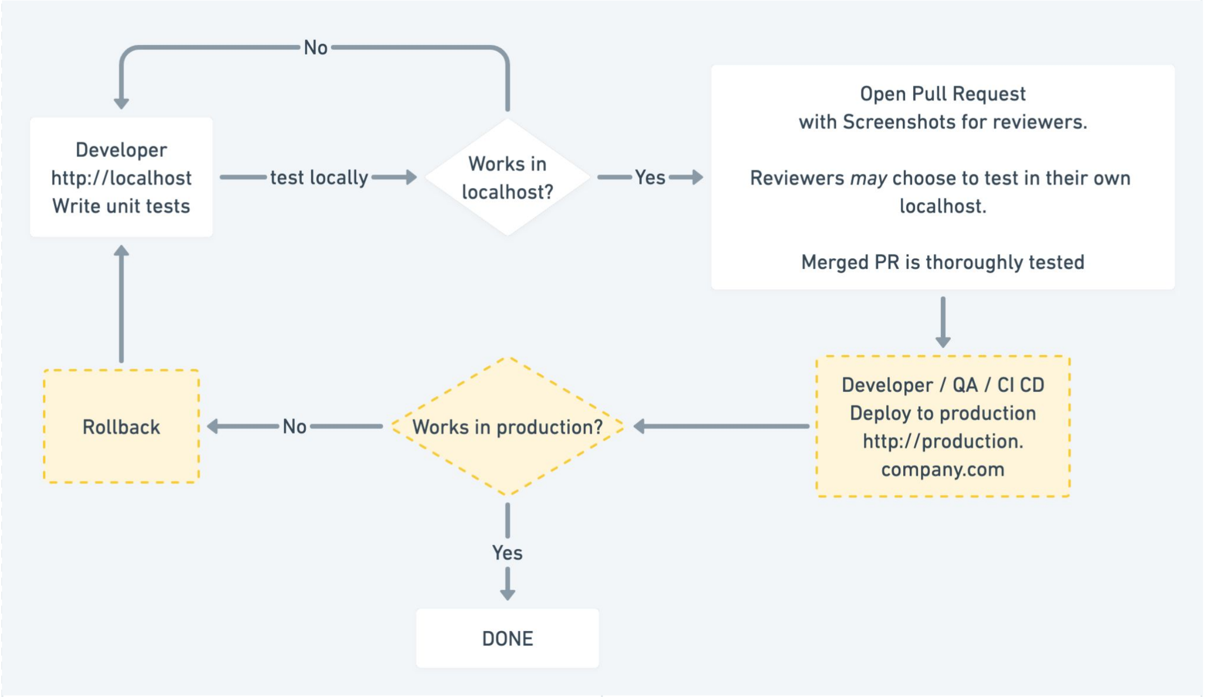
Developer carefully tests in localhost first. Takes screenshots and adds it to the pull request.

After code is merged with approval from reviewer(s), it is manually or automatically deployed to production.

Developer responsible for making sure deployment to production is successful.

Developer then tests all edge cases in production (more on this later).

Developer is involved end 2 end and the main individual responsible.



Testing in production

Mainly 2 technologies go hand in hand.

⛳ Feature flags

and

🧪 Test mode data

Feature flags

The idea is simple.
You define a “toggle switch” which is turned off by default.

You ship everything inside an `if else` condition and the new feature is unreleased by default.

You turn on the feature flag for your own user and check the feature is working correctly.

