DevOps, Docker and Gitlab-CI

Part 1: DevOps

Version 1.1.0 (2023-02-15)

-- Cyril zarak Duval, root CRI/ACU 2020

Introduction

Generic information about the course

Why should you listen to the course ?

- ➔ Being able to code is nice, without the production is useless
- ➔ State of the art notions
- ➔ Having the rights tools makes you more proficient
- ➔ Subject somewhat difficult
- ➔ Getting a decent grade

Who am I ?

- Why this slide ?
  - To understand my profile and my point of view
- EPITA 2020 - CRI/root ACU
  - ~6 years of real DevOps experience
  - I know EPITA and students' POV
- "DevOps engineer" title, in fact more SRE / Ops oriented
  - Not a dev

Plan

1. DevOps concepts
2. DevOps tools
    a. YAML
    b. Docker
    c. Docker-compose
    d. Gitlab-CI
3. Observability

What is DevOps ?

Surprisingly, it’s both Dev and Ops

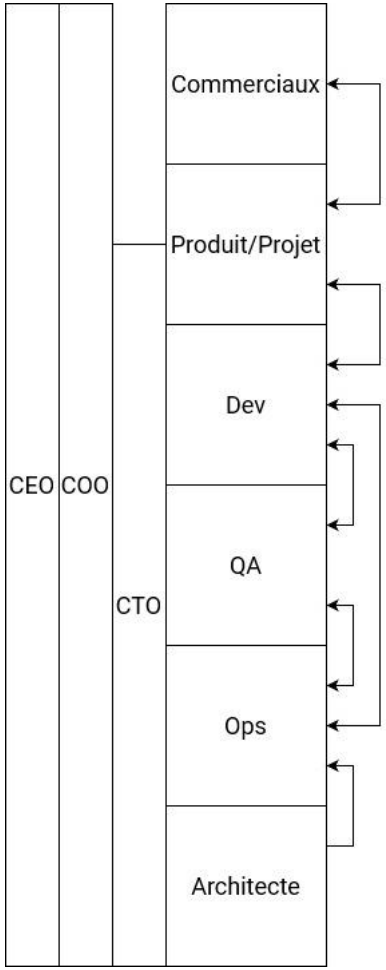
DevOps

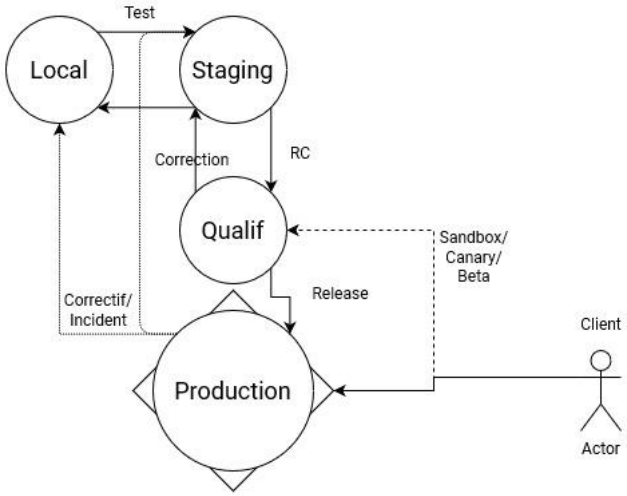
- Contraction of Development and Operations
- Development:
  - Creating code, applications, …
  - A will to introduce changes
  - Work env: your laptop, your IDE
- Operations:
  - Ensuring services are working and available (including aforementioned applications)
  - A will to not break anything (somewhat reluctant to change)
  - Work env: servers/VMs, a text editor and ssh

