The DevSecOps Journey: **Migrating Your Containerized Applications to the AWS Cloud Platform**

Enabling customers to seamlessly implement security in a developer-friendly way

James Bland, Sr. Solutions Architect, AWS
Angel Rivera, Developer Advocate, Circle CI
Jay Yeras, Head of Cloud & Cloud Native Solutions Architecture, Snyk

Presenters

James Bland, AWS Sr. Partner Solutions Architect is a 25+ year veteran in the IT industry helping organizations from startups to ultra large enterprises achieve their business objectives. He has held various roles in software development, worldwide infrastructure automation and operations, and enterprise architecture. James has been practicing DevOps long before the term became popularized. He holds a doctorate in computer science with a focus on leveraging machine learning algorithms for scaling systems.

Angel Rivera, CircleCI's Developer Advocate, discovered his love for technology and software development at the start of his military career. Angel's passions are positive disruption, learning, teaching, mentoring but most of all inspiring all forms of technologists & building awesome tech communities.

Jay Yeras, Head of Cloud Solution Architecture at Snyk, brings over 20 years of experience in various technical roles and is a subject matter expert in building highly technical cloud architectures. His passion for efficiency, process driven development and automation permeates his family life as he is often found organizing kitchen cupboards for optimal use or setting up Alexa skills much to his wife's dismay.

aws

Table of contents

aws

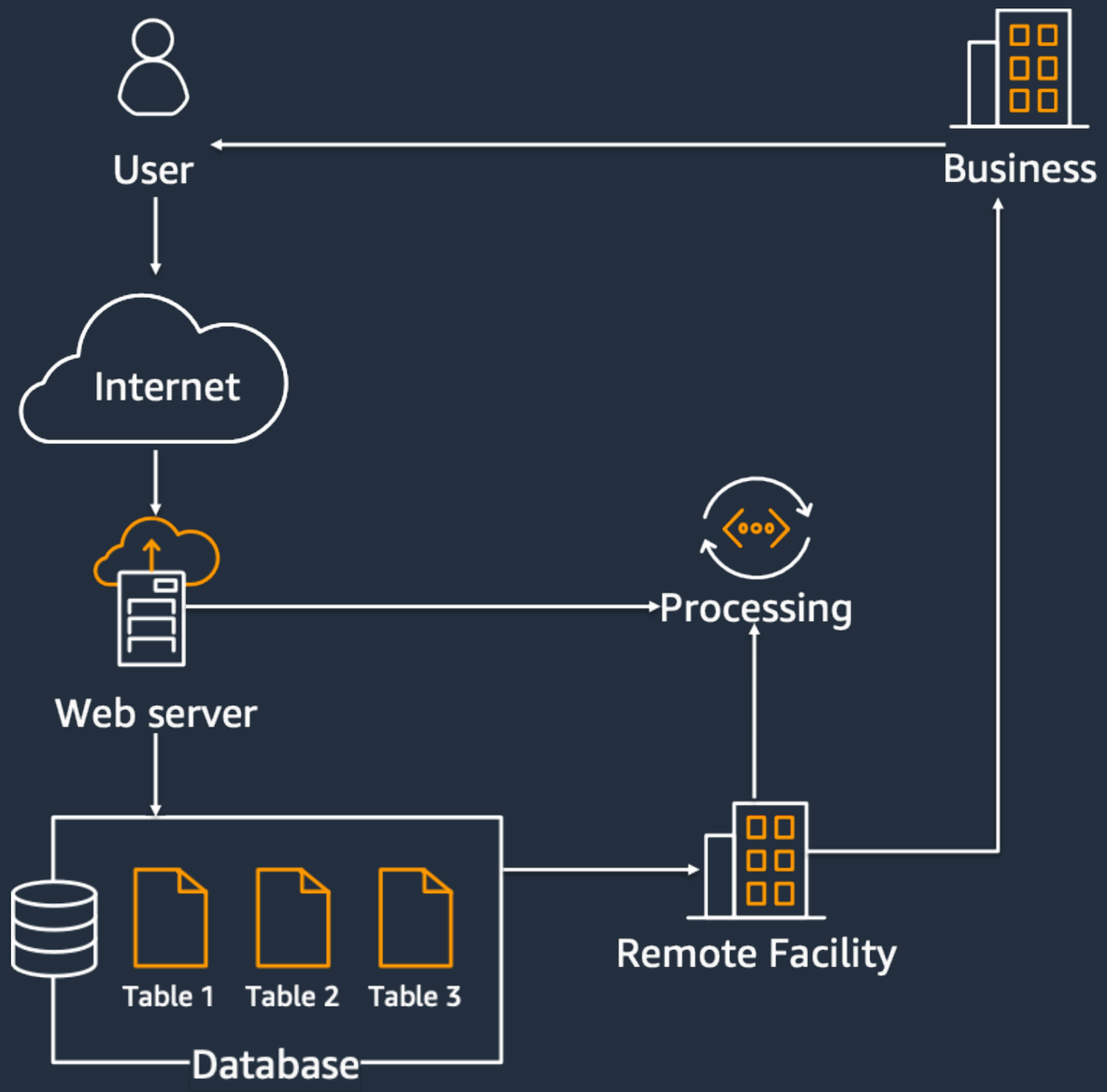
Amazon’s story

An abbreviated version

Just starting out

This is how many web architectures started out, and it's how Amazon started too…

There any many bottlenecks, and scaling of the web server was an immediate factor

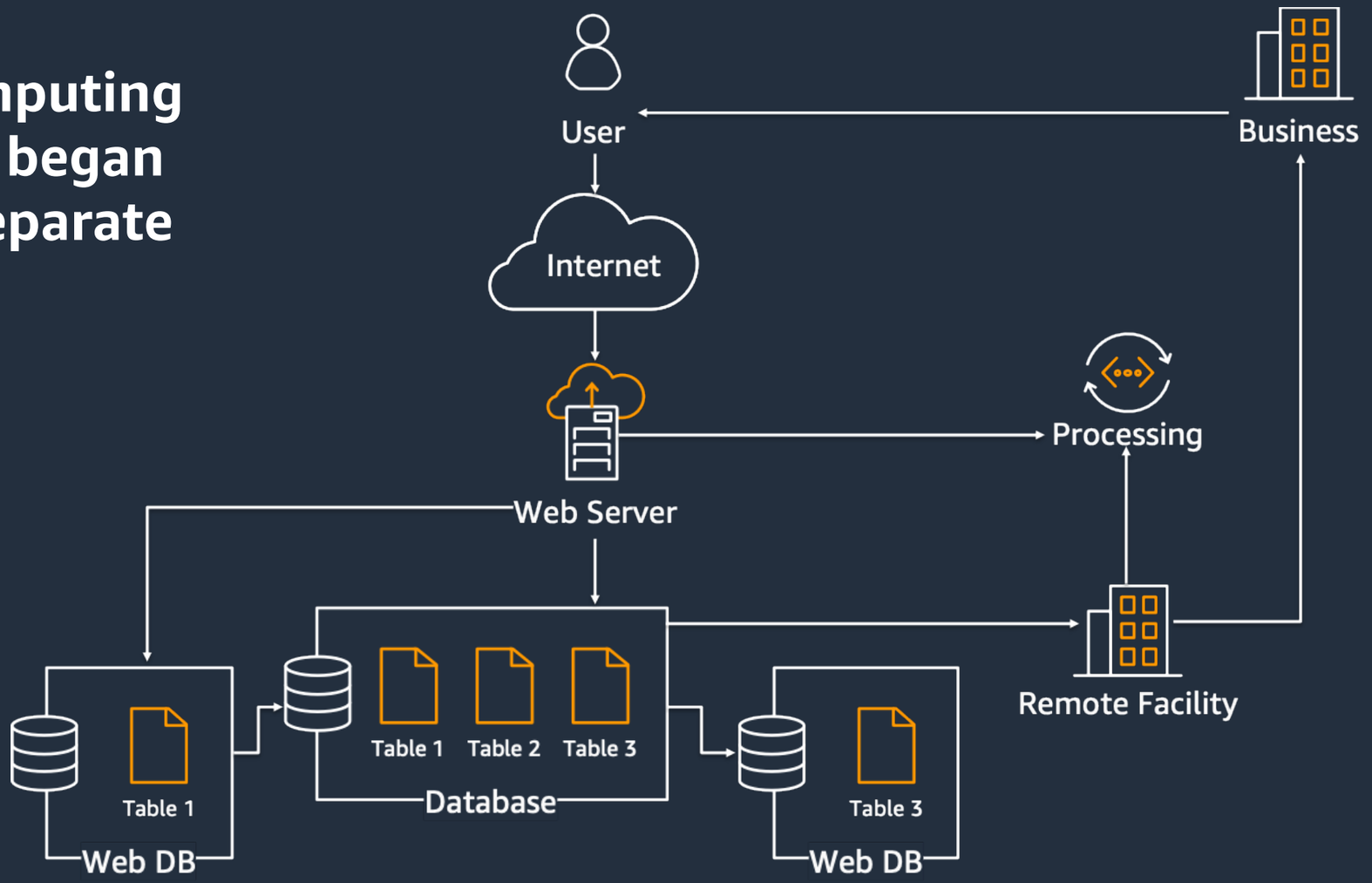


aws

Scaling v1

In 1998 the “Distributed Computing Manifesto” came out and we began breaking things down into separate components…

This was a bit better, still not very scalable

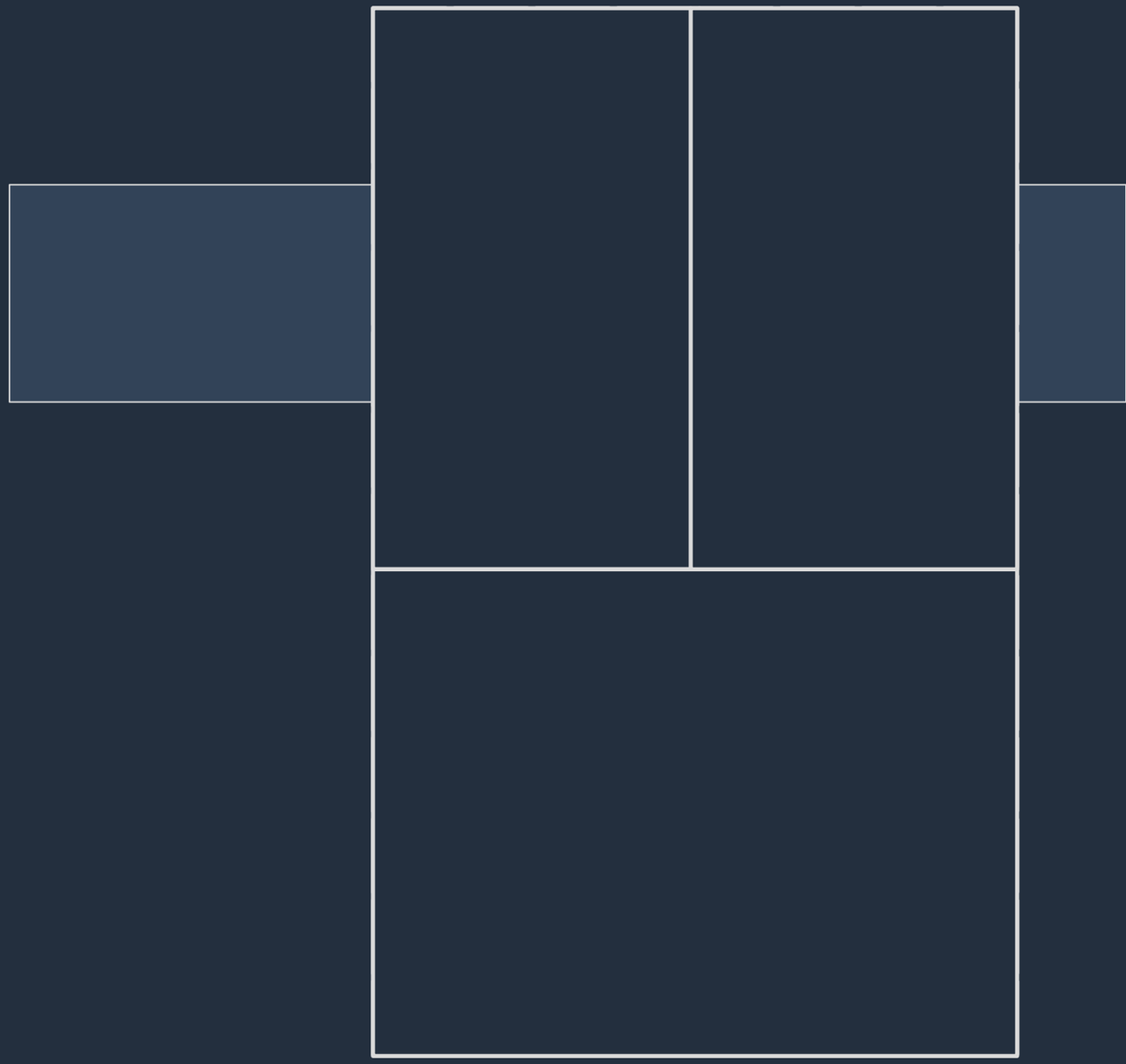


aws

Going further

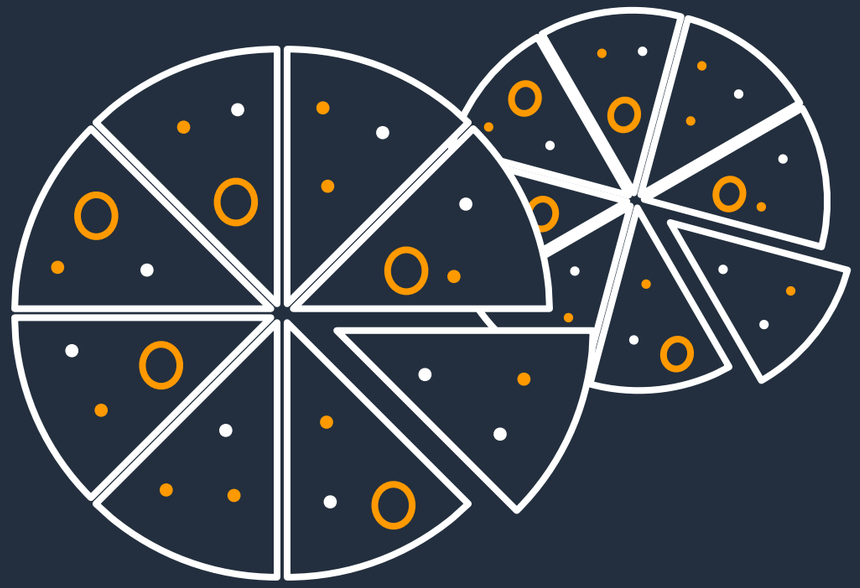
Principles

- **Make units as small as possible (Primitives)**
- **De-couple based on scaling factors, not functions**
- **Each service operates independently**
  “Communication is terrible!” —Jeff Bezos
- **APIs (contracts) between services**