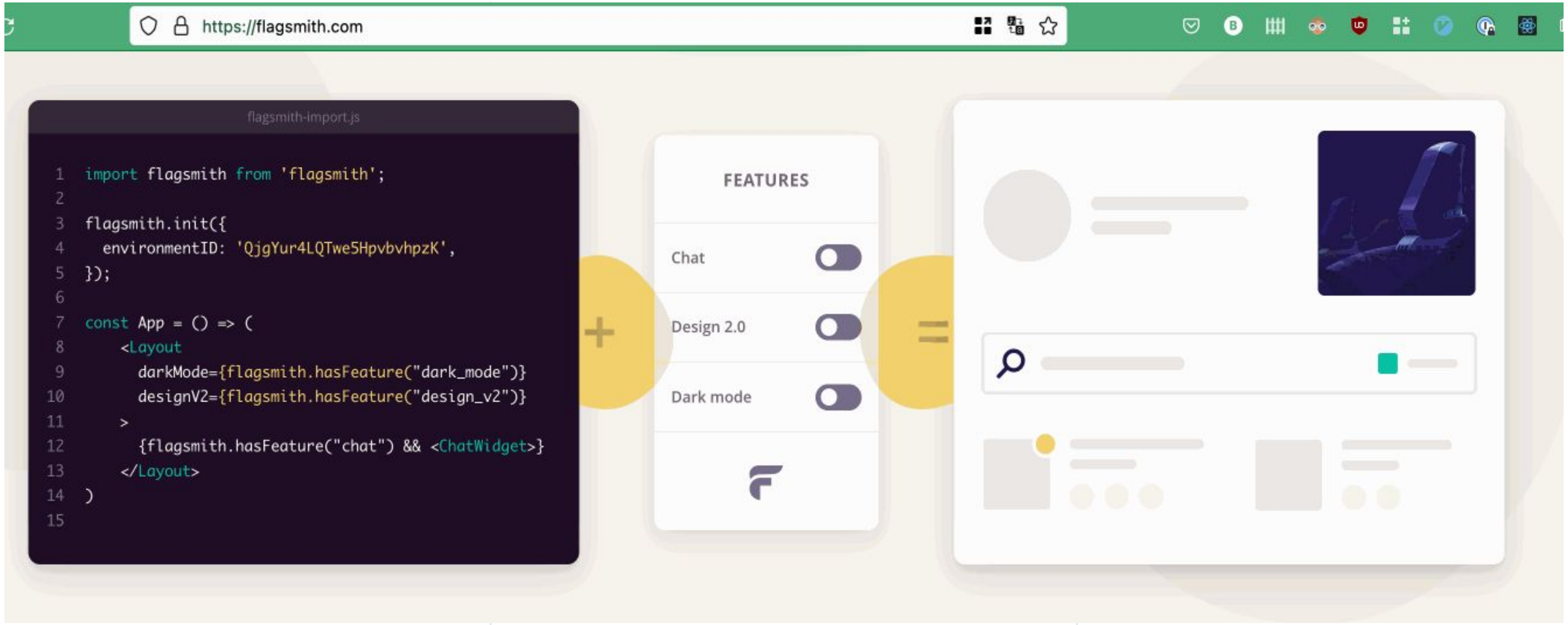
You then turn the feature flag to all users.

Test mode data

Users Table
Username
Password
Email
IsTest

Orders Table
Id
Customer
Total
Tax
IsTest

⛳ Feature flags



https://flagsmith.com

flagsmith-import.js

```
import flagsmith from 'flagsmith';

flagsmith.init({
  environmentID: 'QjgYur4LQTwe5HpvbvhpzK',
});

const App = () => (
    <Layout
      darkMode={flagsmith.hasFeature("dark_mode")}
      designV2={flagsmith.hasFeature("design_v2")}
    >
      {flagsmith.hasFeature("chat") && <ChatWidget>}
    </Layout>
)

```

⛳ Feature flags at Stripe

A new flag is created through admin.

