ingenico ePayments & accenture High performance. Delivered.

The road to enable DevOps beyond Facebook, Spotify, Netflix … within the Payment Industry

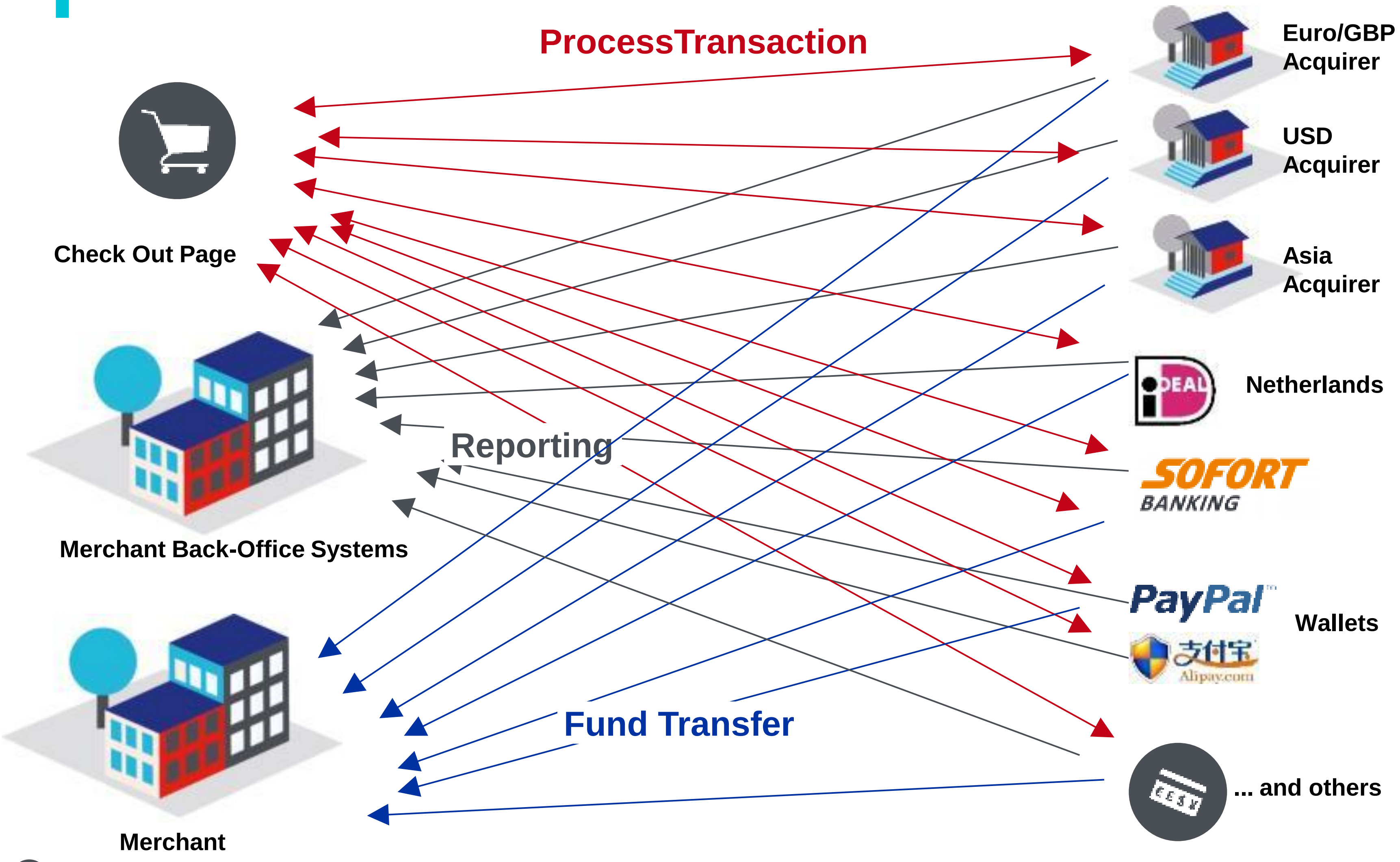
DEVOPS ENTERPRISE SUMMIT

01 July 2016

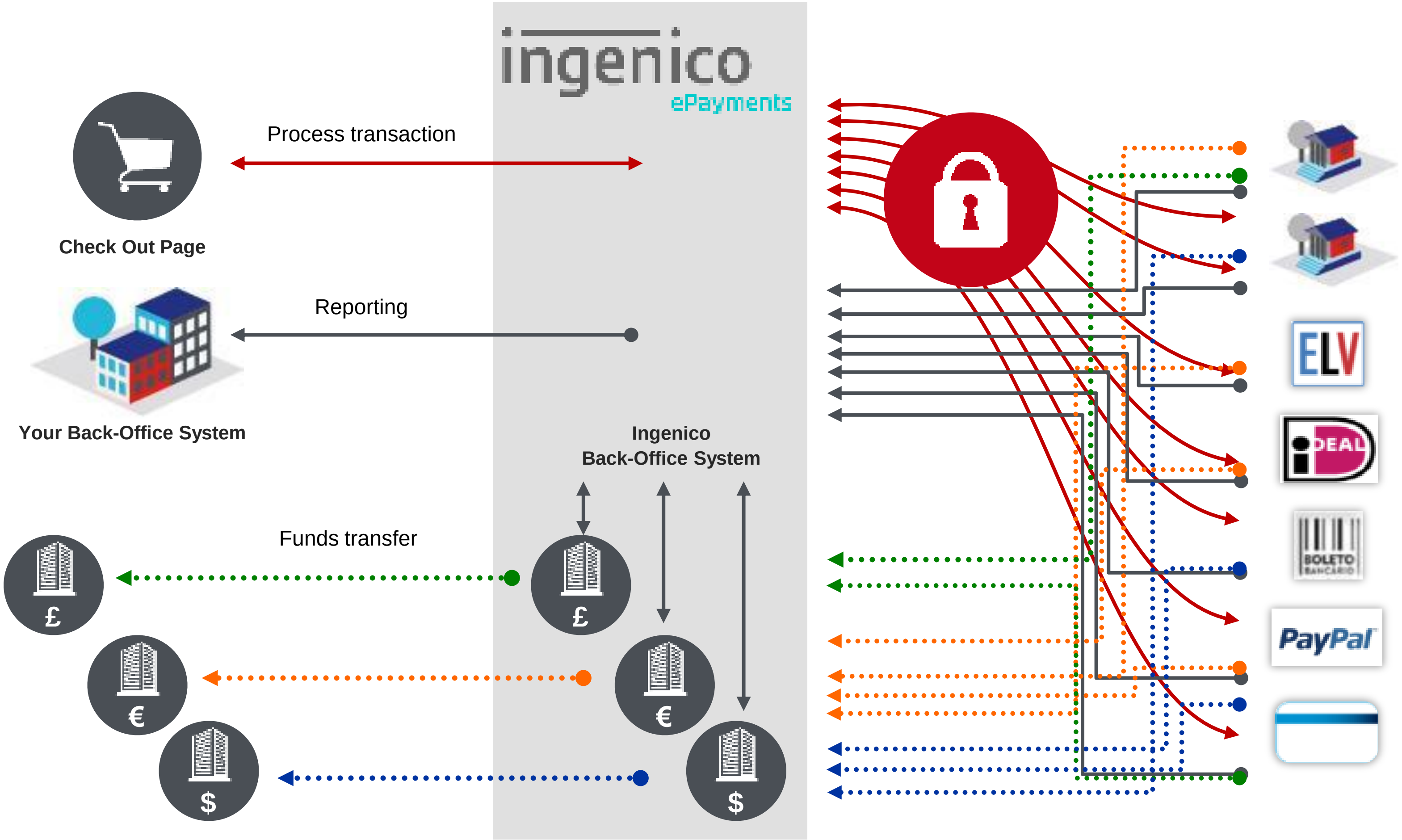
Vincent van Kooten, Ingenico ePayments
Gebrian uit den Bulten, Accenture

What is a “Payment Service Provider” ?

