

# Entregable CICD

1. El enlace al repositorio de GitHub donde se encuentra el código de la aplicación.

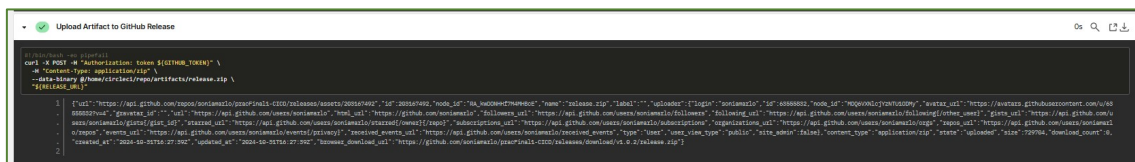
<https://github.com/chris-cmsoft/laravel-kubernetes-example-app.git>

2. El enlace al repositorio de artefactos donde se encuentra el artefacto de la aplicación.

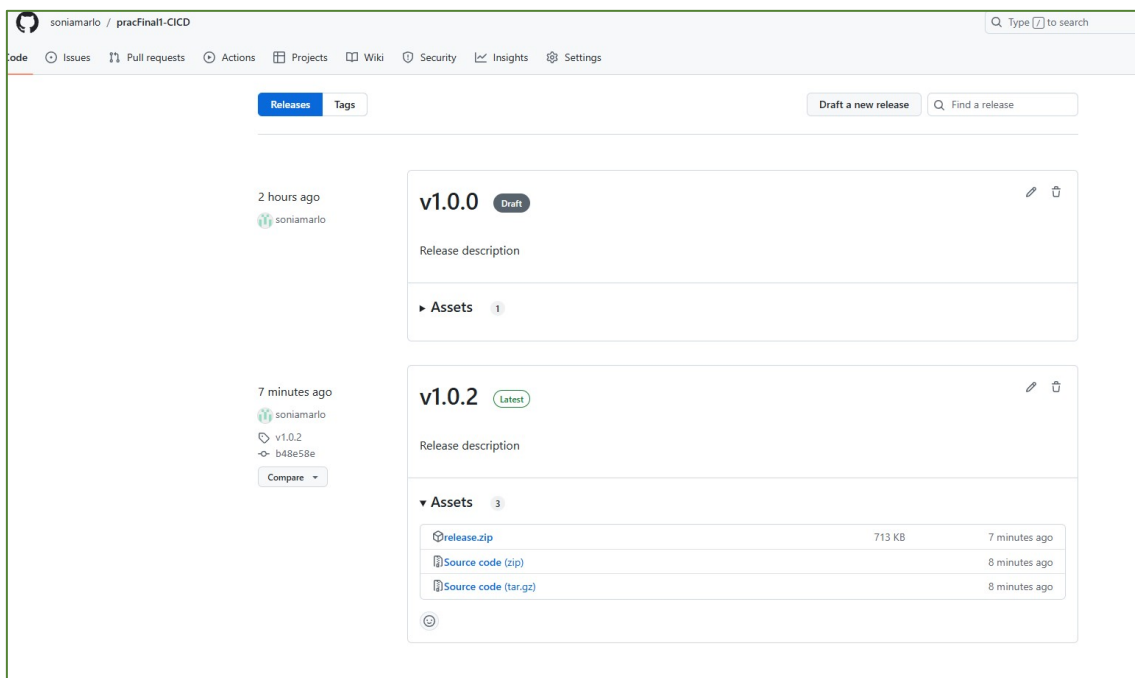
Se encuentra en el repositorio <https://github.com/soniamarlo/pracFinal1-CICD.git>.

Dentro de ese repositorio se encuentran en tags, y después en releases.

<https://github.com/soniamarlo/pracFinal1-CICD/releases>.



```
curl -X POST -H "Authorization: token $(GITHUB_TOKEN)" \
  -H "Content-Type: application/json" \
  --data-binary @/home/cicrict/repos/artifacts/release.zip \
  https://api.github.com/repos/soniamarlo/pracFinal1-CICD/releases/assets
```



### 3. El fichero de configuración del pipeline de CI/CD.

El fichero se encuentra en el repositorio:

<https://github.com/soniamarlo/pracFinal1-CICD.git>

En la carpeta .circleci.

