Empresa Brasil de Comunicação

de **Gitlab** a **Kubernetes**:

Serviços públicos com entrega ágil de sistemas de TIC sustentados por plataforma de orquestração contêineres

Adriano Vieira
adriano.vieira@ebc.com.br

Empresa Brasil de Comunicação

de **Gitlab** a **Kubernetes**

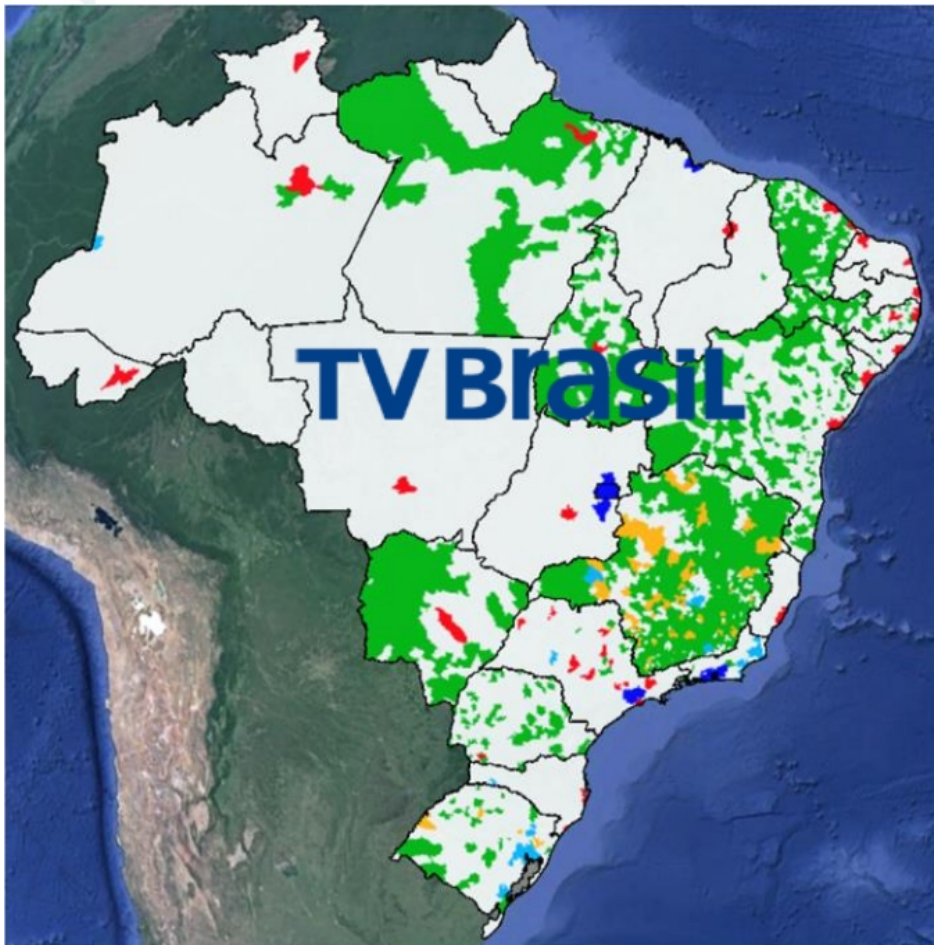
Serviços públicos com entrega ágil de sistemas de TIC sustentados por plataforma de orquestração de contêineres

Empresa Brasil de Comunicação

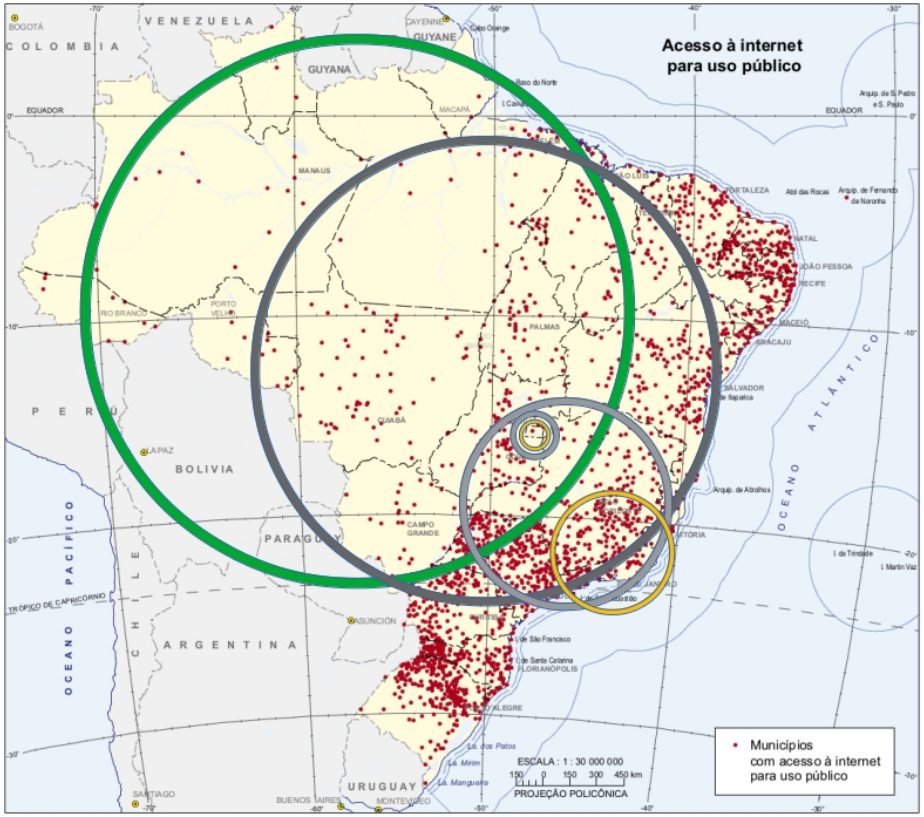
- **Agencia Brasil**: agência pública de notícias
- **NBR Governo Federal**: emissora estatal de televisão do Governo Federal
- **Portais**: Conteúdo público & Comunicação pública.
- **Rádioagência Nacional**: comunicação de conteúdos radiofônicos
- **Rádios**: Sistema EBC de Rádio que cobre todo o território nacional
- **Serviços**: diretamente prestados a clientes publico e privado
- **TV Brasil**: emissora de TV integrante da Rede Pública de Televisão

http://www.ebc.com.br/institucional

Empresa Brasil de Comunicação



Emissoras e parcerias da TV Brasil
(geradoras, retransmissoras, afiliadas)



Fonte: IBGE, Pesquisa de Informações Básicas Municipais 2009.

Rádios

- Nacional da Amazônia
- Nacional de Brasília
- Nacional do Rio de Janeiro
- MEC
- Internet: municípios com acesso para uso público
  (Fonte: IBGE/2009)

Empresa Brasil de Comunicação

http://www.ebc.com.br/institucional

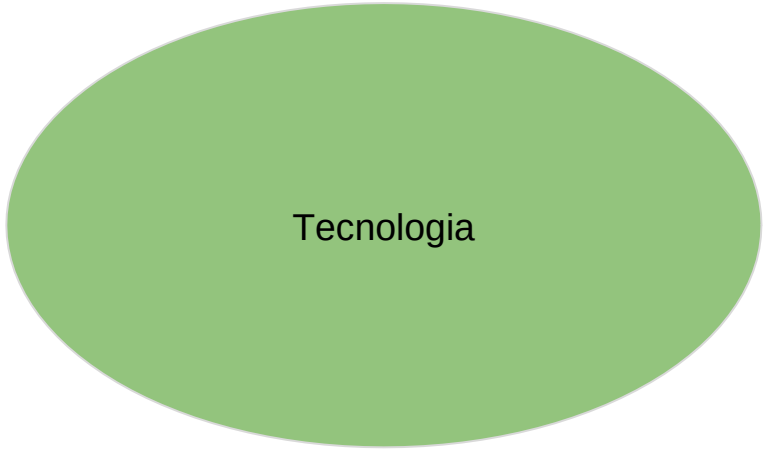
em TIC

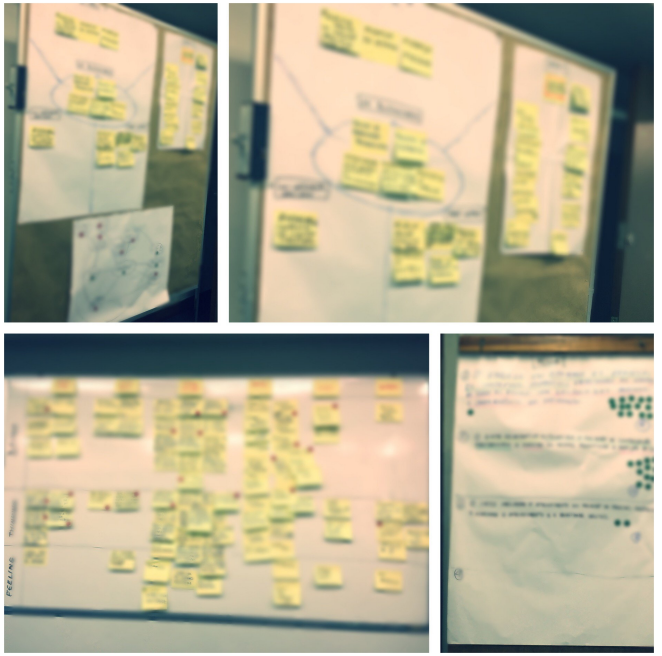
- Equipes: 4 (~30 Dev&Ops)
- Projetos: 300+
  - Java EE, Drupal, RoR, Python, NodeJS, C/C++, pyTorch
- Portais/Serviços/apps
  - Gestão de Ativos Digitais
  - Publicidade Legal
  - Portais EBC
  - EBC Play
  - EBC Rádios

