endava

CONCURRENCY vs PARALLELISM in DEVOPS

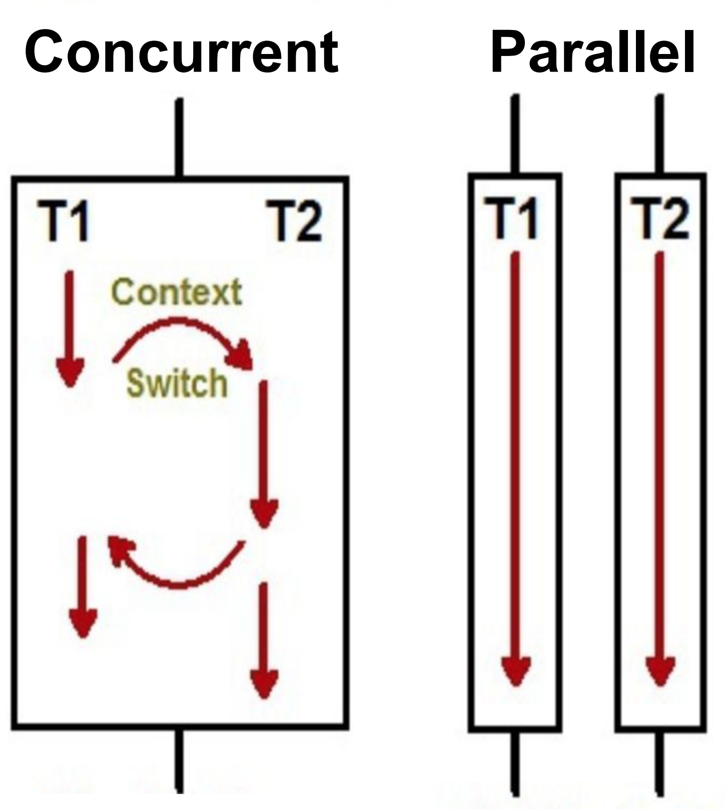
From real-life examples to DevOps tools!

Catherine Cruz. AAD Days 2019

AGENDA

- Concepts definition.
- Real-life case study.
- Coding: Monzo, banking from monolith to microservices.
- Testing: Jenkins X, pipelines going into the cloud.
- Operations: Kubernetes, deployments and health checks.

Concepts Definition



Concurrency is the way we interact with the world!

Concurrency means multiple tasks which start, run, and complete in overlapping time periods, in no specific order.

Parallelism is when multiple tasks or several part of a unique task literally run at the same time, like in a multi-core processor.

*¨Concurrency is about **dealing** with lots of things at once. Parallelism is about **doing** lots of things at once.¨*

Rob Pike - 'Concurrency Is Not Parallelism'

CONCURRENCY

Concurrency is just like the brain works!

Is the composition of independently executing computations.

If you have tasks having inputs and outputs, and you want to schedule them so that they produce correct results, you are solving a concurrency problem.

Some times, there is no specific order between them, so we have multiple alternatives for running it sequentially.

Only when you build things concurrently you can safely execute them in parallel.

Parallelism

Is the simultaneous execution of multiple things.

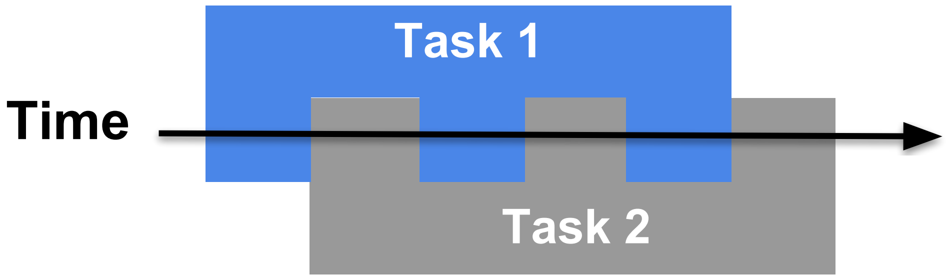
"Two queues and two ATM machines"

Parallelism means that an application splits its tasks up into smaller subtasks which can be processed in parallel, for instance on multiple CPUs at the exact same time.

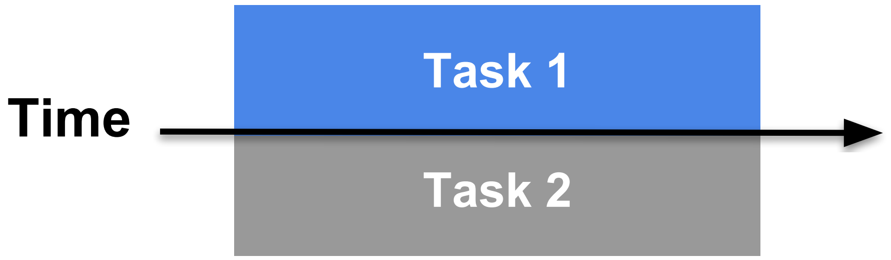
Typically, programs starts sets of child tasks that run in parallel and the parent task only continues once every subtask has finished. This makes parallel programs much easier to debug.