### 4. Screenshots del pipeline de CI/CD

- Definición de orbs y ejecutores

```
version: 2.1

orbs:
  ggshield: gitguardian/ggshield@volatile
  sonarcloud: sonarsource/sonarcloud@2.0.0

executors:
  php-executor:
    docker:
      - image: circleci/php:8.0-cli
    working_directory: ~/repo
    environment:
      APP_ENV: testing
      DB_CONNECTION: sqlite
      DB_DATABASE: ":memory:"
```

- Definición del primer job: instalación de dependencias

```
jobs:
  install_dependencies:
    executor: php-executor
    steps:
      - checkout
      - run:
          name: Install Composer Dependencies
          command: |
            composer install --prefer-dist --no-interaction --no-scripts --no-progress
      - persist_to_workspace:
          root: ~/repo
          paths:
            - vendor
```

- Definición del segundo job: ejecución de los test

```
run_tests:
  executor: php-executor
  steps:
    - checkout
    - attach_workspace:
        at: ~/repo
    - run:
        name: Install and Enable Xdebug
        command: |
          sudo pecl install xdebug
          XDEBUG_INI_FILE=$(php -r "echo PHP_CONFIG_FILE_SCAN_DIR;")/docker-php-ext-xdebug.ini
          echo "zend_extension=$(find /usr/local/lib/php/extensions/ -name xdebug.so)" | sudo tee $XDEBUG_INI_FILE
          echo "xdebug.mode=coverage" | sudo tee -a $XDEBUG_INI_FILE
    - run:
        name: Copy .env File
        command: cp .env.example .env
    - run:
        name: Set Up Application Key
        command: php artisan key:generate
    - run:
        name: Run Database Migrations
        command: php artisan migrate --env=testing --database=sqlite
    - sonarcloud/scan
    - run:
        name: Run PHPUnit Tests with Coverage
        command: |
          mkdir -p ~/repo/artifacts/test-results
          php -d xdebug.mode=coverage ./vendor/bin/phpunit --coverage-clover ~/repo/artifacts/test-results/coverage.xml
    - persist_to_workspace:
        root: ~/repo/artifacts/test-results
        paths:
          - coverage.xml
```

- Definición del tercer job: creación de artefactos

```
create_artifact:
  executor: php-executor
  steps:
    - checkout
    - attach_workspace:
        at: ~/repo
    - run:
        name: Pack Application
        command: |
          composer archive --format=zip --dir=artifacts --file=release
    - attach_workspace:
        at: ~/repo/artifacts/test-results
    - store_artifacts:
        name: Upload Release Package
        path: ~/repo/artifacts/release.zip
        destination: package/release.zip
    - store_artifacts:
        name: Upload Coverage Report
        path: ~/repo/artifacts/test-results/coverage.xml
        destination: coverage
    - run:
        name: Increment Version Tag Automatically
        command: |
          LAST_TAG=$(git describe --tags $(git rev-list --tags --max-count=1))
          NEW_TAG=$(echo $LAST_TAG | awk -F. -v OFS=. '{ $NF += 1 ; print }')
          echo "export NEW_TAG=${NEW_TAG}" >> $BASH_ENV
```

```
- run:
  name: Create GitHub Release
  command: |
    RESPONSE=$(curl -s -X POST -H "Authorization: token ${GITHUB_TOKEN}" \
      -H "Content-Type: application/json" \
      -d '{
        "tag_name": "'"$NEW_TAG"'",
        "target_commitish": "main",
        "name": "'"$NEW_TAG"'",
        "body": "Release description",
        "draft": false,
        "prerelease": false
      }' \
      https://api.github.com/repos/soniamarlo/pracFinal1-CICD/releases)
    echo "GitHub API response: $RESPONSE" # Para depurar
    RELEASE_URL=$(echo "$RESPONSE" | jq -r '.upload_url' | sed -e "s/{?name,label}/?name=release.zip/")
    echo "export RELEASE_URL=${RELEASE_URL}" >> $BASH_ENV

- run:
  name: Upload Artifact to GitHub Release
  command: |
    curl -X POST -H "Authorization: token ${GITHUB_TOKEN}" \
      -H "Content-Type: application/zip" \
      --data-binary @/home/circleci/repo/artifacts/release.zip \
      "${RELEASE_URL}"
```

- El flujo test-and-deploy organiza la secuencia de trabajos

```
workflows:
  Open run panel to run this workflow
  test-and-deploy:
    jobs:
      - install_dependencies
      - run_tests:
          requires:
            - install_dependencies
          context: SonarCloud
      - create_artifact:
          requires:
            - run_tests
          filters:
            branches:
              only: main
      - ggshield/scan:
          name: security_scan
          base_revision: << pipeline.git.base_revision >>
          revision: << pipeline.git.revision >>
          requires:
            - run_tests
```

## 5. Los manifiestos de Kubernetes para el despliegue de la aplicación.

Los archivos de kubernetes se encuentran en el repositorio:

<https://github.com/soniamarlo/pracFinal1-CICD.git>

En la carpeta kubernetes. Dentro de ella estarían las subcarpetas: app, argocd y secrets.

- En la carpeta app estarían los archivos.

