



OWASP

Open Web Application
Security Project

Dracon - Knative Security pipelines

Or how I learned to love the cloud

Itinerary

- \$id
- Qs
- The environment
- The challenge
- Existing Solutions
- Requirements
- Chosen solution
- Demo
- Advantages
- Future work

\$id

Spyros

- Security engineer
- Appsec
- @0xfde



Henry

- Backend engineer
- Blue teamer
- @steakunderscore



Thought Machine

Qs

- Security testing in release pipeline?
- Support more than 4 languages?
- Both static and dynamic testing?
- Config per team?
- Less than 10h per week?

The environment



Microservices



Client - specific



Devops



Kubernetes



Traceability

The challenge



Multi-language



Usable by engineers




Quick release cycle

Existing solutions

Combinations of:

 Static and dynamic analysis

 CI environments + plugins

 Open source

 Scripting

Requirements



On cloud and local



Environment agnostic



Scanners output to various locations.



Mix and match.



Must. play well. with. CI

Introducing Dracon

Security pipelines on Kubernetes

The stack

Kubernetes



Tekton

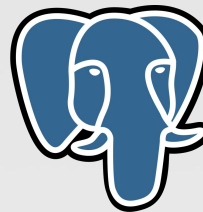


TEKTON

Protobufs

Postgres

Go & Python



Background



Tekton, Knative



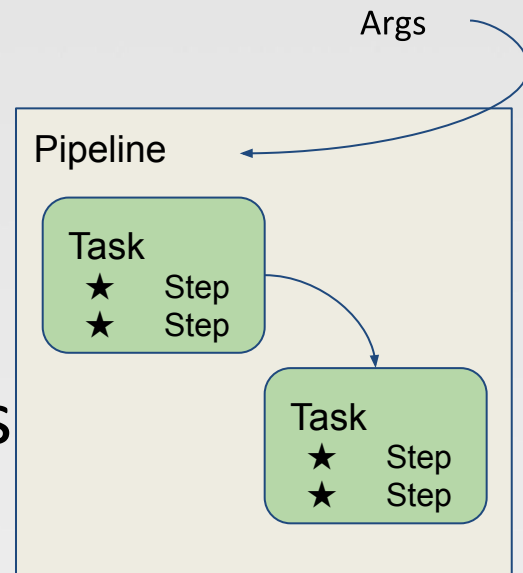
Pipelines with tasks



Tasks with steps



Taskruns with specific arguments



API

Protobufs - Files defining message format

- LaunchTool
- LaunchToolResponse
- Issue
- Vulnerability

API

Protobufs

- LaunchTool
- LaunchToolResponse
- Issue
- Vulnerability

```
message Issue {  
    string target = 1;  
    string type = 2;  
    string title = 3;  
    Severity severity = 4;  
    double cvss = 5;  
    Confidence confidence = 6;  
    string description = 7;  
}
```