All about management and performance

Imagine learning a new programming language by watching a video tutorial. You need to pause the video, code what been said then continue watching. **That's concurrency.**



Now you're a professional programmer. And you enjoy listening to calm music while coding. **That's Parallelism.**



Now, apply all that into our daily life

Create a Bank Account and a Client’s Report

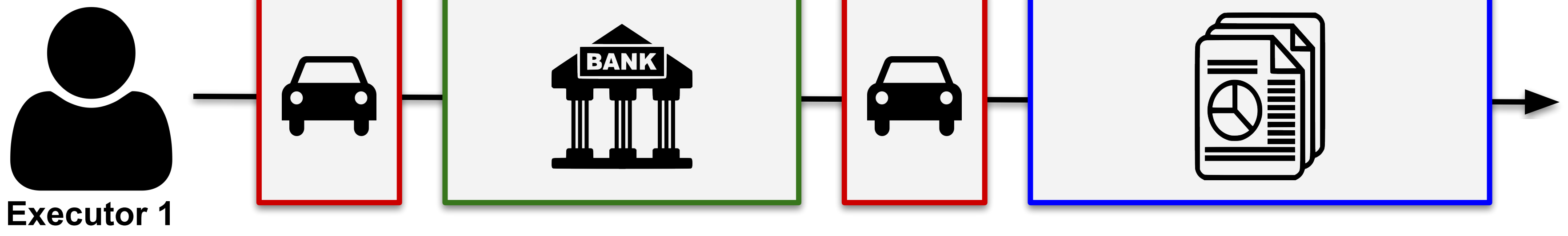
Remember we need to keep it real, so here are the rules:

Complete these two tasks in a day.
You are allowed to work with your team.
Present your ID and sign when creating the account.

Time frames:
New accounts takes 2 hours to complete.
Your Client’s report is for the same day.

Case 1: Sequential Execution

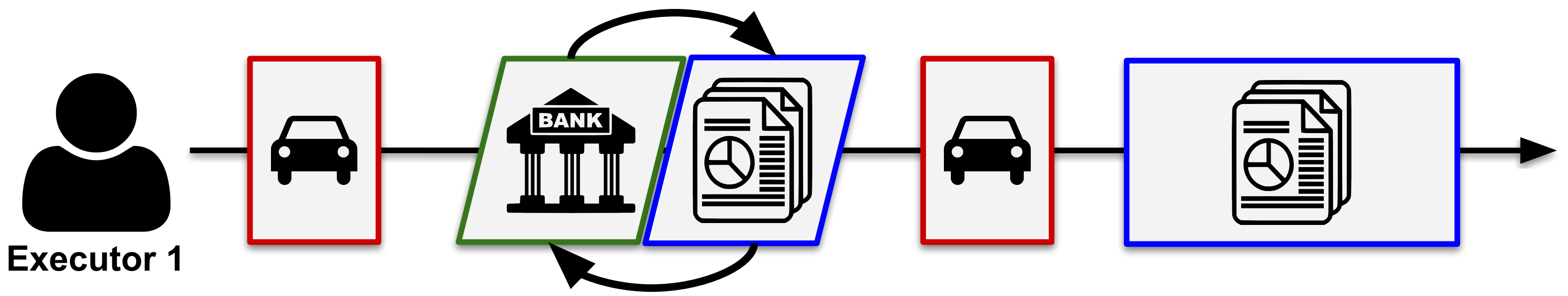
Ordinarily, you will drive to the bank, then wait in line for 1 hour, create the new account, drive back to the office, and invest 3 more hours getting the report done.



Sequential, neither parallel – nor concurrent, which means that it processes all tasks one at a time.