aws

Getting (re)organized



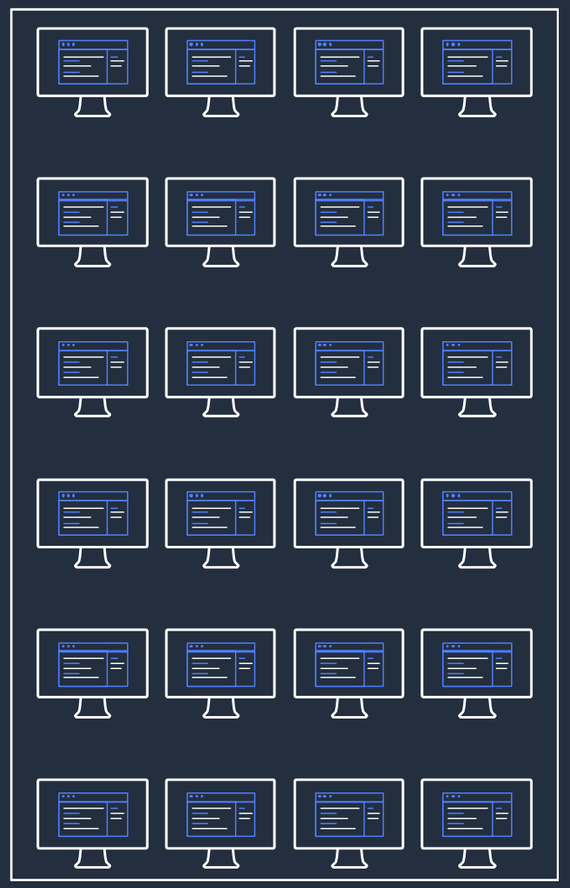
“Two-pizza” teams

- Own a service
- Minimizes social constraints (Conway’s law)
- Autonomy to make decisions