DevOps

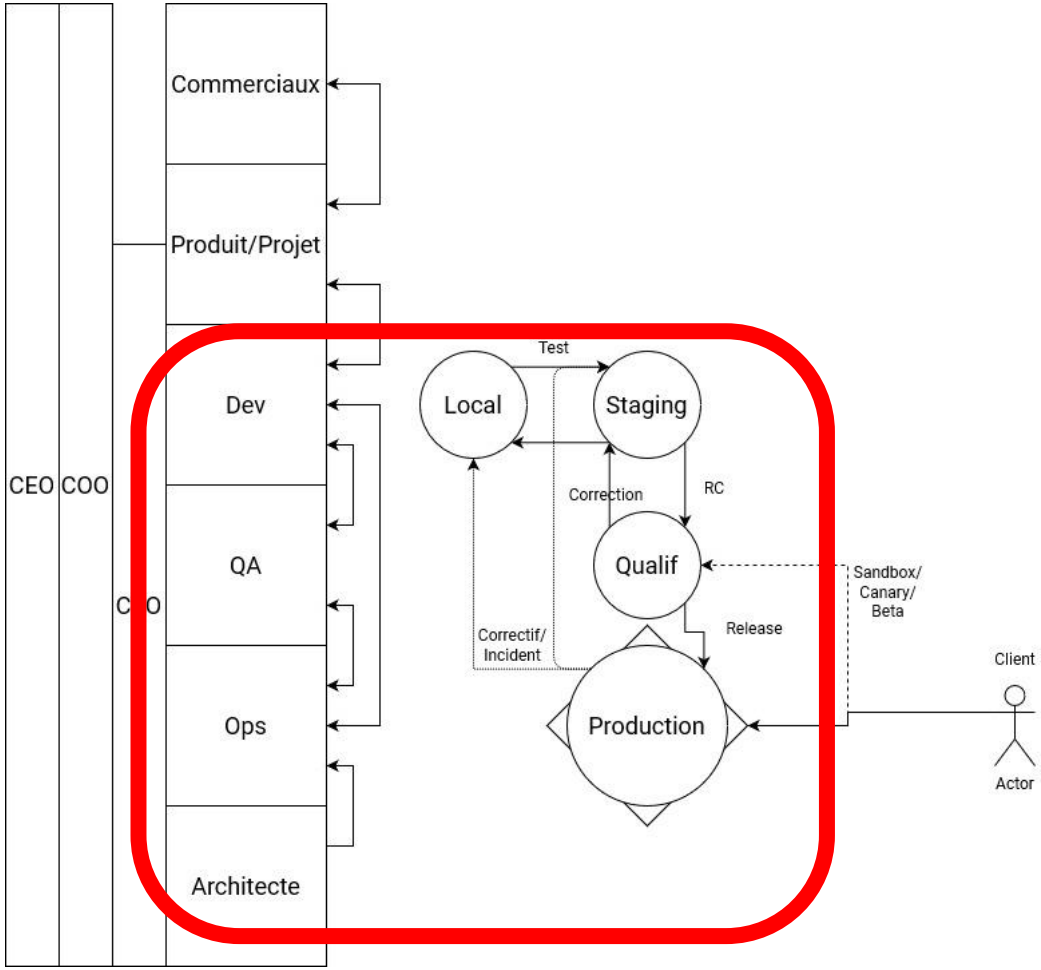
- DevOps is a set of tools and practices
- Aim to reduce – even remove – friction between Devs and Ops
- Devs are doing a bit of Ops
- Ops are doing a bit of Dev
- DevOps is not really a job per-se
- Some DevOps tools are so popular they became new standards

Devs and Ops relation





DevOps scope



What issues are solved by DevOps?

Why is everyone so psyched by DevOps ?

DevOps - Dev POV

- Devs work on their laptop
  - Their OS
  - Their libraries
  - Their environment
  - Their hardware specs
  - Their network stack
  - ...
- How to ship that to production servers seamlessly ?
- What are the meaningful differences ?

DevOps - Dev POV

- How does a Dev introduce architectural changes ?
- How to add a new dependency ?
- How to avoid “but it worked on my laptop :’(“ ?