Empresa Brasil de Comunicação

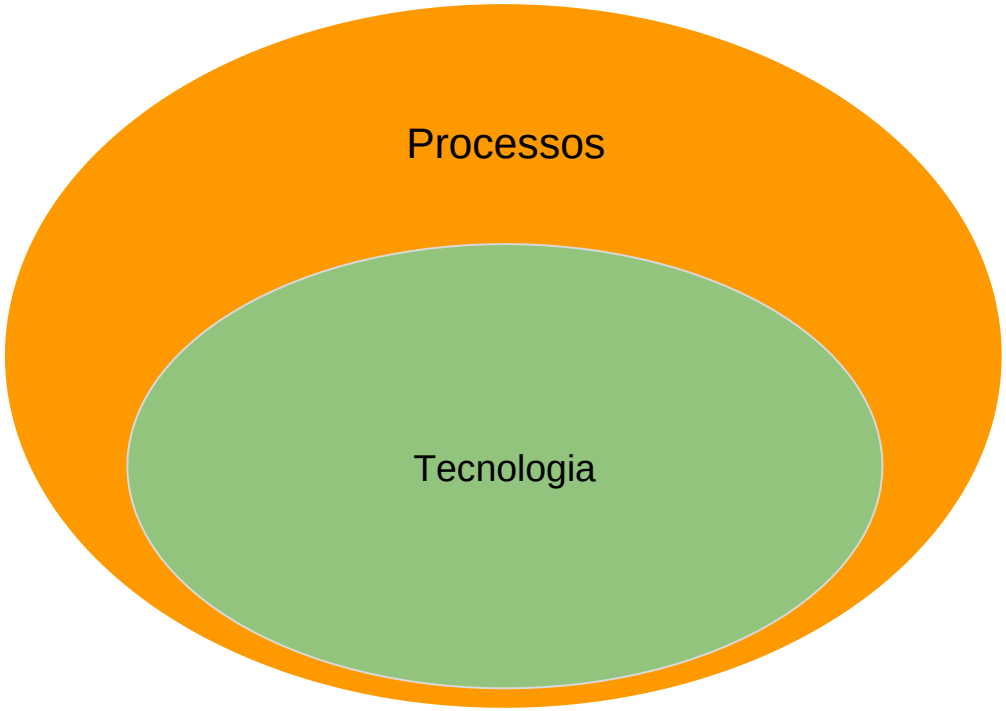
Melhorar e Adaptar

Melhorar e Adaptar

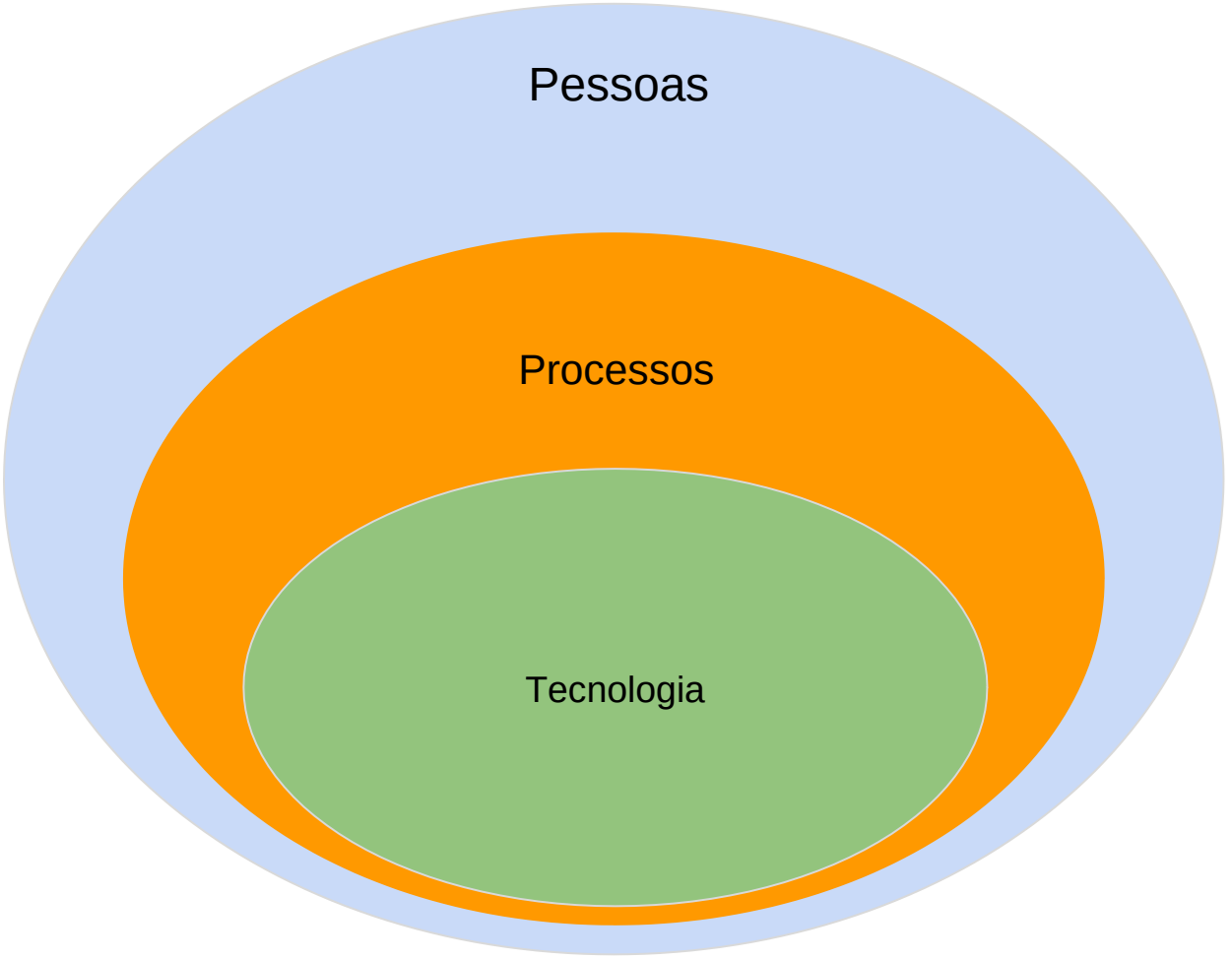




Melhorar e Adaptar

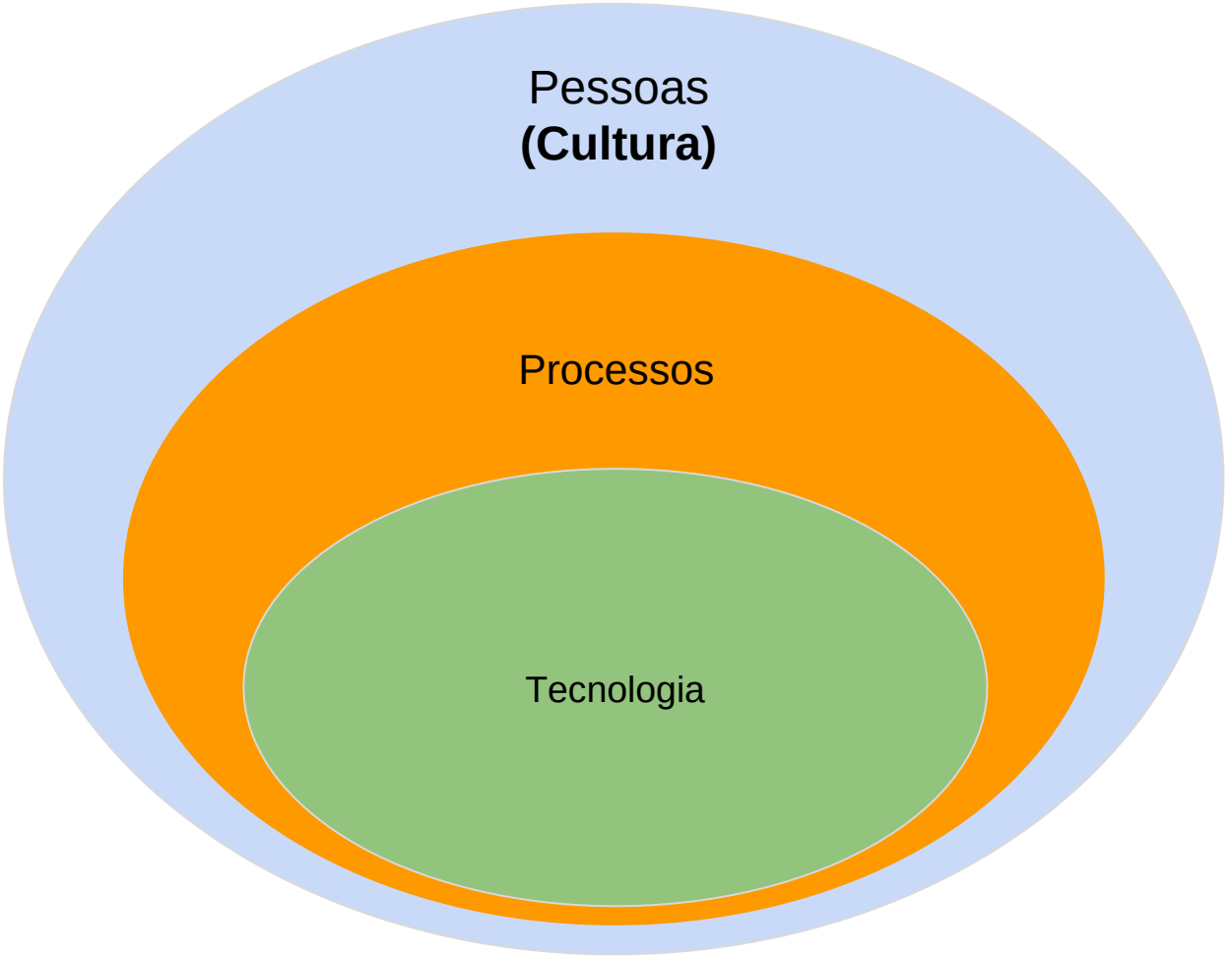


Melhorar e Adaptar



Fonte: Making IT Lean

Melhorar e Adaptar

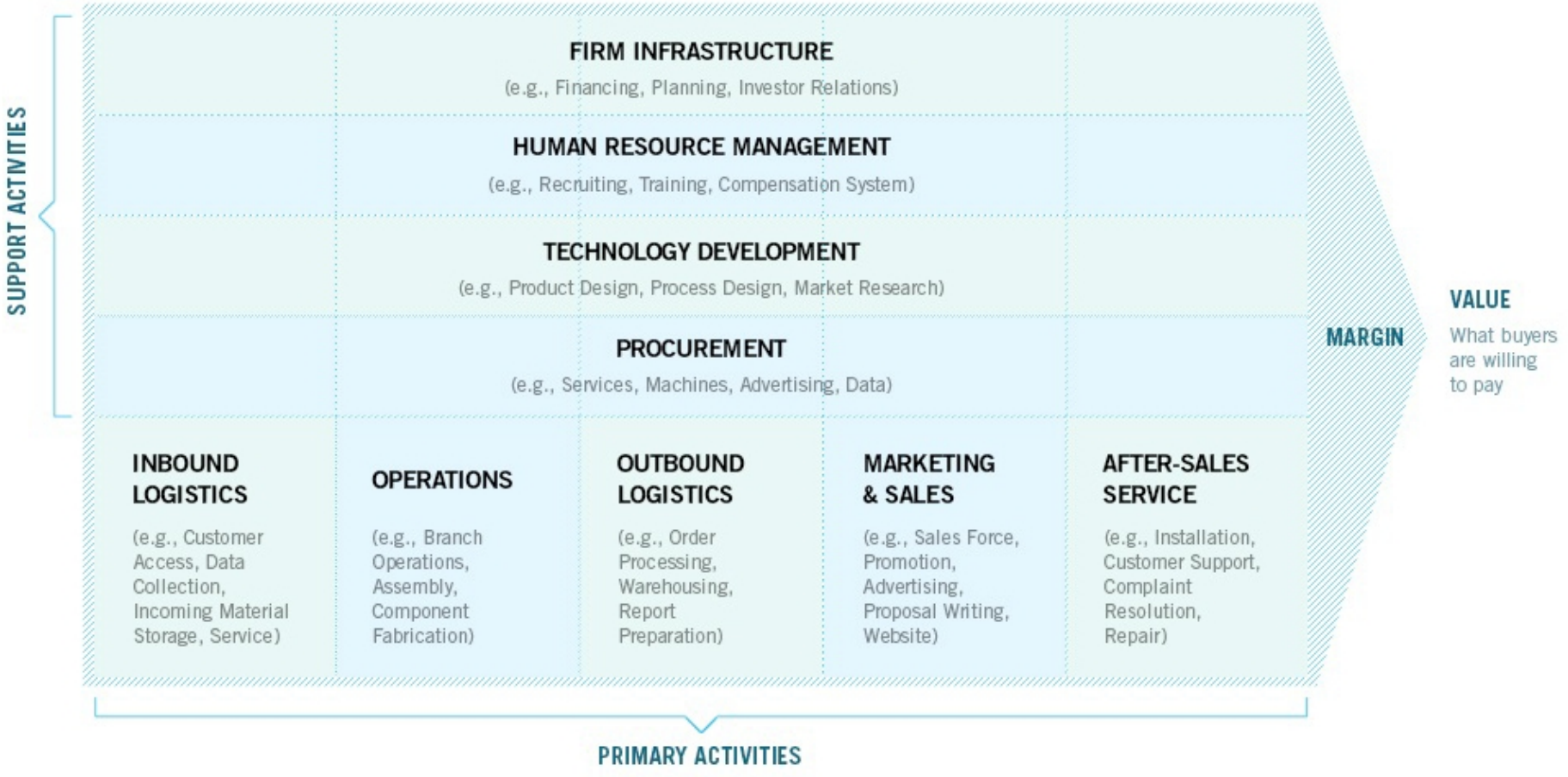


Fonte: Making IT Lean

Melhorar e Adaptar

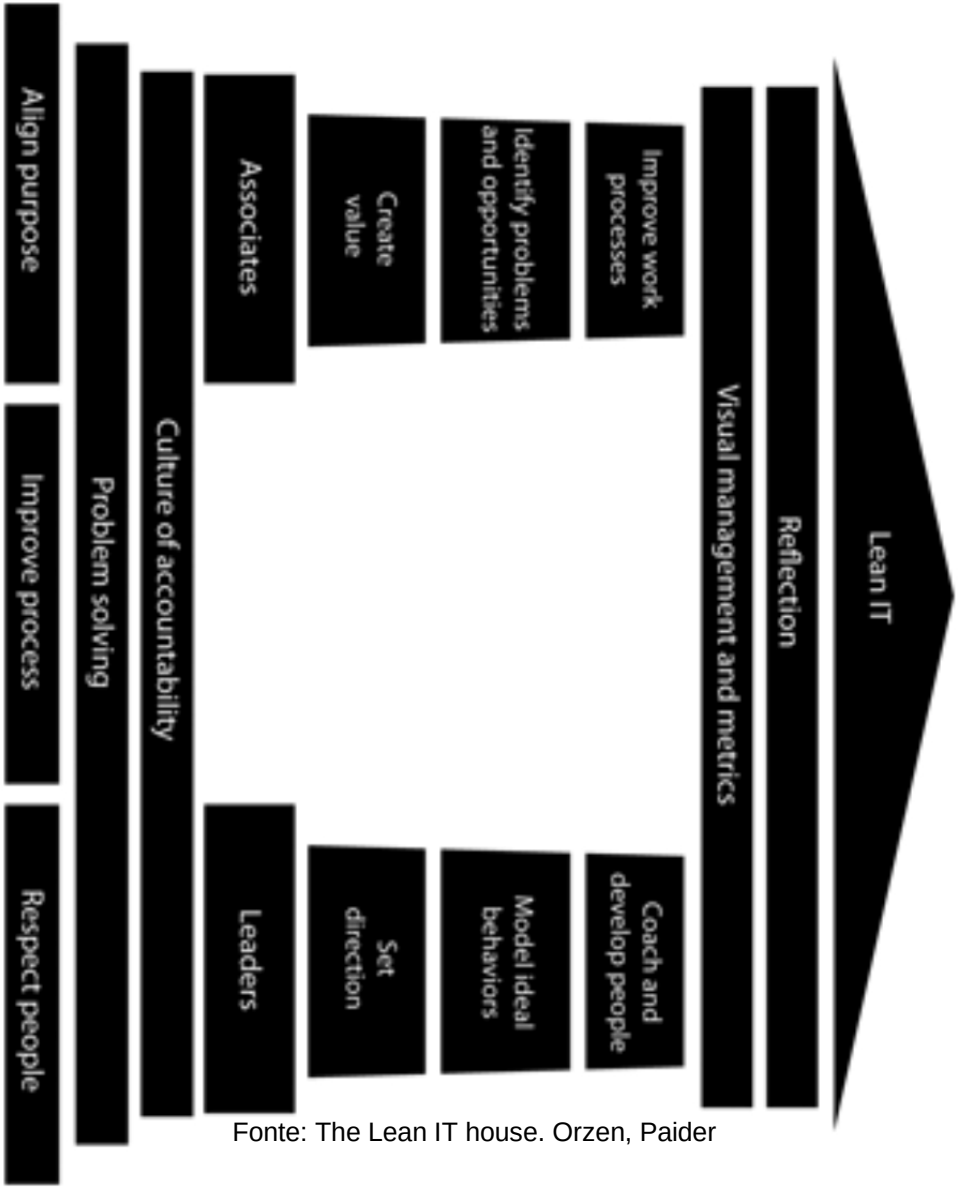
Cadeia de Valor

Michael Porter



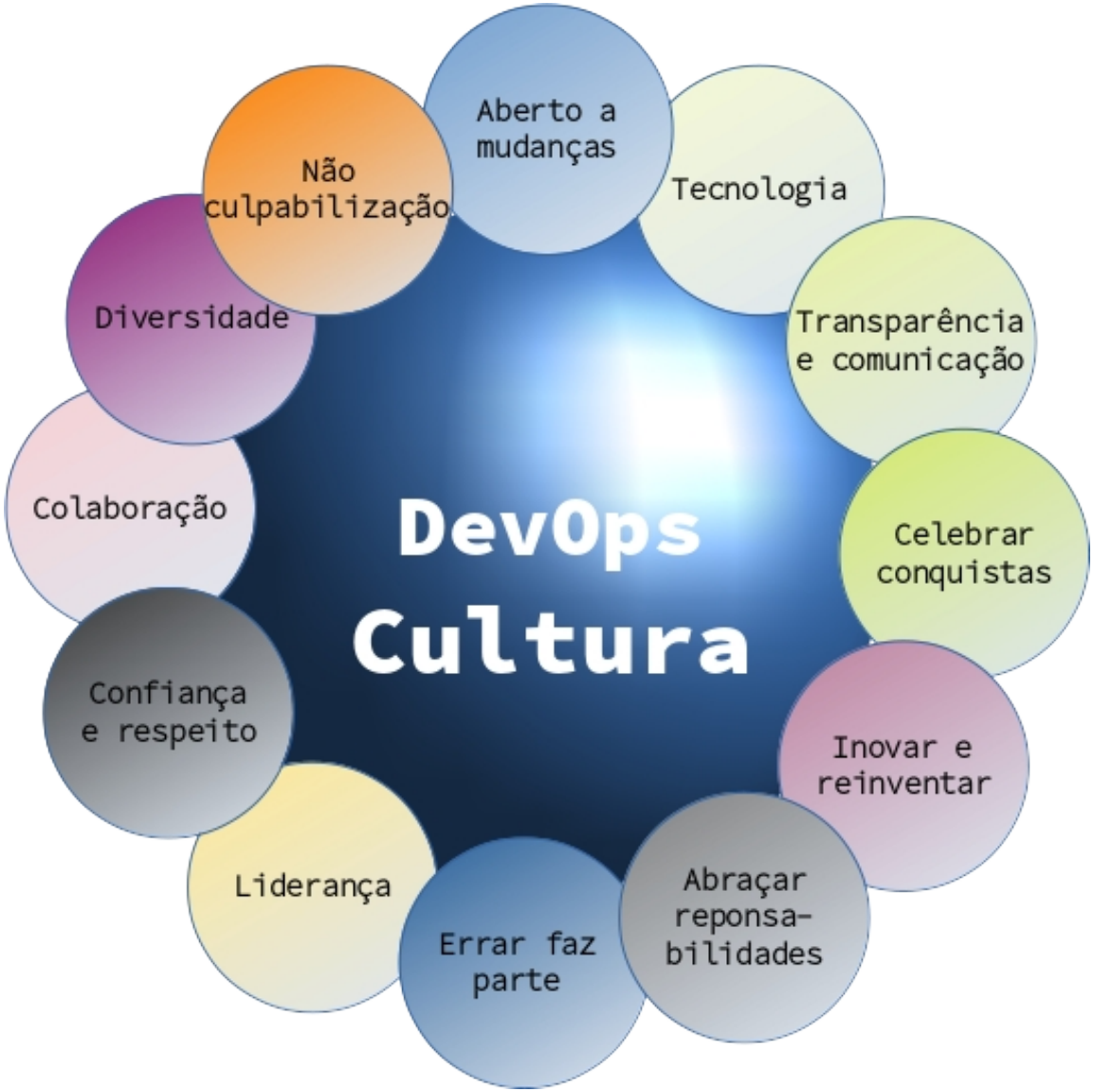
Fonte: HBS. https://www.isc.hbs.edu/strategy/business-strategy/Pages/the-value-chain.aspx