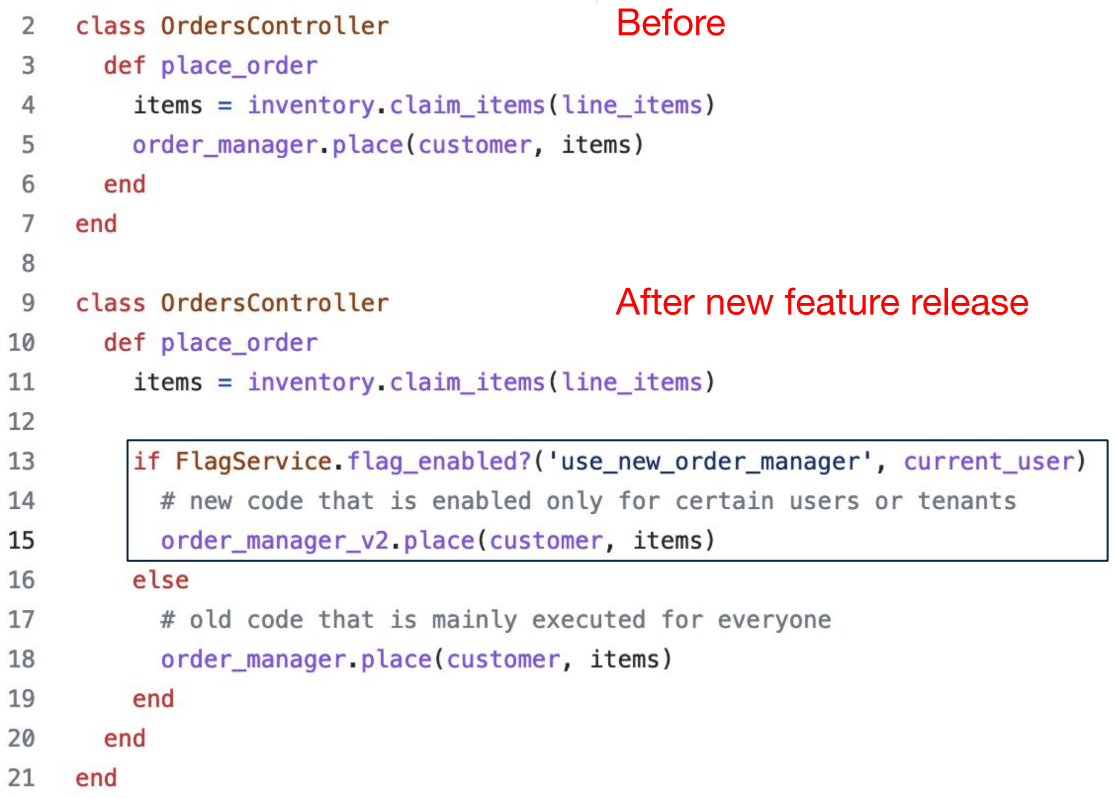
New code is shipped guarded within the flag.

All code paths will take the `else` codepath (old codepath).

Developer who is responsible for the feature, enables the flag for her user, and then makes sure the feature works. All edge cases are tested.

The feature flag is rolled out to all logged in users.

Finally, the `else` code path is deleted in separate PR safely.



Before

```
class OrdersController
  def place_order
    items = inventory.claim_items(line_items)
    order_manager.place(customer, items)
  end
end
```

After new feature release

```
class OrdersController
  def place_order
    items = inventory.claim_items(line_items)

    if FlagService.flag_enabled?('use_new_order_manager', current_user)
      # new code that is enabled only for certain users or tenants
      order_manager_v2.place(customer, items)
    else
      # old code that is mainly executed for everyone
      order_manager.place(customer, items)
    end
  end
end
```