We turn your ceck-out complexity …



… into something easy!



You've done business with us …

€58 Bn
Processed flow

65,000
Clients

230
Acquiring partners and banks

150+

150
International, local and alternative payment methods

170
Countries

Shift away from the monolith

Ingenico ePayments started in ***1994*** *as Payment Service Provider (PSP).*
Being a front-runner back then means dealing with older technologies today …
… so it was time for a ***'big'*** *change.*

Technical goals:

- Refactoring from monolith into micro services architecture
- Maintain 99.999% uptime
- Maintain full automatic regression test (>25k test cases)
- Easily handle over 1000s transactions per second

Delivery goals:

- Move to a Agile / DevOps way of working
- Introduction of the Continuous Delivery pipeline
- Cloud enabled with technologies like containers
- Maintain highest level of PCI compliancy

Image source: http://www.pinterest.com/pin/201958982988814935/

Learn from the best and improve on 3 themes:

organizational & cultural change

architecture enablement

continuous delivery

Inspired by: Spotify NETFLIX facebook

organizational & cultural change

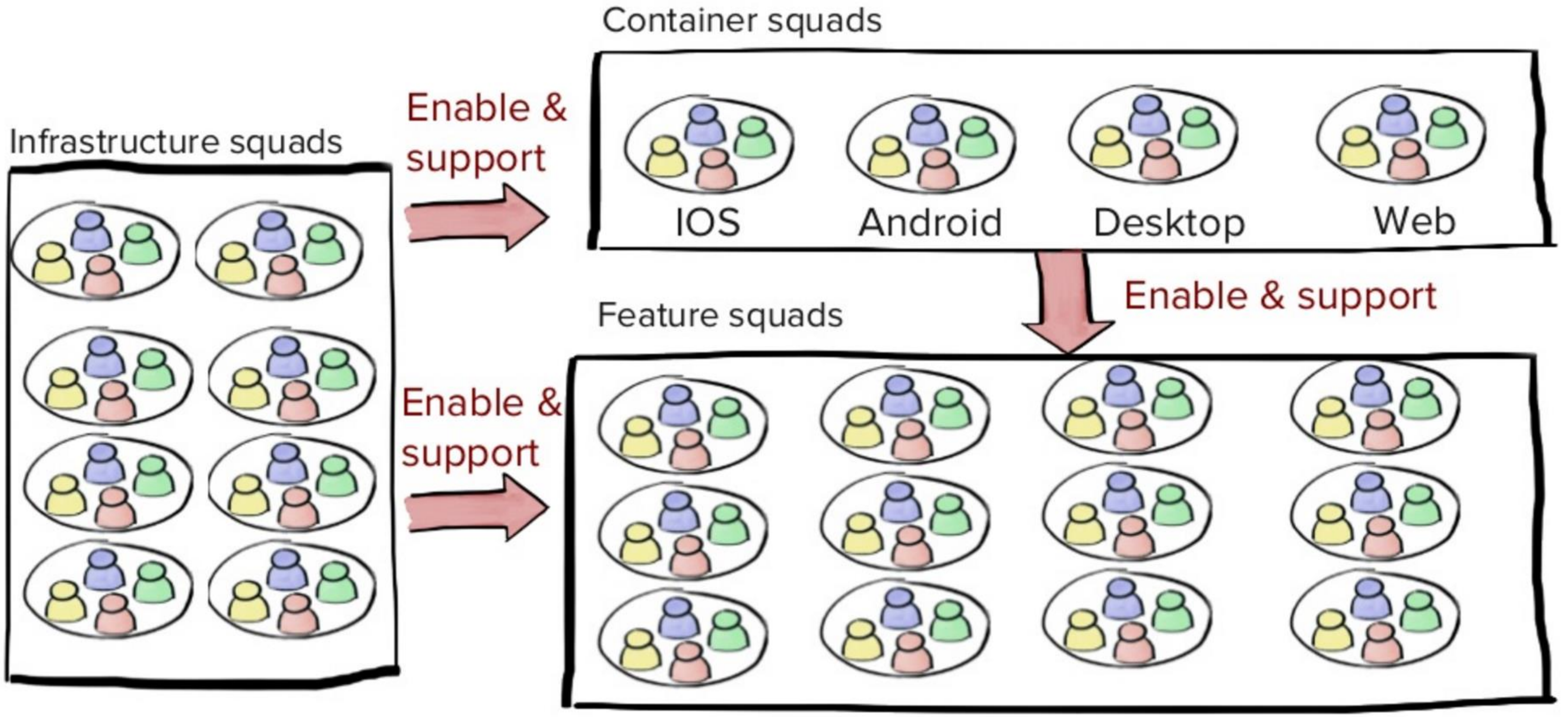
Spotify

Development is a team effort!

Goal: Self-service model

organizational & cultural change

Spotify



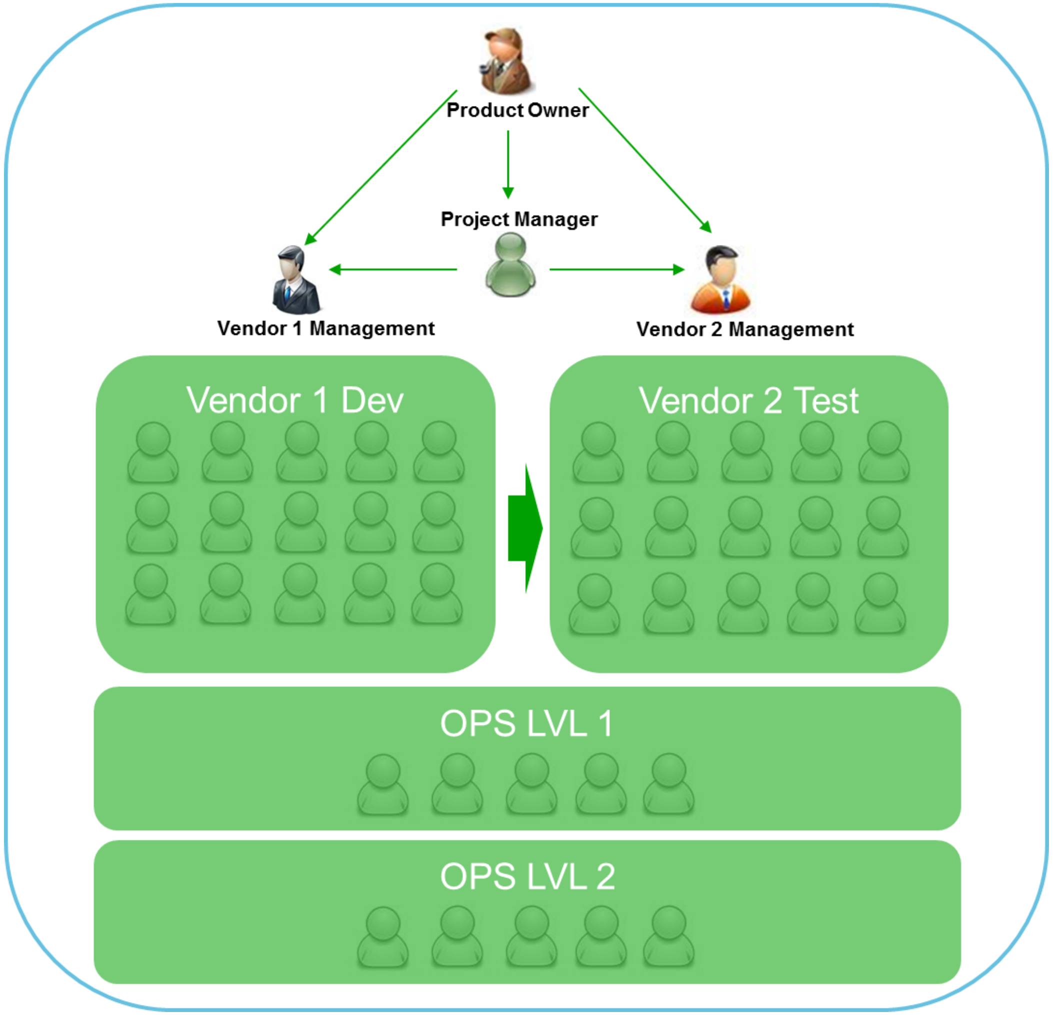
The feature team should feels like a mini startup and should be self organizing and end to end responsible.