Melhorar e Adaptar



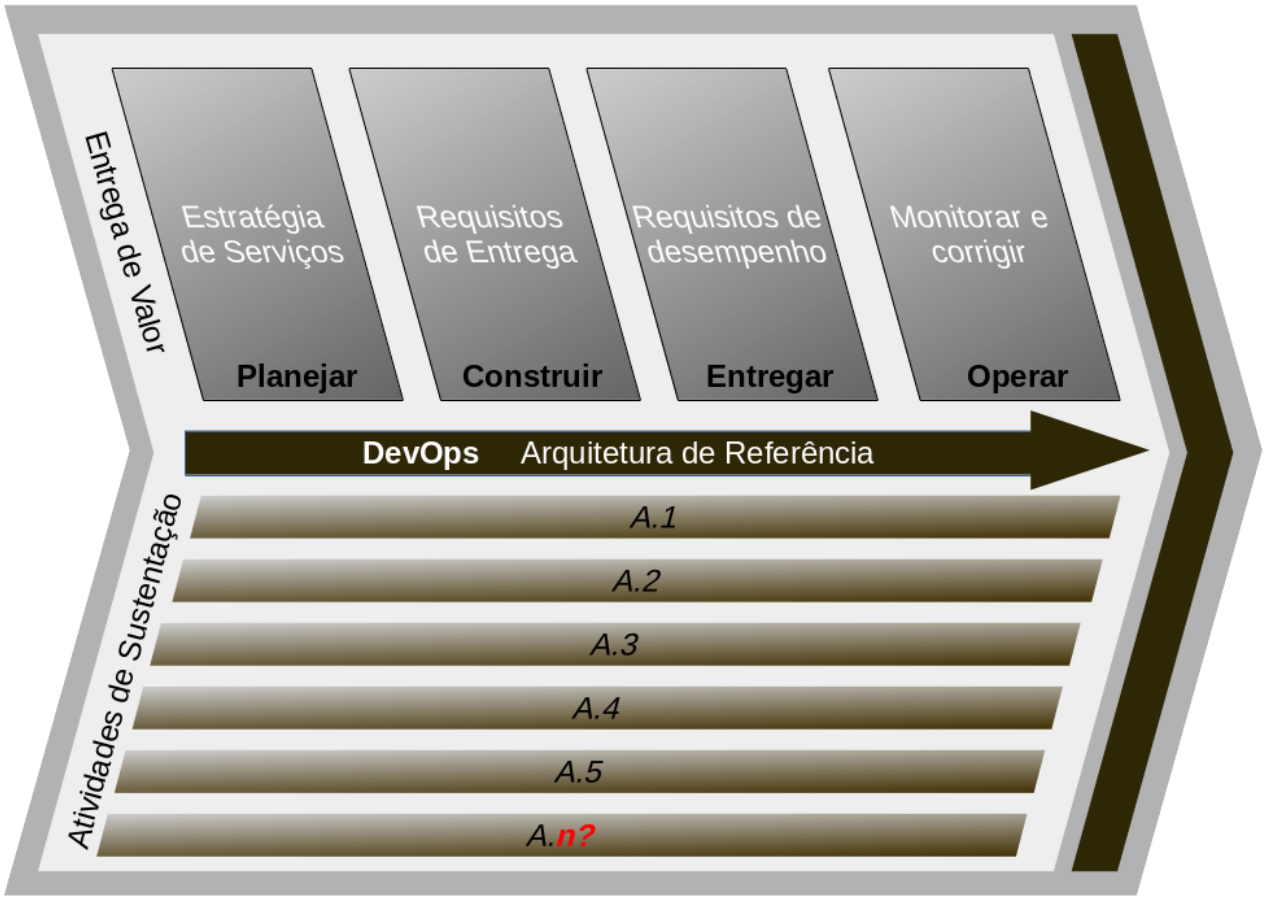
Fonte: The Lean IT house. Orzen, Paider

Melhorar e Adaptar



Melhorar e Adaptar

Cadeia de Valor em TI



Eficiência
Agilidade

Melhorar e Adaptar

Trabalhe em interações curtas
ajustadas ao negócio

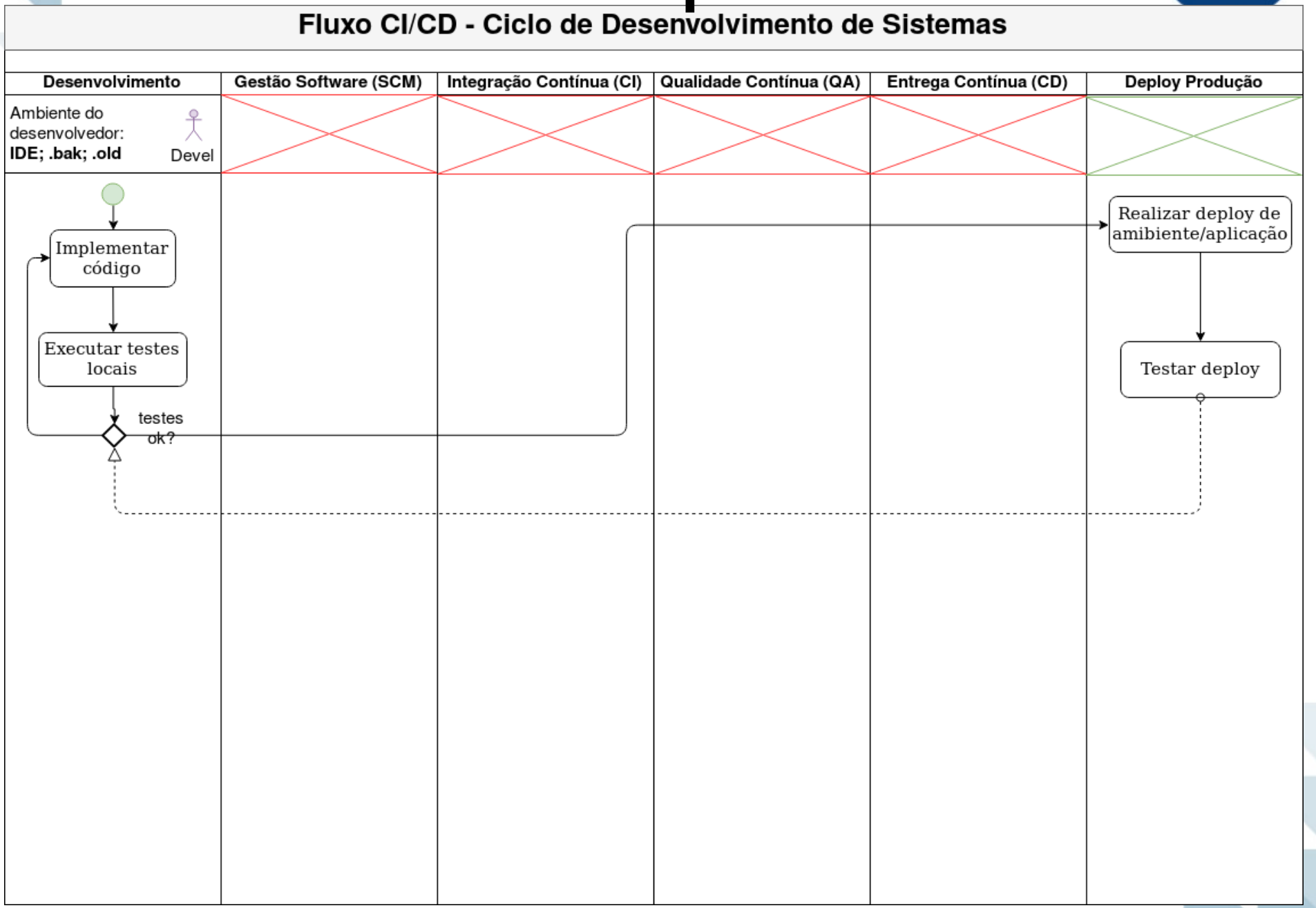
Cultura de experimentação

Melhorar e Adaptar

Feedback constante

Gestão apoiada em observação / monitoramento / KPI

Melhorar e Adaptar

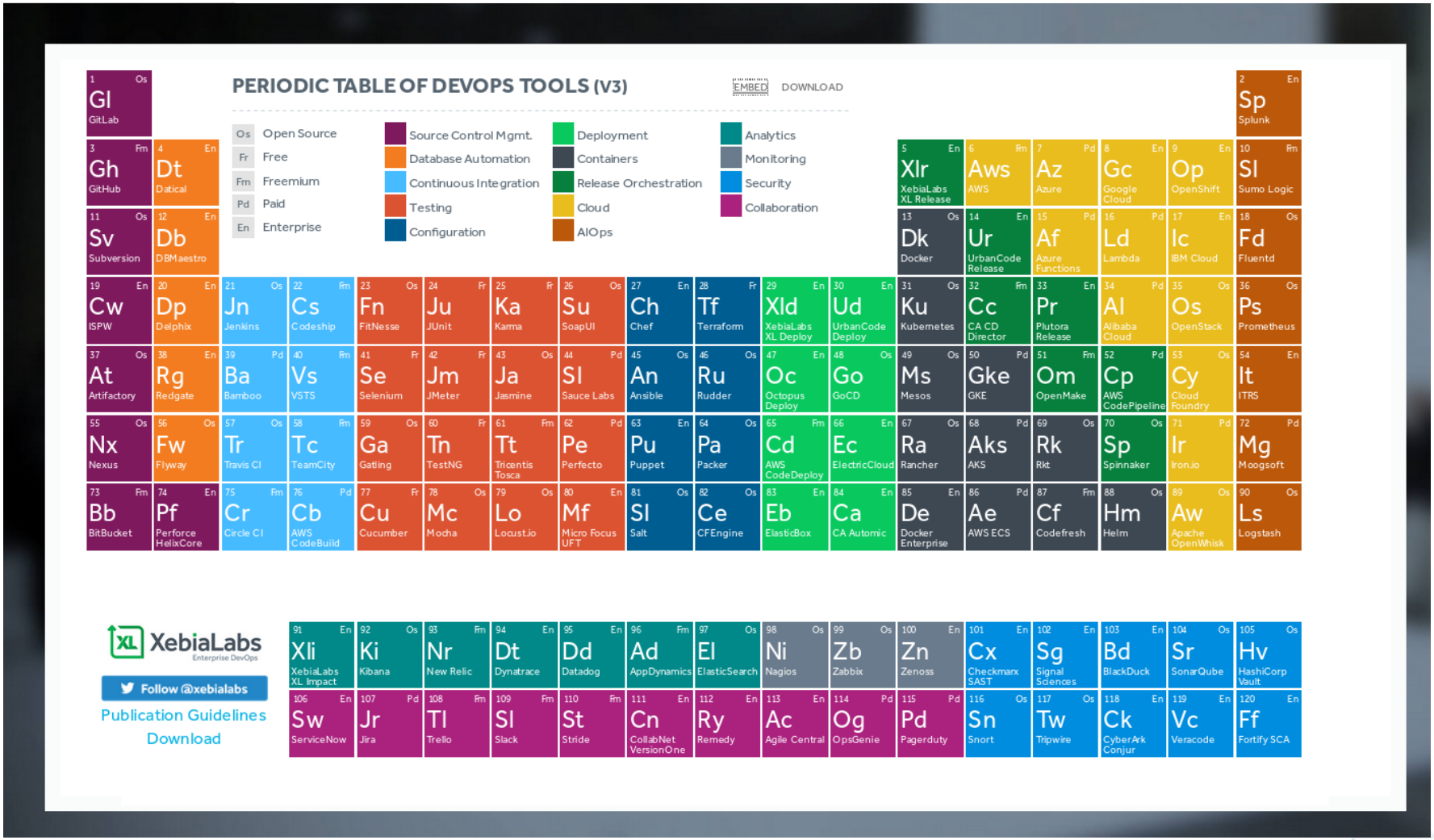


Melhorar e Adaptar

Empresa Brasil de Comunicação

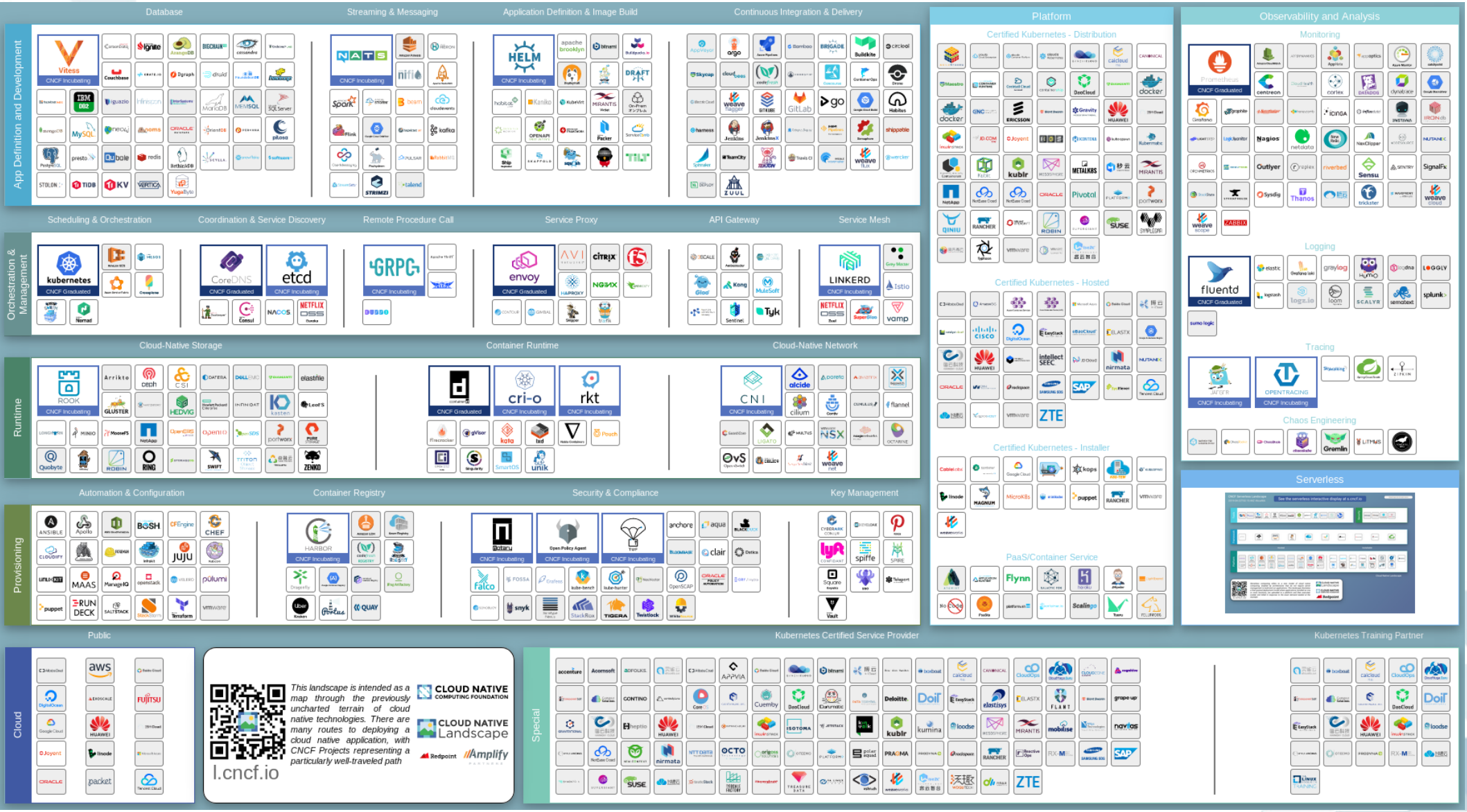
Tecnologias

Tecnologias



Periodic Table of DevOps Tools (v3) from @XebiaLabs https://xebialabs.com/periodic-table-of-devops-tools