But also …

- How to reduce time-to-market ?
- How to easily understand years of dev without you ?
- How to work with coworkers ?
  - That will be different from you

DevOps - Ops POV

- How to deploy Dev's application ?
- How to make sure it's working well ?
- How to know what the dependencies are ?
- How to upgrade things without breaking anything ?
- How to avoid differences between prod and dev env ?
- How to handle config/secrets ?
- How to easily understand years of ops without you ?
- How to work with coworkers ?
  - That will be different from you

DevOps - Solutions

- Devs must have some knowledge on how things are deployed
- Devs should have some control on ops side
  - Limited, scoped, supervised …
  - They aren't ops
  - Control should be application-oriented
- Ops should be involved in the dev process
  - Not necessarily by coding directly
  - Focus on dependencies

How does DevOps solve those issues?

Surely, it isn't magic

DevOps - Concepts

- Devops concepts are built around “12 factors”
- Goals of those 12 factors:
  - Be as declarative as possible
  - Understand interactions between app and system
  - No divergence between dev and prod
  - Try to be platform agnostic
  - Be able to scale up/down

DevOps - 12 factors - focus

- I. Codebase
  - One codebase tracked in revision control, many deploys
- II. Dependencies
  - Explicitly declare and isolate dependencies
- III. Config
  - Store config in the environment
- VII. Port binding
  - Export services via port binding
- X. Dev/prod parity
  - Keep development, staging, and production as similar as possible

DevOps - Practical aspect

- 12 factors is not the magical solution to everything
  - It gives good advices
  - Generic concepts
- DevOps is about mentality and tools
  - Let's look at the tools to understand the mentality

Let's talk about configuration

Let me introduce you to a new job:
YAML engineer

Configuration

- Article 3 of the 12 factors
  - strict separation of config from code
- What does it means ?
- Softwares must not include configuration within the code
- Configuration shall not even be in the code VCS
  - But could and should be stored in another VCS
    - Be careful not to store any secrets in plaintext in a VCS however
- Configuration will be different between dev, staging and prod(s)

What is configuration ?

- What is configuration ?
  - Everything removing genericity
  - Everything that can change from an env to another
- Examples:
  - Address of the database
  - IP/port to bind to
  - Log-level
  - Path(s) to store data
  - ....

How to provide config ?

- How to provide configuration ?
  - env variables
    - Recommended for simple cases, but limited
    - Only Key-Value
  - Config files
    - A bit more complex to setup/provide but advanced
    - Allow lists, mapping, etc
- How to write a config file ?
- Introducing to the most well known format in DevOps: YAML

YAML

- YAML = Yet Another Markup Language
- Used for Docker-compose, Gitlab-CI, Kubernetes, Helm charts, Ansible, Elasticsearch, Argo, Harbor, …..
- A superset of JSON to make it more human readable
  - Meaning that JSON is valid YAML
  - Even small JSON snippets embedded within a YAML file
- .yml or .yaml extension
- Libs in every language to parse it

YAML - the simple way

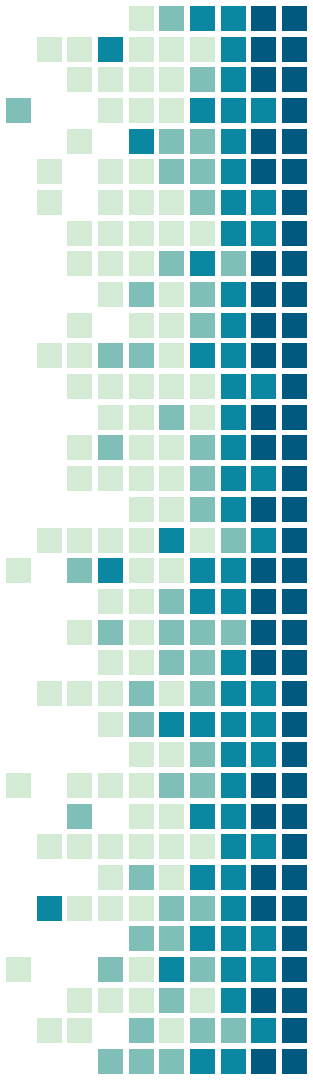
- YAML is a key-value format
- key is a string, value can be :
  - string
  - int
  - boolean
  - list
  - mapping
- Indentation is important
- Quoting can be used