Case 2: Concurrent Execution

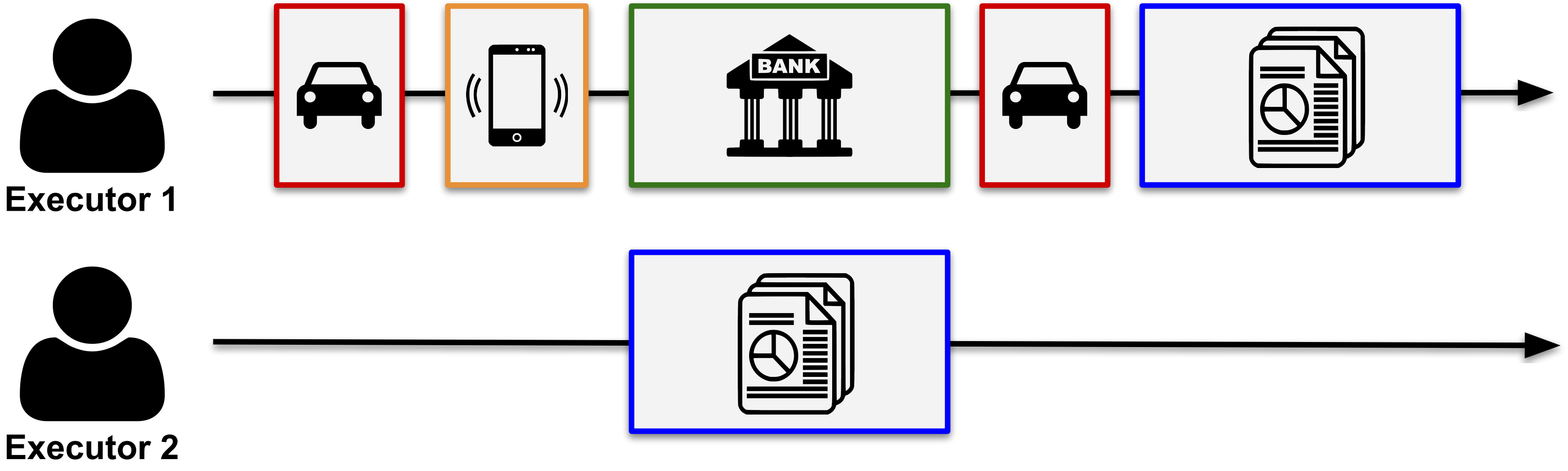
You plan ahead and take your computer with you, so while you waiting in line at the bank, start working on your report. In this case, both task are done by you, but in pieces.



You interrupted your bank task while waiting in the line, then you switch back when your turn is called. The saving in time was essentially possible due to **interruptibility of both the tasks**.

Case 3: Parallel Execution

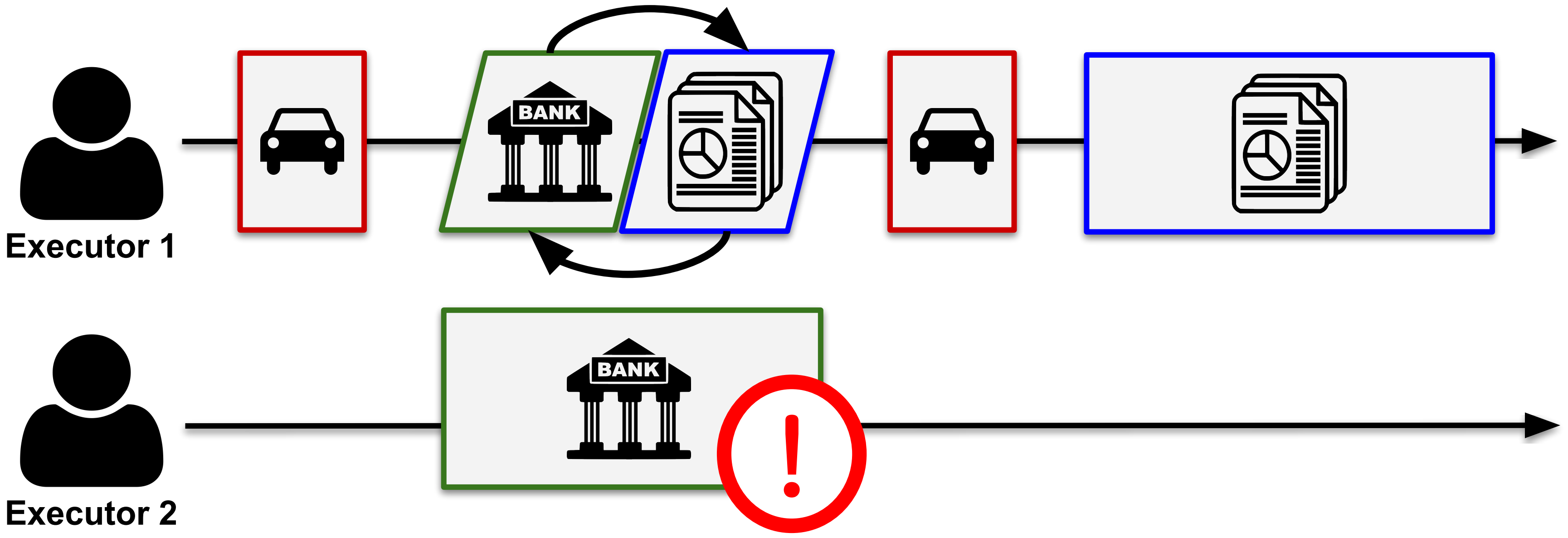
Before going into the bank, you call your assistant at the office, to start a draft of the report.
Once your new account is created, you go back to the office and finalize the report.