Tecnologias



Cloud Native Landscape from @CloudNativefdn https://landscape.cncf.io/

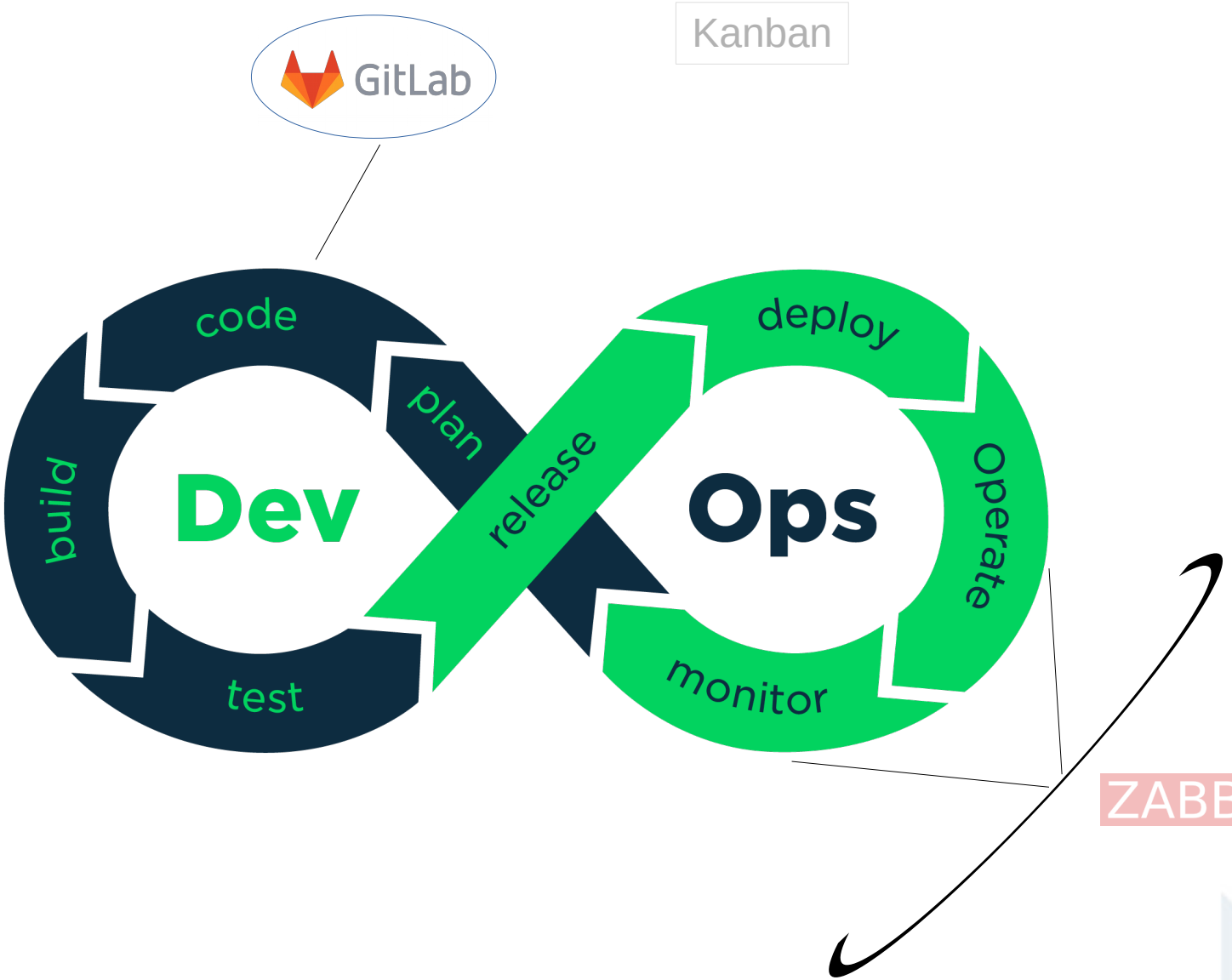
Keep C.A.L.M.S. and having fun!

Tecnologias

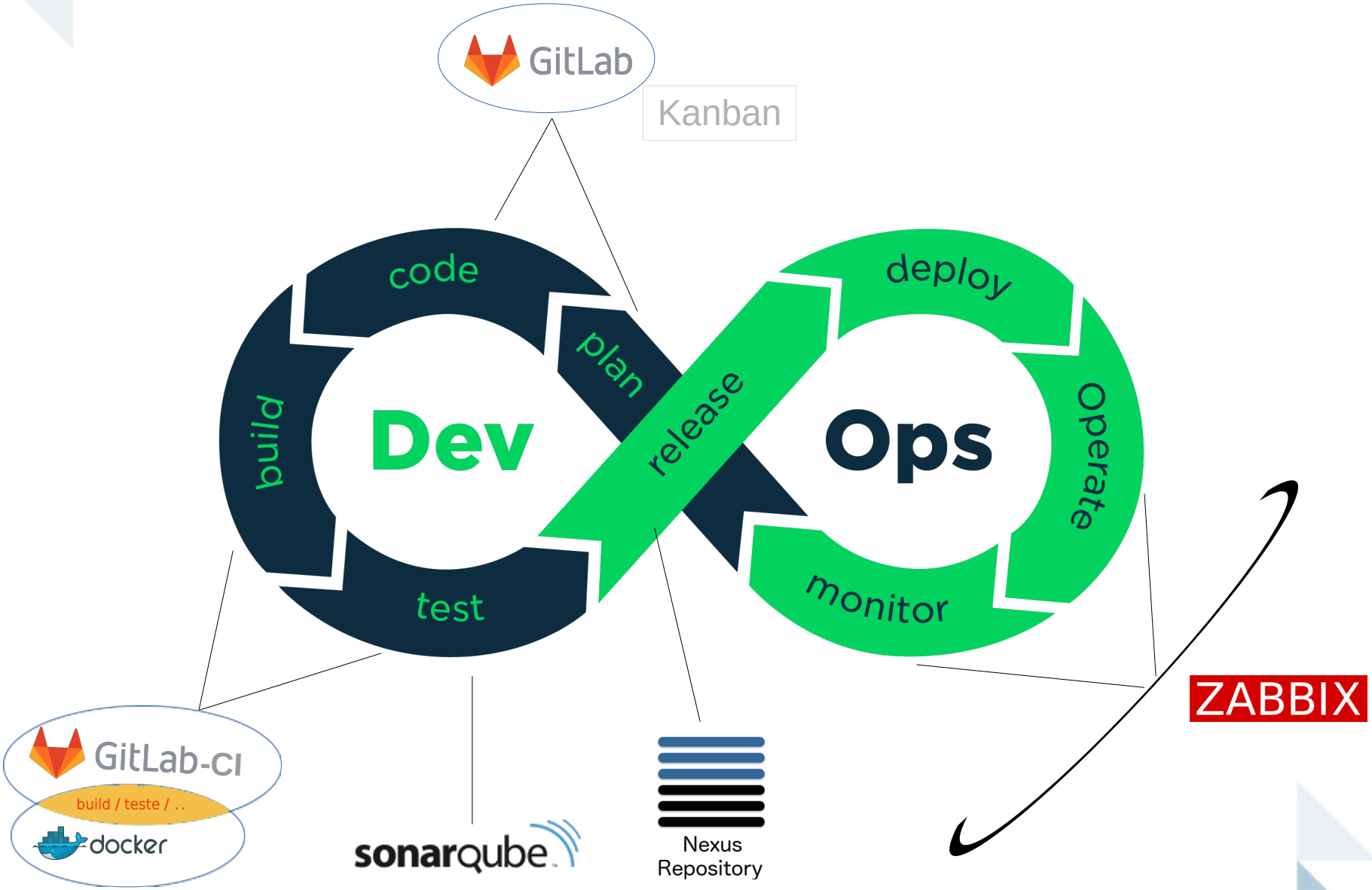
Adote a mais aderente a resolver o problema