Orchestrating

-i <init-step> -iA <init-args>

-p <producer1> -pA <producer-args1>

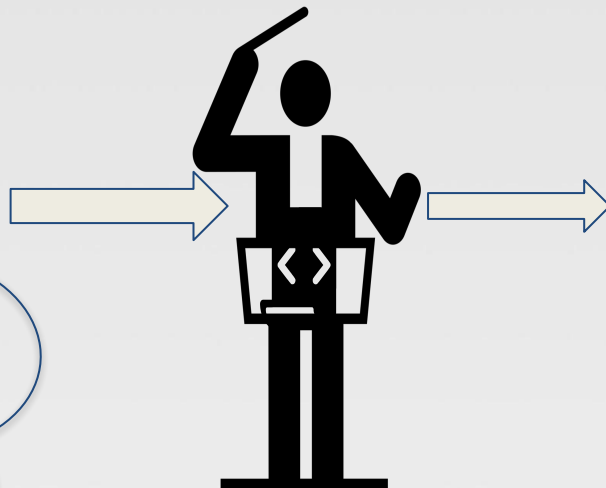
-p <producer2> -pA <producer-args2>

-p <producer3> -pA <producer-args3>

-e enricher

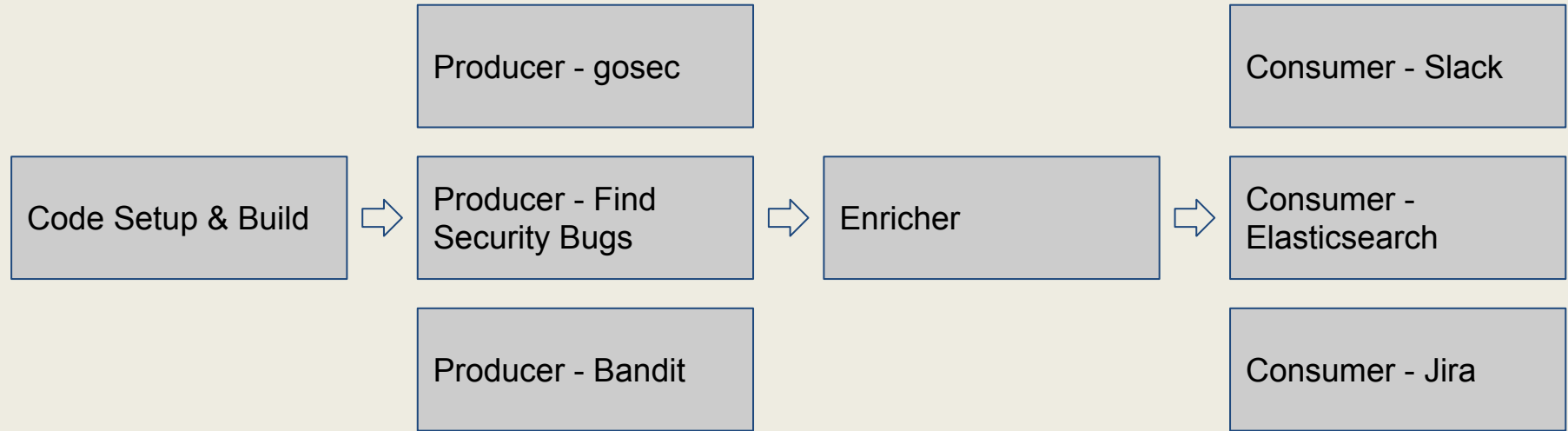
-c <consumer1> -cA <consumer-args1>

-c <consumer2> -cA <consumer-args2>



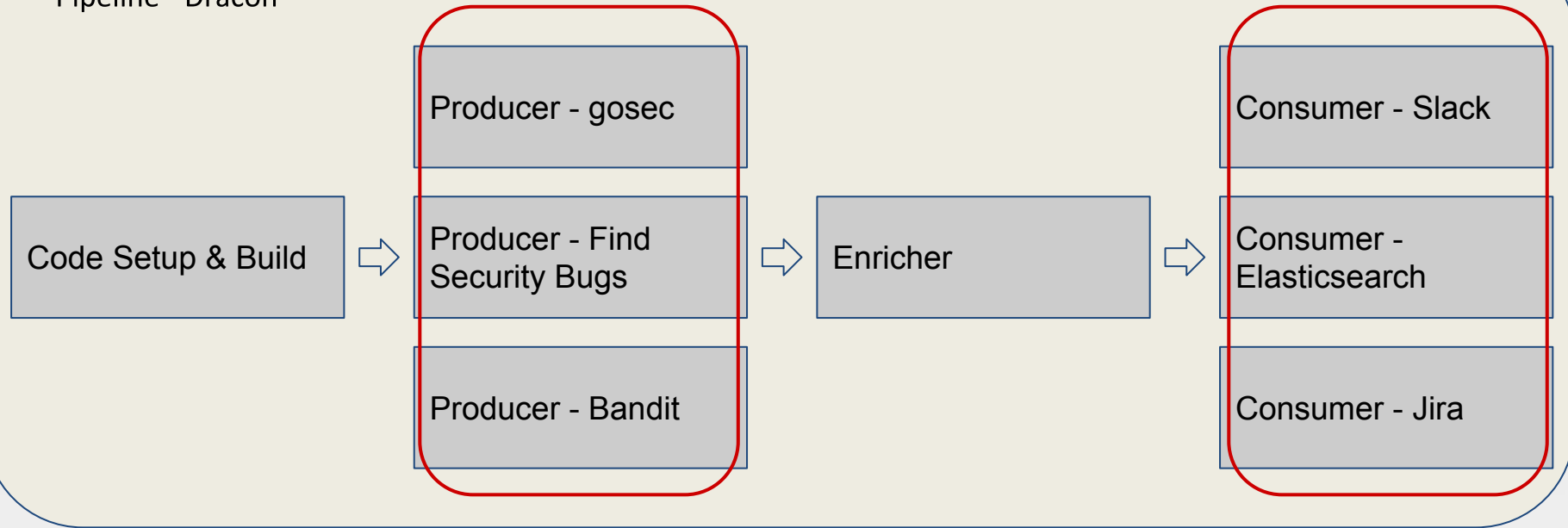
Architecture

Pipeline - Dracon



Architecture