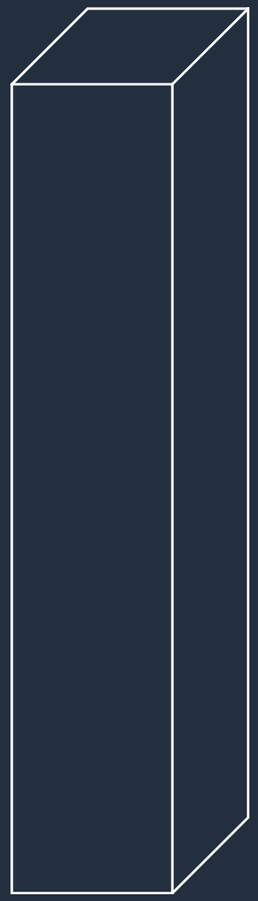
aws

Monolith development lifecycle

Developers



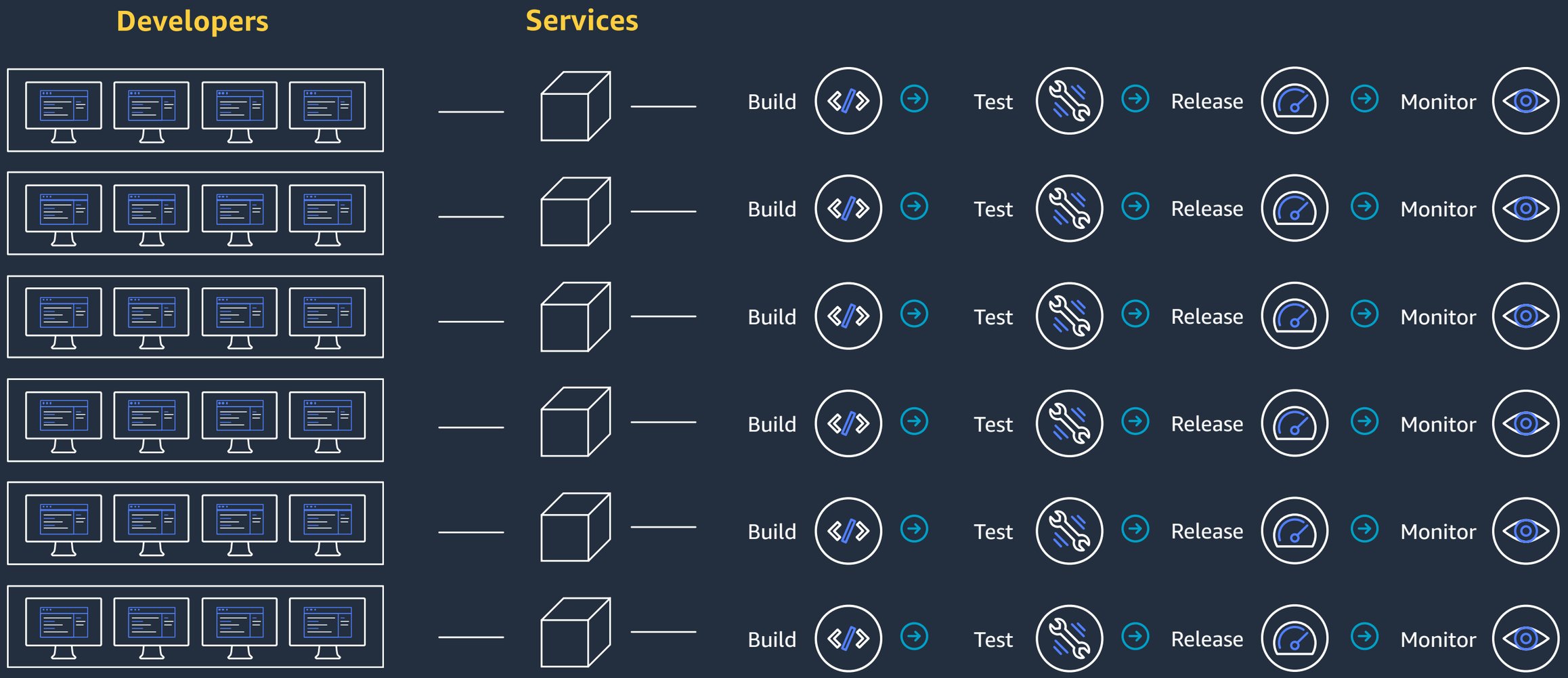
Services



Delivery pipelines

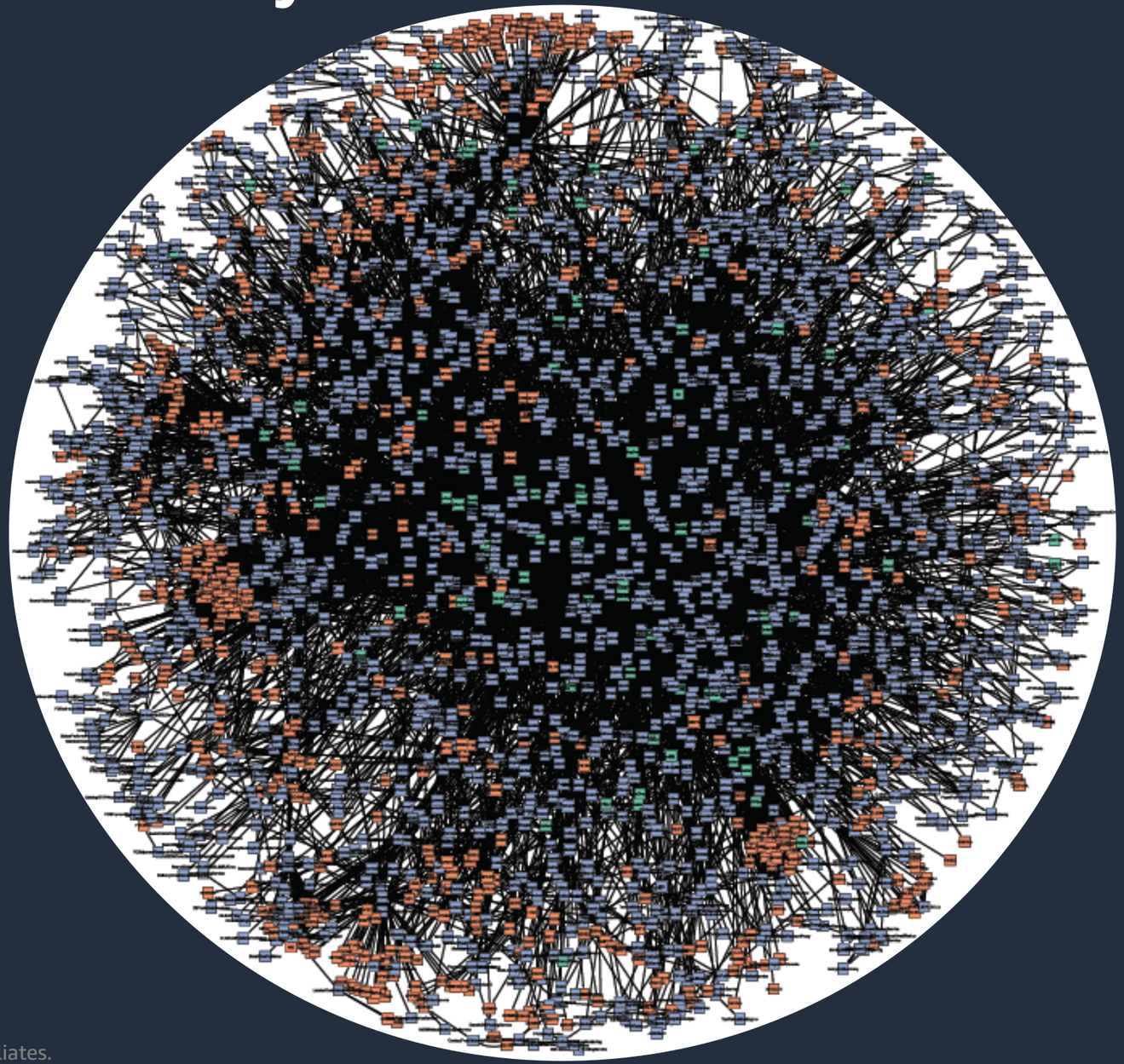
aws

Development lifecycle refactored for speed and agility



aws

Fast forward to today



aws

Want to learn more

aws

The Amazon Builders' Library

How Amazon builds and operates software

Architecture, software delivery, and operations

By Amazon's senior technical executives and engineers

Real-world practices with detailed explanations

Content available for free on the website

aws

AWS Modernization Workshops

Curated workshops created by AWS Partners

https://awsworkshop.io

aws

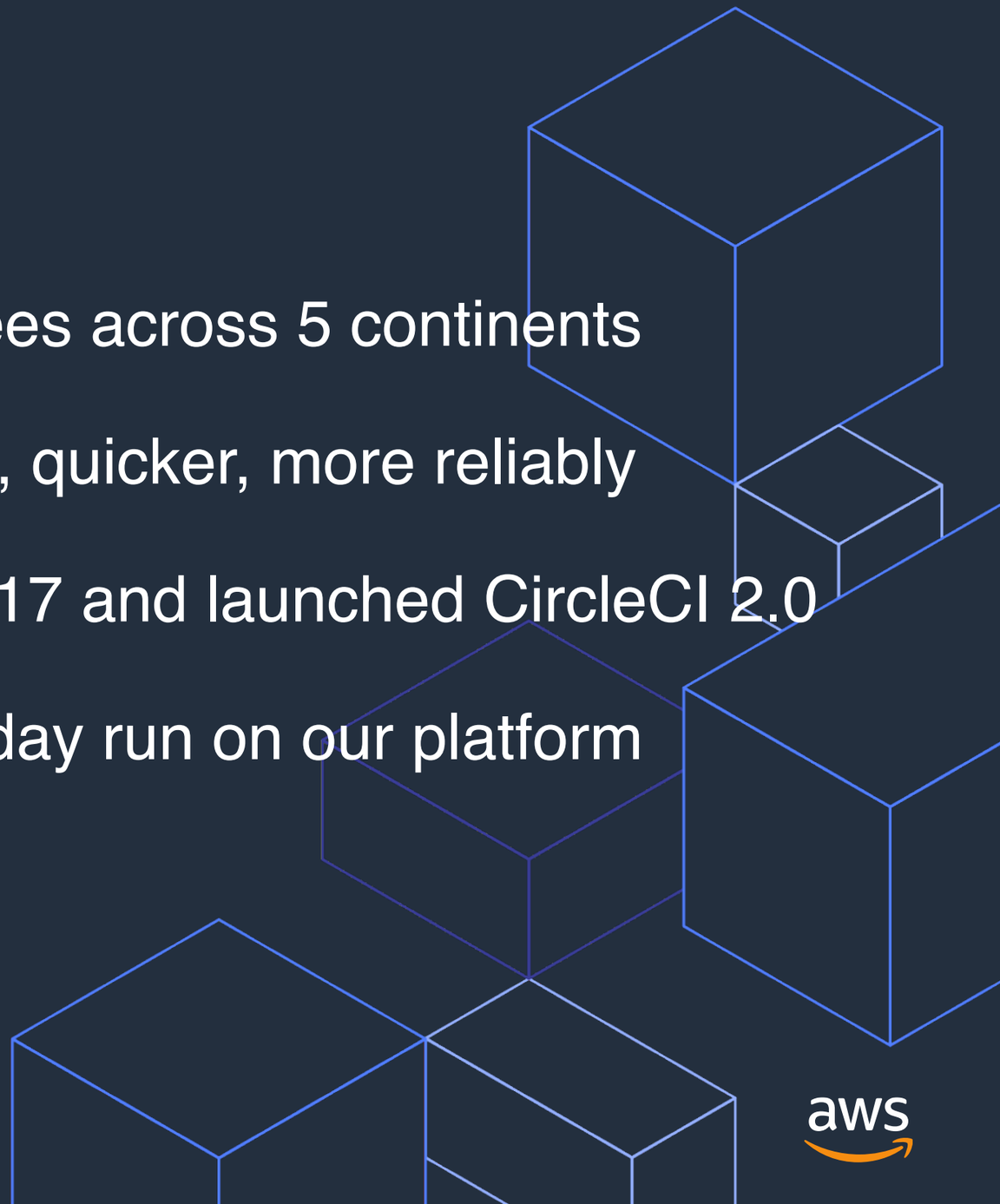
aws

Our mission is to make it as easy as possible to go from idea to delivery.

aws

Introduction to CircleCI

- Founded in 2011 with 280+ employees across 5 continents
- We help teams build better software, quicker, more reliably
- We re-architected the platform in 2017 and launched CircleCI 2.0
- Built for the Cloud - 1M+ builds per day run on our platform



aws

Where CircleCI sits in the toolchain:

CREATION

ORCHESTRATION

LOGISTICS

Code Commit

Ship to Production

Collaborate

Source Control

GitHub

Atlassian Bitbucket

Future support: Microsoft Visual Studio Team Foundation Server

Deliver

Build • Test • Deliver

circleci

Run

Monitor • Operate

docker

HEROKU

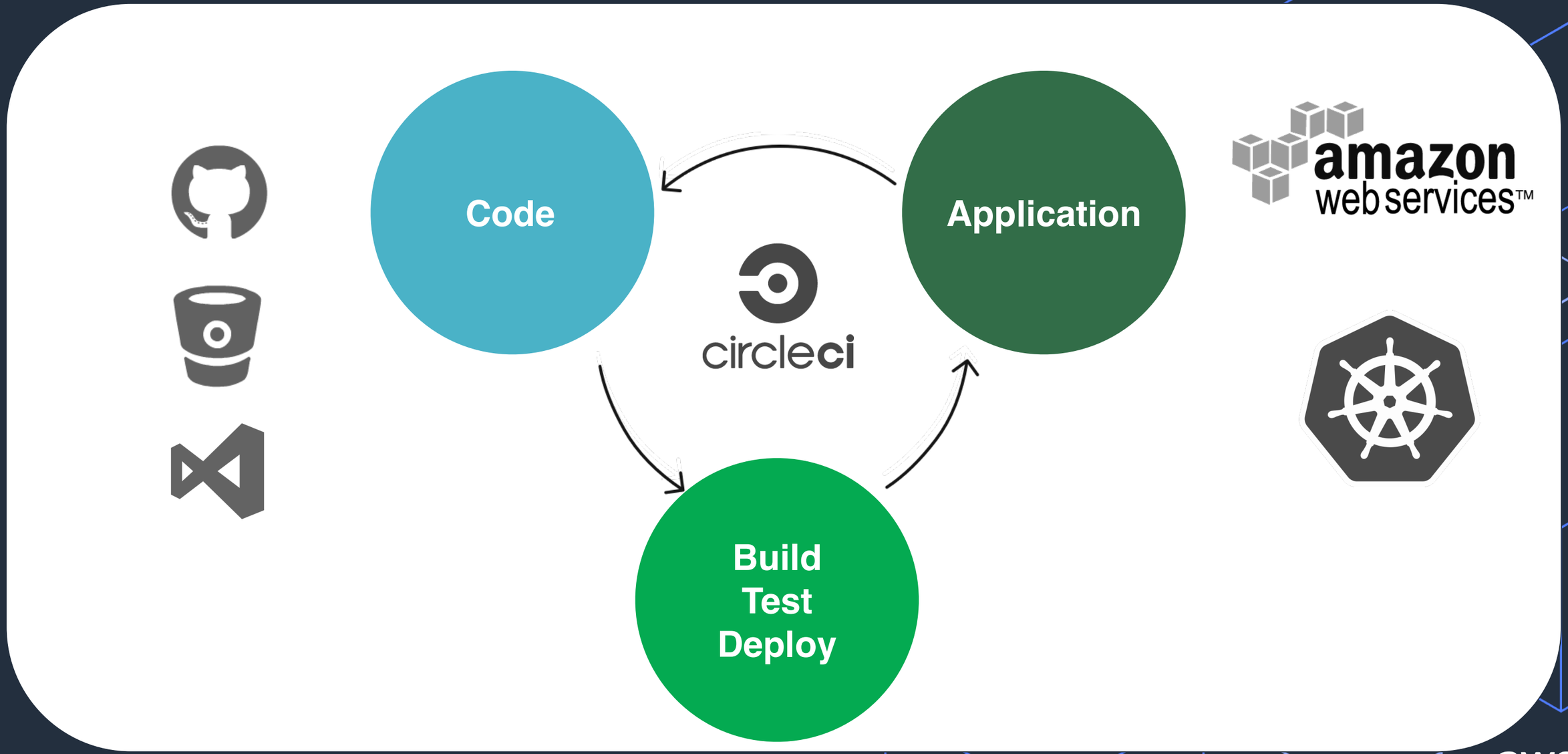
Google Cloud Platform

IBM Bluemix™

amazon web services

aws

CircleCI automates process from code to production software



Orbs. Config Reuse is Here.

/ôrb/ • noun

A reusable, shareable package of CircleCI configuration.

What is an orb?

An orb is a package of CircleCI YAML configuration that contains predefined commands, executors, and jobs. Orbs are shareable across users, teams, and organizations, making it easy to keep your CircleCI configuration up-to-date and in-sync.

orb.yml

```yaml
version: 2.1
description: An orb that says hello
commands:
  say-hello:
    parameters:
      to:
        default: ${CIRCLE_USERNAME}
        type: string
    steps:
      - run: echo "Hello << parameters.to >>"
executors:
  default:
    parameters:
      tag:
        default: latest
        type: string
    docker:
      - image: bash:<< parameters.tag >>
jobs:
  hello-build:
    executor: default
    steps:
      - say-hello
```

Commands

Commands are parameterized sets of steps that can be invoked as a step in a job.

Executors

Executors define the runtime environment settings for executing jobs.

Jobs

Jobs in orbs are invoked similarly to local jobs in config and can include parameters.

CircleCI Pipeline Config using Orbs

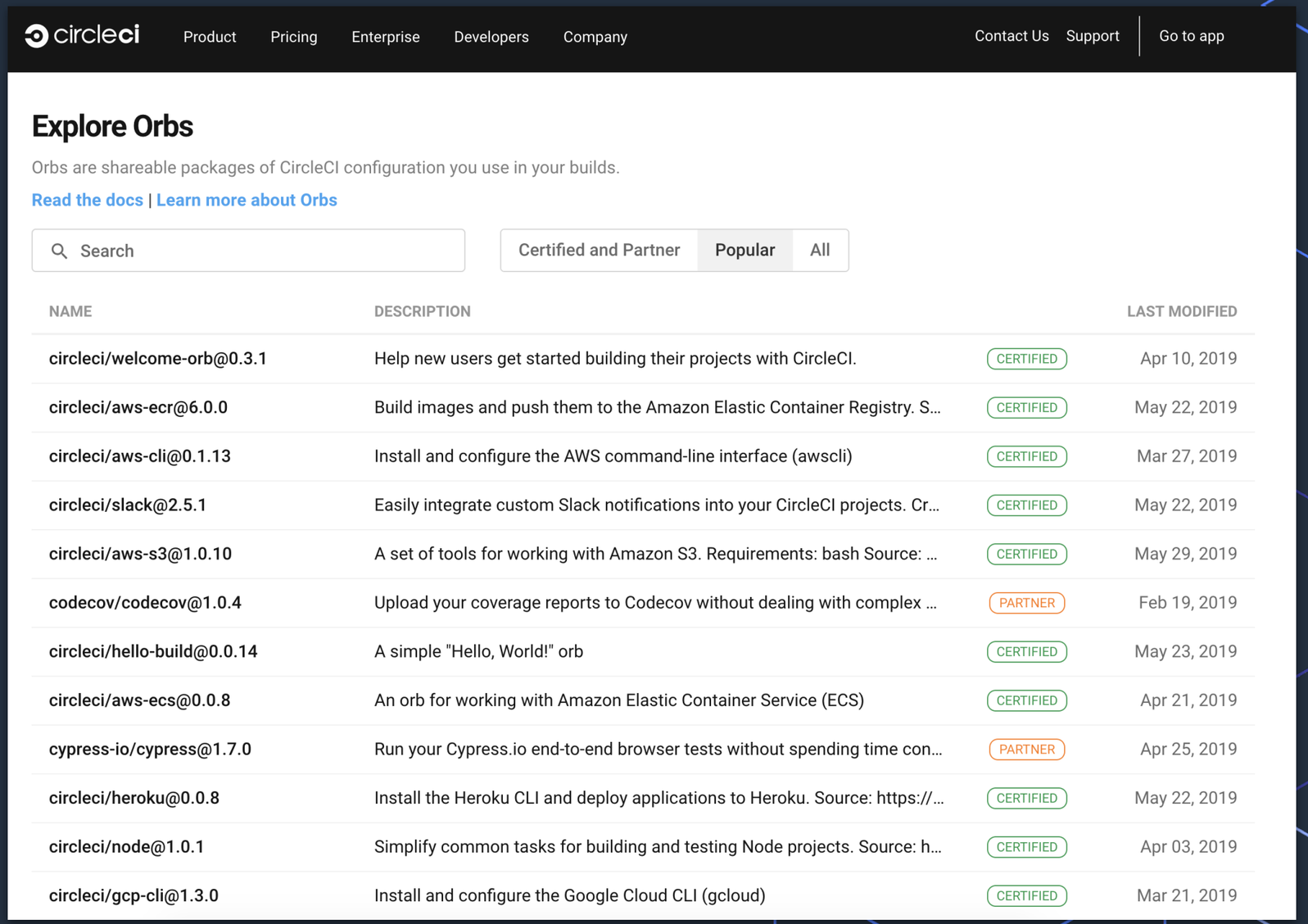
```yaml
version: 2.1
orbs:
  aws-ecr: circleci/aws-ecr@0.0.4

workflows:
  build_test_deploy:
    jobs:
      - build_test
      - docker_hub_build_push_image:
          requires:
            - build_test
      - aws-ecr/build_and_push_image:
          region: us-east-1
          account-url: ${AWS_ECR_ACCOUNT_URL}
          repo: ${CIRCLE_PROJECT_REPONAME}
          tag: ${CIRCLE_BUILD_NUM}
          requires:
            - build_test
jobs:
  build_test:
    docker:
      - image: circleci/python:2.7.14
    steps:
      - checkout
      - run:
          name: Setup VirtualEnv
          command: |...
      - run:
          name: Run Tests
          command: |...
```