Feature flags at Stripe

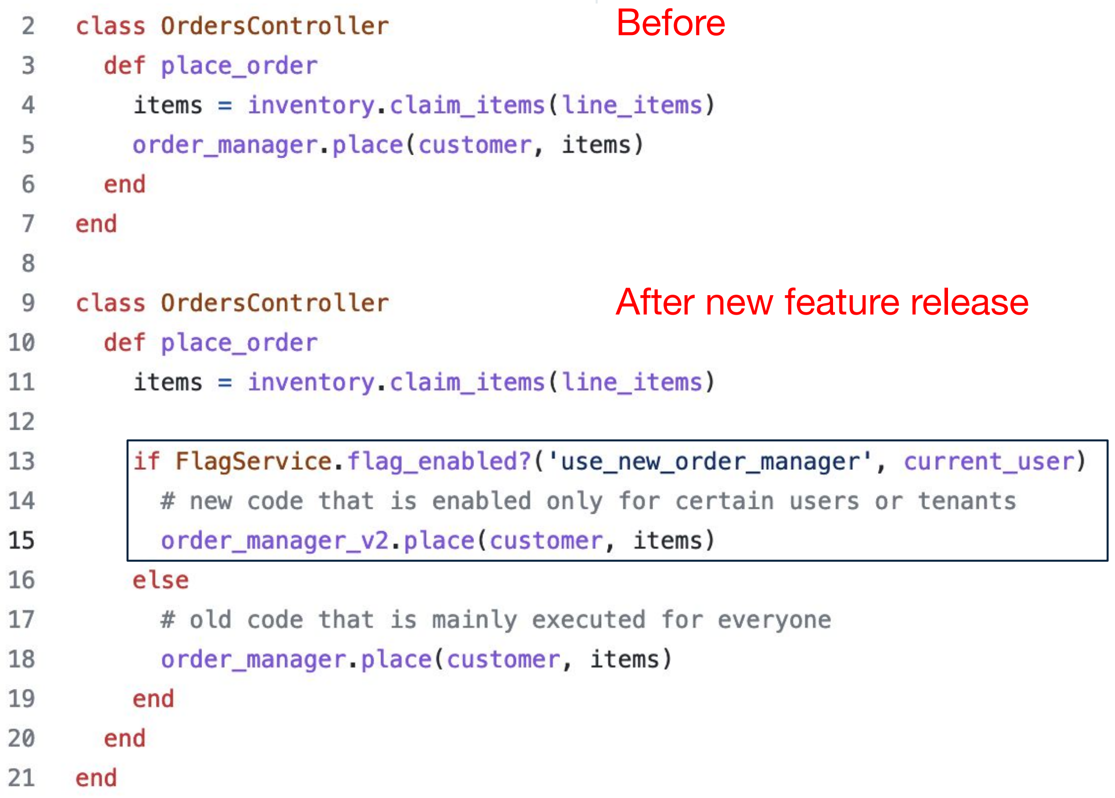
Stripe's feature flag service is in-house built.

Supports rollout based on User Id, or based on Client Ids.

Supports rollout based on percentage '%' so new features can be slowly shipped. Rollout can look like 1%, 5%, 20%, 100% as the developer looks for bugs or errors.

Feature flags can also be used as circuit breaker. If we want to immediately disable something, a feature flag can be toggled.

Heavily used across all teams.



Before

```ruby
class OrdersController
  def place_order
    items = inventory.claim_items(line_items)
    order_manager.place(customer, items)
  end
end
```

After new feature release

```ruby
class OrdersController
  def place_order
    items = inventory.claim_items(line_items)

    if FlagService.flag_enabled?('use_new_order_manager', current_user)
      # new code that is enabled only for certain users or tenants
      order_manager_v2.place(customer, items)
    else
      # old code that is mainly executed for everyone
      order_manager.place(customer, items)
    end
  end
end
```

⛳ Feature flags - open source servers

Some results from google search.

I have not used them personally but looking forward to integrate `**unleash**` in an upcoming project.