Tasks independent ability allows performing both tasks at the same time by **two different executioners.**

Case 4: Concurrent but not Parallel

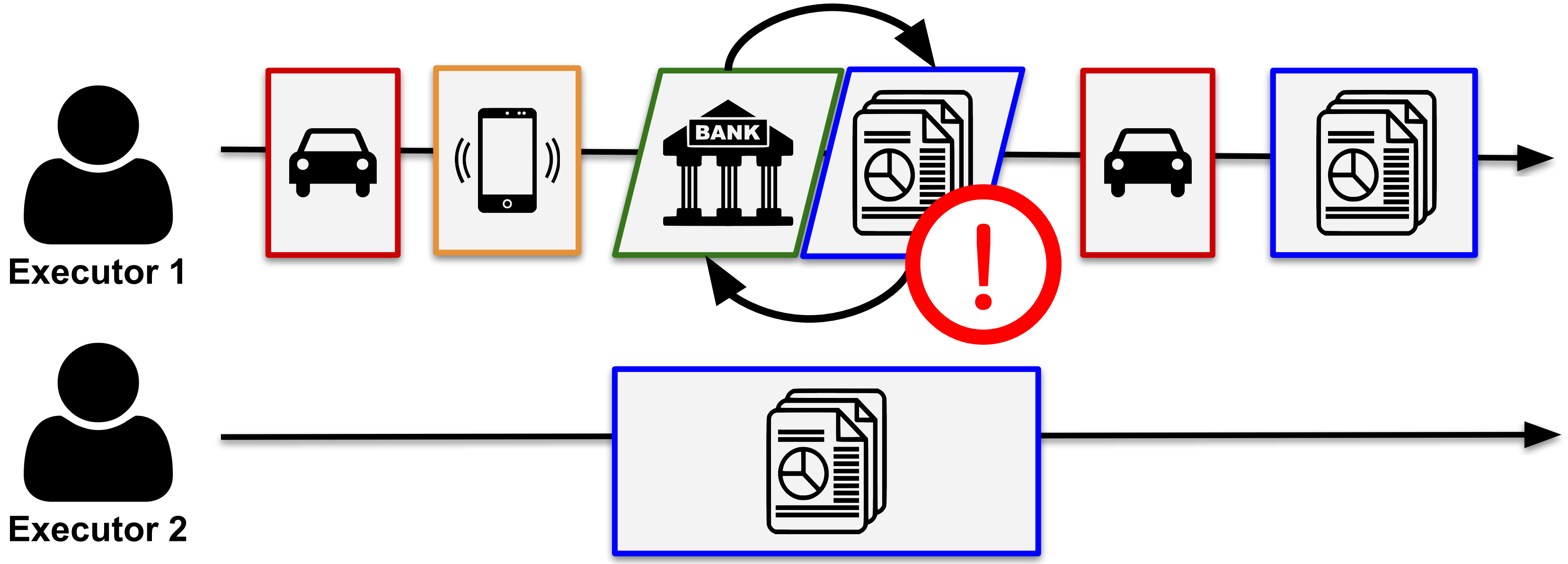
For security reasons the bank will check your ID, so this task can be executed only by you.



Bank task has interruptibility (you can pause it while waiting in the line, then resume it later when your turn is called), **but no independent ability** (your assistant cannot wait in your stead).