Image source: https://labs.spotify.com/2014/03/27/spotify-engineering-culture-part-1/

Previous Organization Setup

organizational & cultural change

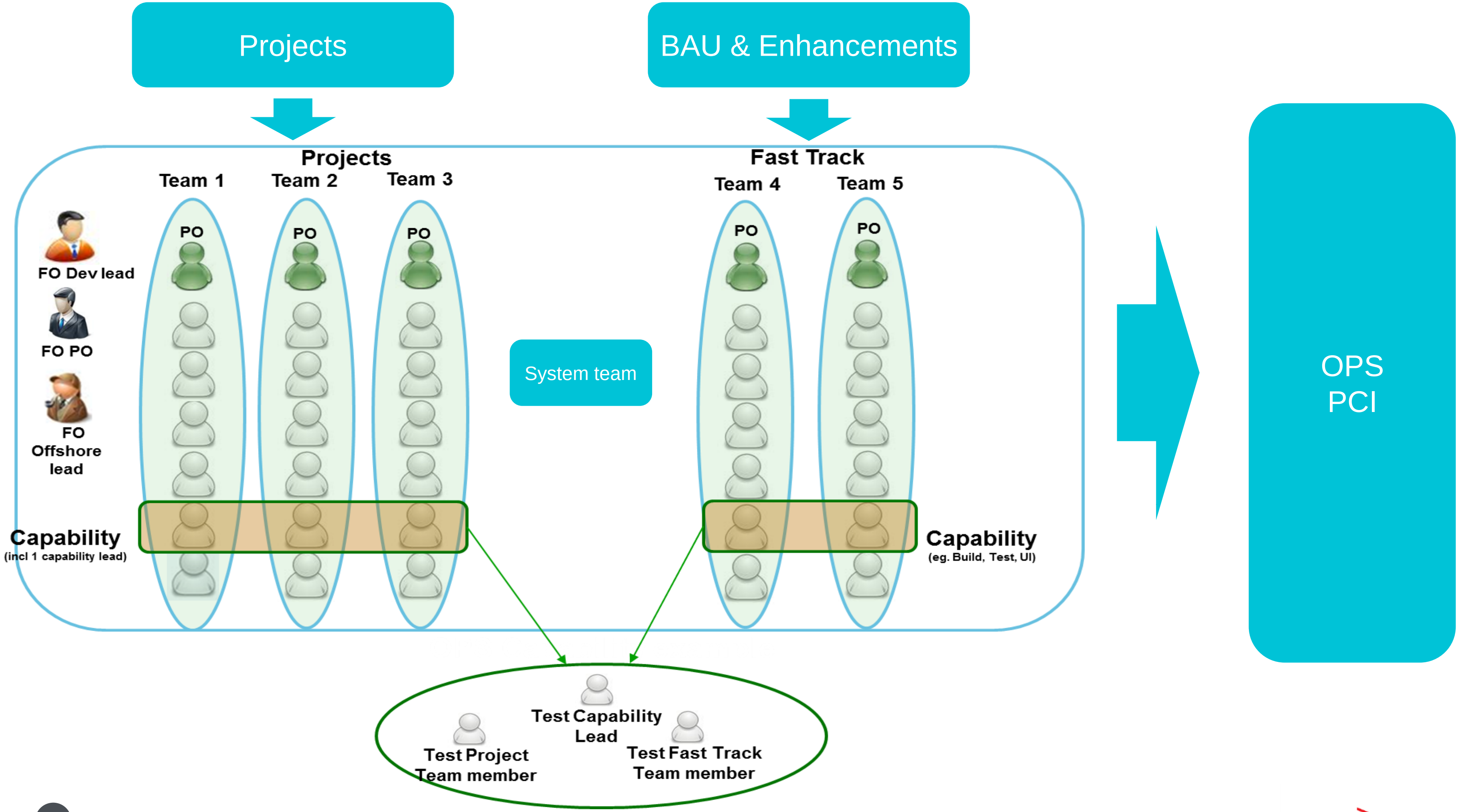
Spotify



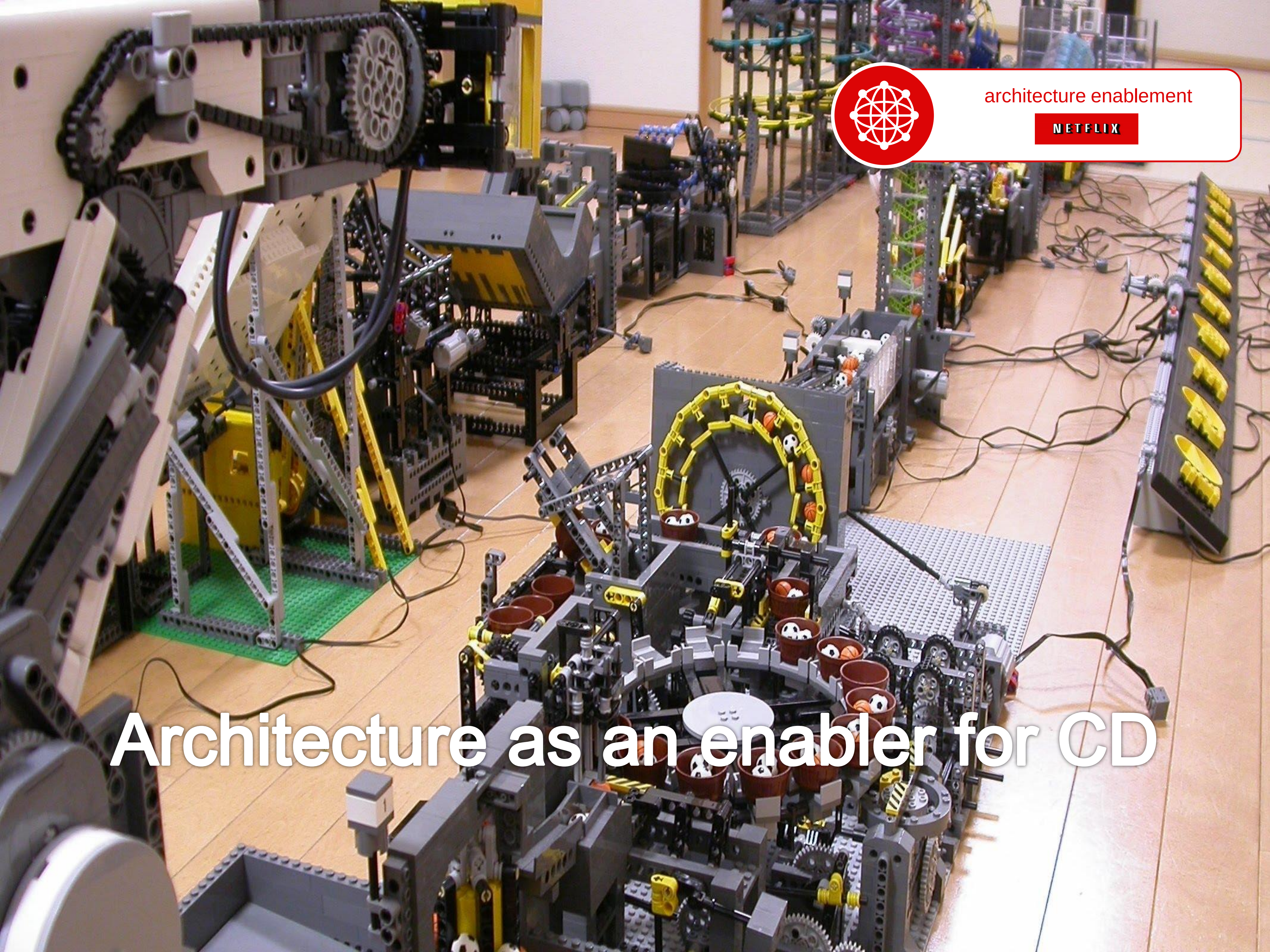
Current Organization Setup

organizational & cultural change

Spotify



Architecture as an enabler for CD

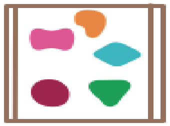


Goal: move to micro-services

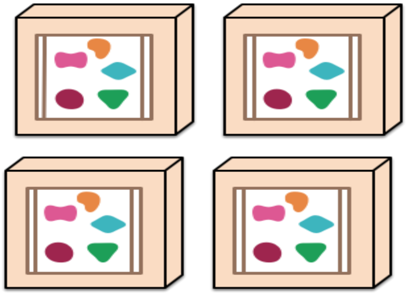
architecture enablement

NETFLIX

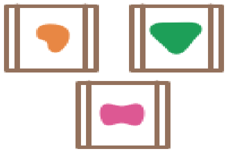
A monolithic application puts all its functionalities into a single process



and scales by duplicating the monoliths



A micro-service architecture puts each element on functionality into a separate service



and scales by distributing these services across servers

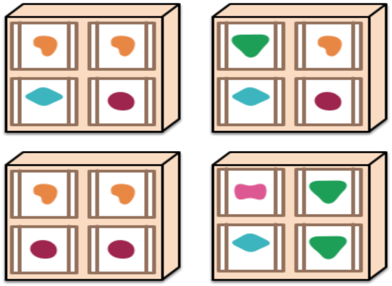


Image source: http://martinfowler.com/articles/microservices.html

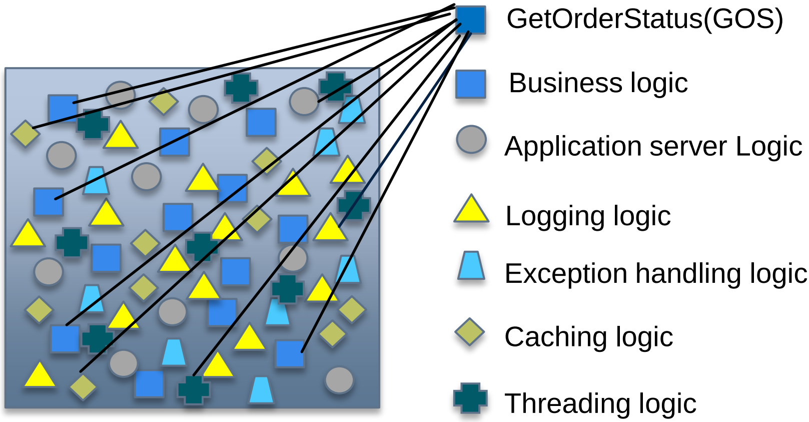
ingenico ePayments

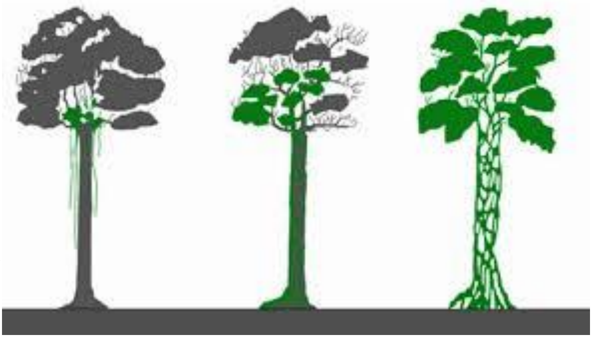
accenture High performance. Delivered.

Setup the basic core architecture and validate this on production

architecture enablement

Step 1:



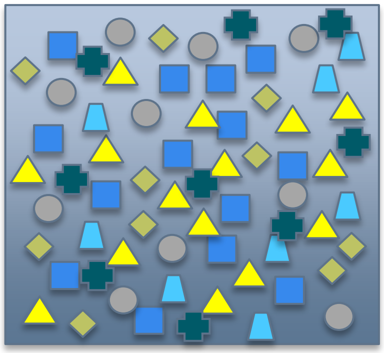


Strangler Pattern

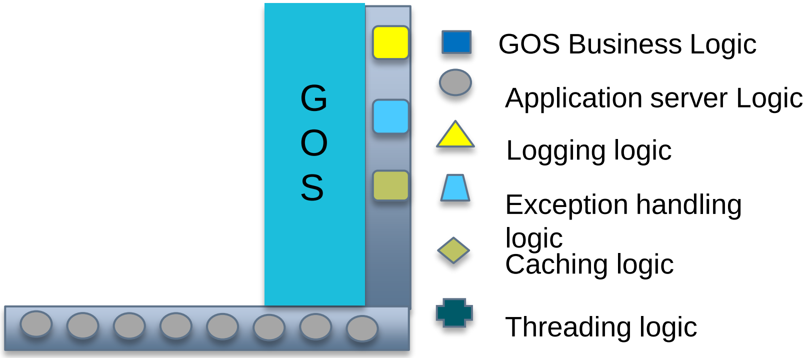
Image source: http://www.martinfowler.com/bliki/StranglerApplication.html

Step 2:

Legacy



Refactored

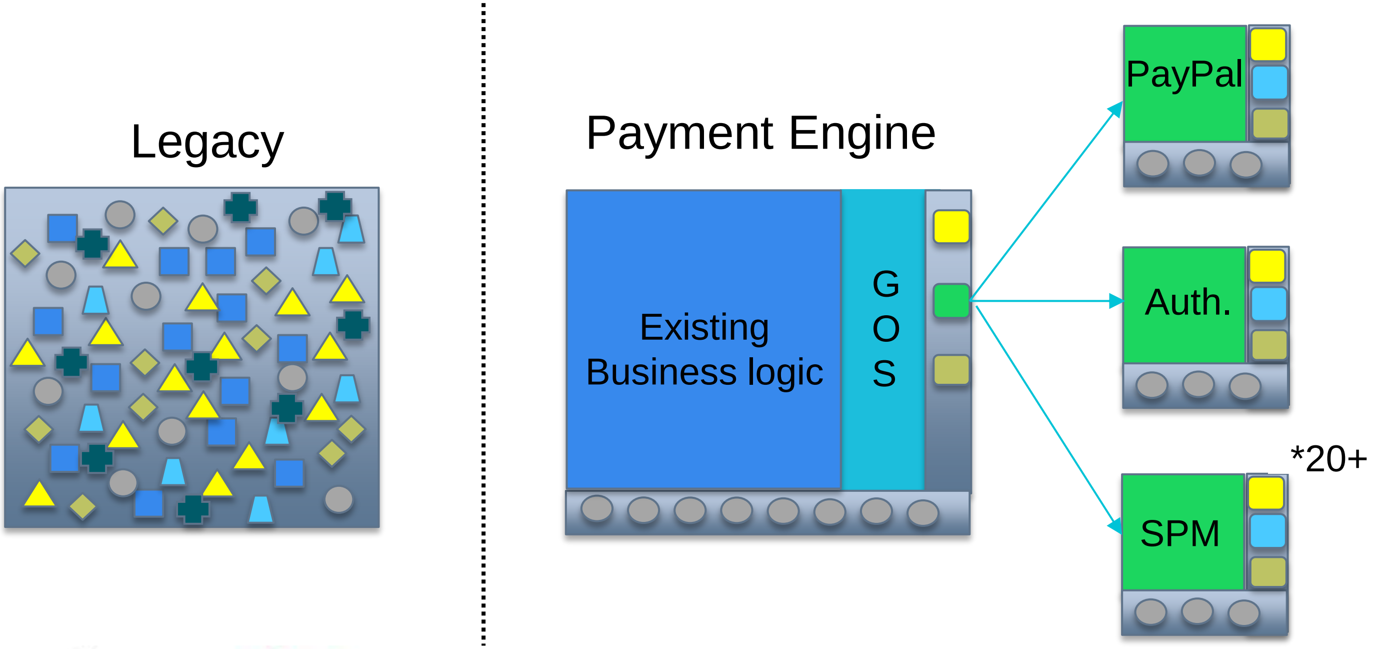


Move existing logic to the new platform by using the strangler pattern

architecture enablement

NETFLIX

Step 3:



Legacy

Payment Engine

Existing Business logic

G O S

PayPal

Auth.

SPM

*20+

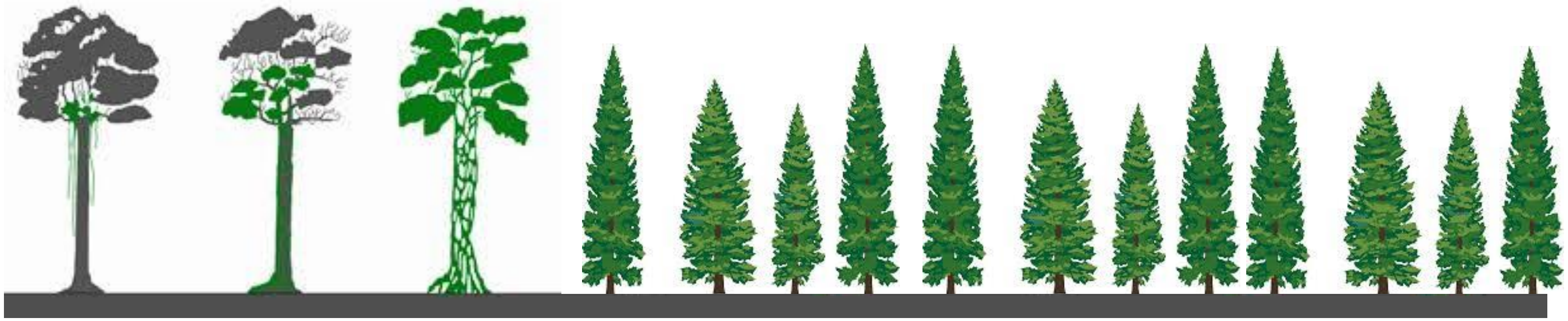
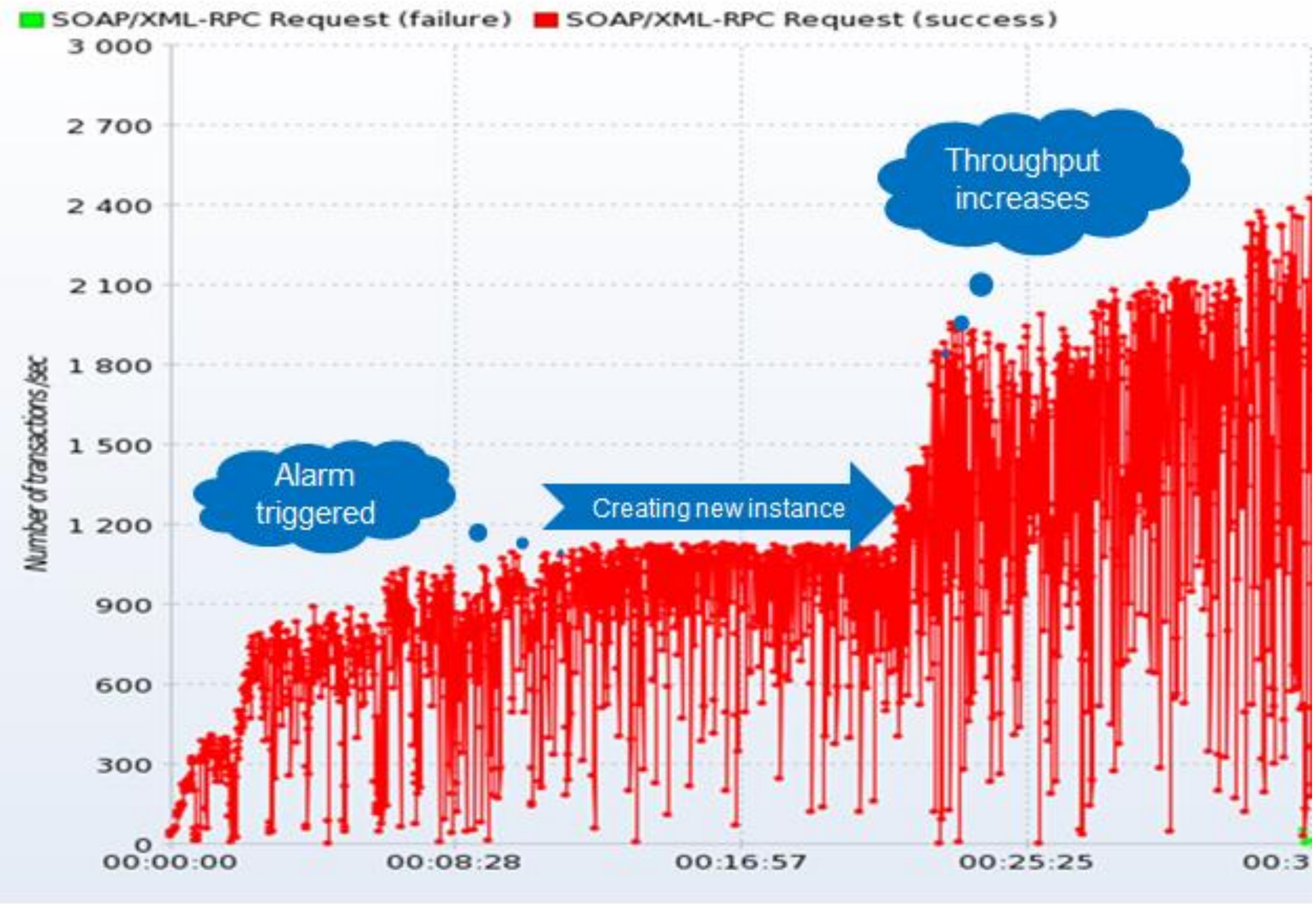