Important
Keep a close eye on errors during deployment and some time after deployment. *Developer also wears devops hat.*

Be ready to rollback PRs asap.

unleash

Feature toggle management solution with 12 official and 10+ community SDKs
Trusted by thousands of companies all over the world

Flagsmith

FeatureHub

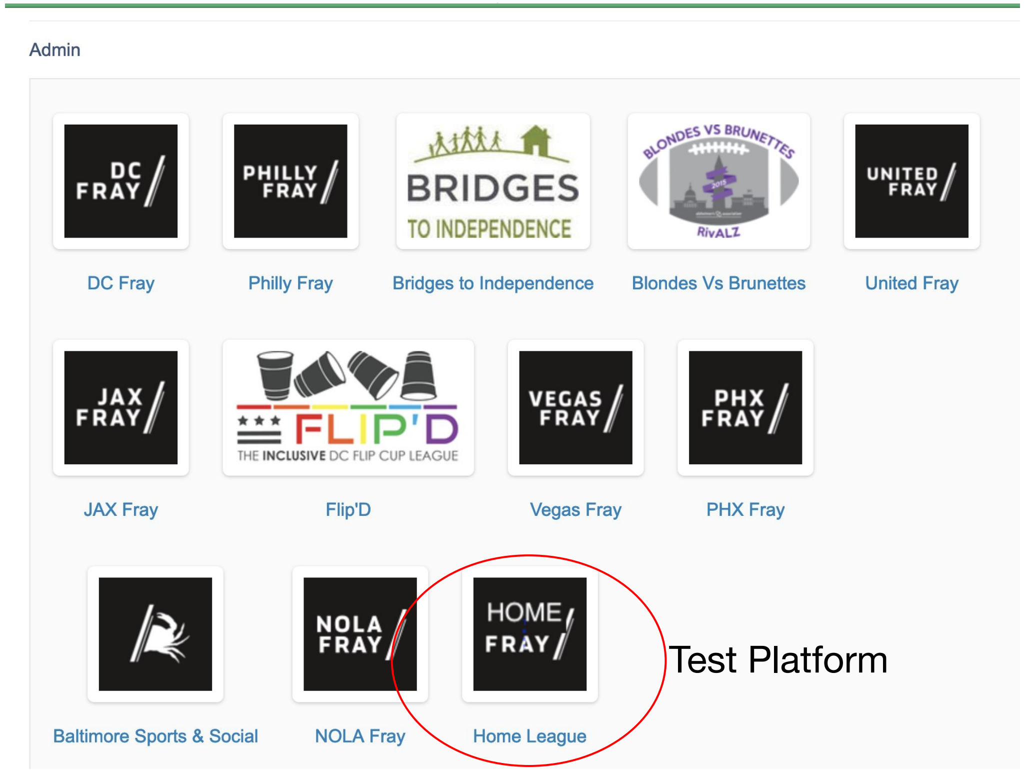
⛳ Testmode data in production

Similar to staging server, you need some way to enter data in the *production database* but mark it as test data. eg:

In a multi-tenant system, you can create a test-tenant that is used purely for testing purpose. (**Best solution**)

You can also create special `test` users. user.is_test=true. All records created by a `test` user also contains `is_test` flag set to true.

You can filter out `is_test` records from any kind of reports, exports, numbers, etc.



Admin

DC Fray

Philly Fray

Bridges to Independence

Blondes Vs Brunettes

United Fray

JAX Fray

Flip'D

Vegas Fray

PHX Fray

Baltimore Sports & Social

NOLA Fray

Home League

Test Platform

Challenges

Requires a mindset shift.

Requires initial investment in integration of feature flag server and test_mode data awareness.

Requires tools to monitor logs, errors and alerting and paying close attention to system's health.

Requires having good test coverage to prevent regression so one doesn't have to test everything.

BUT … it is worth it.

These are all best practices for development.

It enables developer to ship faster. Additional "cruft" filled processes and dependencies are removed.

It allows developer to own her work end-2-end (features, bug fixes, etc) and therefore builds accountability.

And overall increases team efficiency.

How do you ship features in your company?

Anyone wants to share their experience?

Any feedback for the presentation?

Any questions on the ideas presented?

Ask away … or email me at bibek@hey.com