Case 5: Parallel but not Concurrent

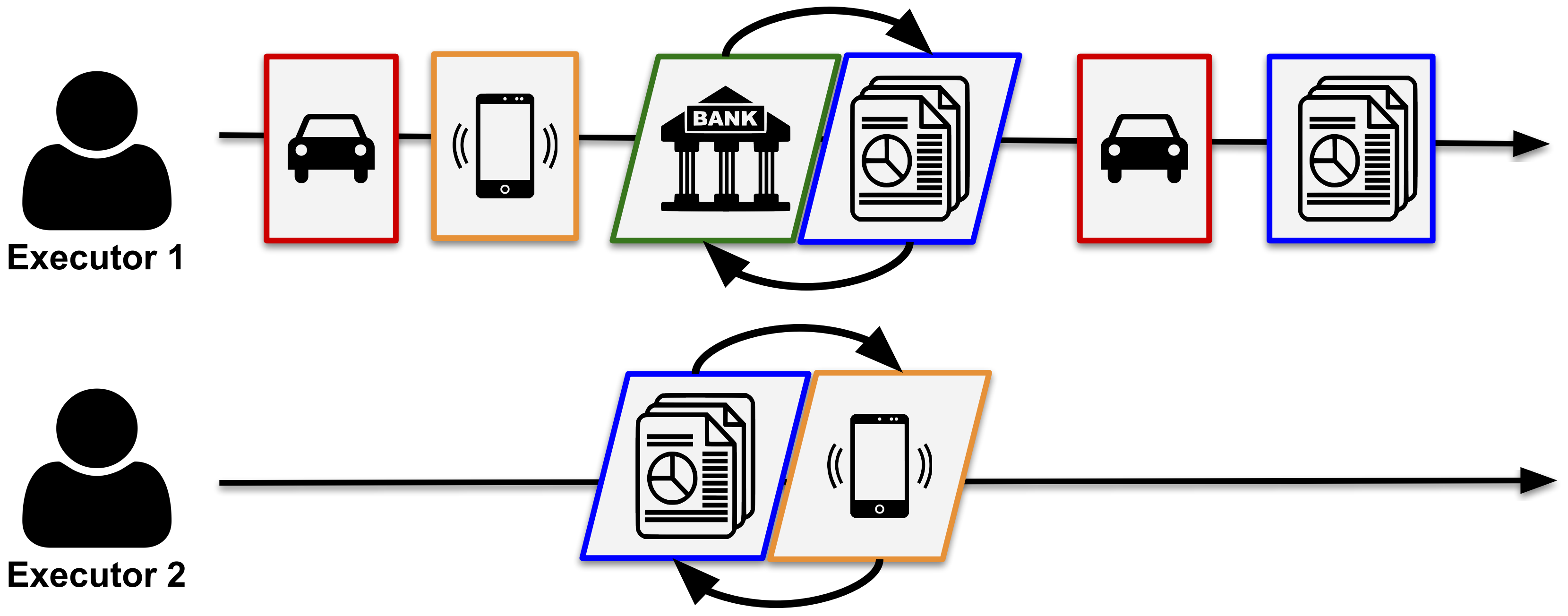
This is a bank after all, and it has security checks on the premises, where you cannot use your phone or laptop inside, so you have to finish this task before any other execution.



In this, case, the bank task is **neither independentable nor interruptible**. Even if you are waiting in the line, you cannot work on something else because you do not have necessary equipment.

Case 6: Concurrent and Parallel

Here, you call your assistant to start working with in a shared doc, also have a laptop with you. While waiting, you work together. So back at the office instead of 3 hours, you just need 15 minutes to finish.



Because report task has independent ability (either one of you can do it) and interruptibility (you can stop it and resume it later). You **concurrently and parallelly executed both tasks.**

Banking from monolith to microservices

MONZO Bank Ltd, is a digital, mobile-only bank

Based in the United Kingdom.
Originally operating through as a mobile prepaid debit card,
in April 2017 their UK banking licence restrictions were lifted,
enabling them to offer baking account services.

One of the earliest new app-based challenger banks in the UK.

https://monzo.com/

Banking as a monolith system

That is Guatape, yes, it’s a big rock!

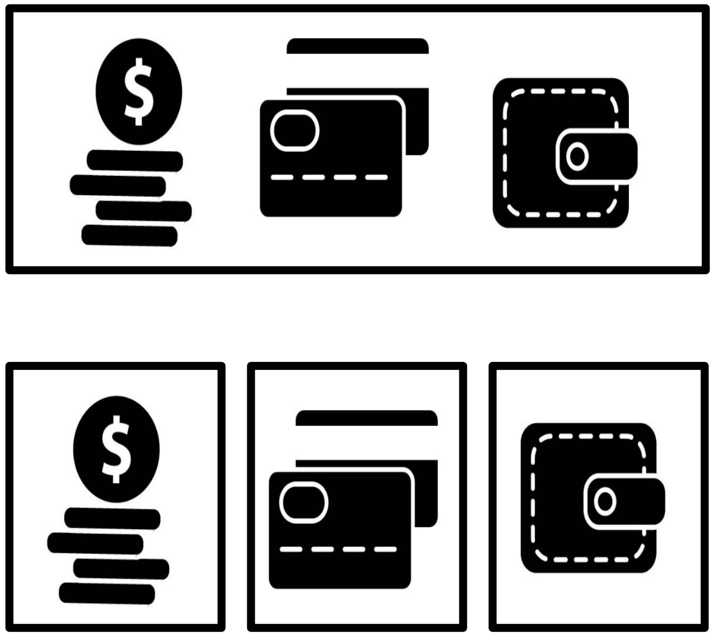
Baking core platforms are typically batch-based, largely monolithic, and ancient by modern standards.

Sequential execution is expected, so the challenge typically surfaces when applications must evolve faster.