### 1. Deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: laravel-app
  labels:
    app: laravel-app
spec:
  replicas: 2
  selector:
    matchLabels:
      app: laravel-app
  template:
    metadata:
      labels:
        app: laravel-app
    spec:
      containers:
        - name: laravel-app
          image: nginx:latest
          ports:
            - containerPort: 80
          volumeMounts:
            - name: html
              mountPath: /usr/share/nginx/html
      volumes:
        - name: html
          emptyDir: {}
```

## 2. Service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: laravel-service
  namespace: argocd-cicd
  labels:
    app: laravel-app
spec:
  type: NodePort
  ports:
    - port: 80
      targetPort: 80
  selector:
    app: laravel-app
```

## 3. Ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: laravel-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  ingressClassName: nginx
  rules:
    - host: laravel-app.local
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: laravel-service
                port:
                  number: 80
```

#### 4. mysql-deployment.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mysql
  namespace: argocd-cicd
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
        - name: mysql
          image: mysql:8.0
          envFrom:
            - secretRef:
                name: laravel-secret
          ports:
            - containerPort: 3306
```

## 5. mysql-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: mysql-service
  namespace: argocd-cicd
  labels:
    app: mysql
spec:
  ports:
    - port: 3306
      targetPort: 3306
  selector:
    app: mysql
```

- En la carpeta argocd estaría el archivo:

### 1. Argoapp.yaml

```
apiVersion: argoproj.io/v1alpha1
kind: Application
metadata:
  name: laravel-app
  namespace: argocd
  finalizers:
    - resources-finalizer.argocd.argoproj.io
spec:
  project: default
  source:
    repoURL: https://github.com/soniamarlo/pracFinal1-CICD.git
    targetRevision: main
    path: kubernetes/app
  destination:
    server: https://kubernetes.default.svc
    namespace: argocd-cicd
  syncPolicy:
    automated:
      prune: true
      selfHeal: true
```



- En la carpeta secrets estarían los archivos:

1. My-secret.yaml
2. Sealed.yaml

Por temas de protección de datos no incluyo el contenido de ninguno.

Se adjuntan también imágenes del despliegue de los manifests de Kubernetes en la aplicación K9s.

- Deployments

```

Context: kind-argo          <0> all          <ctrl-d> Delete
Cluster: kind-argo         <1> argocd-cicd  <d>      Describe
User:    kind-argo         <2> default     <e>      Edit
K9s Rev: v0.32.5           <?>           <?>      Help
K8s Rev: v1.30.4           <l>           <l>      Logs
CPU:      n/a              <p>           <p>      Logs Previous
MEM:      n/a
  
```

Deployments(argocd-cicd)[2]				
NAME↑	READY	UP-TO-DATE	AVAILABLE	AGE
laravel-app	2/2	2	2	39m
mysql	1/1	1	1	39m

- Pod

```

Context: kind-argo          <0> all          <a>      Attach
Cluster: kind-argo         <1> argocd-cicd  <ctrl-d> Delete
User:    kind-argo         <2> default     <d>      Describe
K9s Rev: v0.32.5           <e>      Edit
K8s Rev: v1.30.4           <?>      Help
CPU:      n/a              <shift-j> Jump Owner
MEM:      n/a
  
```

Pods(argocd-cicd)[3]							
NAME↑	PF	READY	STATUS	RESTARTS	IP	NODE	AGE
laravel-app-678c775655-kbf55	•	1/1	Running	0	10.244.0.84	argo-control-plane	35m
laravel-app-678c775655-lnq4d	•	1/1	Running	0	10.244.0.85	argo-control-plane	35m
mysql-564c4fd96b-xvws8	•	1/1	Running	0	10.244.0.83	argo-control-plane	35m

- Services

```

Context: kind-argo          <0> all          <b>      Bench Run/Stop
Cluster: kind-argo         <1> argocd-cicd  <ctrl-d> Delete
User:    kind-argo         <2> default     <d>      Describe
K9s Rev: v0.32.5           <e>      Edit
K8s Rev: v1.30.4           <?>      Help
CPU:      n/a              <l>      Logs
MEM:      n/a
  
```

Services(argocd-cicd)[2]					
NAME↑	TYPE	CLUSTER-IP	EXTERNAL-IP	PORTS	AGE
laravel-service	NodePort	10.96.141.178		80→31134	35m
mysql-service	ClusterIP	10.96.143.41		3306→0	35m

- Ingress