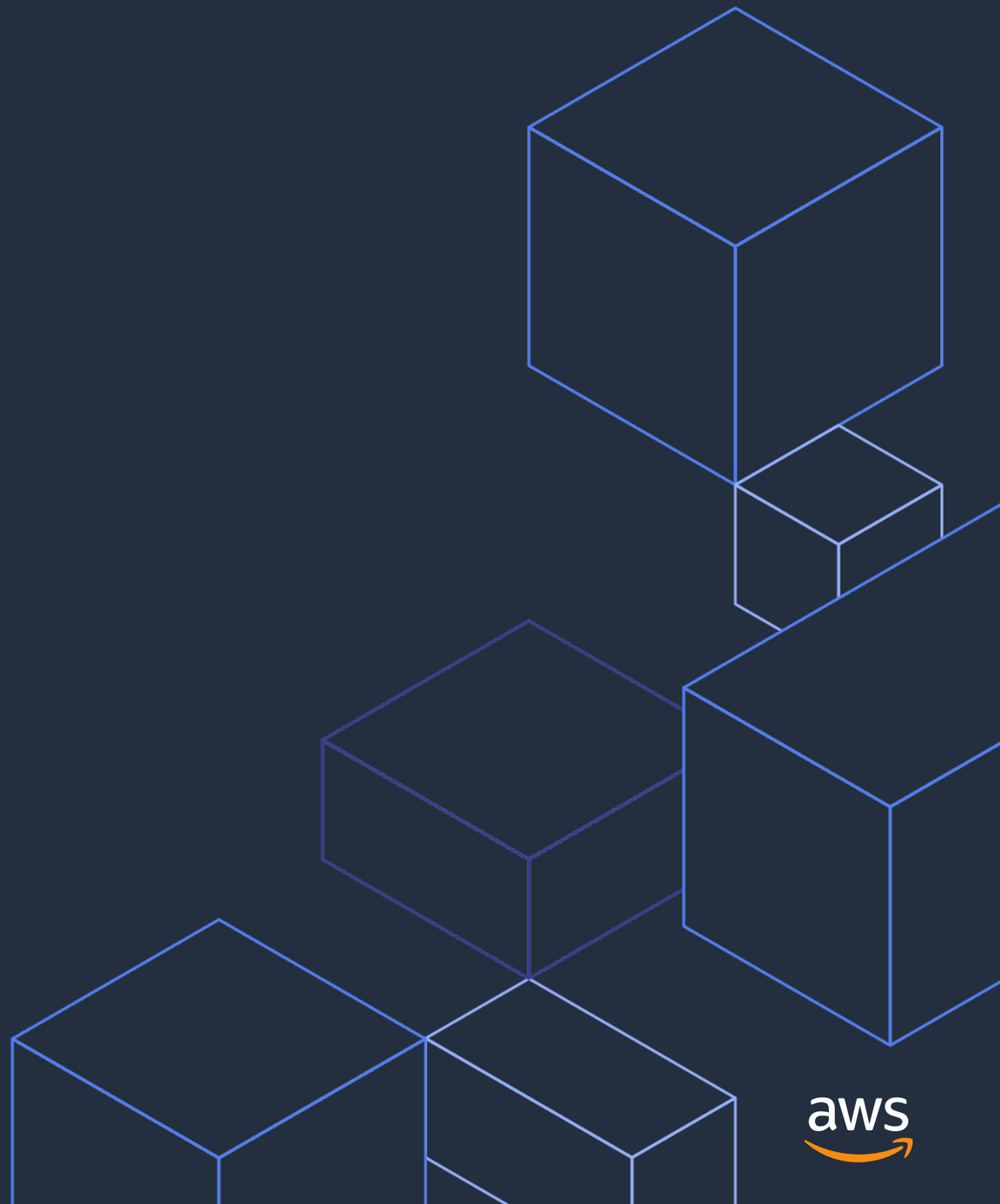
```yaml
docker_hub_build_push_image:
  docker:
    - image: circleci/python:2.7.14
  steps:
    - checkout
    - setup_remote_docker:
        docker_layer_caching: false
    - run:
        name: Build and push Docker image to Docker Hub
        command: |
          echo 'export TAG=0.1.${CIRCLE_BUILD_NUM}' >> ${BASH_ENV}
          echo 'export IMAGE_NAME=${CIRCLE_PROJECT_REPONAME}' >> ${BASH_ENV}
          source ${BASH_ENV}
          docker build -t ${DOCKER_LOGIN}/${IMAGE_NAME} -t
            ${DOCKER_LOGIN}/${IMAGE_NAME}:${TAG} .
          echo ${DOCKER_PWD} | docker login -u ${DOCKER_LOGIN}
            --password-stdin
          docker push ${DOCKER_LOGIN}/${IMAGE_NAME}
```


aws

CircleCI Orbs Registry



Snyk Demo



Free self-paced workshop!

Solutions

Home

SNYK ACADEMY

Getting Started with Snyk

Snyk Open Source

Snyk Container

PATTERNS LIBRARY

Amazon Web Services

CircleCI

Sign up for Snyk

Securing Kubernetes Workloads on AWS

Getting started

Create EKS cluster

CircleCI Configuration

config.yml

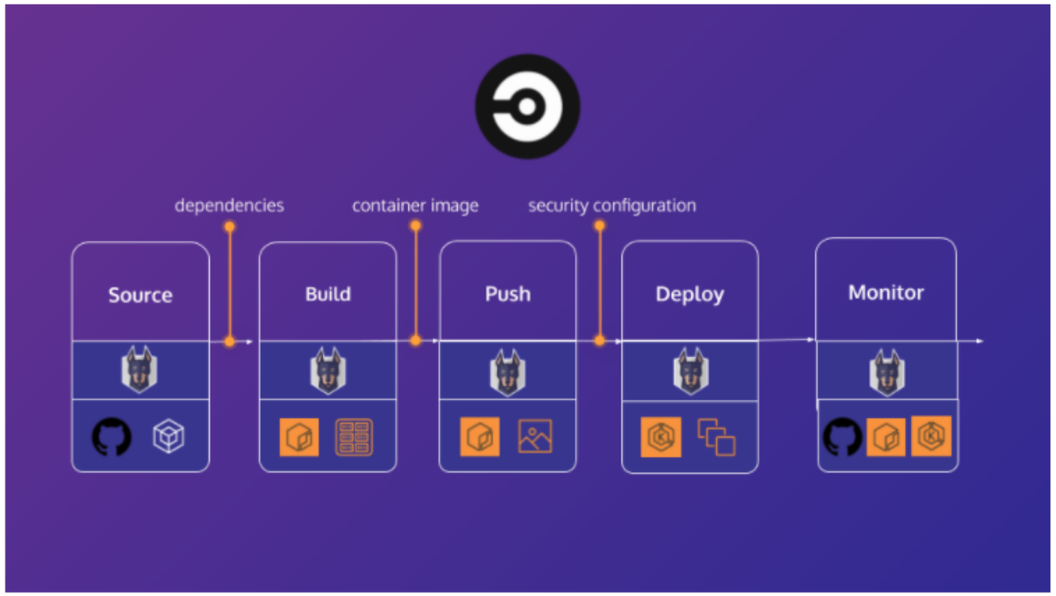
Kubernetes manifests

CircleCI Project

Test deployment

Interpret scan results

Securing Kubernetes Workloads on AWS

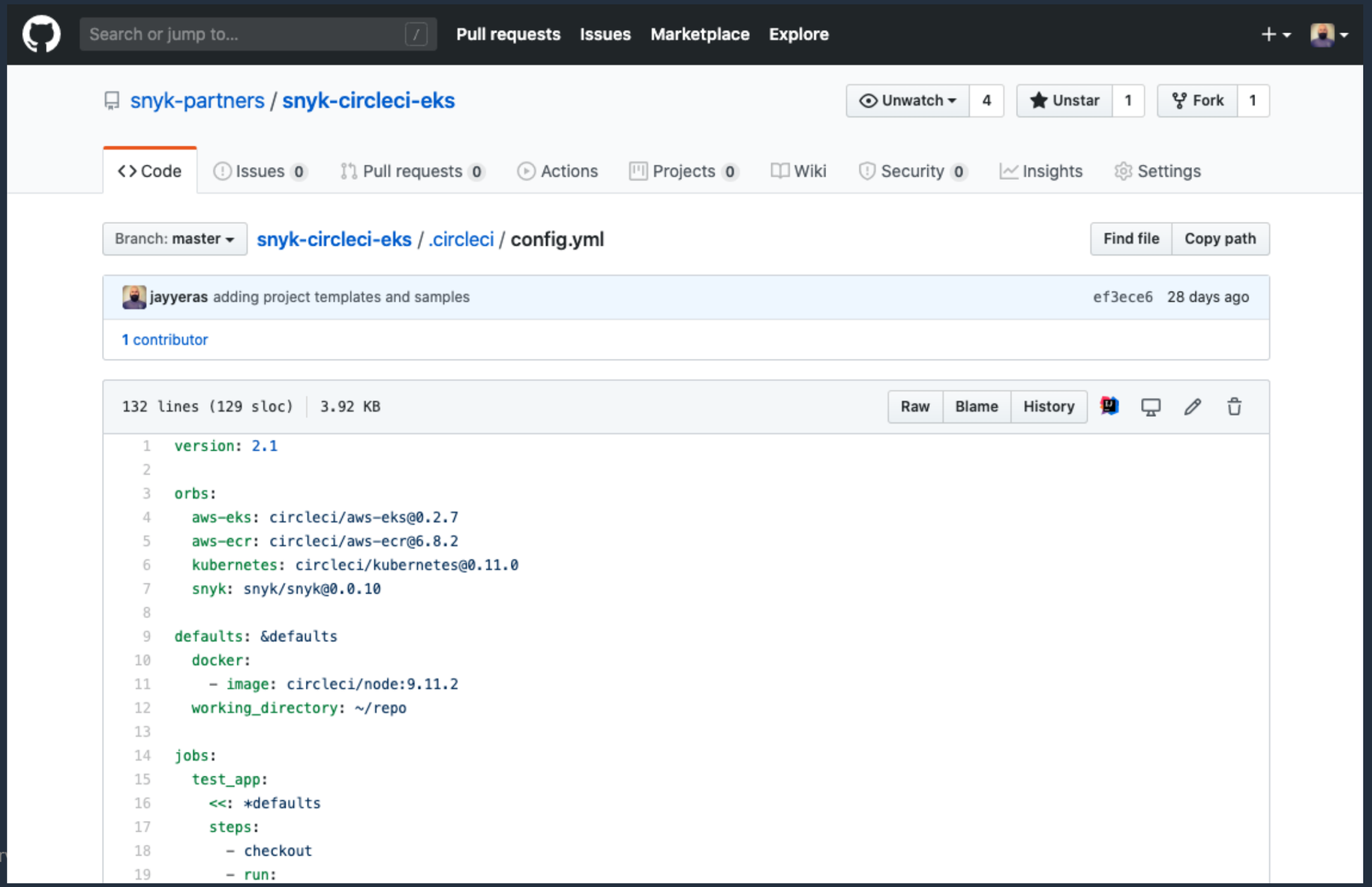


The following exercises will walk you through building a CI/CD pipeline using CircleCI and various Orbs on the AWS Cloud. You will also be guided on how to leverage Snyk to scan each phase of your development process. Along the way, you will receive instruction on how to interpret scan results and the impact of these vulnerabilities through hands-on exercises. Lastly, you will receive guidance on how Snyk can help you fix these.

Let's get started!

aws

Free samples on GitHub!

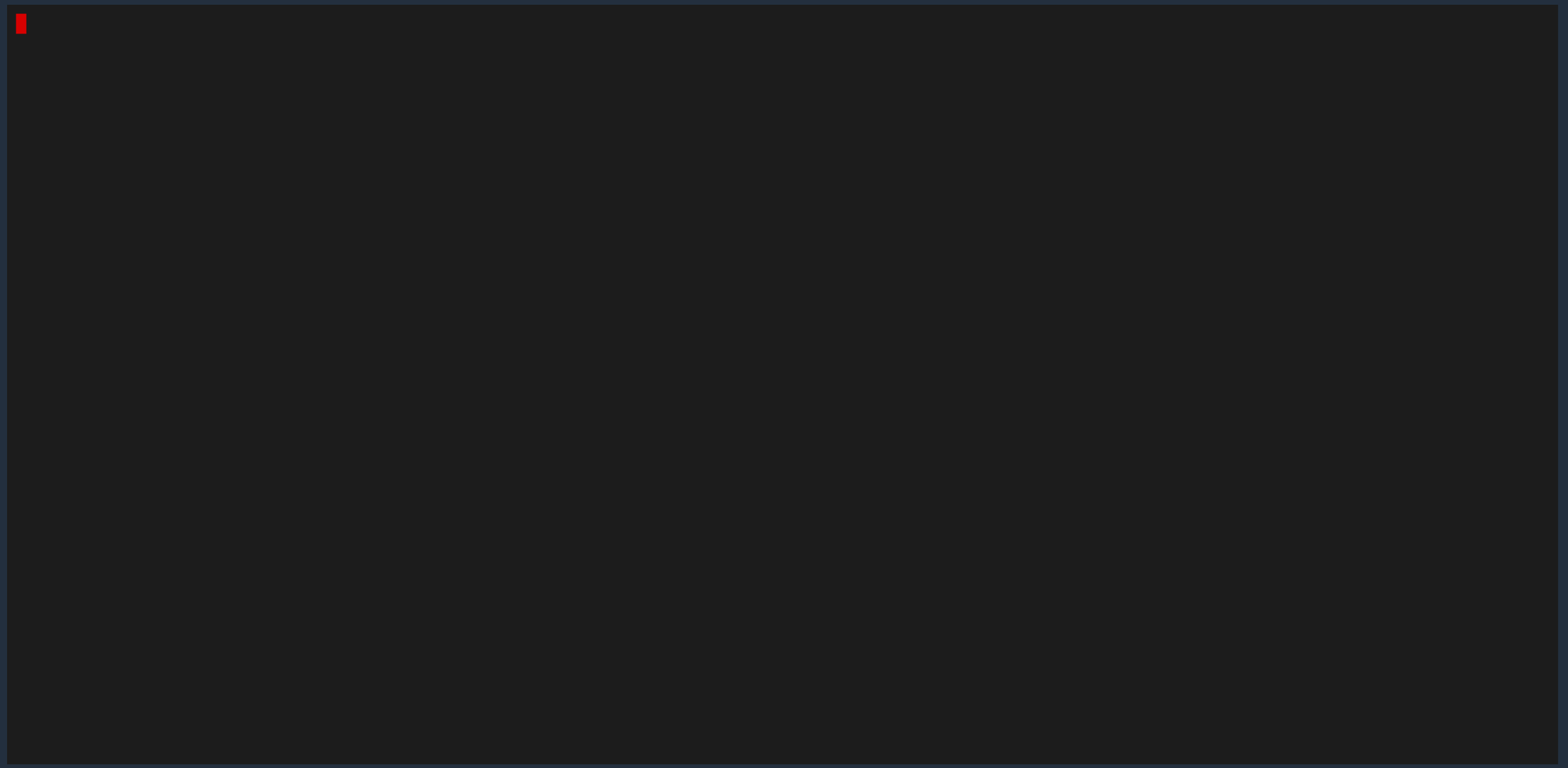


Workshop flow

1. Create an EKS cluster
2. Clone the sample Git repository
3. Create an AWS IAM user
4. Configure your CircleCI project
    a. Define environment variables for AWS & Snyk
5. Enable Snyk integrations
6. Deploy application
7. Find vulnerabilities
8. Fix them! (Homework 😁)