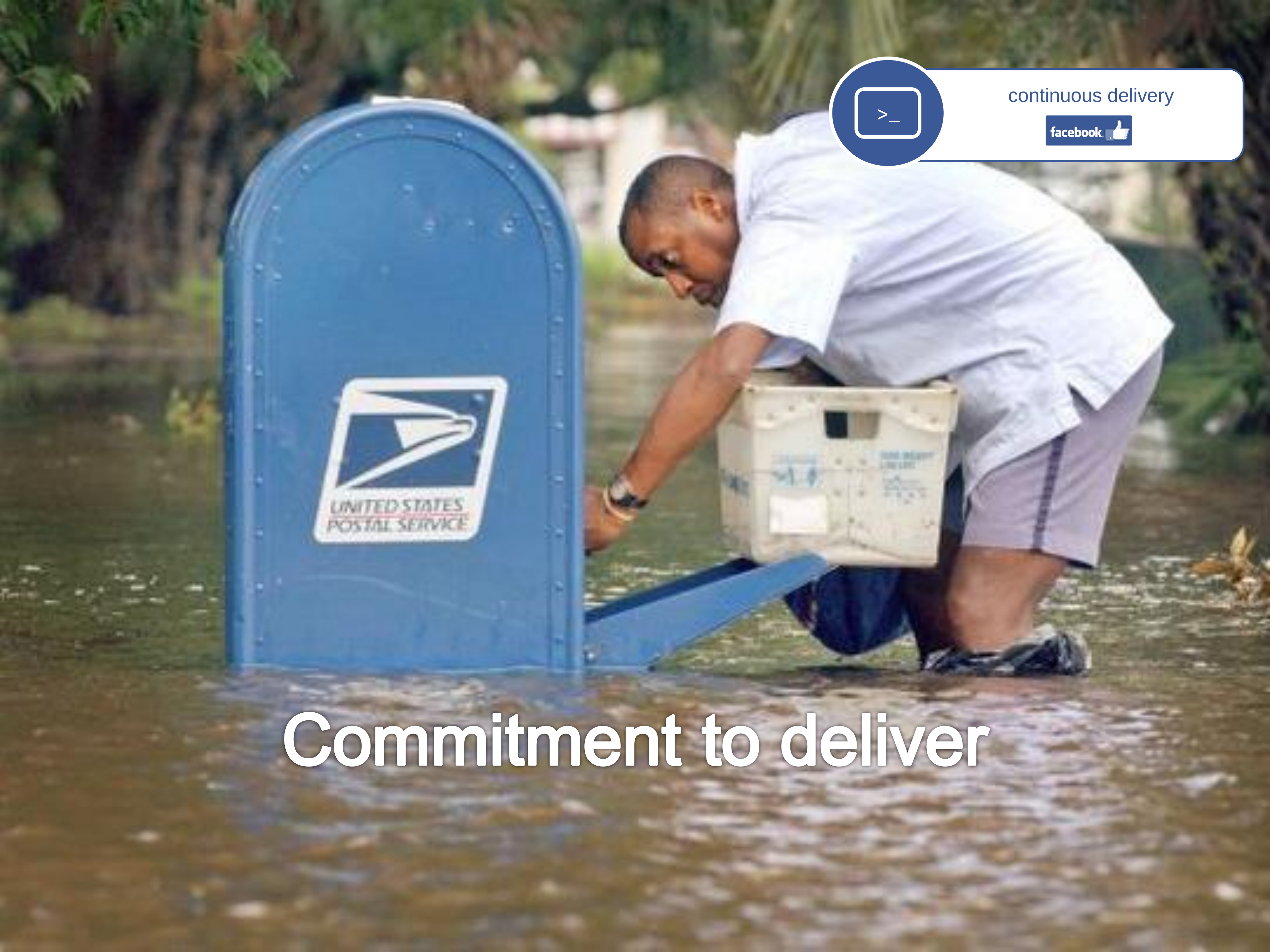
Image source: http://cliparts.co/redwood-tree-clip-art