```

Context: kind-argo          <0> all          <ctrl-d> Delete
Cluster: kind-argo         <1> argocd-cicd  <d>      Describe
User:    kind-argo         <2> default     <e>      Edit
K9s Rev: v0.32.5           <?>           <?>      Help
K8s Rev: v1.30.4           <y>           <y>      YAML
CPU:      n/a
MEM:      n/a
  
```

Ingresses(argocd-cicd)[1]					
NAME↑	CLASS	HOSTS	ADDRESS	PORTS	AGE
laravel-ingress	nginx	laravel-app.local	localhost	80	36m

- Secret

```
Context: kind-argo
Cluster: kind-argo
User: kind-argo
K9s Rev: v0.32.5
K8s Rev: v1.30.4
CPU: n/a
MEM: n/a
```

NAME↑	TYPE	DATA	AGE
laravel-secret	Opaque	9	13h

## 6. Enlace o screenshot de la aplicación desplegada.

- Análisis del pipeline con éxito.

Dashboard Project Branch Workflow

All Pipelines > pracFinal1-CICD > main > test-and-deploy

**test-and-deploy** Success

Duration / Finished: 1m 32s / 3m ago

Branch: main

Commit: b48e58e

Author & Message: fix: automatically tag version

```

graph LR
    A[install_dependencies 19s] --> B[run_tests 50s]
    B --> C[security_scan 7s]
    B --> D[create_artifact 15s]
  
```

- Instalación de dependencias

**install\_dependencies** Success Rerun ...

Duration / Finished: 8s / 3m ago

Queued: 0s

Executor / Resource Class: Docker / Large

Branch: main

Commit: a21748b

Author & Message: fix: config.yml

⚠ You're using a deprecated Docker convenience image. Upgrade to a next-gen Docker convenience image.

STEPS TESTS TIMING ARTIFACTS RESOURCES

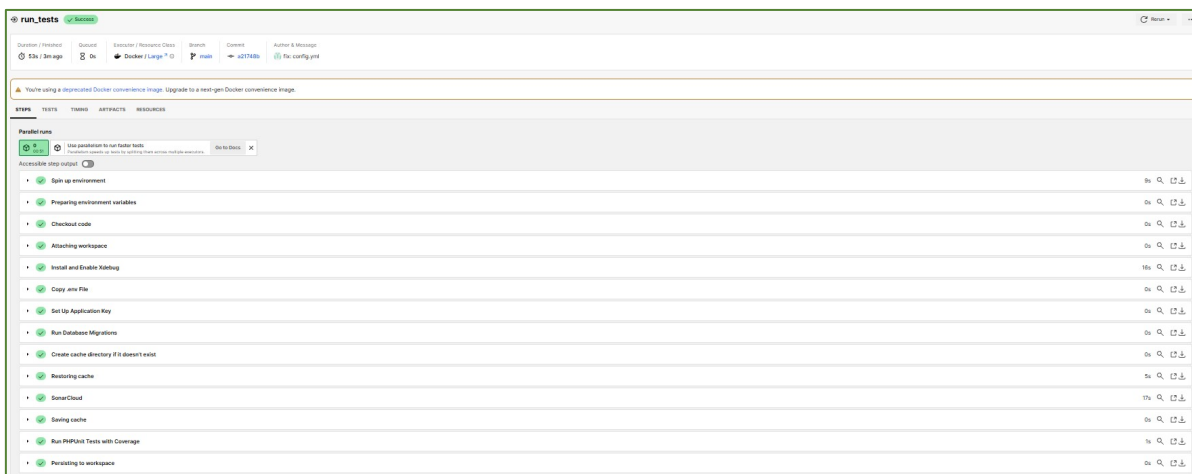
**Parallel runs**

0 00:07 Use parallelism to run faster tests Parallelism speeds up tests by splitting them across multiple executors. Go to Docs X

Accessible step output ☐

- Spin up environment 0s
- Preparing environment variables 0s
- Checkout code 0s
- Install Composer Dependencies 3s
- Persisting to workspace 2s