Even if your banking application uses an agile methodology to go to market, change sets that impact core banking will delay the whole train from getting to the station on time.

The Rock of Guatapé Colombia

Divide and Conquer!



Moving into multiple small, independent services

In short, microservices stems from the same ideology as Agile and DevOps, by seeking to break down slow moving, monolithic systems into multiple small, highly decoupled **independent services.**

This breaking down into manageable pieces that can **operate independently in parallel,** allows large organizations to be more flexible to move at the speed of their smaller competitors.

Microservices enable concurrency and parallelism development.

Path to microservices - Rimantas Benetis

What Netflix, Amazon, and Twitter taught to Monzo

Follow the leads...

- Single responsibility principle **(Microservices)**

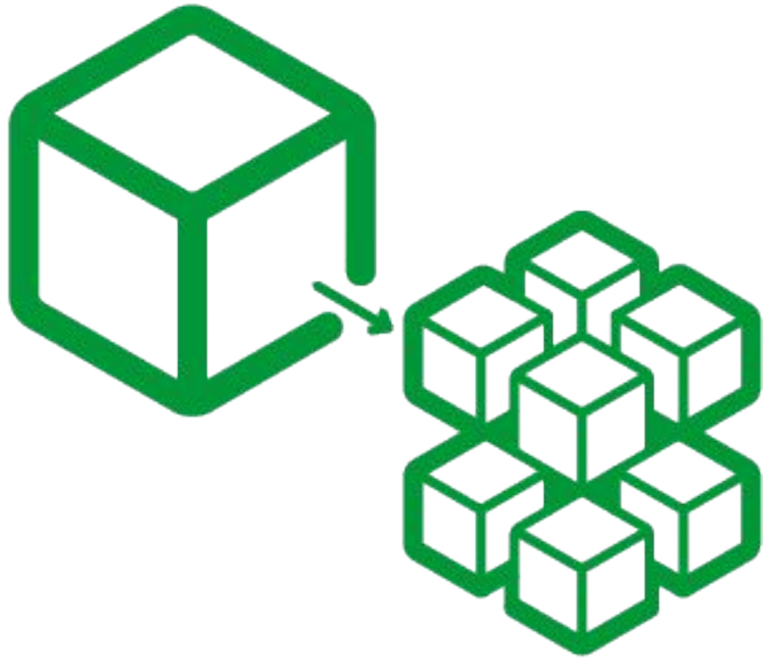
  "Do an specific job, and do it very well!"

- Bounded context **(Concurrency)**

  "Small chunks of business logic isolated to easier management"

- Well defined interfaces **(Parallelism)**

  "My transaction info should not know about the merchant info"



GOTO 2016 • Microservices in Go • Matt Heath

Pipelines going into the cloud

Jenkins X

Provides pipeline automation, built-in GitOps

Initially is a distribution of the Core Jenkins with a custom kubernetes configuration and some additional built in plugins (e.g. jx pipelines plugin) packaged as a Helm chart.

Planned to move to a more Cloud Native, by using Kubernetes resources to store jobs, runs and credentials. So it will be easier to support things like multi-master or one shot masters.