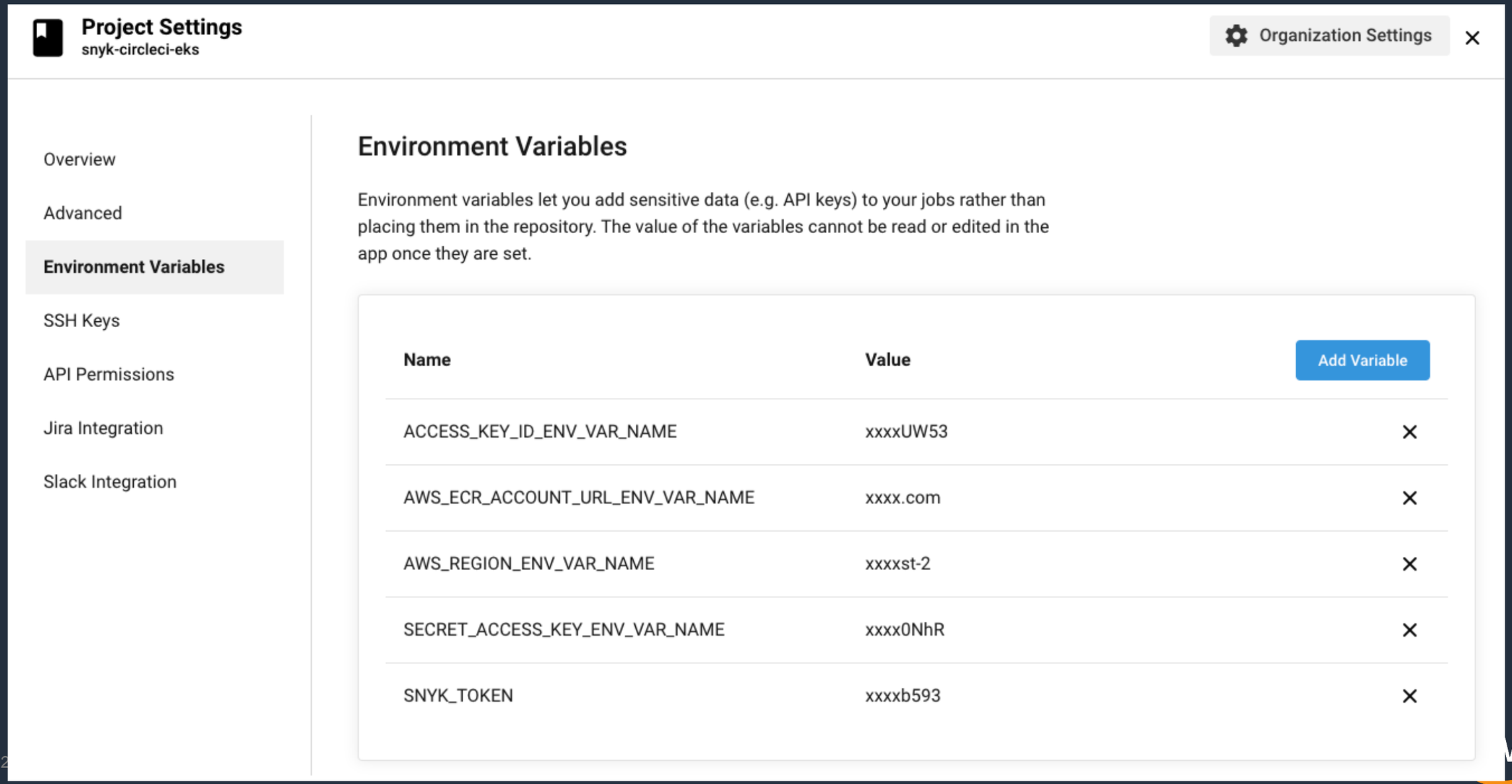
aws

Create EKS cluster



aws

CircleCI project



config.yml

```
version: 2.1


orbs:
 aws-eks: circleci/aws-eks@0.2.7
 aws-ecr: circleci/aws-ecr@6.8.2
 kubernetes: circleci/kubernetes@0.11.0
 snyk: snyk/snyk@0.0.10
```

- CircleCI Certified & Partner Orbs
- Reduce complex lines of code

aws

Application scanning

```
scan_app:
- snyk/scan:
fail-on-issues: false
monitor-on-build: true
project: '${CIRCLE_PROJECT_REPONAME}/${CIRCLE_BRANCH}-app'
severity-threshold: high
token-variable: SNYK_TOKEN
target-file: ./submodules/goof/package.json
```


aws

Container image scanning

```
build_and_scan_image:
- aws-ecr/build-image:
account-url: AWS_ECR_ACCOUNT_URL_ENV_VAR_NAME
dockerfile: Dockerfile
path: ./submodules/goof/
repo: ${CIRCLE_PROJECT_REPONAME}
tag: ${CIRCLE_SHA1}
- snyk/scan:
docker-image-name: '$AWS*/${CIRCLE_PROJECT_REPONAME}:${CIRCLE_SHA1}'
project: '${CIRCLE*}/${CIRCLE_BRANCH}-container'
severity-threshold: high
target-file: ./submodules/goof/Dockerfile
token-variable: SNYK_TOKEN
```


aws

Push to ECR

```
build_and_push_image:
- setup_remote_docker
- aws-ecr/build-and-push-image:
    account-url: AWS_ECR_ACCOUNT_URL_ENV_VAR_NAME
    aws-access-key-id: ACCESS_KEY_ID_ENV_VAR_NAME
    aws-secret-access-key: SECRET_ACCESS_KEY_ENV_VAR_NAME
    region: AWS_REGION_ENV_VAR_NAME
    repo: ${CIRCLE_PROJECT_REPONAME}
    create-repo: true
    checkout: true
    dockerfile: Dockerfile
    path: ./submodules/goof/
    tag: ${CIRCLE_SHA1}
```

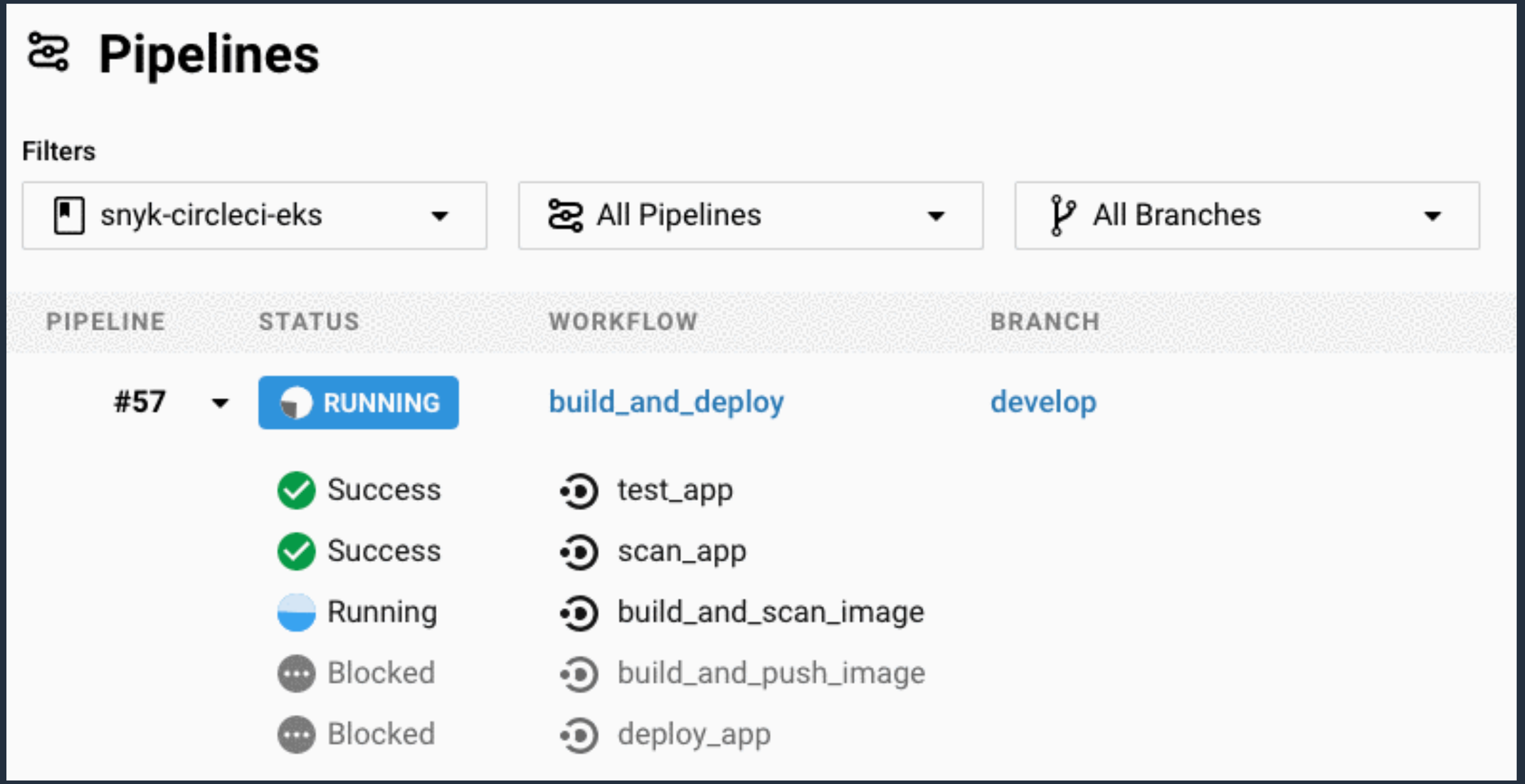

aws

Deploy to EKS

```
deploy_app:
steps:
- run:
    command: |
      BUILD_DATE=$(date '+%Y%m%d%H%M%S')
      cat deployment/goof-deployment-template.yaml |\
        sed "s|DOCKER_IMAGE_NAME|<< parameters.docker-image-name >>|g"\
          > deployment/goof-deployment.yaml
- kubernetes/create-or-update-resource:
    resource-file-path: "deployment/goof-deployment.yaml"
- kubernetes/create-or-update-resource:
    resource-file-path: "deployment/goof-service.yaml"
```

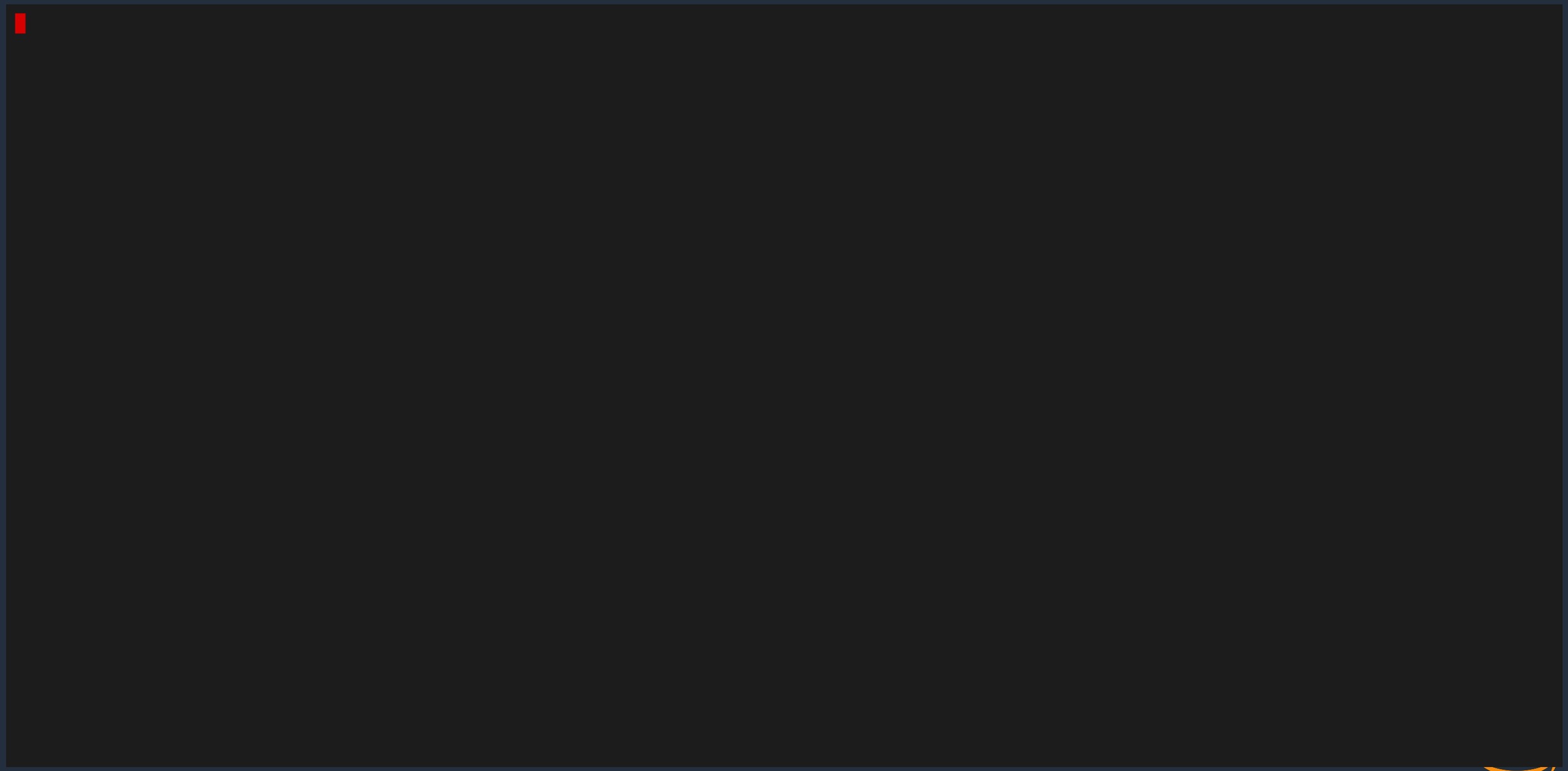

aws

CircleCI pipeline run!