## - Ejecución de los test



**run\_tests** Success

Duration / Finished: 53s / 3m ago | Queued: 0s | Executor / Resource Class: Docker / Large | Branch: main | Commit: a21748b | Author & Message: fix: config.yml

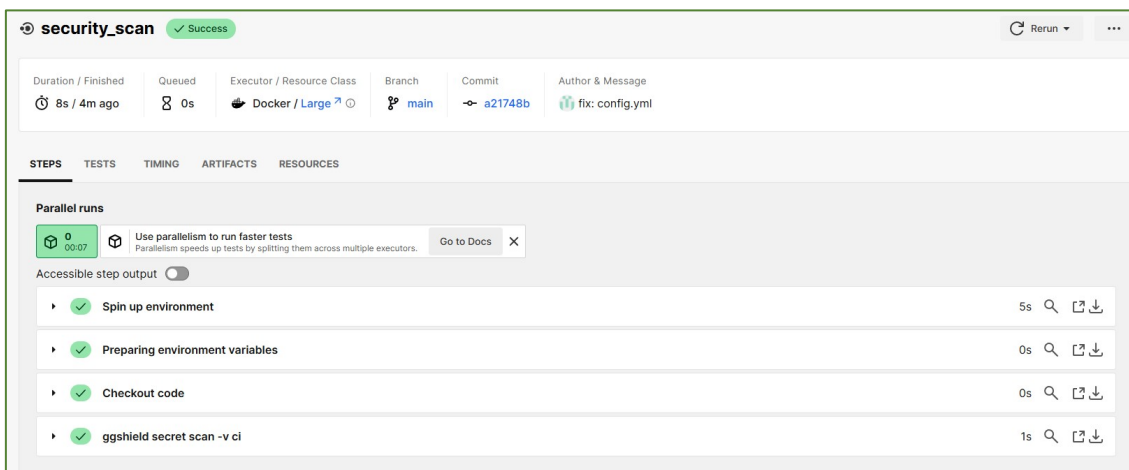
**STEPS** TESTS TIMING ARTIFACTS RESOURCES

**Parallel runs** 0 00:07 Use parallelism to run faster tests Go to Docs X

Accessible step output 0

- Spin up environment 5s
- Preparing environment variables 0s
- Checkout code 0s
- Attaching workspace 0s
- Install and Enable Xdebug 15s
- Copy env file 0s
- Set Up Application Key 0s
- Run Database Migrations 0s
- Create cache directory if it doesn't exist 0s
- Resolving cache 5s
- SonarCloud 10s
- Saving cache 0s
- Run PHPUnit Tests with Coverage 1s
- Persisting to workspace 0s

## - Escaneo de seguridad



**security\_scan** Success

Duration / Finished: 8s / 4m ago | Queued: 0s | Executor / Resource Class: Docker / Large | Branch: main | Commit: a21748b | Author & Message: fix: config.yml

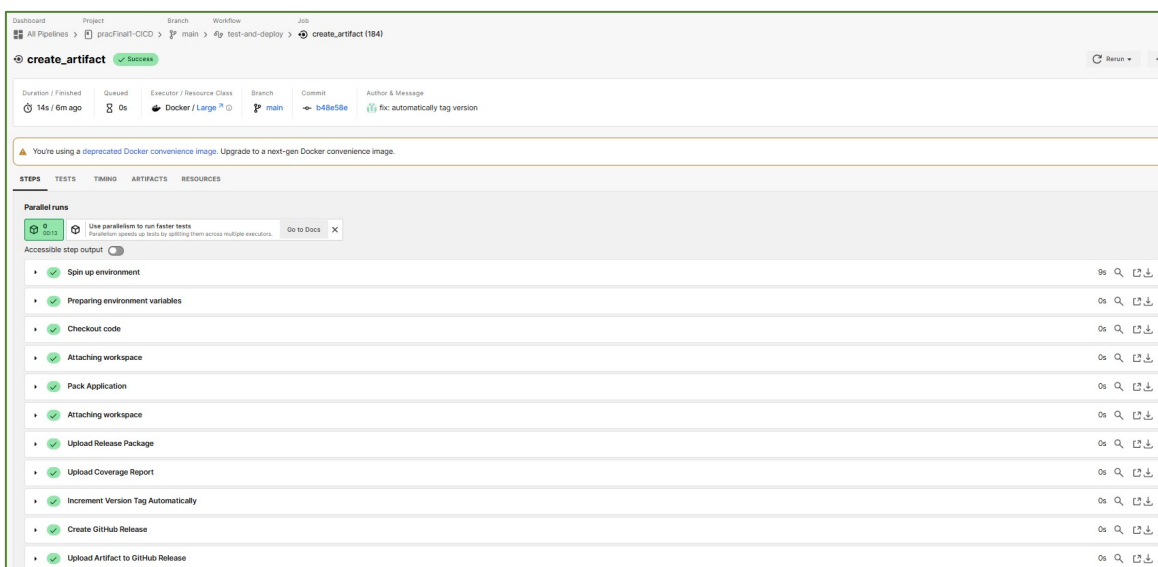