https://jenkins-x.io/

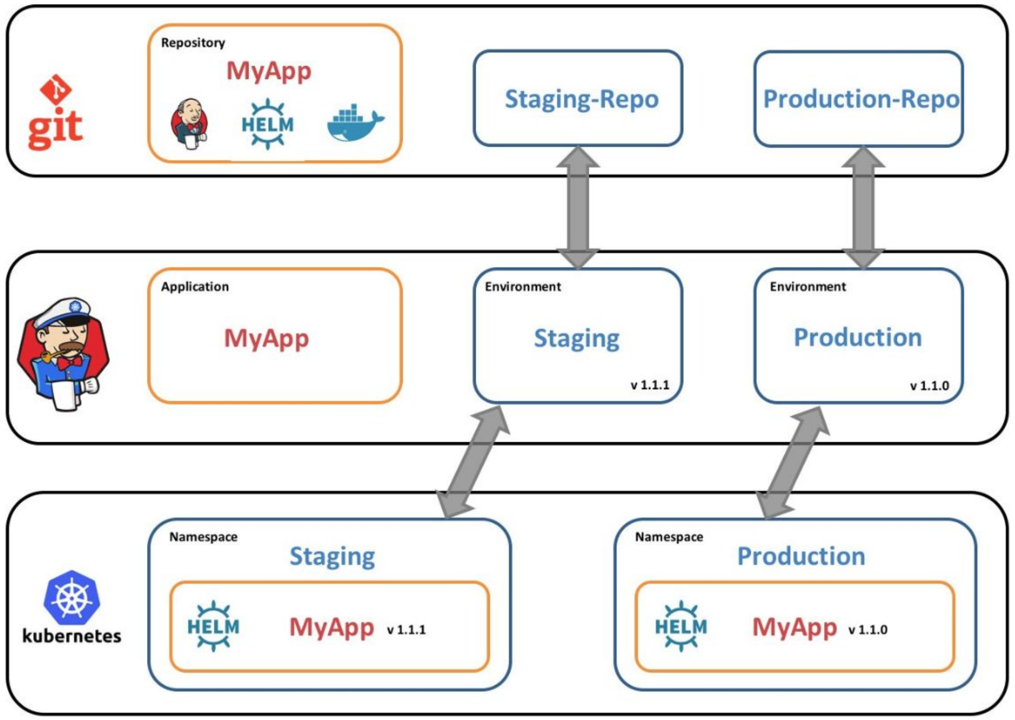
Deploying from Jenkins X

With Jenkins X, is not necessary to understand or perform comprehensive K8s operations, and can significantly improve the shipping rate of a product in the K8s cluster.

DevOps engineers no longer have to deal with Docker files, tweak Helm charts, or write a jenkinsfile.

Aims to enable cloud-native functionalities, recommended to be use in a cloud provider.

Enterprise grade CI/CD with GitOps



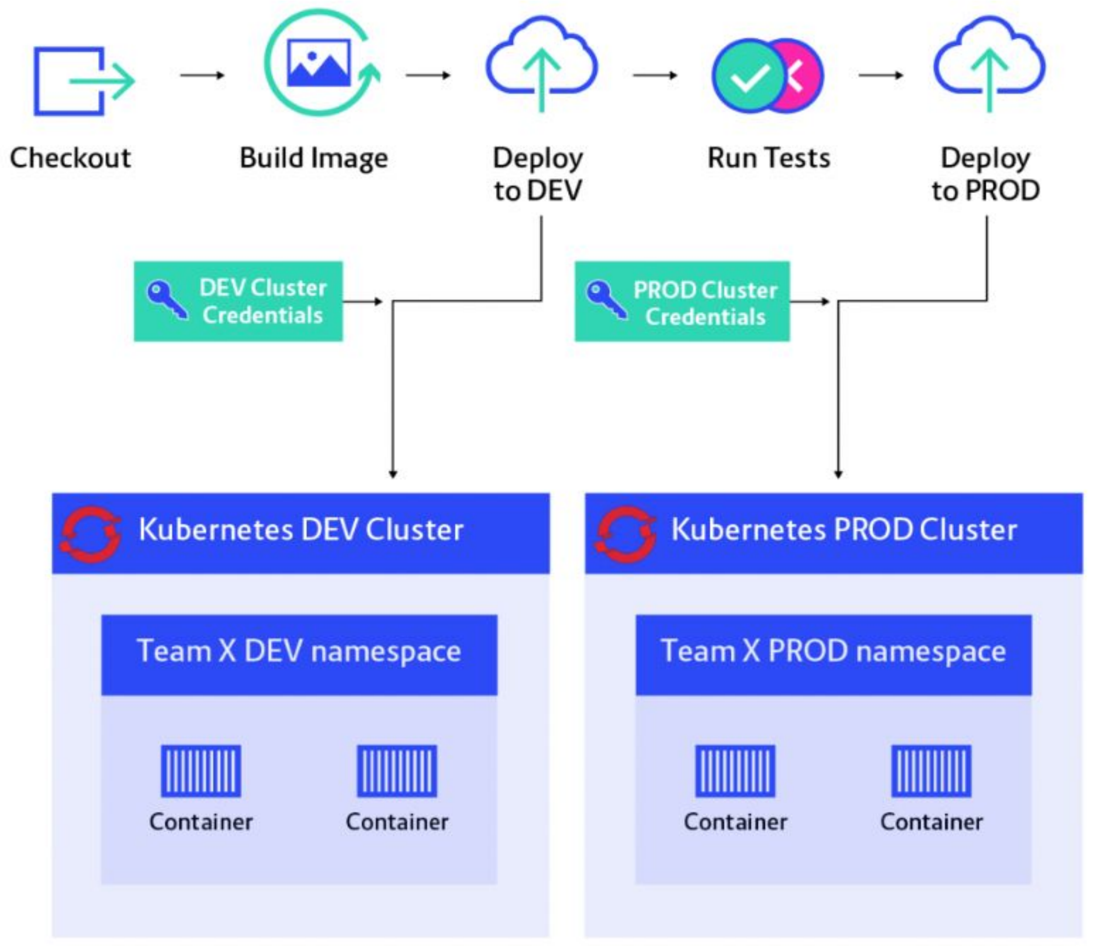
Cloud Native Concurrency and Parallelism

Concurrency will be managed by Jenkins X master, calling a Docker image building when a commit occurs, and also, by promote between environments when a tagget PR is approved.