Cultura de experimentação & domínio da equipe

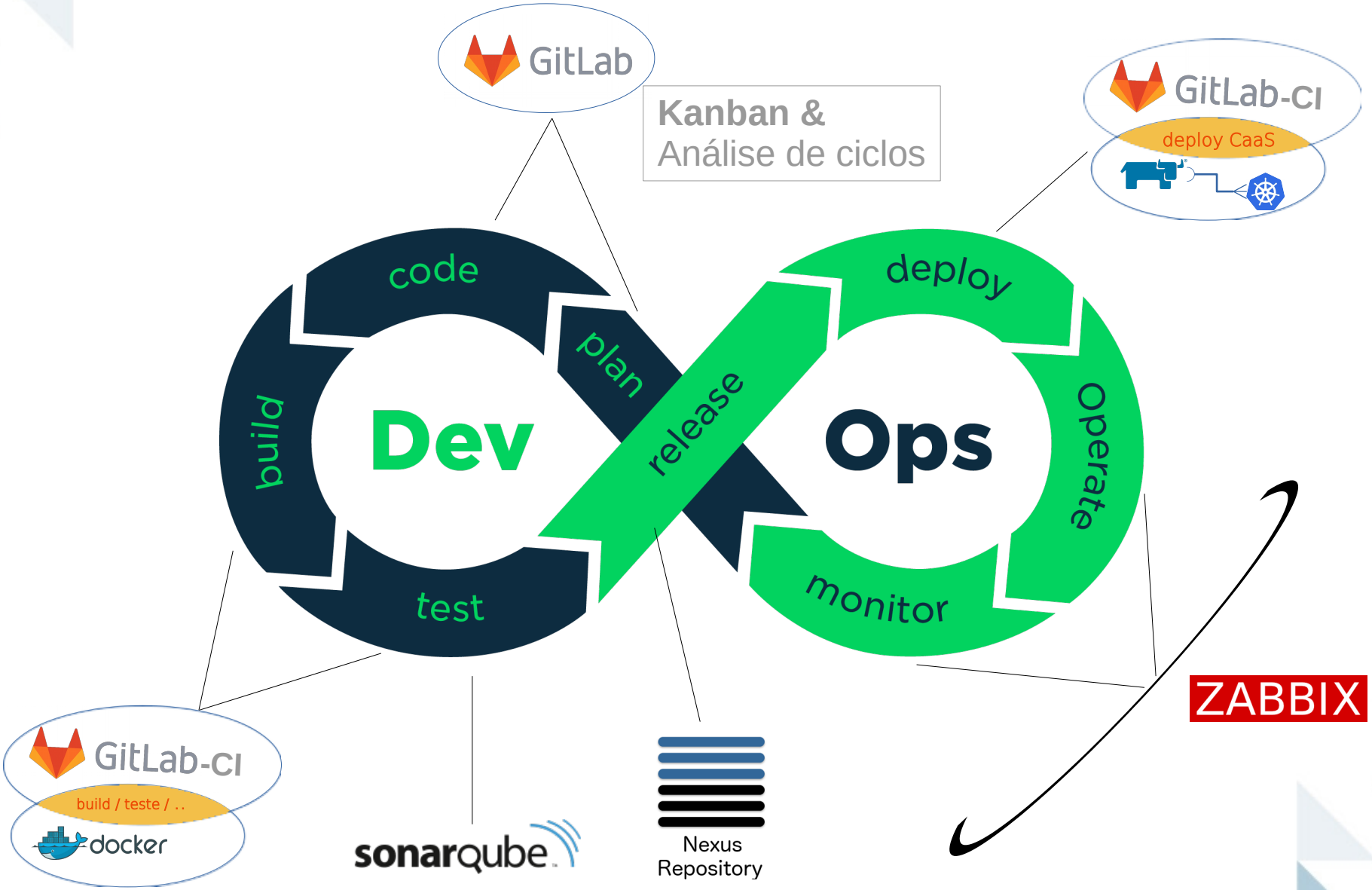
Tecnologias



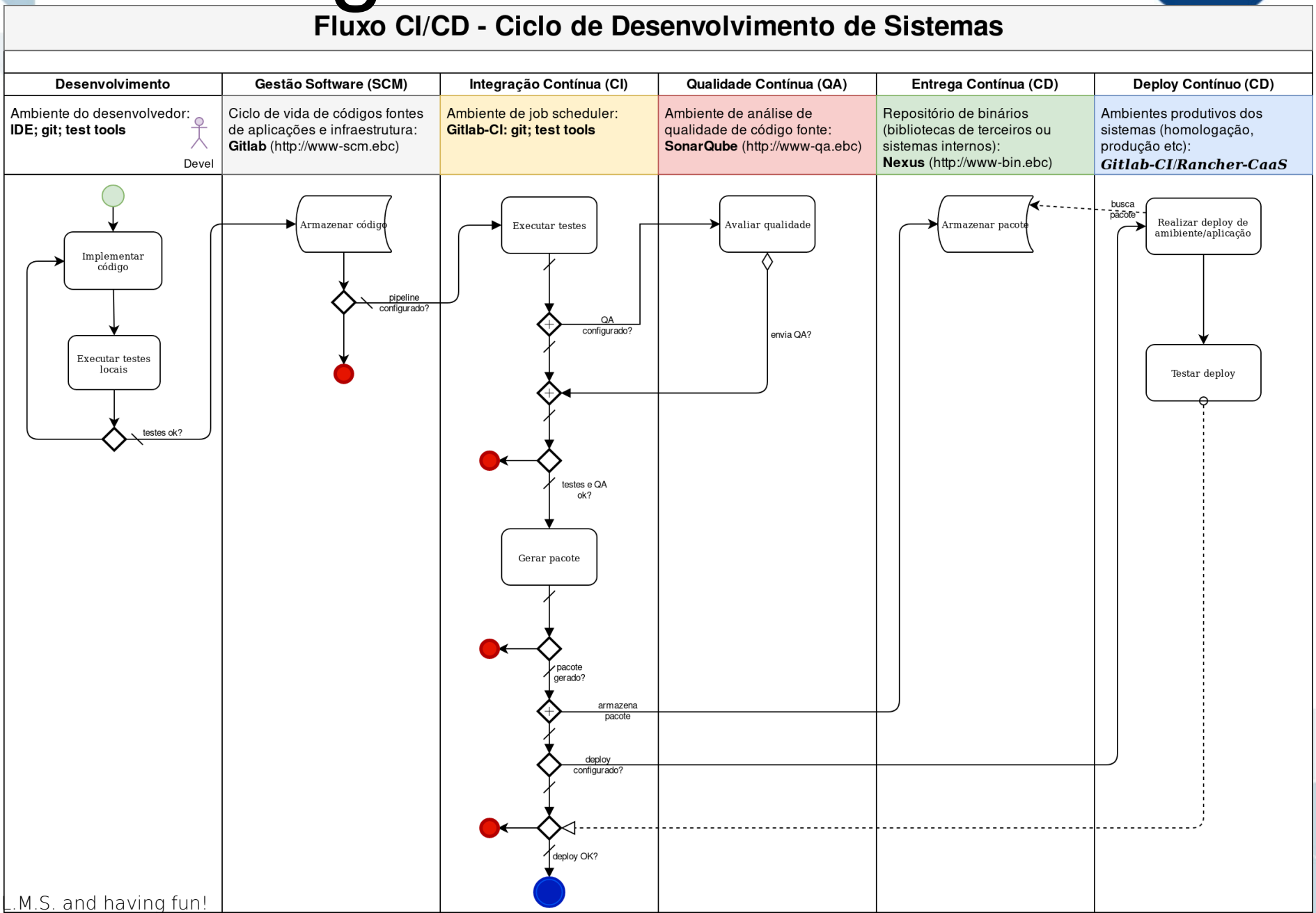
Tecnologias



Tecnologias



Tecnologias

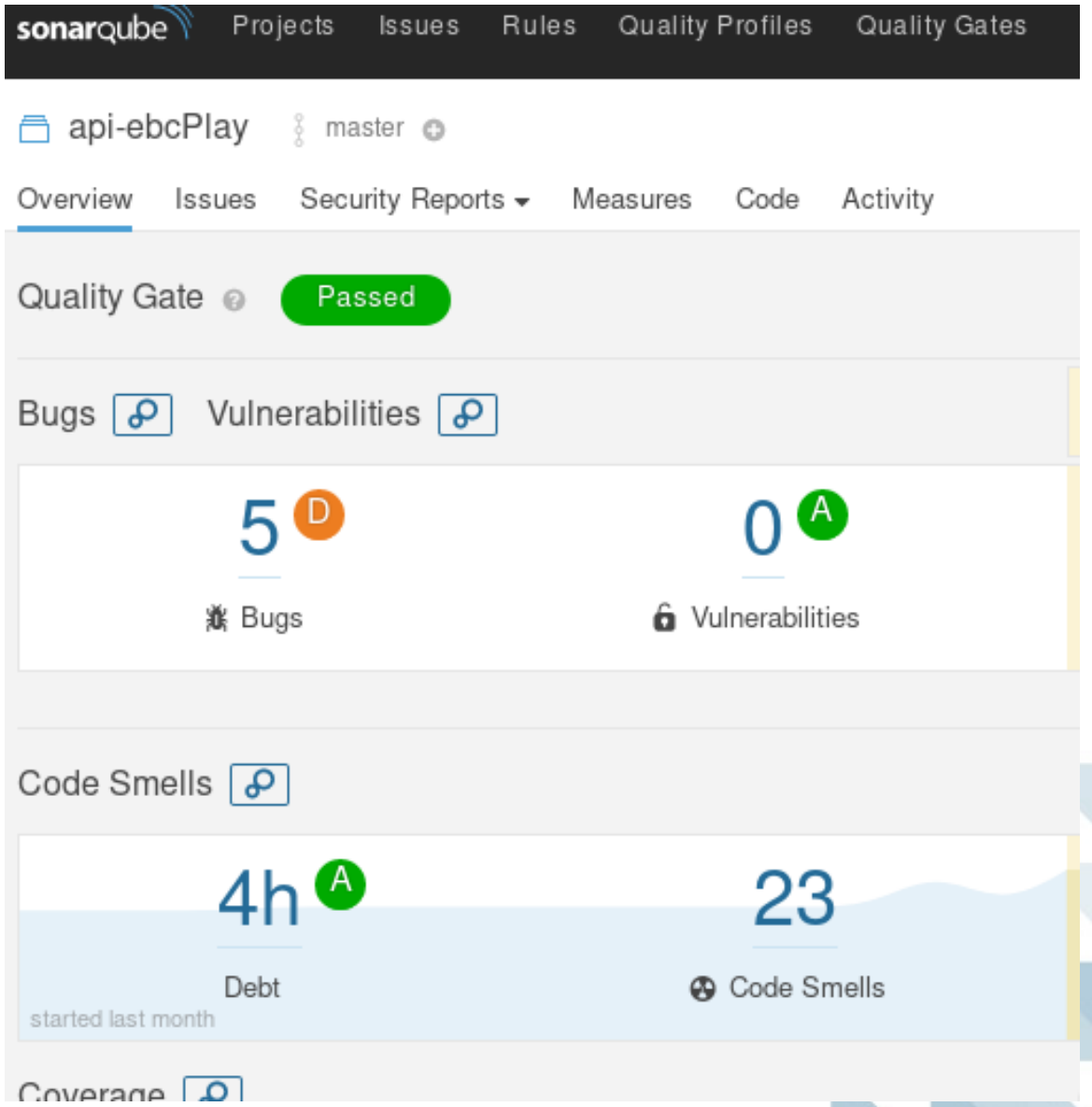


Tecnologias

- Gitlab (http://www-scm.ebc)
- SonarQube (http://www-qa.ebc)
- Nexus (http://www-bin.ebc)
- Gitlab-CI Runn
  - Docker
  - PostgreSQL
  - Rancher

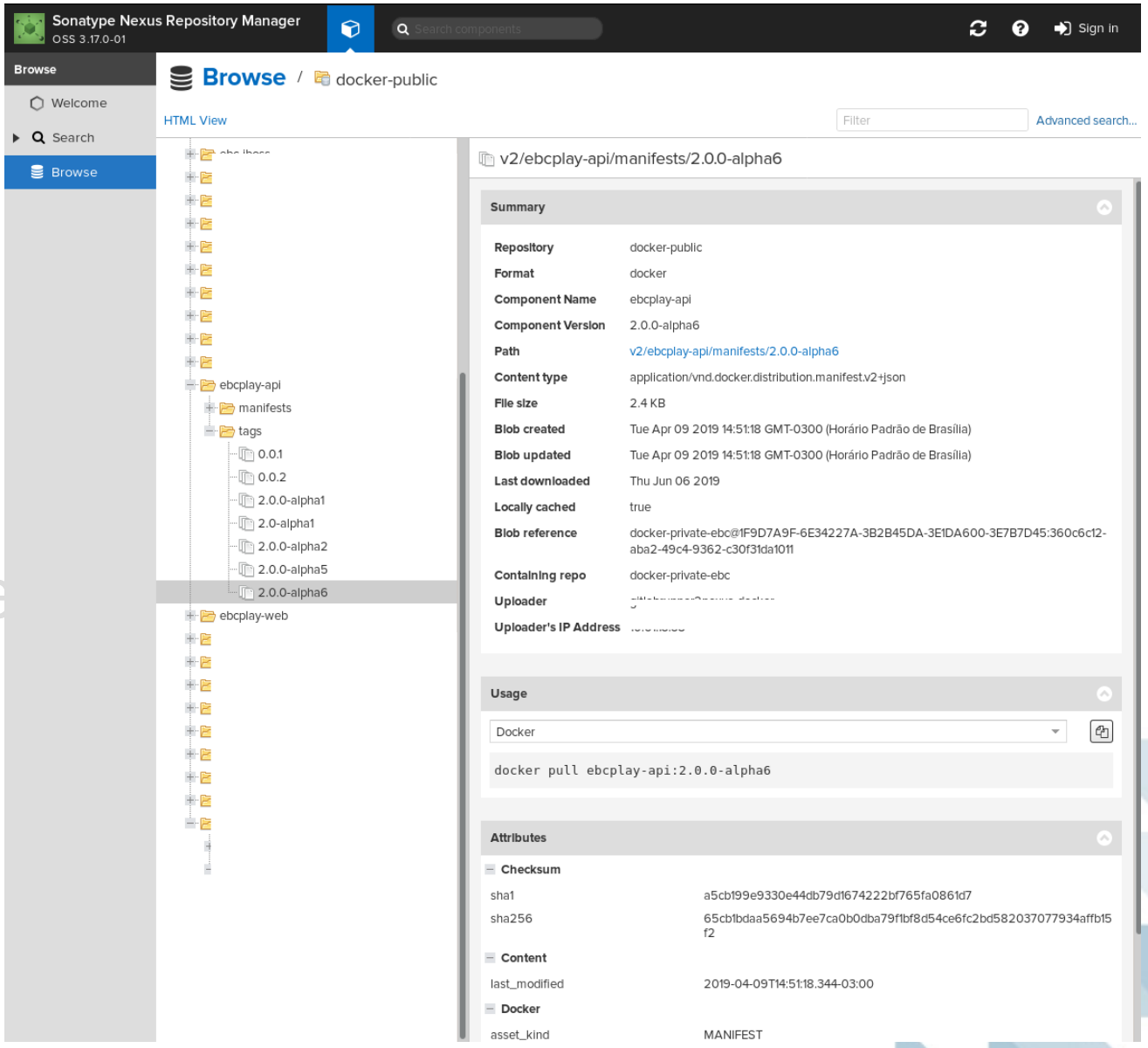
Tecnologias

- Gitlab
  (http://www-scm.ebc)
- SonarQube
  (http://www-qa.ebc)
- Nexus
  (http://www-bin.ebc)
- Gitlab-CI Runner
  - Docker
  - PostgreSQL
  - Rancher



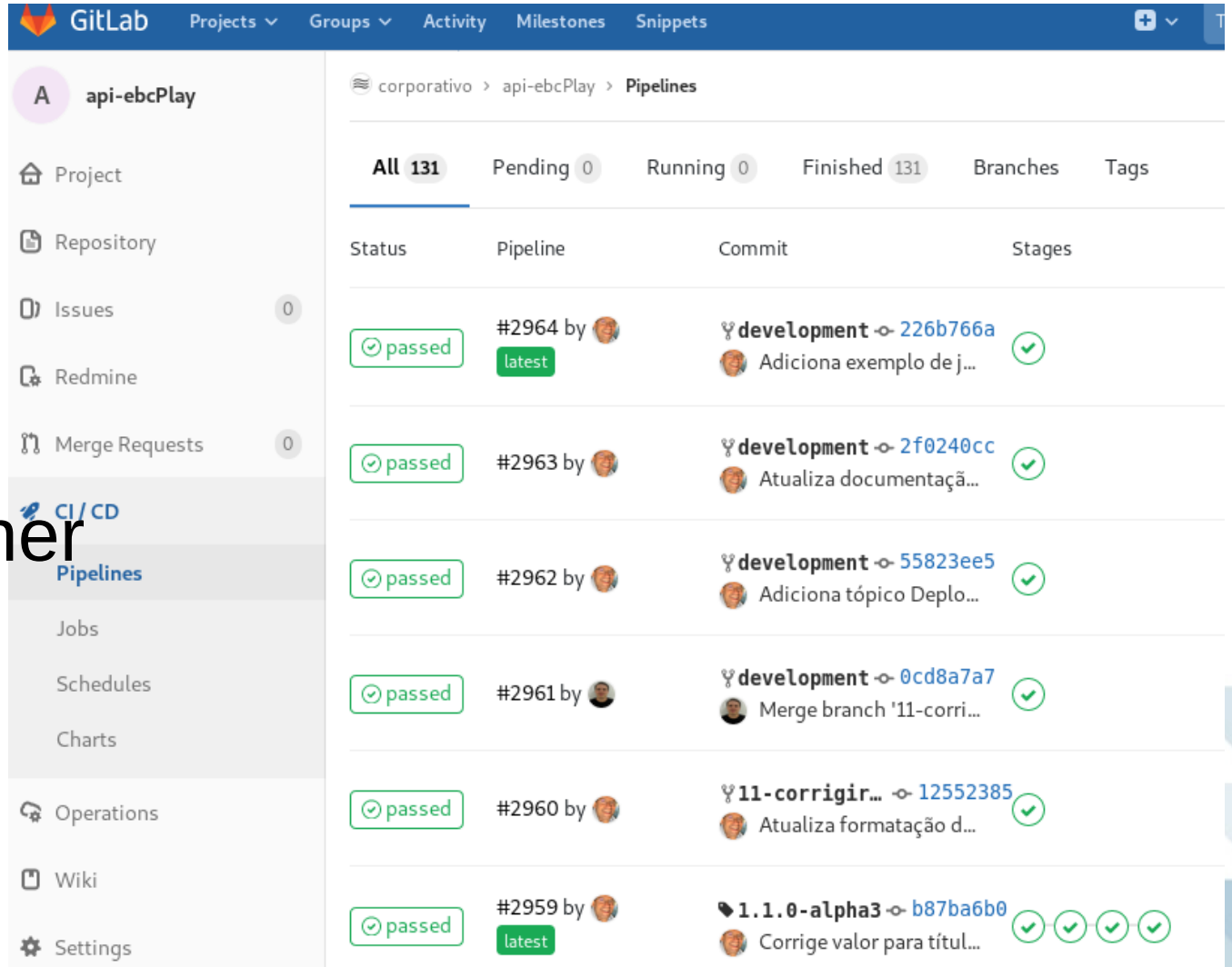
Tecnologias

- Gitlab (http://www-scm.ebc)
- SonarQube (http://www-qa.ebc)
- Nexus (http://www-bin.ebc)
- Gitlab-CI Runne
  - Docker
  - PostgreSQL
  - Rancher



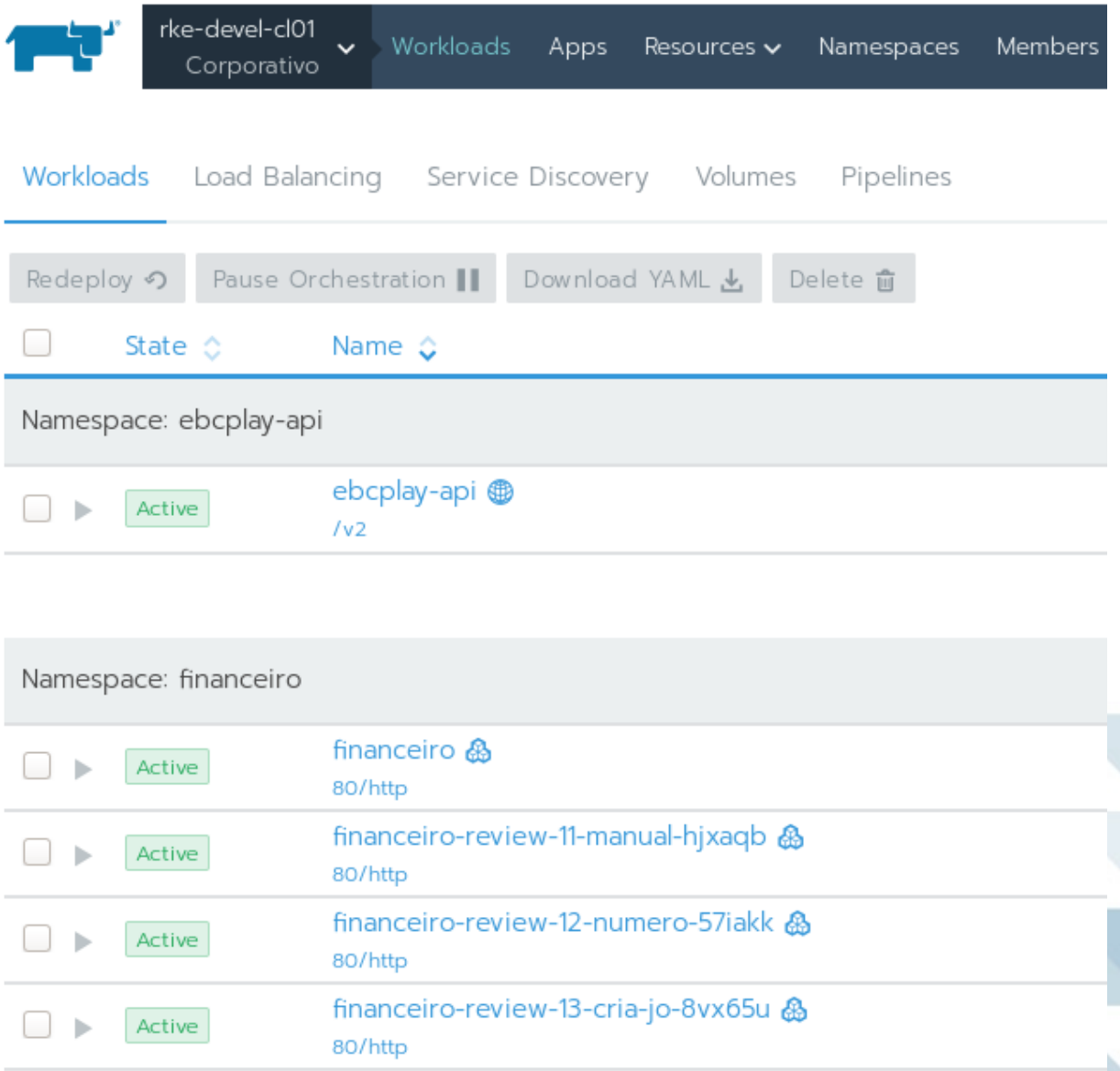
Tecnologias

- Gitlab (http://www-scm.ebc)
- SonarQube (http://www-qa.ebc)
- Nexus (http://www-bin.ebc)
- Gitlab-CI Runner
  - Docker
  - PostgreSQL
  - Rancher