AWS & Docker to enable growth

architecture enablement

NETFLIX



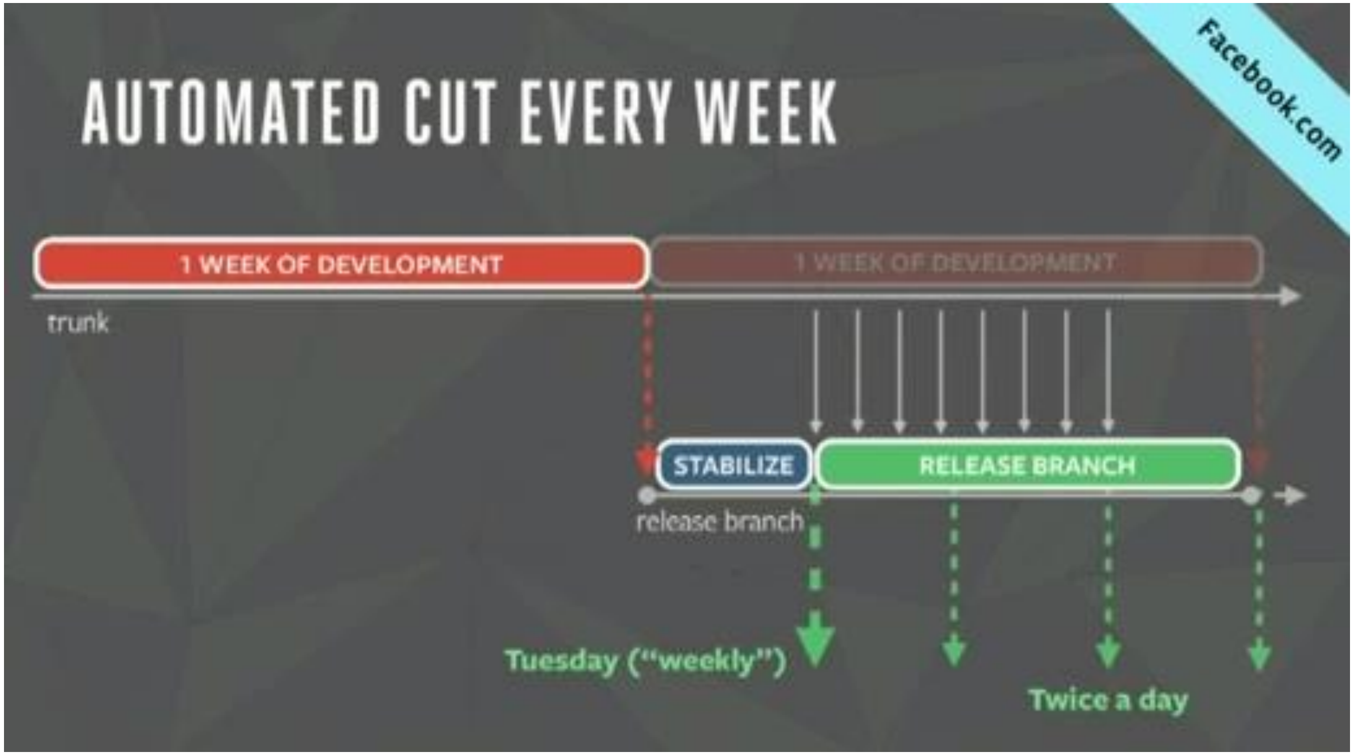


Commitment to deliver

Goal: Being able to release to production every day

continuous delivery

facebook



https://www.infoq.com/presentations/Facebook-Release-Process

Image source http://www.agile-minds.com/facebook-mobile-release-process/