Parallelism will be provided by Kubernetes on each namespace (Production, Staging, etc) or cluster when working with federated environments.

CI/CD in a GitOps environment.

Enterprise grade CI/CD with GitOps



Deployments and health checks

Open-source container orchestration system

Provides automating application deployment, scaling, and management. It was originally designed by Google and is now maintained by the Cloud Native Computing Foundation.

Many of the key developers were previously working on Google’s daily in-house orchestration tasks.

Designed as a concurrent and parallel management tool.

https://kubernetes.io/

Deployments in a nutshell

The **deployment instructs kubernetes** how to create and update instances of your application.

Once created, the kubernetes master schedules the application instances into **individual Nodes in the cluster.**

Features:

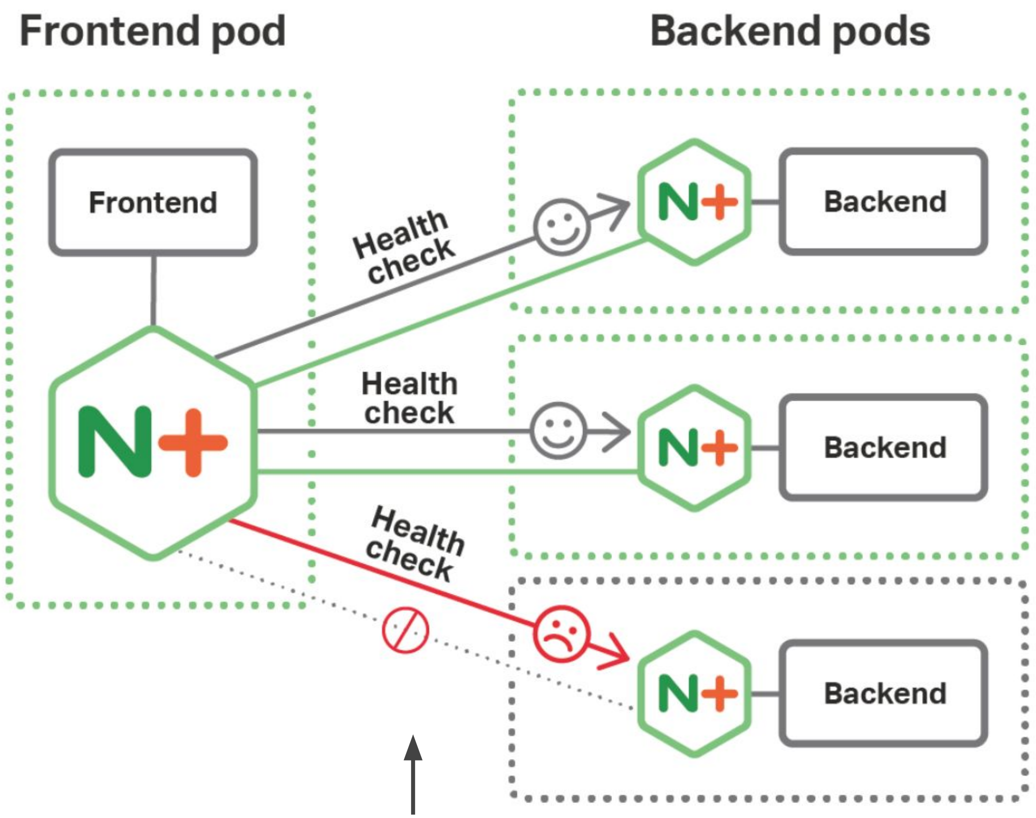
- Ensures that X number of pods are always running.
- Consist of a Pod template and a number of replicas.
- Supports redeployments by changing pod template.
- Horizontal scaling available by adding more pods.

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: <deployment-name>
spec:
  replicas: 2
  template:
    metadata:
      labels:
        <key>: <value>
    spec:
      containers:
      - name: <container-name>
        image: <container-image>
        ports:
        - containerPort: 80
        env:
        - name: <key>
          value: <value>
```

Parallelism

Concurrency

Health checks: Liveness and Readiness probes



Liveness check as parallel executors
and Readiness for concurrent checks

Health checks in Kubernetes, are carried out by the kubelet to determine when to restart a container.

The purpose of **liveness probes** are to indicate that your application is running. (Independent ability)

Readiness probes meant to check if your application is ready to serve traffic. (Dependent Interruptibility)

Using Kubernetes Health Checks

One last musical example...

An orchestral performance of a symphony is concurrency.

A chorus singing in unison is operating in parallel.

Thank you!

endava

Catherine Cruz

Backend Senior Engineer

AAD Days Belgrade / March 2019