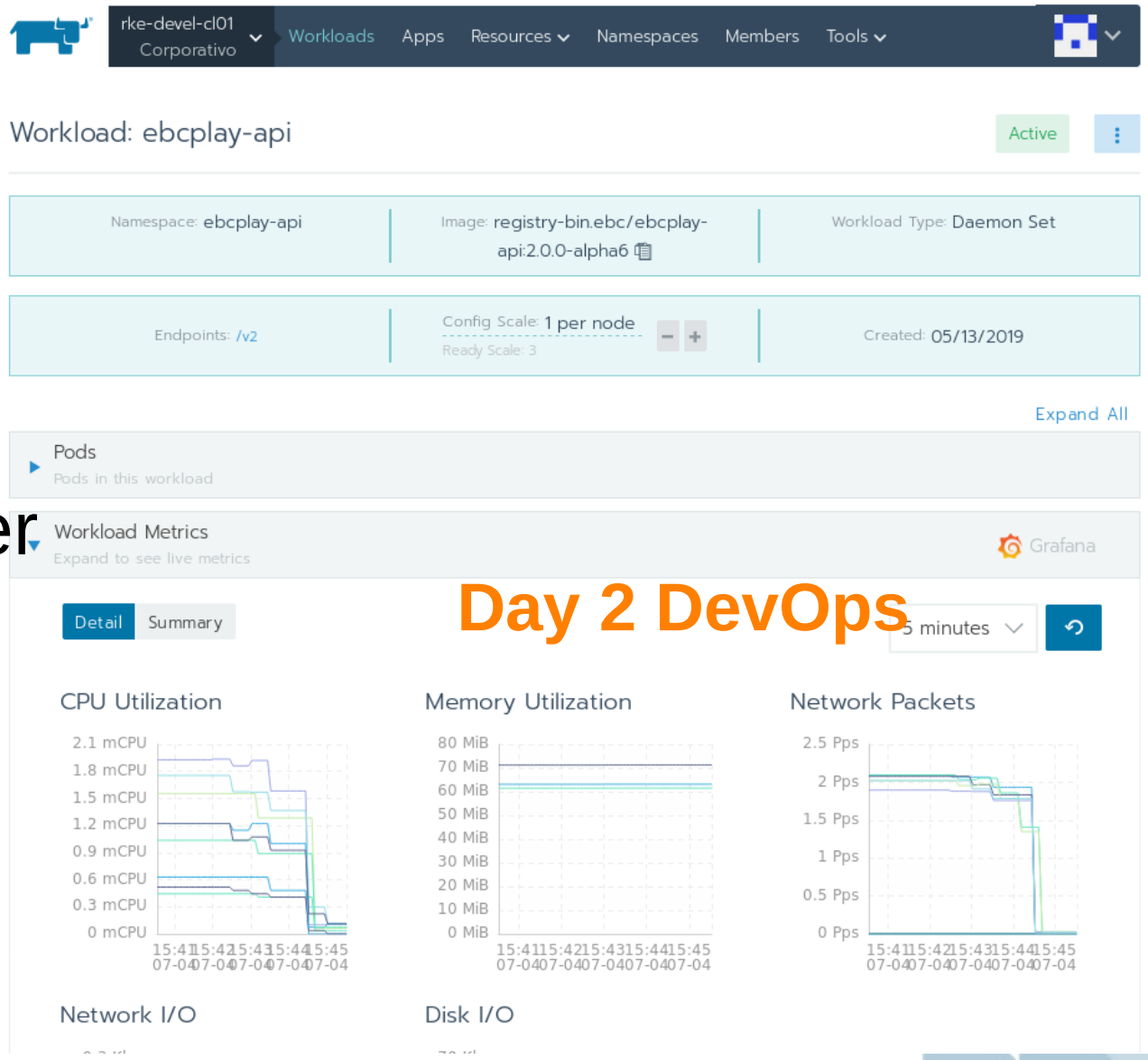
Tecnologias

- Gitlab
  (http://www-scm.ebc)
- SonarQube
  (http://www-qa.ebc)
- Nexus
  (http://www-bin.ebc)
- Gitlab-CI Runner
  - Docker
  - PostgreSQL
  - **Rancher**



Tecnologias

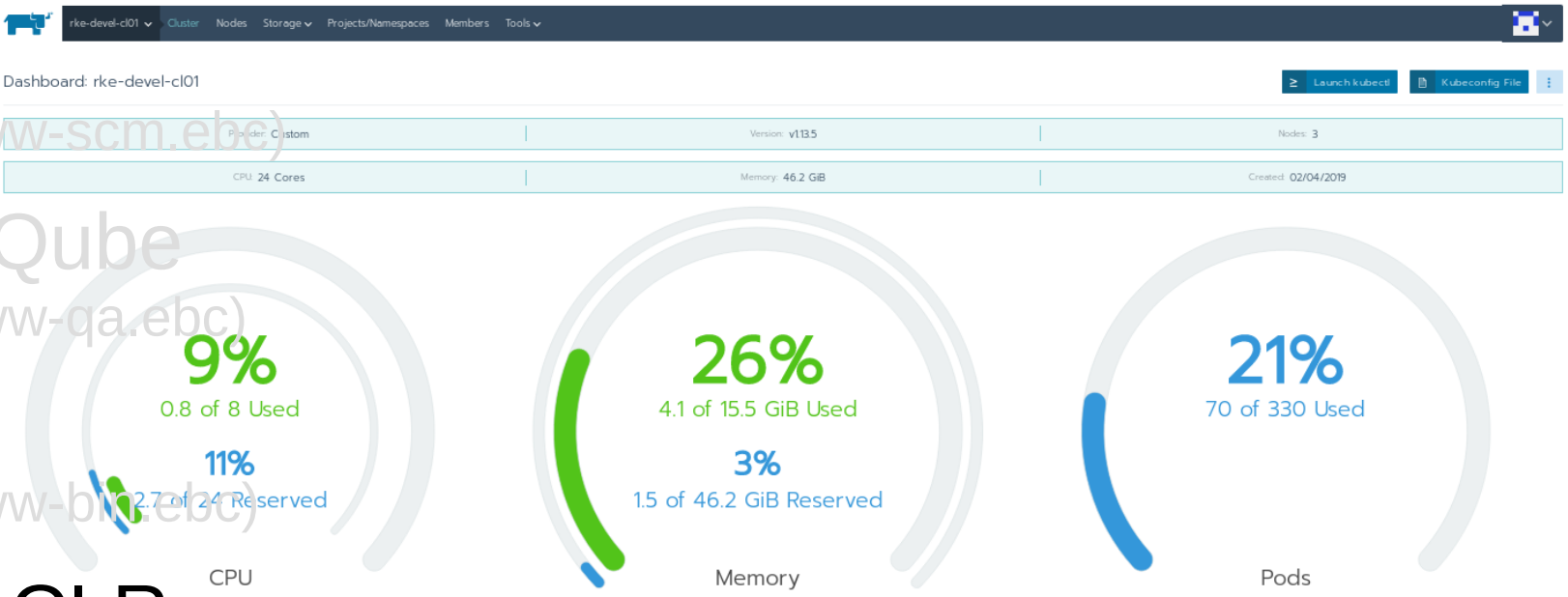
- Gitlab (http://www-scm.ebc)
- SonarQube (http://www-qa.ebc)
- Nexus (http://www-bin.ebc)
- Gitlab-CI Runner
  - Docker
  - PostgreSQL
  - **Rancher**

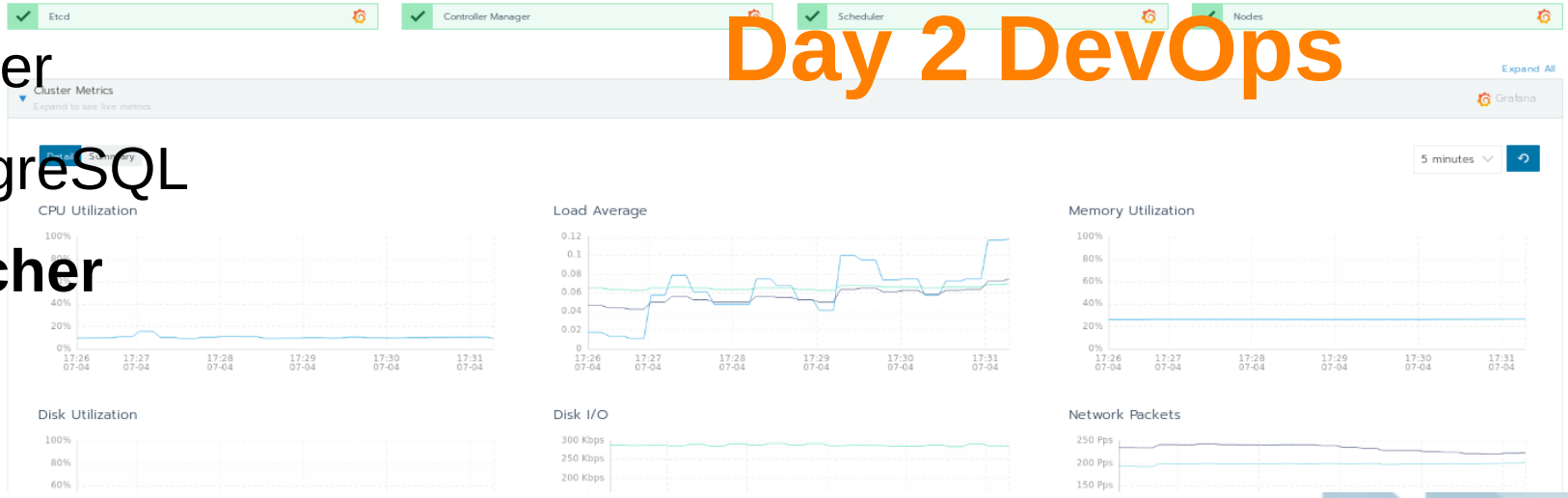


Day 2 DevOps

Tecnologias

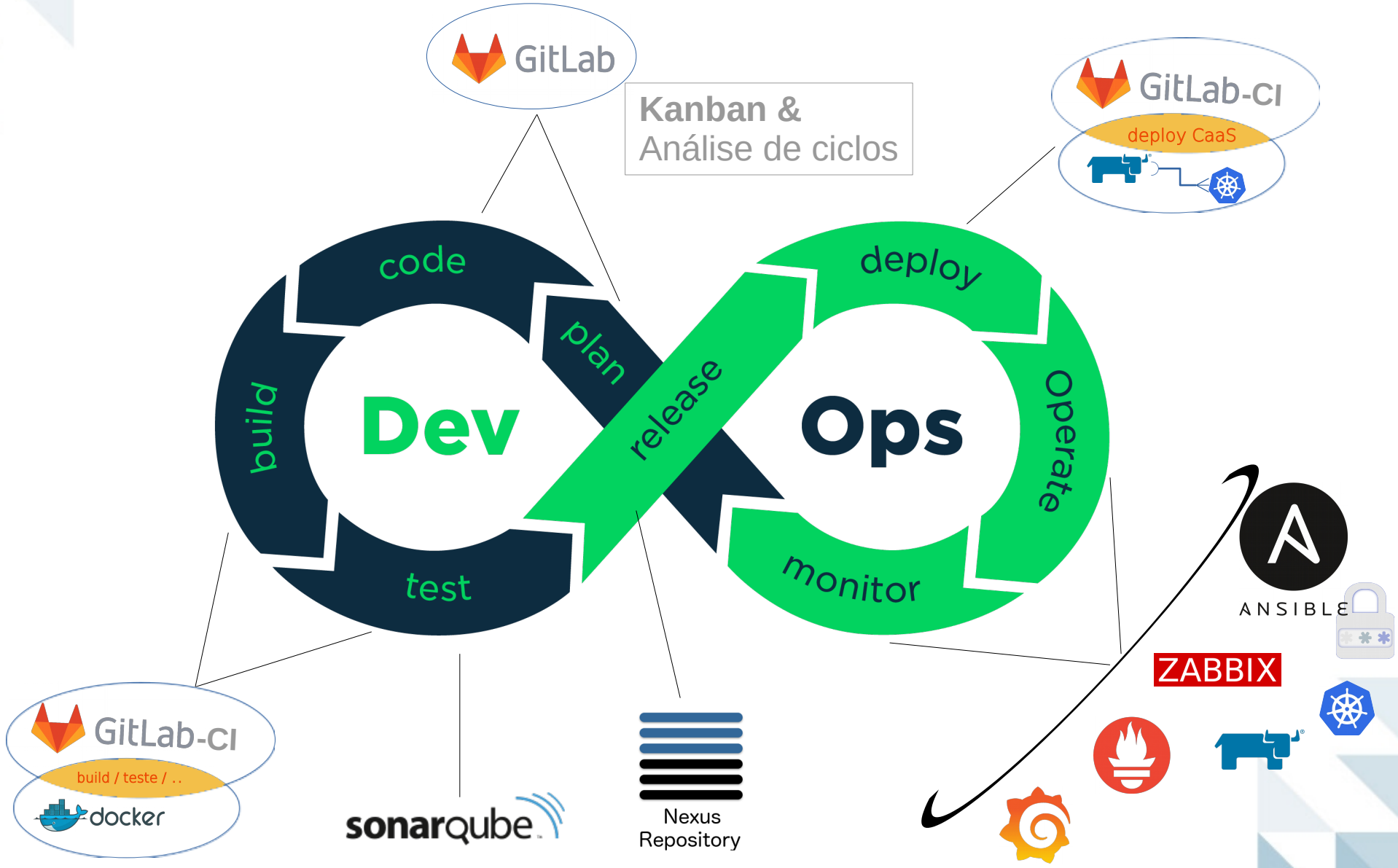
- Gitlab (http://www-scm.ebc)
- SonarQube (http://www-qa.ebc)
- Nexus (http://www-bin.ebc)
- Gitlab-CI Runner
  - Docker
  - PostgreSQL
  - **Rancher**