Previously releasing from multiple teams to production

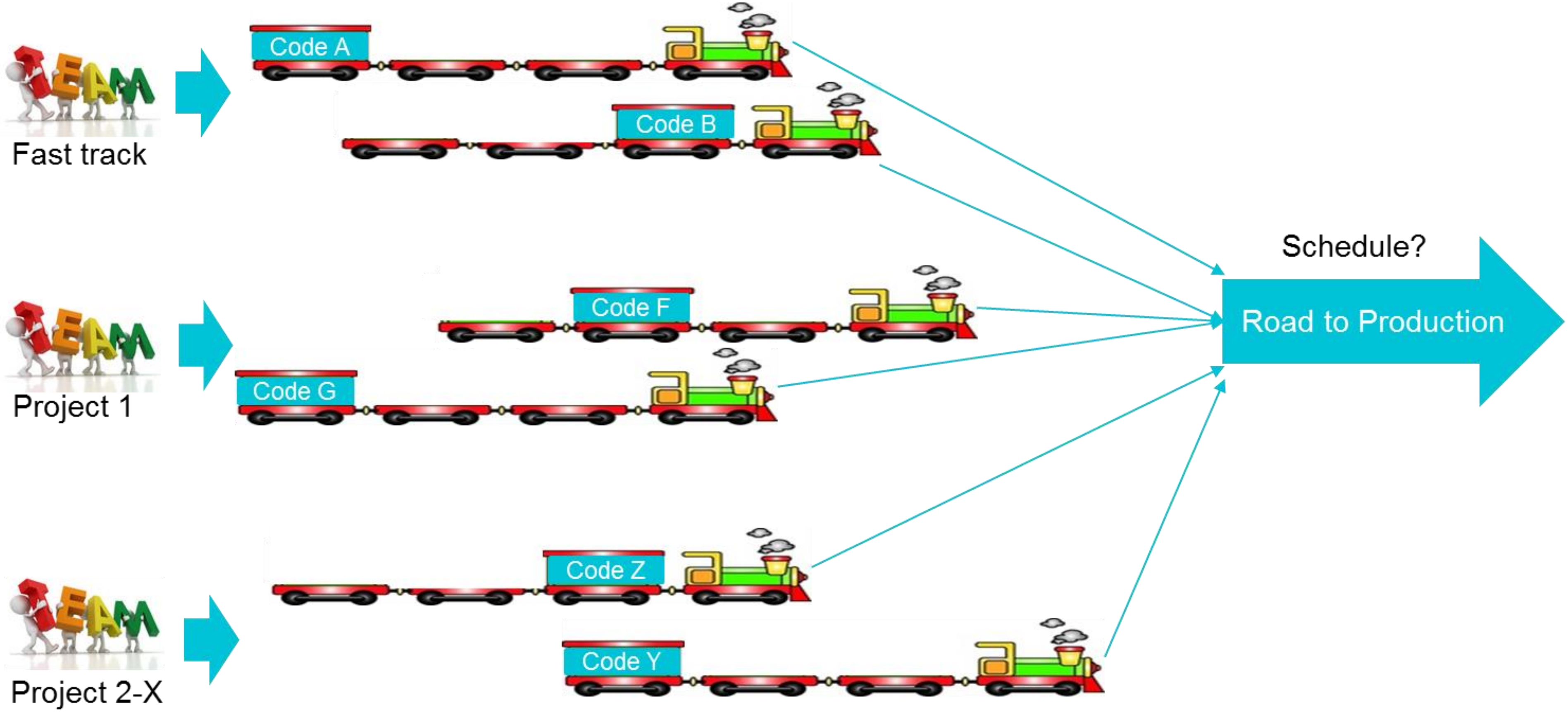
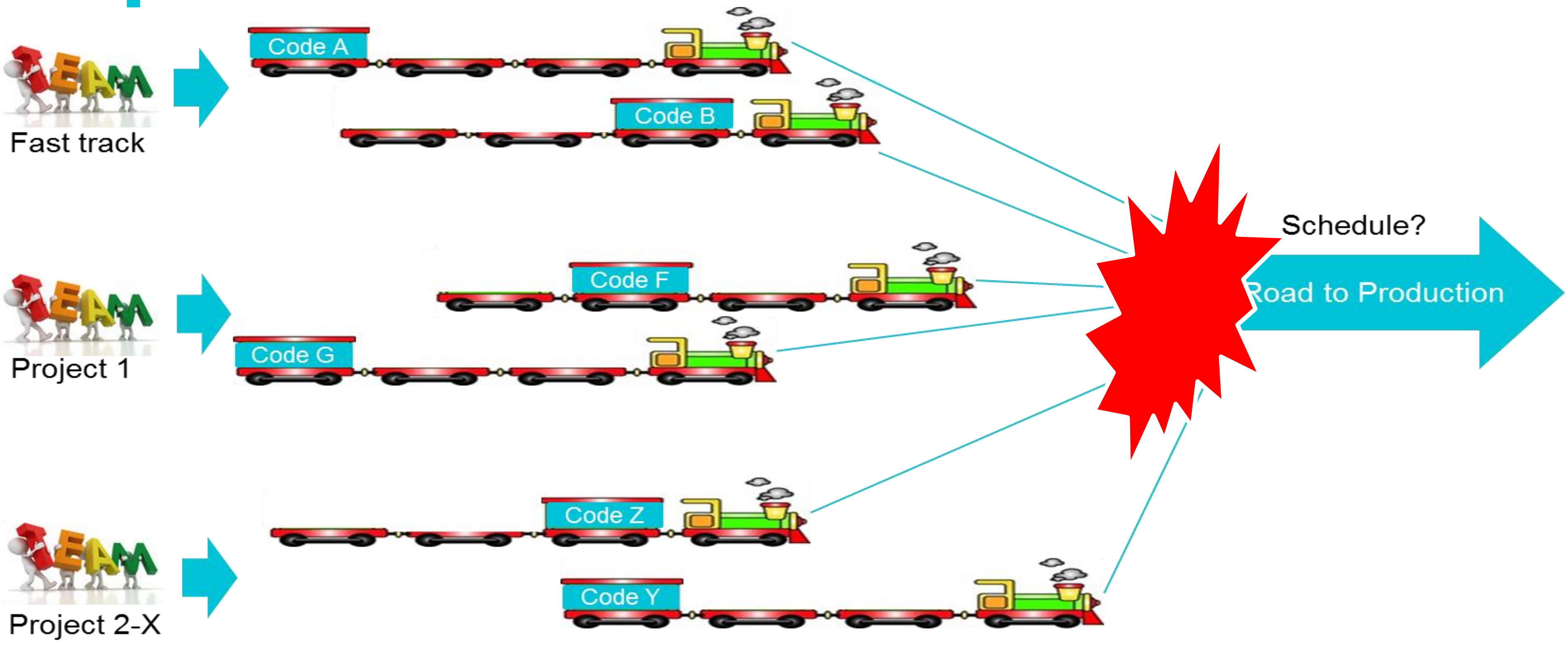


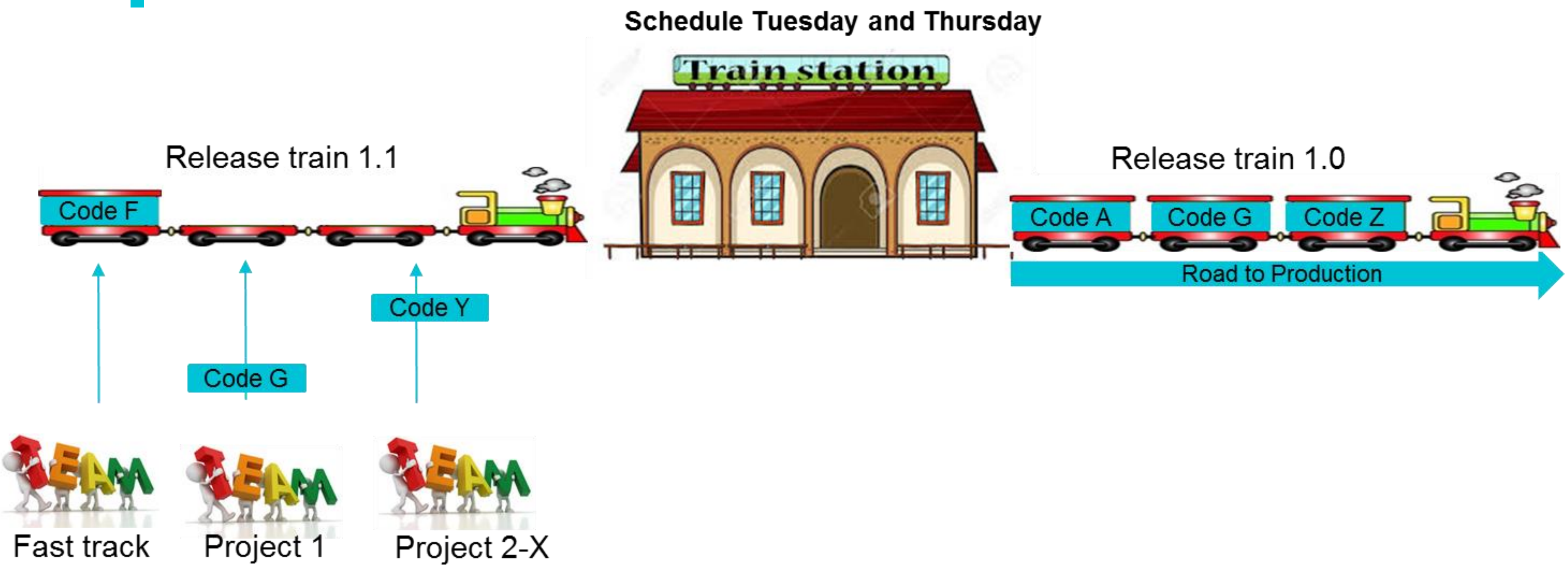
Image source train: http://www.cartoon-clipart.com/cartoon_clipart_images/colorful_cartoon_choochoo_train_locomotive_steam_engine_toy_0515-1005-2200-3354.html
Image source team: http://mediapoliseuropa.com/index.php/team