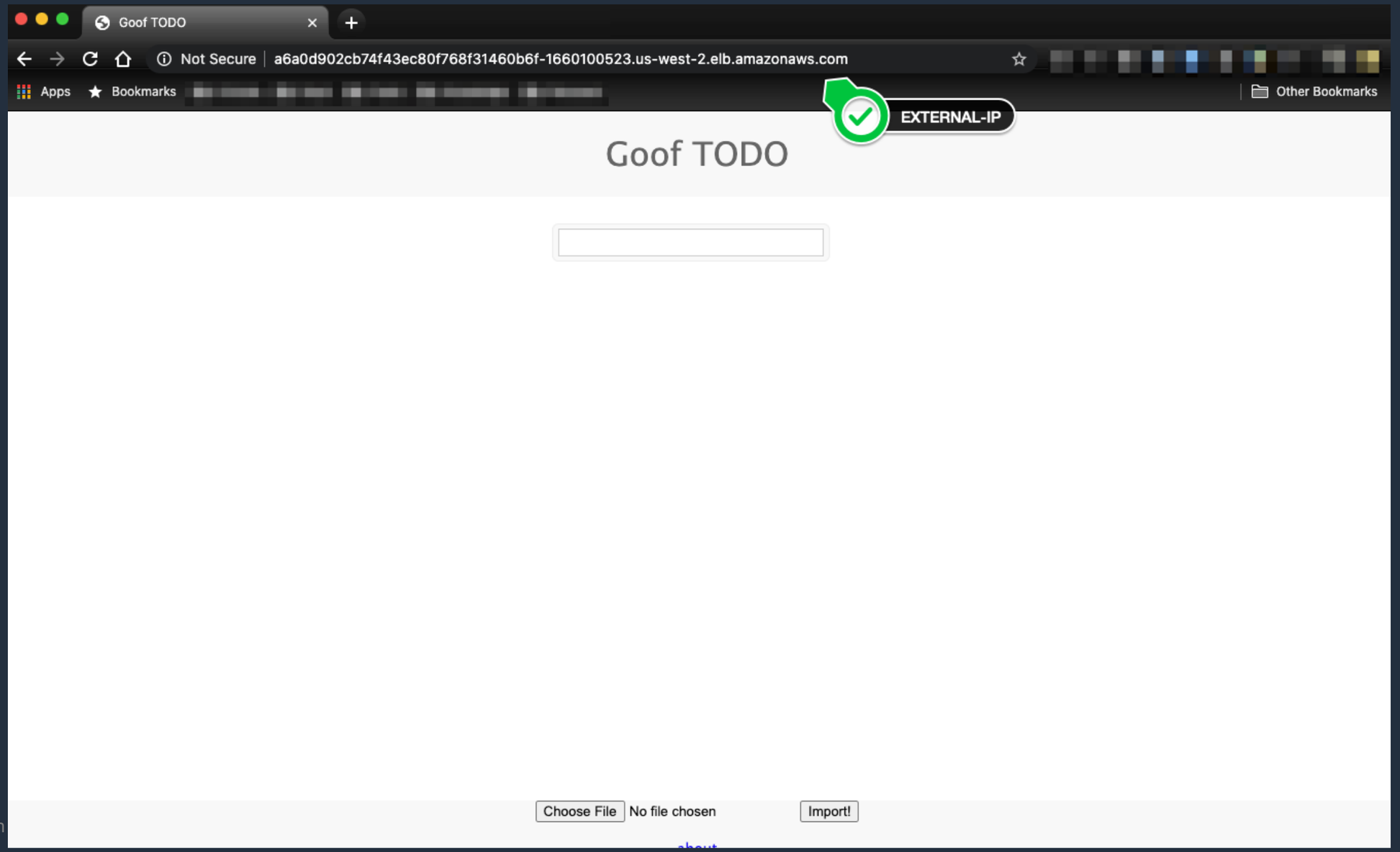


aws

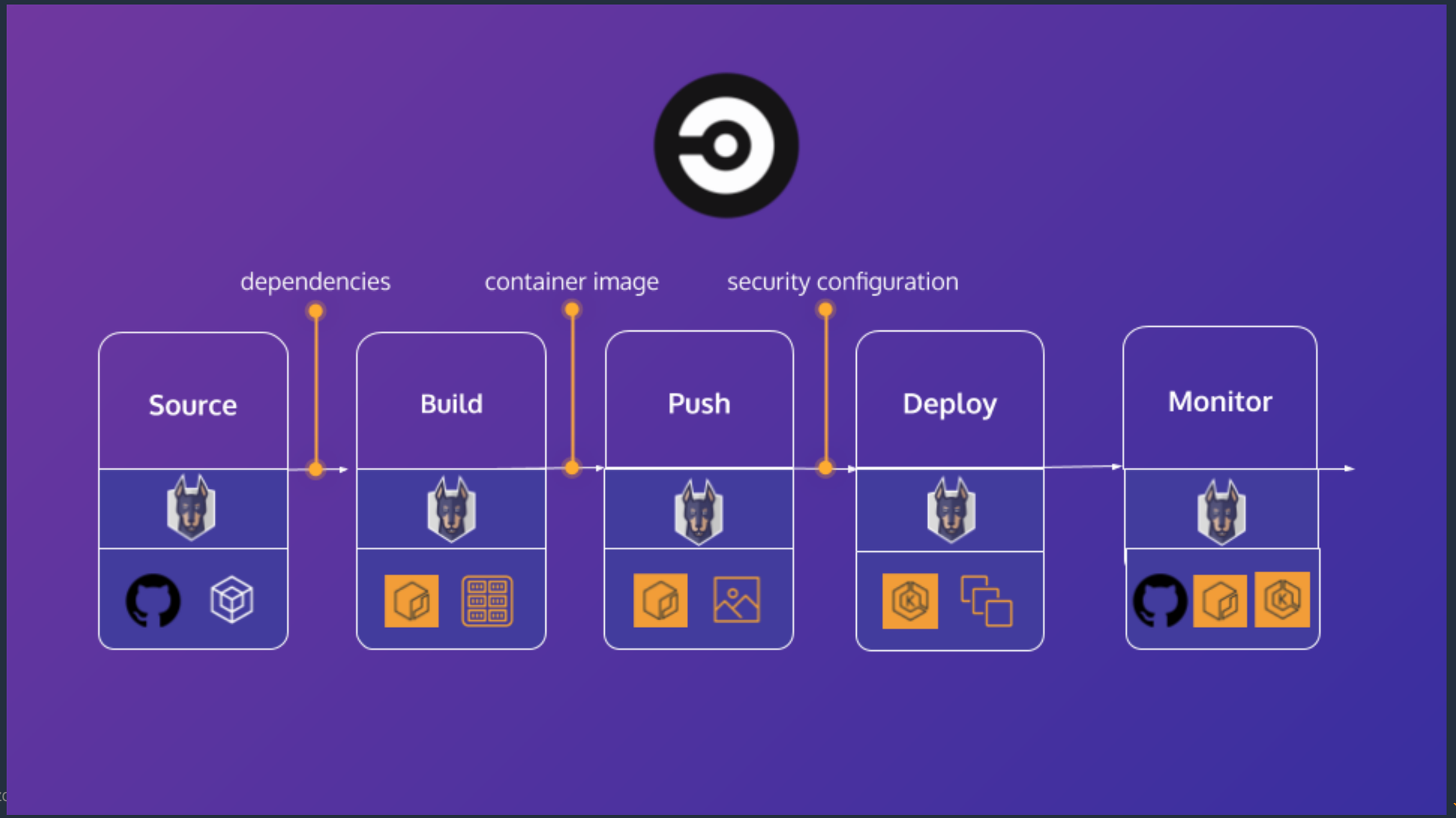
Get external IP



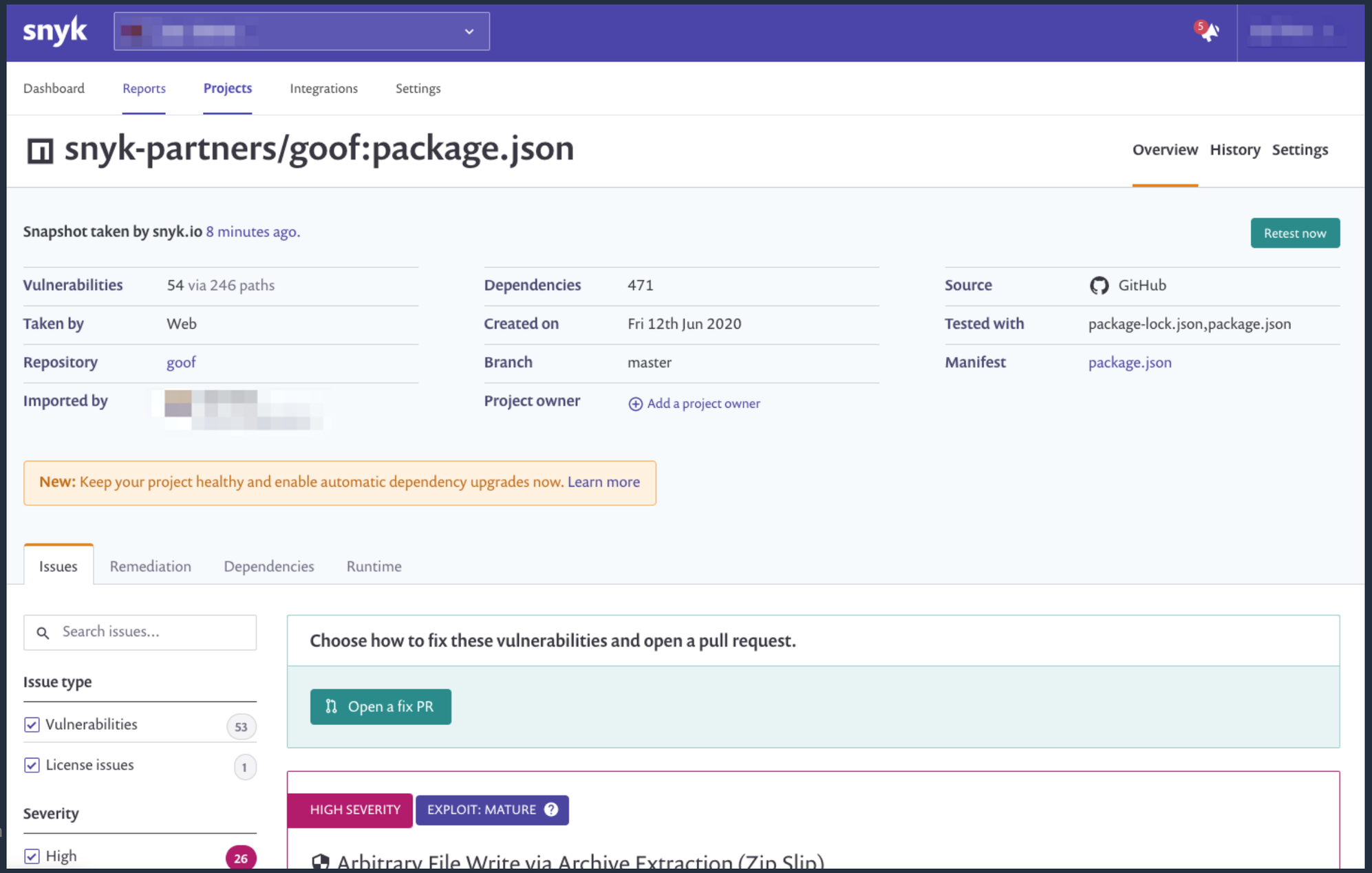
Our Goof application



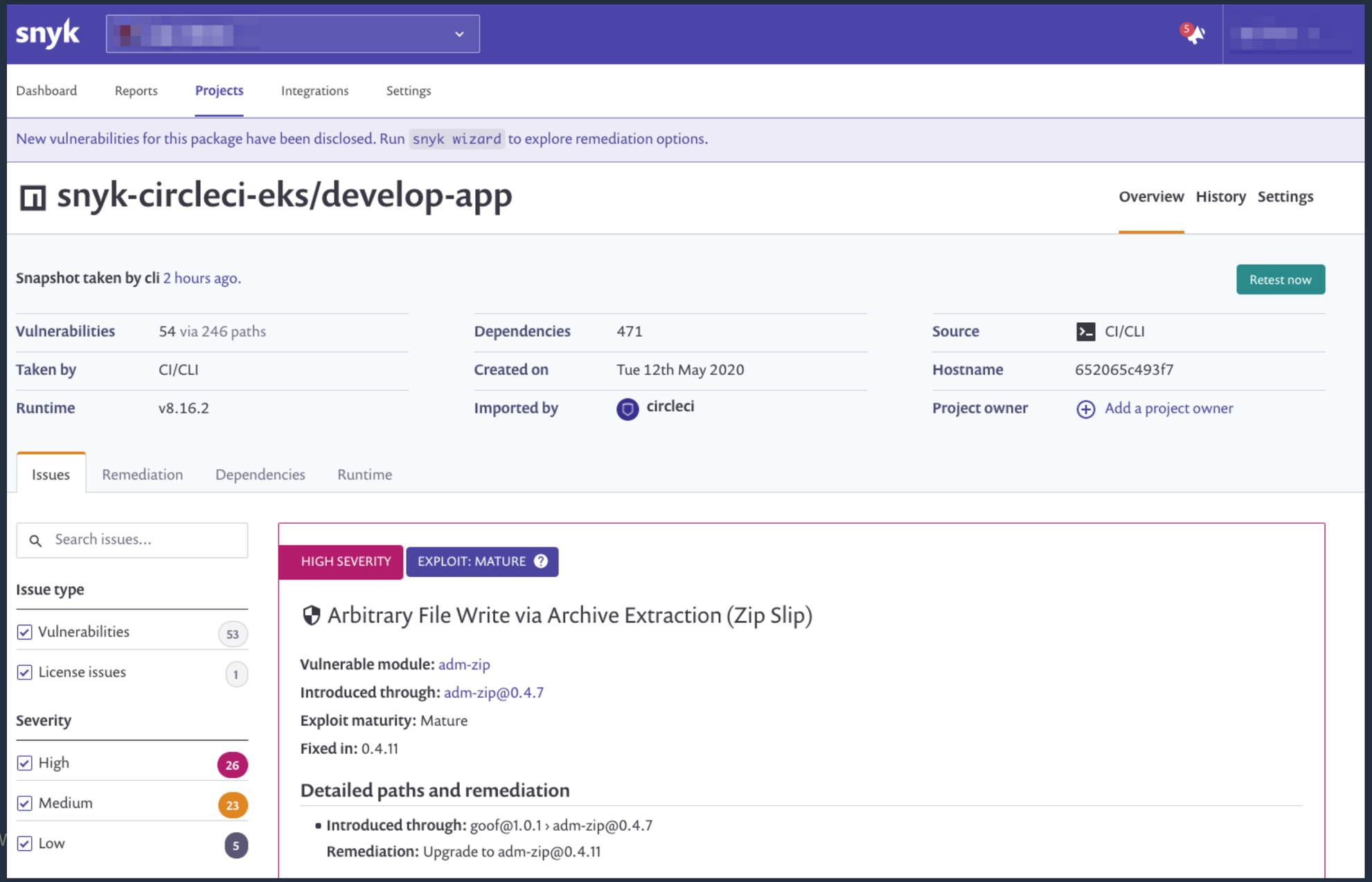
Remember our flow?