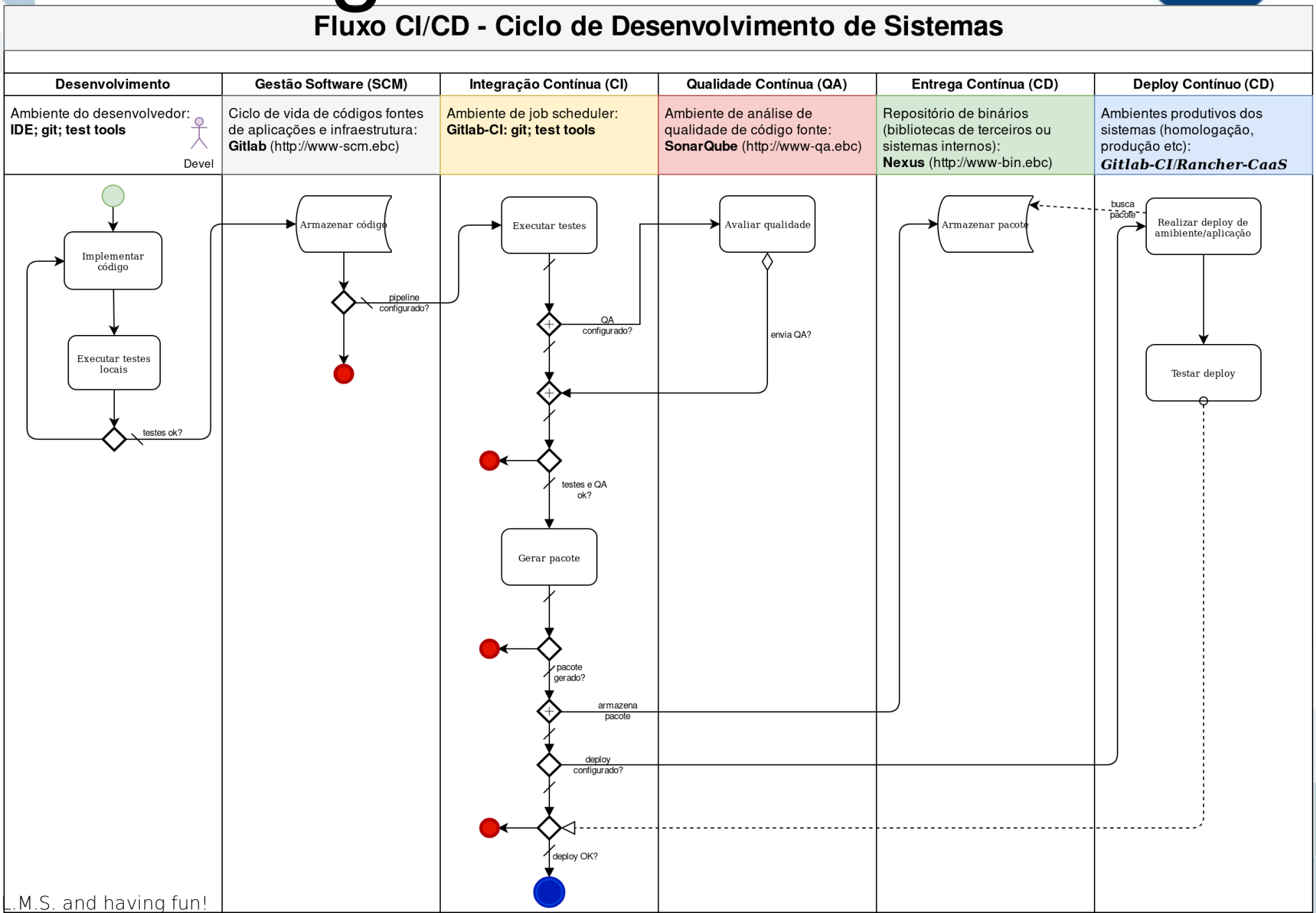


Day 2 DevOps

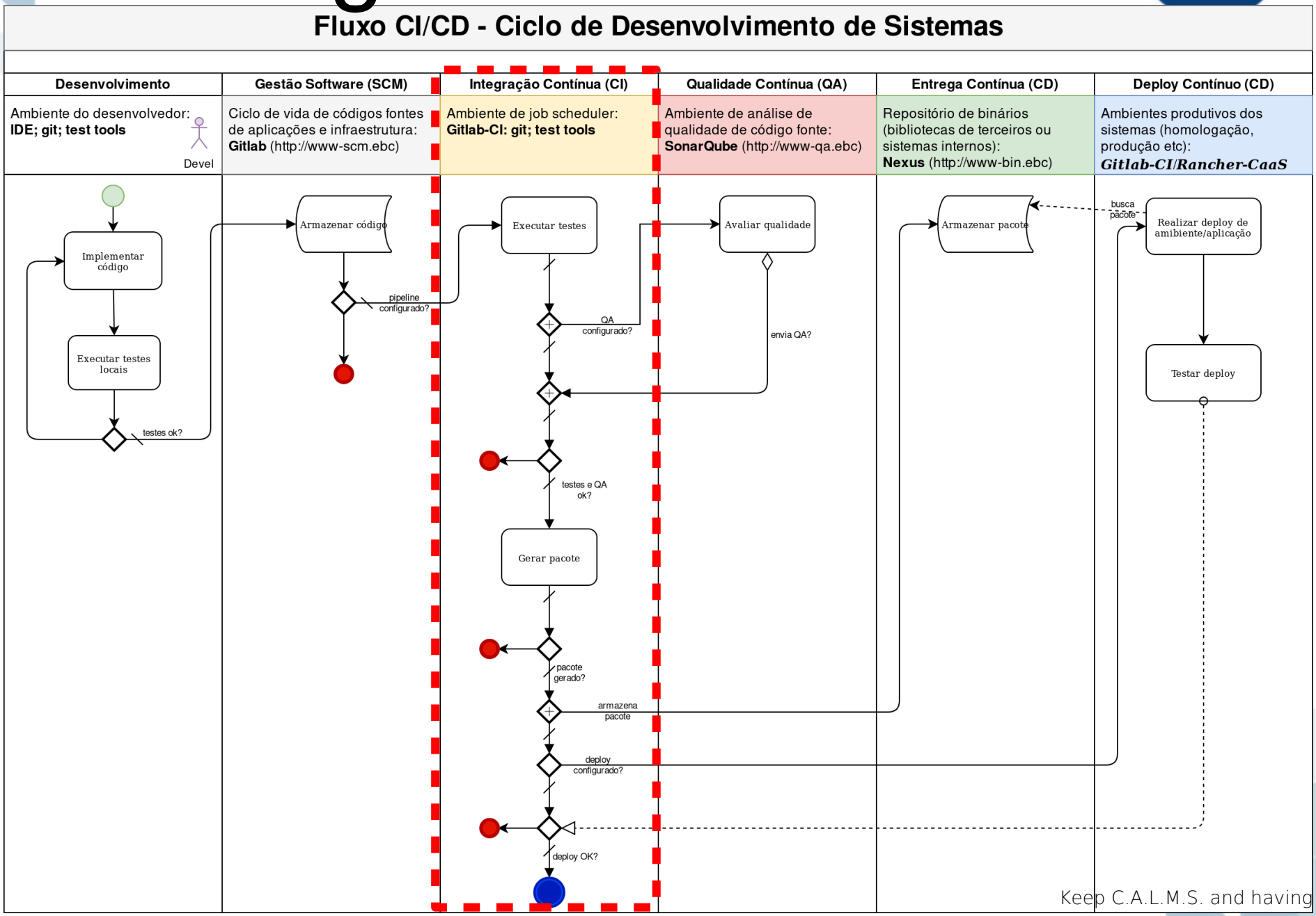
Tecnologias



Tecnologias



Tecnologias



Fluxo CI/CD aplicado

- DSL do pipeline como código
  - ➔ `.gitlab-ci.yml` + `docker`
- DSL deploy (rancher + k8s)
  - ➔ `deployment.yml`
    + `service.yml`
    + `loadbalancing.yml`

.gitlab-ci.yml 3.47 KB

```
image: registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1

services:
  - name: registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0
    alias: dind-pgserver.runner

variables:
  # Configura postgres service (https://hub.docker.com/_/postgres/)
  # Estes valores usados somente para o aprovisioar o serviço aqui na esteira
  POSTGRES_DB: db_prateleira
  POSTGRES_USER: prateleira
  POSTGRES_PASSWORD: prateleira
  PGPASSWORD: ${POSTGRES_PASSWORD}
  PORT: 5555
  URL: http://localhost
  DATABASE_HOST: dind-pgserver.runner
  DATABASE_NAME: ${POSTGRES_DB}
  DATABASE_USERNAME: ${POSTGRES_USER}
  DATABASE_PASSWORD: ${POSTGRES_PASSWORD}
  DATABASE_URL: postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@${DATABASE_HOST}/

stages:
  - build_test
  - code_QA
  - release
  - deploy

build_project_test:
  stage: build_test
  before_script:
    - mkdir -p logs
    - npm install && logs/npm-install.log
    - psql -h "dind-pgserver.runner" -U "$POSTGRES_USER" -d "$POSTGRES_DB" < test/se
    - ./node_modules/forever/bin/forever start -p ./logs -d -v ./bin/server.js
  script:
    - ./node_modules/forever/bin/forever list
    - npm test
    - ./node_modules/forever/bin/forever stop ./bin/server.js
  artifacts:
    when: always # on_failure | on_success
    paths:
      - logs/*.log
    expire_in: 1 day
  tags:
    - nodejs

SonarQube Continuous Code Quality:
  stage: code_QA
  before_script:
    - sed -e "s/0.0.0-VERSION/${CI_COMMIT_REF_NAME}/" -i package.json
  script:
    - sonar-scanner
        -Dsonar.host.url=${EBC_SONARQUBE_QA_URL}
        -Dsonar.projectKey=${PROJECT_SONAR_KEY}
        -Dsonar.login=${PROJECT_SONAR_TOKEN}
        -Dsonar.sources=.
  only:
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-alpha\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-beta\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-RC\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)$/
  tags:
    - nodejs

npm publish release:
  stage: release
  before_script:
    - sed -e "s/0.0.0-VERSION/${CI_COMMIT_REF_NAME}/" -i package.json
    - npm config set registry ${EBC_NEXUS_BIN_NPM_REGISTRY}
    - npm config set init.author.name ${GITLAB_USER_LOGIN}
    - npm config set init.author.email ${GITLAB_USER_EMAIL}
    - npm config set email ${GITLAB_USER_EMAIL}
    - npm config set always-auth=true
    - npm config set _auth $(echo -n "${EBC_NEXUS_BIN_DEPLOYMENT_USERNAME}:${EBC_NE
  script:
    - npm publish
  only:
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-alpha\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-beta\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-RC\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)$/
  except:
    - master

ansible deploy desenvolvimento:
  image: registry-bin.ebc/ebc-ansible:2.7-ebc-1.0
  stage: deploy
  variables:
    ANSIBLE_HOST_KEY_CHECKING: 'False'
  before_script:
    - sed -e "s/0.0.0-VERSION/${CI_COMMIT_REF_NAME}/" -i package.json
    - ansible-playbook --syntax-check deploy/ansible/playbook.yml
    - ansible all -m ping -i deploy/ansible/inventory/desenvolvimento.yml --become
                  --extra-vars "${PROJECT_DEVELENV_ANSIBLE_EXTRAVARS}"
  script:
    - ansible-playbook deploy/ansible/playbook.yml
                       -i deploy/ansible/inventory/desenvolvimento.yml
                       --become
                       --extra-vars "${PROJECT_DEVELENV_ANSIBLE_EXTRAVARS}"
  environment:
    name: desenvolvimento
  only:
    # somente branches ou tags (ex: 1.0.0-alpha1)
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-alpha\d+$/
```

Fluxo CI/CD aplicado

- DSL do pipeline como código

  `.gitlab-ci.yml` + docker

```yaml
image: registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1

services:
  - name: registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0
    alias: dind-pgserver.runner

variables:
  # Configura postgres service (https://hub.docker.com/_/postgres/)
  # Estes valores usados somente para o aprovisioar o serviço aqui na
  POSTGRES_DB: db_prateleira
  POSTGRES_USER: prateleira
  POSTGRES_PASSWORD: prateleira
  PGPASSWORD: ${POSTGRES_PASSWORD}
  PORT: 5555
  URL: http://localhost
  DATABASE_HOST: dind-pgserver.runner
  DATABASE_NAME: ${POSTGRES_DB}
  DATABASE_USERNAME: ${POSTGRES_USER}
  DATABASE_PASSWORD: ${POSTGRES_PASSWORD}
  DATABASE_URL: postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@${

stages:
  - build_test
  - code QA
  - release
  - deploy
```

```yaml
build_project_test:
  stage: build_test
  before_script:
    - mkdir -p logs
    - npm install &> logs/npm-install.log
    - psql -h "dind-pgserver.runner" -U "$POSTGRES_USER" -d "$POSTGRES_DB" < test/sql/massadetest
    - ./node_modules/forever/bin/forever start -p ./logs -d -v ./bin/server.js
  script:
    - ./node_modules/forever/bin/forever list
    - npm test
    - ./node_modules/forever/bin/forever stop ./bin/server.js
  artifacts:
    when: always # on_failure | on_success
    paths:
      - logs/*.log
    expire_in: 1 day
  tags:
    - nodejs

SonarQube Continuous Code Quality:
  stage: code QA
  before_script:
    - sed -e "s/0.0.0-VERSION/${CI_COMMIT_REF_NAME}/" -i package.json
  script:
    - sonar-scanner
        -Dsonar.host.url=${EBC_SONARQUBE_QA_URL}
        -Dsonar.projectKey=${PROJECT_SONAR_KEY}
        -Dsonar.login=${PROJECT_SONAR_TOKEN}
        -Dsonar.sources=.
  only:
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-alpha\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-beta\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-RC\d+$/
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)$/
  tags:
    - nodejs
```