**STEPS** TESTS TIMING ARTIFACTS RESOURCES

**Parallel runs** 0 00:07 Use parallelism to run faster tests Go to Docs X

Accessible step output 0

- Spin up environment 5s
- Preparing environment variables 0s
- Checkout code 0s
- ggshield secret scan -v ci 1s

## - Creación de artefactos



**create\_artifact** Success

Duration / Finished: 14s / 6m ago | Queued: 0s | Executor / Resource Class: Docker / Large | Branch: main | Commit: b48e58e | Author & Message: fix: automatically tag version

**STEPS** TESTS TIMING ARTIFACTS RESOURCES

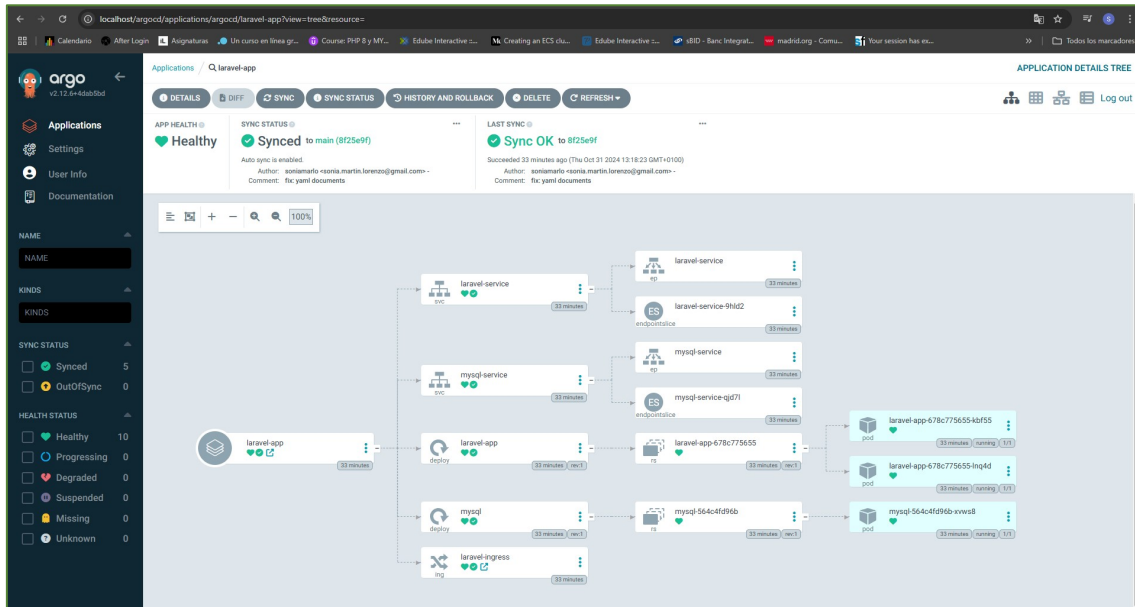
**Parallel runs** 0 00:13 Use parallelism to run faster tests Go to Docs X

Accessible step output 0

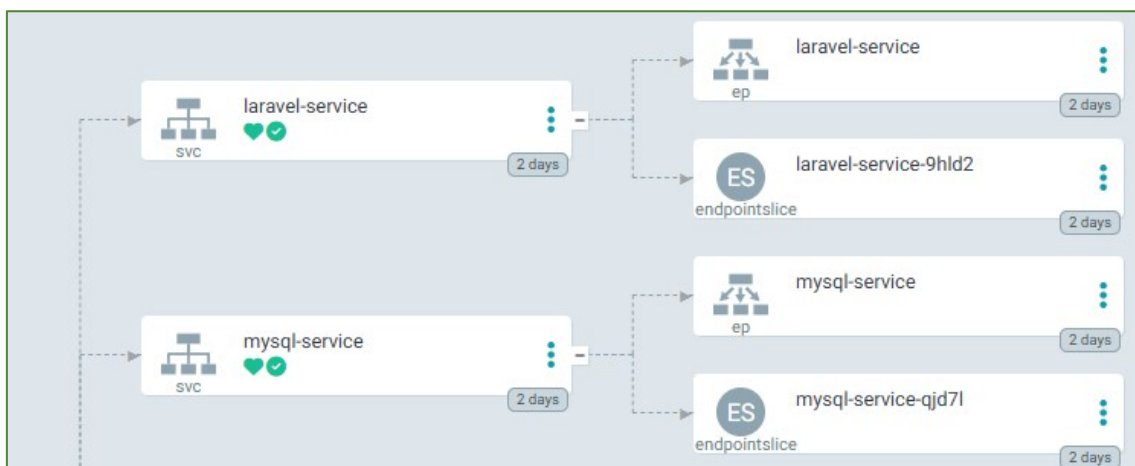
- Spin up environment 0s
- Preparing environment variables 0s
- Checkout code 0s
- Attaching workspace 0s
- Pack Application 0s
- Attaching workspace 0s
- Upload Release Package 0s
- Upload Coverage Report 0s
- Increment Version Tag Automatically 0s