Previously releasing from multiple teams to production



- Releasing to production <10 times a year
- Takes months from code commit to move to production
- High defect ratio in deliverables
- Deployment of the platform to one environment could take day's

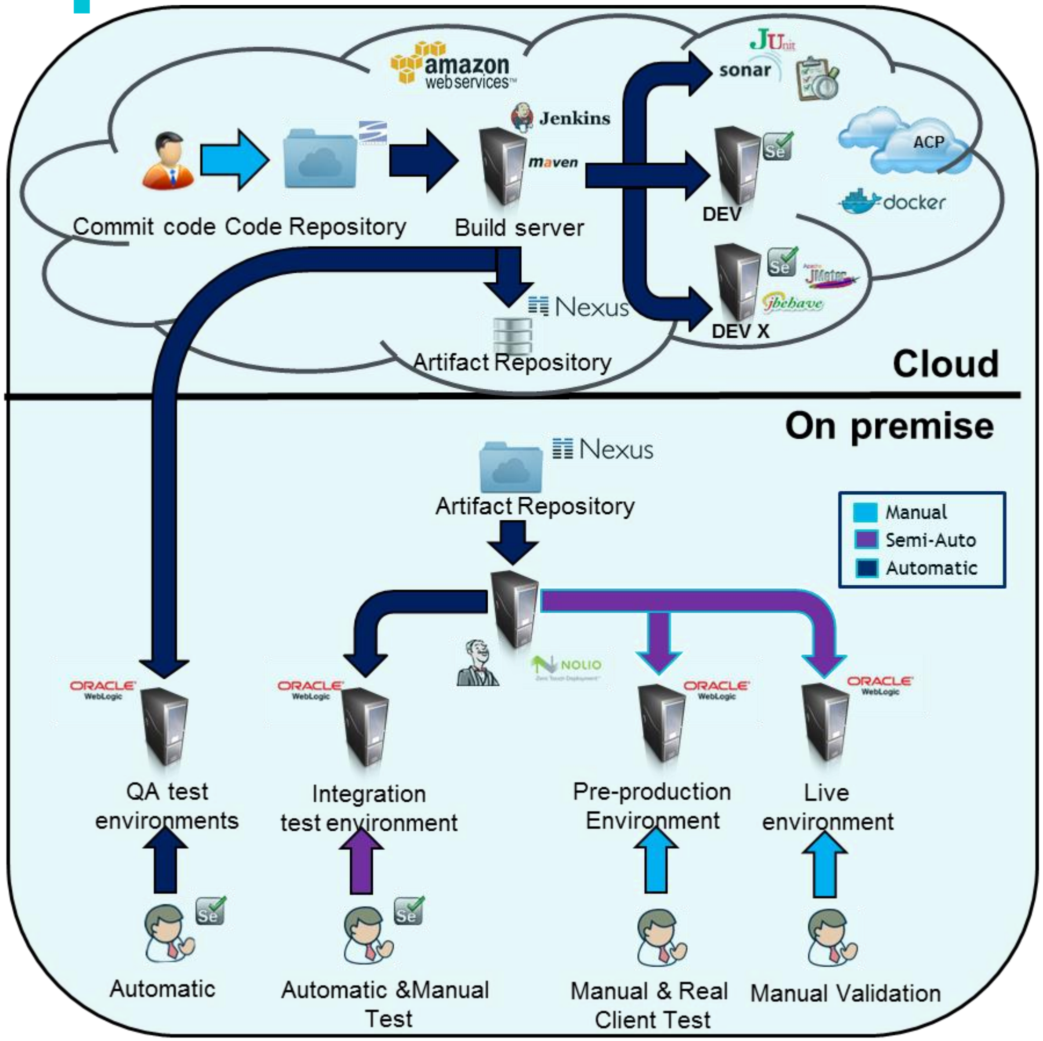
Releasing to Production



- Releasing to production every week
- On average a release would take around a week from code commit to production (this can be speed up to 1-2 days if required)
- Significant lower defect count in releases
- Deployment to one environment of the platform takes less then an hour

Image source internet http://www.nairaland.com/attachments/2833327_internet3_jpeg9cec76723a59485ff8cc19f80be8ccf3
Image source: http://1.static.upcoming.nl/static/images/9b30b08b95_1452776196_Around-the__quiz-qlst.jpg

Current environment setup



Environment	Time	Tests
Build Server	On Commit	• Unit test • Integration unit test
Development	On Commit & Daily	• Automatic Regression test • Manual test
Test	On Demand & Daily	• Manual test • Performance test
Trunk	On Release & Daily	• Automatic Regression test • Performance test • Manual smoke test
Integration Test	On Demand & Daily	• Automatic Regression test • User acceptance test
Pre-production	On Release	• User acceptance test • Real client test
Production	On Release	• Validation test

Some lessons learned

- Work closely together as one team instead of creating a vendor vs client relationship
- No separation of teams based on onshore vs offshore. Mixed onshore & offshore is the key
- Create mixed feature teams where business, dev. and ops are present
- Additional management required for offshore (Team lead/ Scrum master Onshore and Offshore).
- Set clear guidelines on development, testing, automation, quality and enforce this through automatic tooling.
- Mainline branching strategy is the key to enable continuous delivery
  don't use feature branches!!
- Don't be afraid to choose a different direction if this fits better for your setup (e.g. introduce additional management on top of the scrum teams, or lowering delivery speed to let the rest of the organization catch up)

Image source http://covermaker.net/facebook-cover/2880/No+Regrets,+Just+Lessons+Learned.html

Next steps

- Focus on creating feature teams
- Increase development speed
- Investigate cloud possibilities
- Improve metrics
- Continuous improvement!

CONWAY'S LAW

"Any organization that designs a system will inevitably produce a design whose structure is a copy of the organization's communication structure."
Melvin E. Conway, 1967

What can we still learn:

- PCI compliancy in the cloud
- Docker production experience on high profile, high load platforms, PCI compliant environments

http://www.slideshare.net/RachelDavies/tech-talk-implicationsconwayslaw

Don’t be afraid to change!

DEVOPS
ENTERPRISE
SUMMIT

Contact information:
Vincent van Kooten – vincent.vankooten@ingenico.com
Gebrian uit de Bulten – g.uit.de.bulten@accenture.com

Image source: https://www.theguardian.com/sport/picture/2016/may/26/sport-picture-of-the-day-sparks-fly-in-monaco-for-max-verstappen#img-1