Fluxo CI/CD aplicado

- DSL do pipeline como código

  `.gitlab-ci.yml` + docker

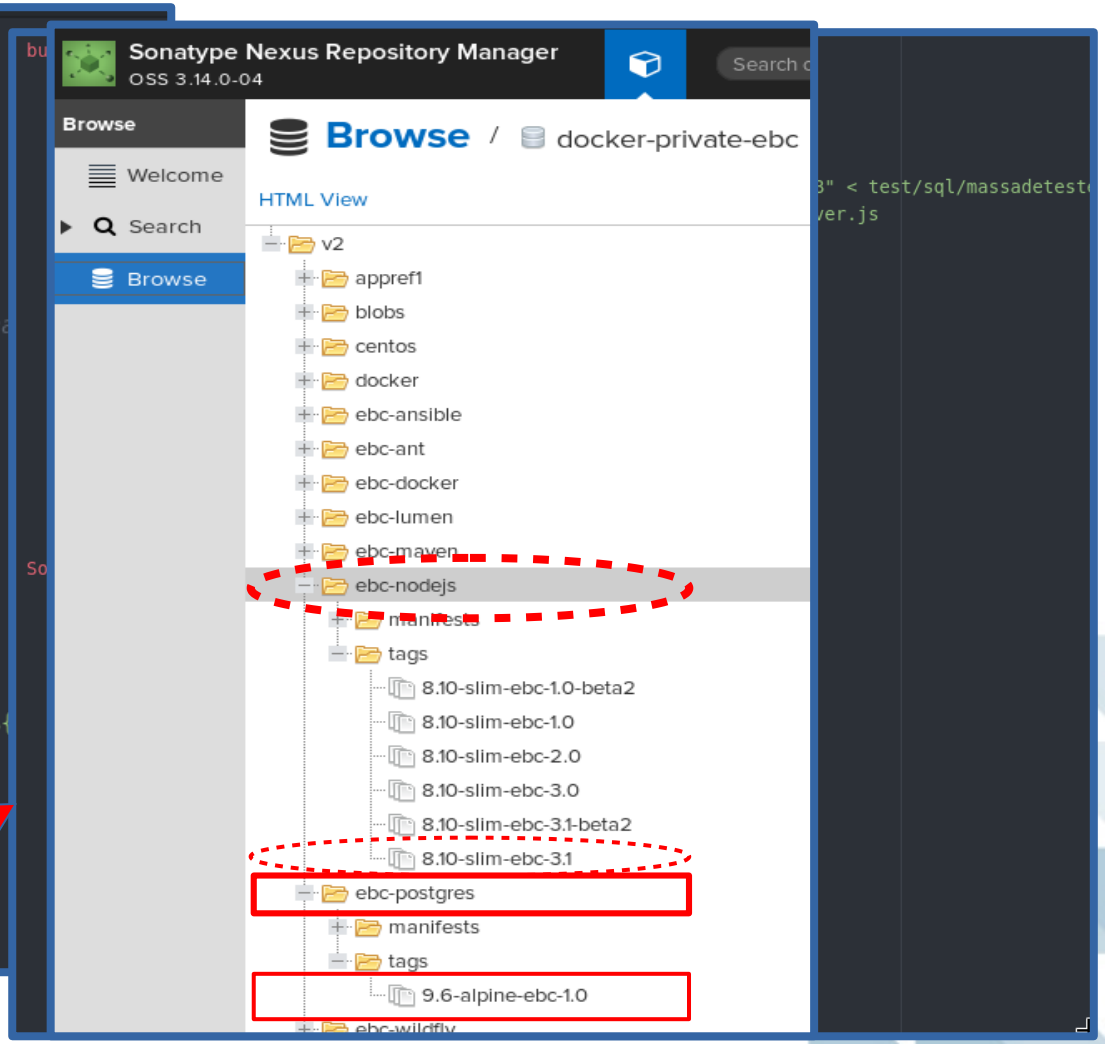
```
image: registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1

services:
  - name: registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0
    alias: dind-pgserver.runner

variables:
  # Configura postgres service (https://hub.docker.com/_/postgres/)
  # Estes valores usados somente para o aprovisioar o serviço aqui na
  POSTGRES_DB: db_prateleira
  POSTGRES_USER: prateleira
  POSTGRES_PASSWORD: prateleira
  PGPASSWORD: ${POSTGRES_PASSWORD}
  PORT: 5555
  URL: http://localhost
  DATABASE_HOST: dind-pgserver.runner
  DATABASE_NAME: ${POSTGRES_DB}
  DATABASE_USERNAME: ${POSTGRES_USER}
  DATABASE_PASSWORD: ${POSTGRES_PASSWORD}
  DATABASE_URL: postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@$

stages:
  - build_test
  - code QA
  - release
  - deploy
```

em tempo de CI, instancia serviço de BD em docker



Fluxo CI/CD aplicado

- DSL do pipeline como código

  ➔ `.gitlab-ci.yml` + `docker`

Testes: Preparação

```
Running with gitlab-runner 11.1.1 (7bd2a384)
  on git-runner01-prod01.ebc-corporativo-runner01 f82629a1
Using Docker executor with image registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1 ...
Starting service registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0 ...
Pulling docker image registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0 ...
Using docker image sha256:ca7199e23213b124274f2625034ac9e399ca1040ce4b9362220c336718dee3fb for registry-bin.ebc/ebc-postgres:9.6-alpine-ebc-1.0 ...
Waiting for services to be up and running...
Pulling docker image registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1 ...
Using docker image sha256:429e7adc18fe9c94b2831b85d31cca9a3ee2896bebf5d720045903d3514f4475 for registry-bin.ebc/ebc-nodejs:8.10-slim-ebc-3.1 ...
Running on runner-f82629a1-project-288-concurrent-0 via git-runner01-prod01.ebc...
Cloning repository...
Cloning into '/builds/corporativo/api-ebcPlay'...
Checking out 18ff5be8 as development...
Skipping Git submodules setup
$ mkdir -p logs
$ npm install &> logs/npm-install.log
$ psql -h "dind-pgserver.runner" -U "$POSTGRES_USER" -d "$POSTGRES_DB" < test/sql/massadeteste.sql > logs/sql-test.log
$ ./node_modules/forever/bin/forever start -p ./logs -d -v ./bin/server.js
warn:    --minUptime not set. Defaulting to: 1000ms
warn:    --spinSleepTime not set. Your script will exit if it does not stay up for at least 1000ms
info:    Forever processing file: ./bin/server.js
$ ./node_modules/forever/bin/forever list
info:    Forever processes running
data:        uid  command             script        forever pid id logfile       uptime
data:    [0] sNX5 /usr/local/bin/node bin/server.js 56      68     logs/sNX5.log 0:0:0:0.125
```

Fluxo CI/CD aplicado

Testes:Execução

```
$ npm test

> api-ebcPlay@0.0.0-VERSION test /builds/corporativo/api-ebcPlay
> ./node_modules/mocha/bin/mocha



  Teste API play.ebc.com.br
    ✓ Deve retornar ao menos 10 conteudos (542ms)
    ✓ Deve retornar pelo menos 10 conteudos usando o termo (1685ms)
    ✓ Retornar os conteudos do tema de ID 3(Entretenimento) - Deve receber ao menos 3 conteudo
    ✓ consultar um conteudo especifico
    ✓ endpoint /populares deve retornar pelo menos 10 conteudos (436ms)
    ✓ endpoint /priorizados deve retornar pelo menos 1 conte0do (93ms)
    ✓ endpoint /conteudos/id/episodios deve retornar pelo menos 3 episódios
nr_segundos_exibidos 6331
st_exibicao 2
    ✓ fazer um post de conteudos em exibicao (39ms)

  Teste API play.ebc.com.br /streaming
    ✓ Endpoint streaming deve responder status 200
    ✓ Endpoint streaming deve retornar pelo menos um conte0do
    ✓ Streaming sort nm_plataforma e order desc nome da plataforma e TV Brasil


  11 passing (3s)

$ ./node_modules/forever/bin/forever stop ./bin/server.js
info:    Forever stopped process:
    uid  command             script        forever pid id logfile       uptime
[0] sNX5 /usr/local/bin/node bin/server.js 56      68     logs/sNX5.log 0:0:0:5.284
Uploading artifacts...
logs/*.log: found 3 matching files
Uploading artifacts to coordinator... ok            id=3707 responseStatus=201 Created token=piksFzKy
Job succeeded
```

Fluxo CI/CD aplicado

- DSL deploy `(.gitlab-ci.yml)`

`(Rancher+K8s) deployment.yml + service.yml + ingress-lb.yml`

```
deploy::review da aplicação no caas:
  stage: deploy
  image: registry-bin.ebc/ebc-rkecli:3.0
  before_script:
    - rancher login --skip-verify
                    --token ${EBC_CAAS_TOKEN}
                    --context ${PROJECT_CAAS_DEVEL_CONTEXT}
                     ${EBC_CAAS_URL}

    - rancher kubectl delete --ignore-not-found=true --namespace ebcplay-api
                             deployment ebcplay-api-$CI_ENVIRONMENT_SLUG
  script:
    - rancher kubectl apply --namespace ebcplay-api --filename deploy/k8s/deployment.yml
    - rancher kubectl apply --namespace ebcplay-api --filename deploy/k8s/service.yml
    - rancher kubectl apply --namespace ebcplay-api --filename deploy/k8s/loadbalancing.yml
  environment:
    name: review/$CI_COMMIT_REF_NAME
    url: http://ebcplay-api-$CI_ENVIRONMENT_SLUG.local
  only:
    - branches
  except:
    - master
    - tags
```

```
deploy::staging da api no caas:
  stage: deploy
  image: registry-bin.ebc/ebc-rkecli:1.0.0
  environment:
      name: staging
      url: http://api.local/v2
  before_script:
    - rancher login --skip-verify --token ${EBC_CAAS_TOKEN}
                    --context ${PROJECT_CAAS_CONTEXT} ${EBC_CAAS_URL}
  script:
    - rancher kubectl apply --namespace ebcplay-api
                            --filename deploy/k8s/deployment.yml
  only:
    # somente branches ou tags (ex: 1.0.0-alpha1)
    - /^(?:(\d+)\.)?(?:(\d+)\.)?(\*|\d+)-alpha\d+$/
```

Fluxo CI/CD aplicado

- DSL deploy `(.gitlab-ci.yml)`

`(Rancher+K8s) deployment.yml + service.yml + ingress-lb.yml`

```yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
  labels:
    app: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
spec:
  replicas: 1
  selector:
    matchLabels:
      app: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
  template:
    metadata:
      labels:
        app: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
    spec:
      containers:
      - name: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
        image: registry-bin.ebc/ebcplay-api:{{CI_COMMIT_REF_NAME}}
        imagePullPolicy: Always
        env:
          - name: APP_DEBUG
            value: "true"
          - name: APP_ENV
            value: "local"
        envFrom:
        - secretRef:
            name: ebcplay-api
            optional: false
      imagePullSecrets:
      - name: registry-bin-ebc
```

```yaml
---
apiVersion: v1
kind: Service
metadata:
  name: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
  labels:
    run: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
spec:
  type: ClusterIP
  ports:
  - port: 80
    protocol: TCP
    targetPort: 80
  selector:
    app: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
```

```yaml
---
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}{{DOMAIN_NAME}}"
    http:
      paths:
      - backend:
          serviceName: "ebcplay-api-{{CI_ENVIRONMENT_SLUG}}"
          servicePort: 80
  tls:
    - secretName: ebcplay-api-cert-tls
```

Desafios

Identificar cadeia de valor

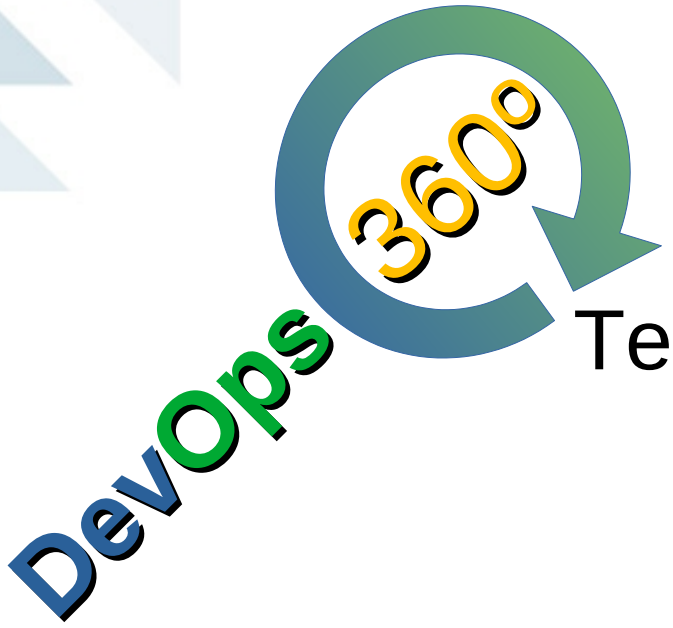
Testes incipientes e manuais

Domínio da infraestrutura

Deploy dia(s)

Monitoramento de serviços baixa maturidade

Desafios



Entender a cadeia de valor

Testes automatizados dev*.*ops

Domínio da infraestrutura

n Deploy/dia

Monitoramento de serviços alta maturidade

Resultados

- agilidade na entrega: ↑
  - de: dia(s) ↑ para: < 1h ↓
- maturidade de testes ↑
  - de: manual ↓ para: automatizado ↑
- entregas com mais qualidade ↑
  - Débito técnico: < 5%

Oportunidades

- Aprimorar cobertura de testes
- Ampliar uso da esteira CI/CD para mais projetos
- Aumentar a qualidade do código
- Amplificar monitoramento de comportamento de aplicações e ambientes

"Eficiência & Agilidade" (~5min),
https://www.youtube.com/watch?v=u00S-hCnmFY

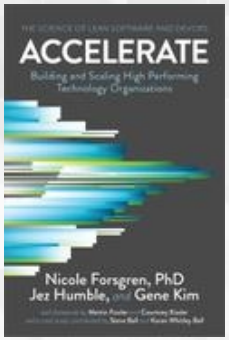
Fome de poder / The Founder

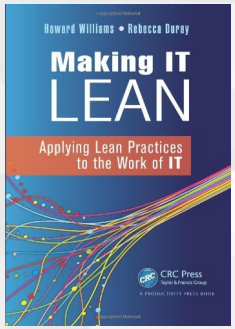
Empresa Brasil de Comunicação

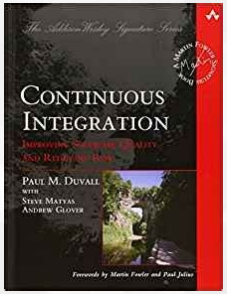
Smart Risk
Cadeia de valor
Burnout syndrome
Débito técnico
Automação
Pipeline CI/CD
Testes
APM
Melhoria Contínua
Docker
Agilidade
DevSecOps
KPI
PDCA
Cultura organizacional
Experimentação
Observability
Tecnologia
Perguntas?
Processos
Colaboração
Pessoas
Confiança
DevOps
Gitlab
Kubernetes
Lean
Kaban
Feedback
Liderança
Day 2 DevOps
Kata
Lean IT
Ansible
Security
Métricas
Integração Contínua
Continuous learning
Qualidade Contínua
Infraestrutura
Entrega Contínua

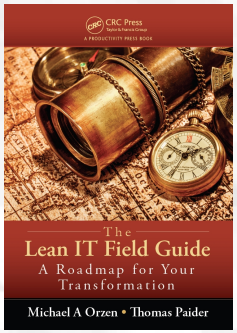
Empresa Brasil de Comunicação

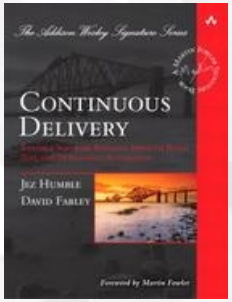
Referências











- Accelerate. Nicole Forsgren, Gene Kim, Jez Humble. 2018
- Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Jez Humble, David Farley. 2010
- Continuous Integration: Improving Software Quality and Reducing Risk. Andrew Glover, Steve Matyas, Paul M. Duvall. 2007
- The Lean IT Field Guide. Thomas A. Paider, Michael A. Orzen. 2017
- Making IT Lean. Rebecca Duray, Howard Williams. 2012
- GitLab Continuous Integration (GitLab CI/CD), https://docs.gitlab.com/ce/ci
- Continuous integration, https://en.wikipedia.org/wiki/Continuous_integration
- Ansible IT automation tool documentation, https://docs.ansible.com/
- Docker Documentation, https://docs.docker.com
- SonarQube documentation, https://docs.sonarqube.org/

@adriano_vieira
speakerdeck.com/adrianovieira
gitlab.com/adrianovieira

Keep C.A.L.M.S. and having fun!

Adriano Vieira

Entusiasta em integração de equipes e agilidade em entrega de resultados, atua como agente influenciador na implementação de infraestruturas ágeis e na internalização de práticas Lean & DevOps.

Profissional de TI com mais 20 anos de experiência trabalhando e gerenciando equipes em ambientes de Datacenter, tendo atuado em desenho, análise e desenvolvimento de sistemas de baixa a alta complexidade como ERP. Também atuou como líder do desenvolvimento de projeto *open-source* e contribui para projetos desta natureza.

Atualmente trabalha na Empresa Brasil de Comunicação (EBC) como Coordenador de Desenvolvimento e Distribuição de Conteúdos. É formado em engenharia mecânica e em MBA.