- Create GitHub Release 0s
- Upload Artifact to GitHub Release 0s

## 7. Enlace o screenshot del proyecto en ArgoCD.

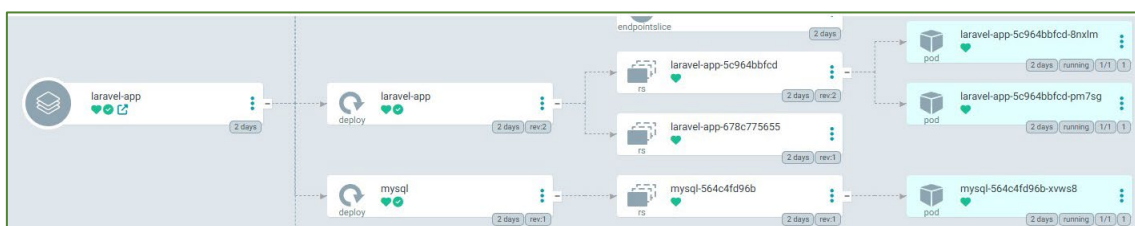
- Proyecto en general desplegado



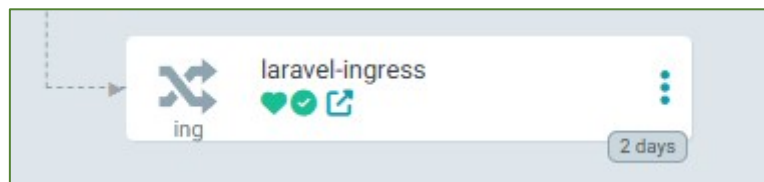
- Servicios



- Deployments

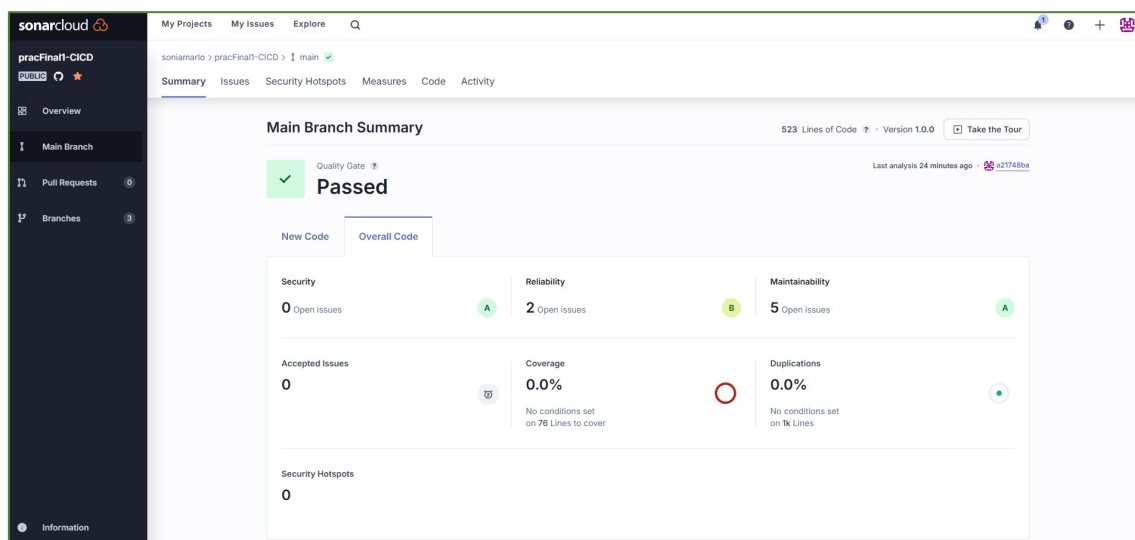
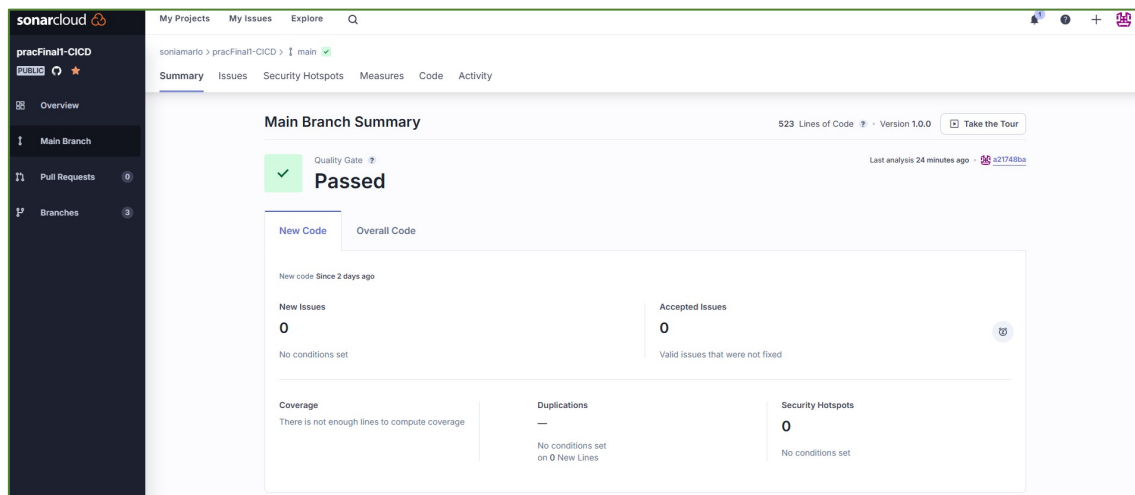


- Ingress



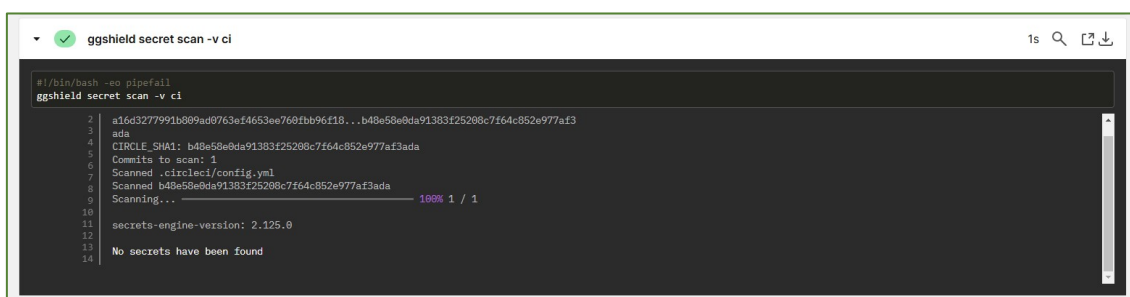
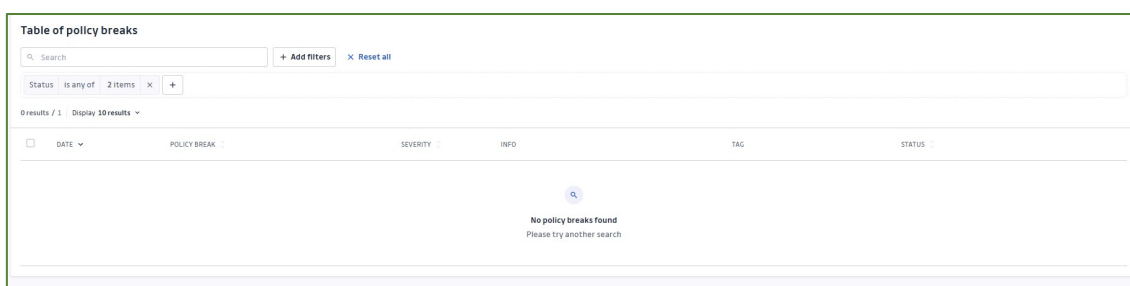
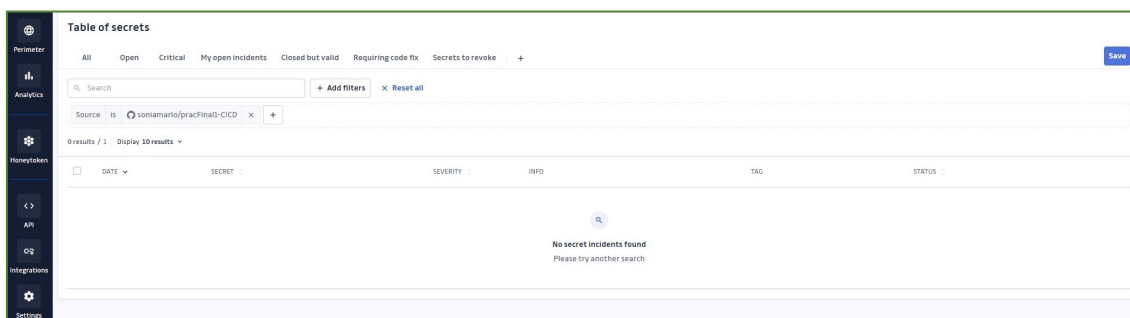
## 8. Enlace o screenshot del proyecto en SonarCloud.

- Análisis estático de código con SonarCloud



## 9. Enlace o screenshot del proyecto en Snyk o GitGuardian.

- Análisis de vulnerabilidades con GitGuardian



## 10. Enlace a un vídeo de Youtube donde se explique la práctica.

El enlace al vídeo es este:

<https://youtu.be/xdaoxgx-qjA>

Está en privado, pero supuestamente he dado permiso a tu correo de Gmail. Si hay algún problema, me lo puedes comunicar y vemos como dar más permisos.