PROGRESSIVE DELIVERY
CONTINUOUS DELIVERY THE RIGHT WAY
Carlos Sanchez / csanchez.org / @csanchez
PROGRESSIVE DELIVERY
Progressive Delivery, a History.... Condensed

By Adam Zimman - August 6, 2018

4326
JAMES GOVERNOR'S MONKCHIPS

Towards Progressive Delivery

By James Governor | @monkchips | August 6, 2018
Progressive Delivery is a term that includes deployment strategies that try to avoid the pitfalls of all-or-nothing deployment strategies.
New versions being deployed do not replace existing versions but run in parallel for an amount of time receiving live production traffic, and are evaluated in terms of correctness and performance before the rollout is considered successful.
Continuous Delivery is hard
Progressive Delivery makes Continuous Delivery easier to adopt
reduces the risk associated with Continuous Delivery
• Avoiding downtime
• Limit the blast radius
• Shorter time from idea to production
PROGRESSIVE DELIVERY TECHNIQUES
CANARY DEPLOYMENT
we put **toggle points** in our code to switch behavior for the new feature

several toggle points for the same feature use a single **toggle router** to determine their state

the **toggle router** may need to consider **toggle context** (e.g. which user is making the request) in order to make a routing decision

**a toggle router** is controlled via the **toggle configuration** for this environment.

Martin Fowler [martinfowler.com/articles/feature-toggles.html](http://martinfowler.com/articles/feature-toggles.html)
MONITORING IS THE NEW TESTING

Know when users are experiencing issues in production

React to the issues automatically
Progressive Delivery requires a good amount of metrics
To make error is human. To propagate error to all server in automatic way is #devops.
If you haven't automatically destroyed something by mistake, you are not automating enough
JENKINS X
KUBERNETES

KUBERNETES EVERYWHERE
Pipeline engine in Kubernetes

Uses Pods and containers to run the pipeline steps
Implements ChatOps
Handles GitHub webhooks
Package manager for Kubernetes
Build Docker images with multiple backends:

- Docker build
- Kaniko
- Google Cloud Build
- Jib (Maven/Gradle)
Generates Dockerfile and Helm charts for your project
PROGRESSIVE DELIVERY
WITH JENKINS X
jenkins-x.io/docs/managing-jx/tutorials/progressive-delivery
Istio

Connect, secure, control, and observe services.

Connect
Intelligently control the flow of traffic and API calls between services, conduct a range of tests, and upgrade gradually with red/black deployments.

Secure
Automatically secure your services through managed authentication, authorization, and encryption of communication between services.

Control
Apply policies and ensure that they're enforced, and that resources are fairly distributed among consumers.

Observe
See what's happening with rich automatic tracing, monitoring, and logging of all your services.
PROMETHEUS

A systems monitoring and alerting toolkit
FLAGGER

flagger.app

automates the promotion of canary deployments by using Istio’s traffic shifting and Prometheus metrics to analyse the application’s behaviour during a controlled rollout
Add the `canary` section to our application Helm chart values.yaml

```yaml
...
  canary:
    enable: true
    service:
      hosts:
        - croc-hunter.istio.us.g.csanchez.org
    gateways:
      - jx-gateway.istio-system.svc.cluster.local
  canaryAnalysis:
    interval: 60s
    threshold: 5
    maxWeight: 50
    stepWeight: 10
```
metrics:
- name: request-success-rate
  # minimum req success rate (non 5xx responses)
  # percentage (0-100)
  threshold: 99
  interval: 60s
- name: request-duration
  # maximum req duration P99
  # milliseconds
  threshold: 500
  interval: 60s
PROFIT!

jx promote croc-hunter-java \
--version 0.0.130 \
--env production
podinfo.test
New revision detected, starting canary analysis.

Target
Deployment/podinfo.test

Traffic routing
Weight step: 5 max: 50

Failed checks threshold
10

Progress deadline
60s

podinfo.test
Canary analysis completed successfully, promotion finished.

podinfo.test
Progress deadline exceeded deployment does not have minimum availability for more than 60s

podinfo.test
Failed checks threshold reached 10
A Kubernetes Native Java stack tailored for GraalVM & OpenJDK HotSpot, crafted from the best of breed Java libraries and standards
THE DEVOPS 2.6 TOOLKIT

Jenkins X

Viktor Farcie