YAML – the simple way

```yaml
---

key_1: value
key_2: 140
key_3: 1.0
key_4: "1.0"
key_5:
    - 0
    - 1
    - 2
key_6:
    subkey_1: value with a "quote"
    subkey_2: '"full quoted"'
    subkey_3:
        - subsubkey_1: value1
          subsubkey_2: value2
        - subsubkey_1: value3
          subsubkey_2: value4
key_7:
    bool_1: True
    bool_2: False
    bool_3: yes
    bool_4: no
    bool_5: on
    bool_6: off
```

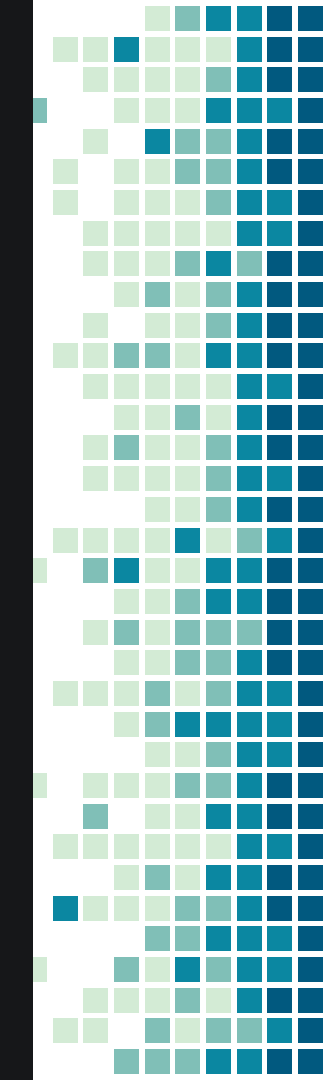
YAML - Advanced features

```yaml
---
common: &common
  toto: {"key": "value"}
  titi: tutu

a:
  - <<: *common
  - <<: *common
    titi: something else

key: |
    multi
    line
    value

key2: |-
    "another way to escape quotes

key3: |
    2

key4: >
    lorem
    ipsum
    sit

            amet dolor
```



```yaml
---
common: &common
  toto: {"key": "value"}
  titi: tutu

a:
  - <<: *common
  - <<: *common
    titi: something else

key: |
    multi
    line
    value

key2: |-
    "another way to escape quotes

key3: |
    2

key4: >
    lorem
    ipsum
    sit

            amet dolor
```

```
$ cat /tmp/file.yaml | yq
{
  "common": {
    "toto": {
      "key": "value"
    },
    "titi": "tutu"
  },
  "a": [
    {
      "toto": {
        "key": "value"
      },
      "titi": "tutu"
    },
    {
      "toto": {
        "key": "value"
      },
      "titi": "something else"
    }
  ],
  "key": "multi\nline\nvalue\n",
  "key2": "\"another way to escape quotes",
  "key3": "2\n",
  "key4": "lorem ipsum sit\n\n amet dolor\n"
}
```



YAML - tools

- ➔ yq is a useful tool to parse yaml in bash
- ➔ yamllint detects linting errors, inconsistencies and warn you about possible misuage

YAML - pitfalls

- YAML by its nature can mislead users
- Here are some points to be careful on:
  - JSON being valid YAML syntax
  - Strings can be unquotted
    - Be careful with numbers
  - The Norway problem
    - Case insensitive booleans in [Yes, True, On, Y]
  - Indentation
  - Mixup of “complex” types (mapping of lists of mappings, …)

YAML – some help

- ➔ https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

Thanks !

Questions ?

Slides available on zarak.fr/

Contact: cyril@cri.epita.fr

zarak production#5492