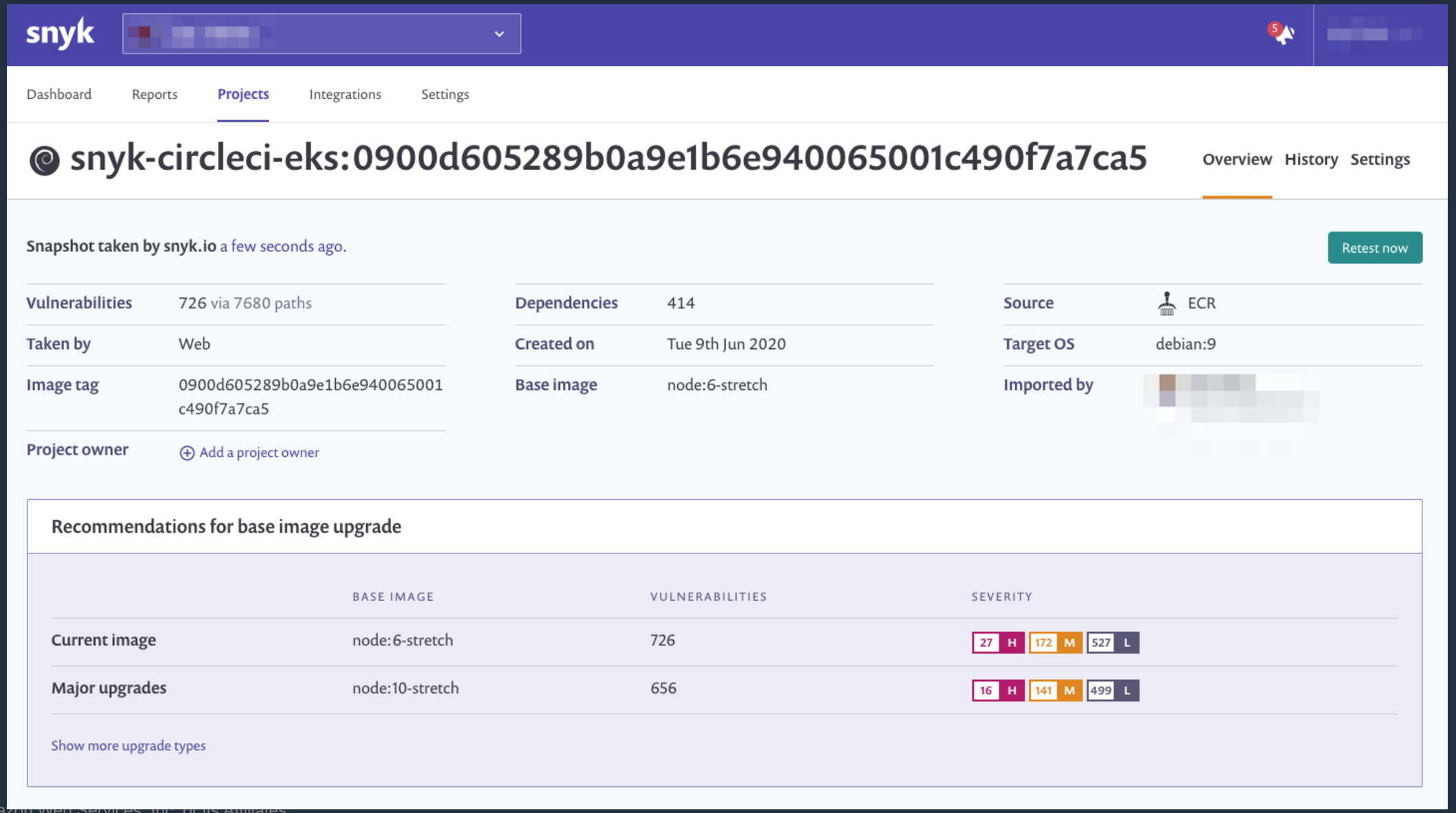
GitHub



Application scan

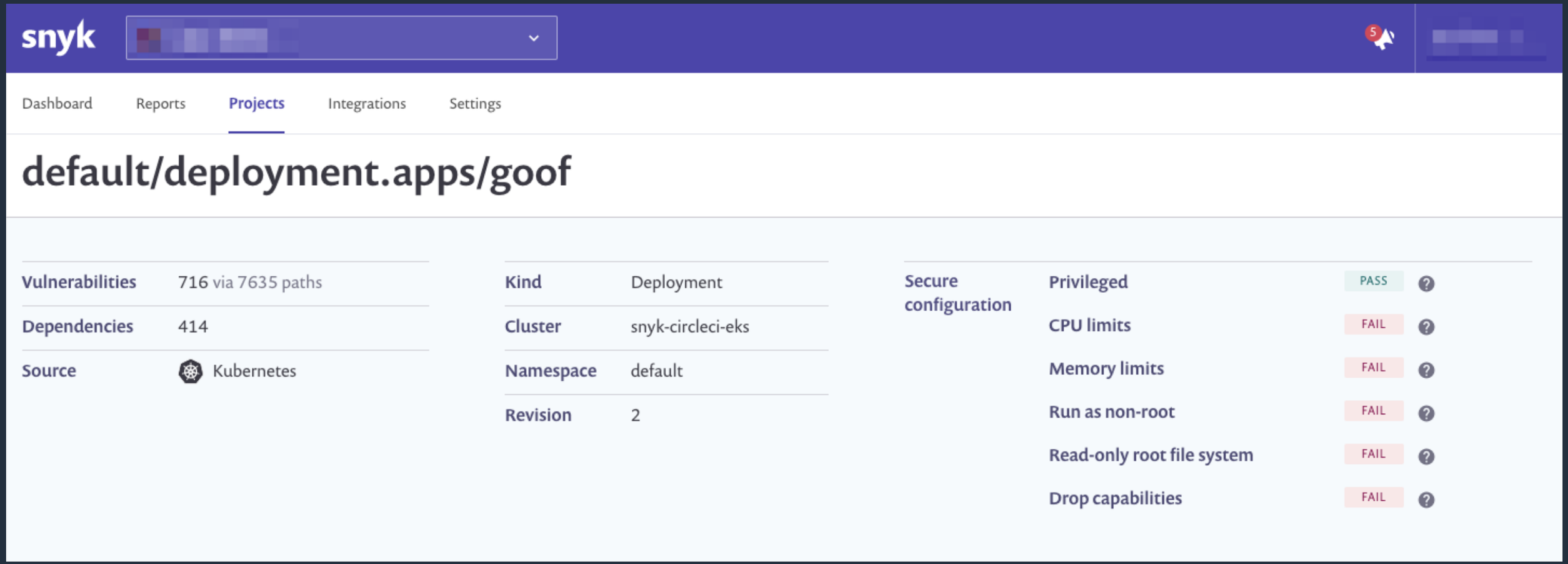


Container registry scan



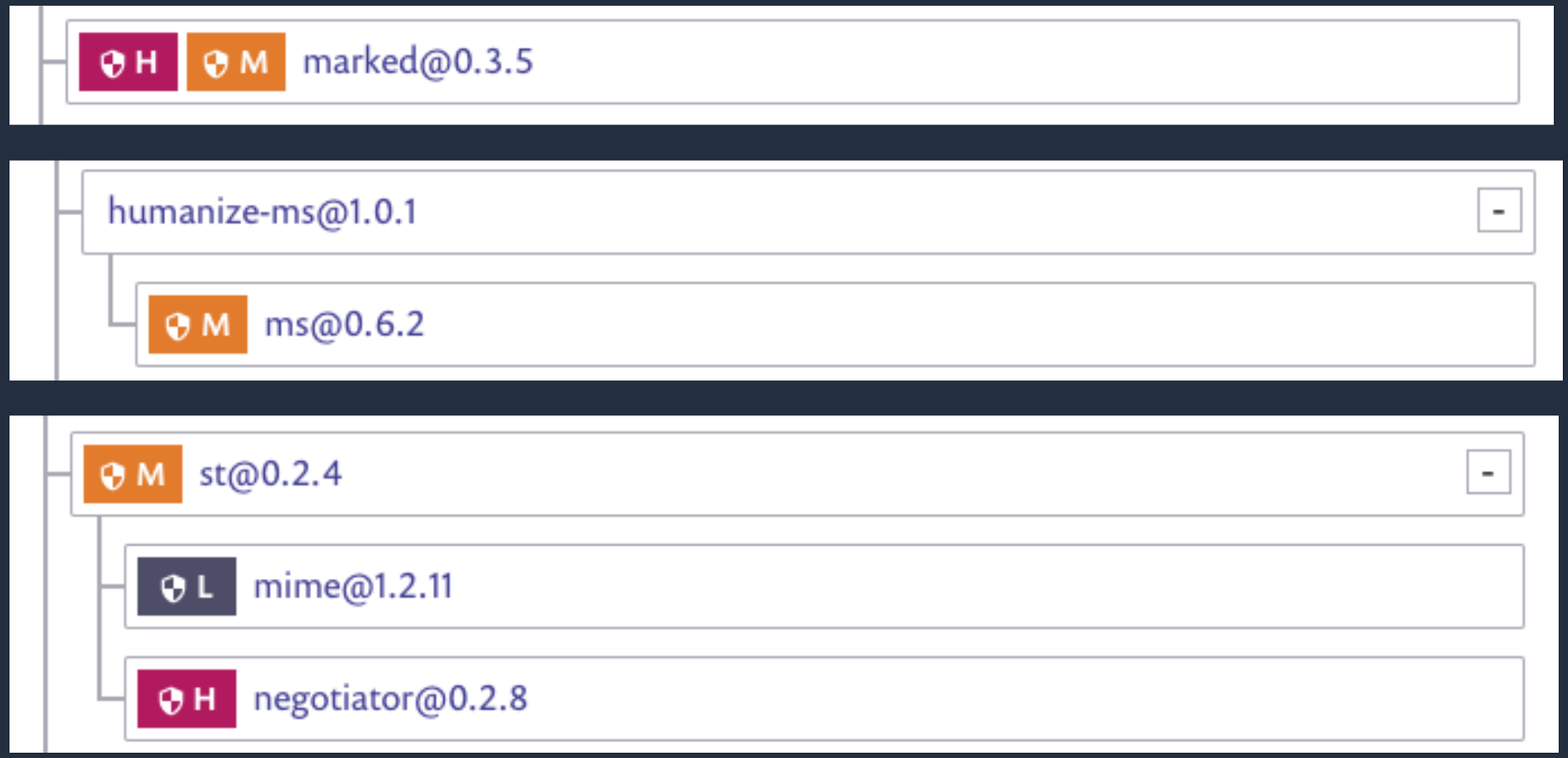
aws

Kubernetes scan



aws

Three example vulnerabilities

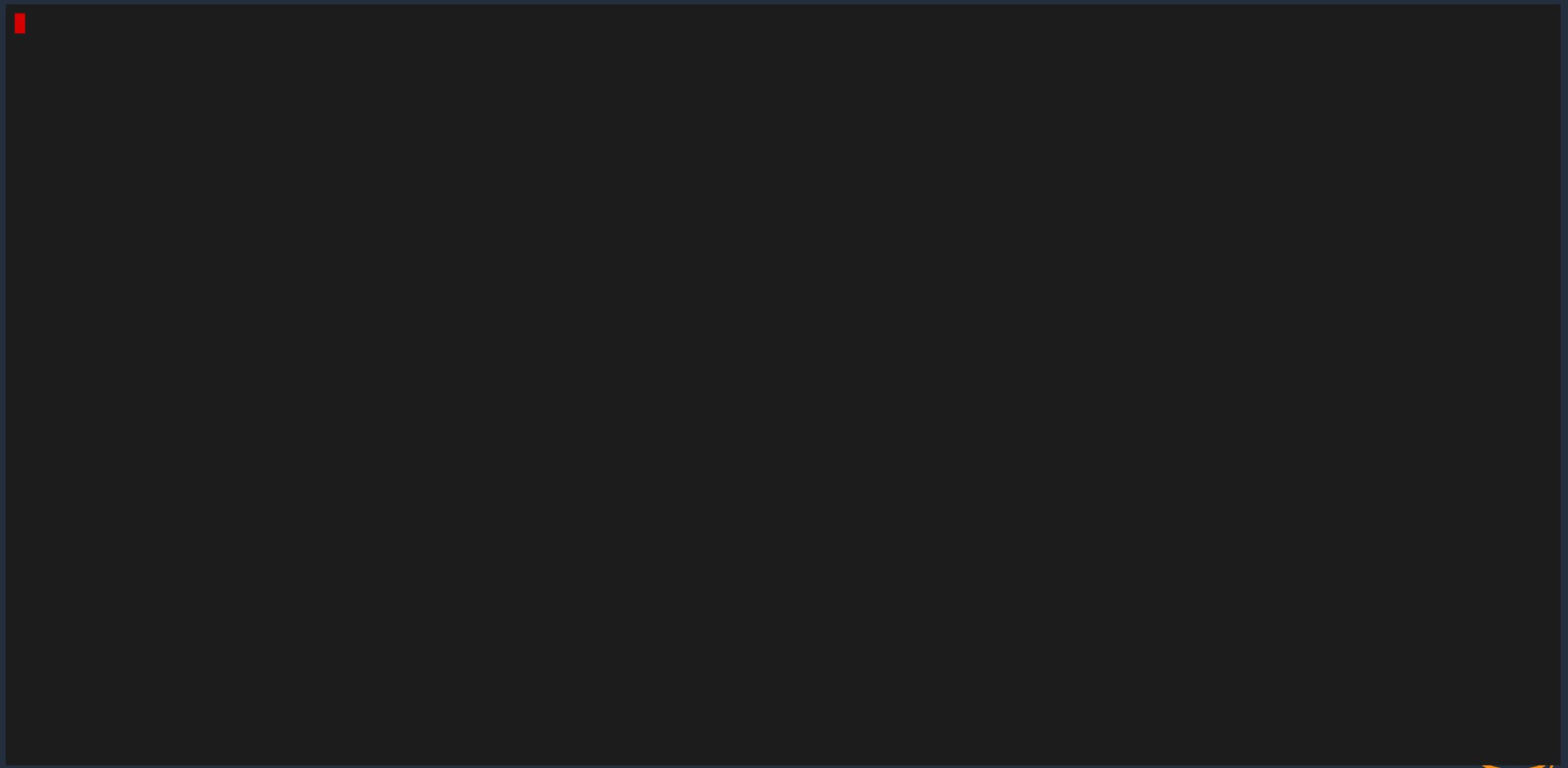


aws

Static pages & the st package vulnerability

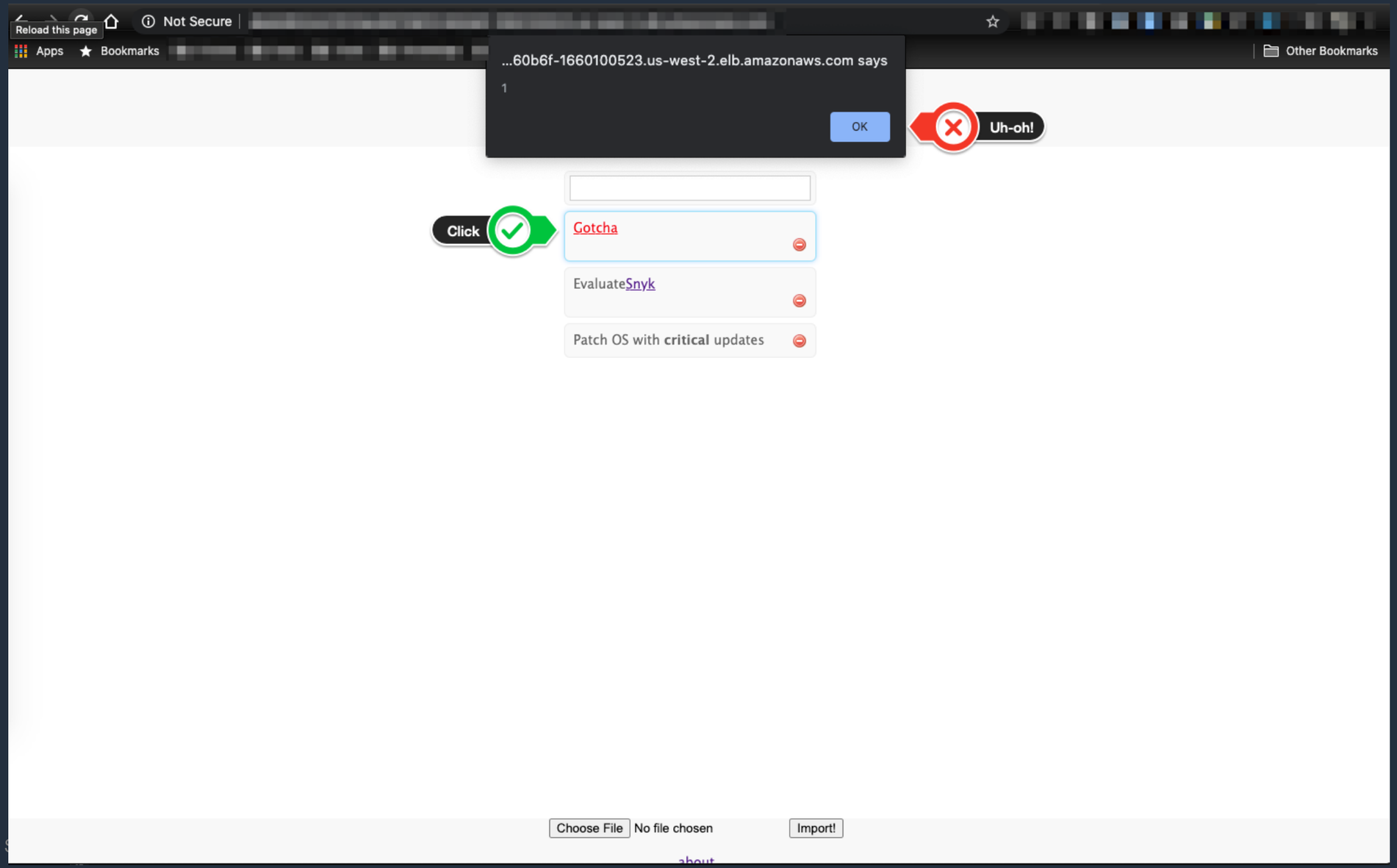
aws

Directory traversal

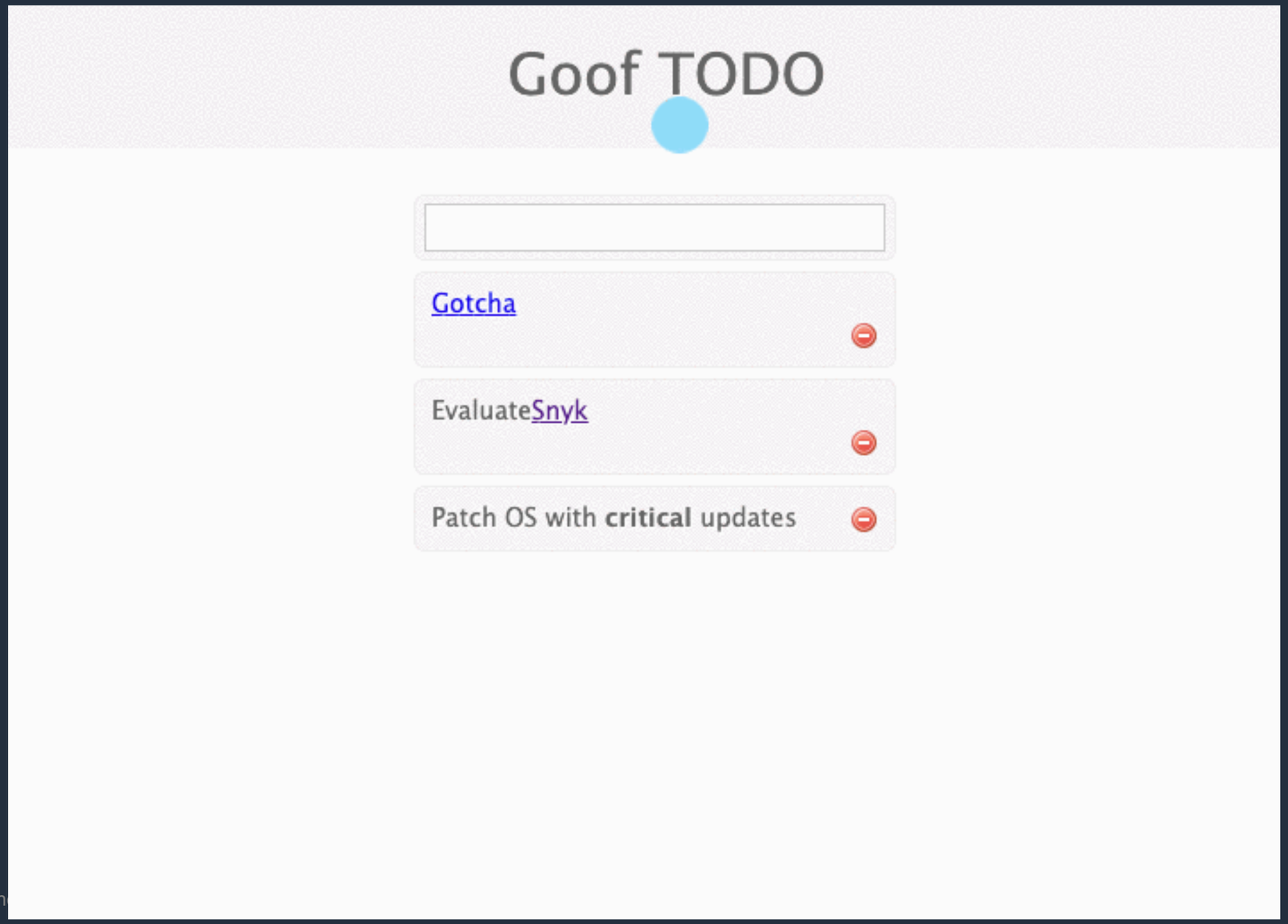


Markdown with marked package

Cross-site Scripting (XSS)



Time format conversion with ms package



aws

(ReDos) Regular Expression Denial of Service

```
echo 'content=Reboot server in '\`printf
"%.0s5" {1..60000}\`' minutea' | http --
form $GOOF_HOST/create -v
```

aws

What did we learn?

- It’s easy to build security into your workflows.
- Secure your pipeline end to end.
- Monitor configuration drifts that may expose you to potential attacks.
- Finding vulnerabilities is great, fixing them before it’s a problem is better!

aws

Where do you go to learn more?

https://solutions.snyk.io/

aws

Q&A

Thank You