Pipeline - Dracon

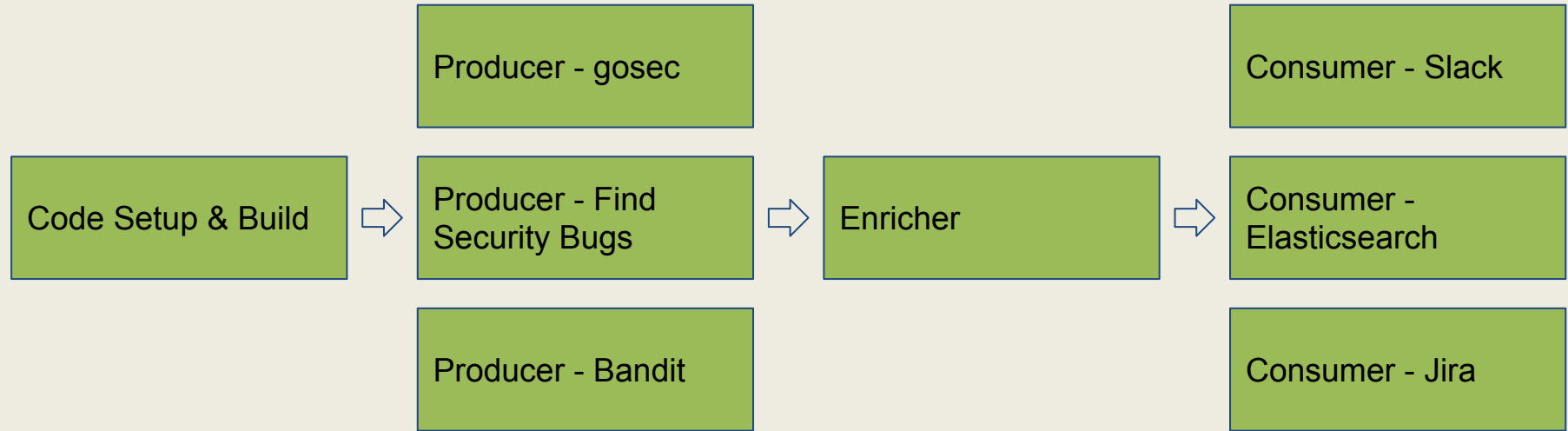


Producers & Consumers

- Gosec
- Find Security Bugs
- Bandit
- OWASP Dependency Check
- Pip Safety
- Elasticsearch
- Slack
- Defect Dojo
- Phabricator

Architecture

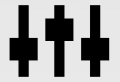
Pipeline - Dracon



Catalogue



Tasks



Producers: wrapped tools producing protobufs

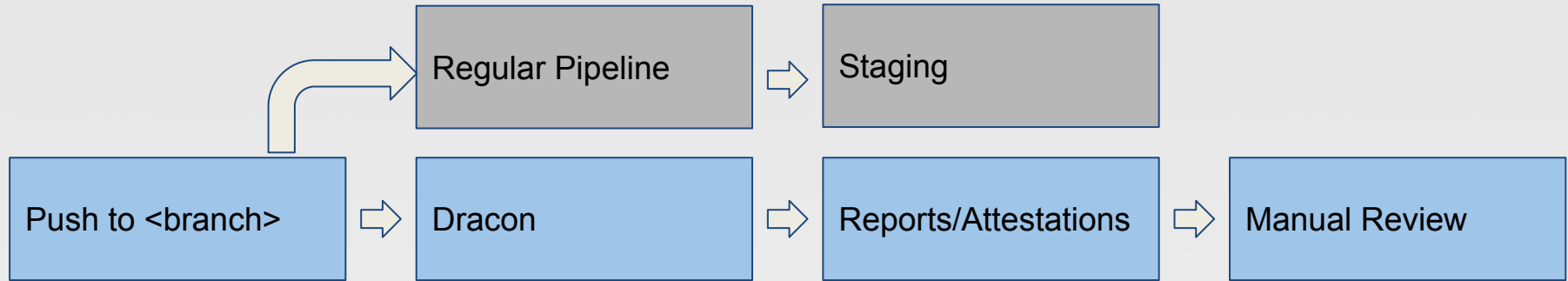


Consumers: wrapped tools consuming

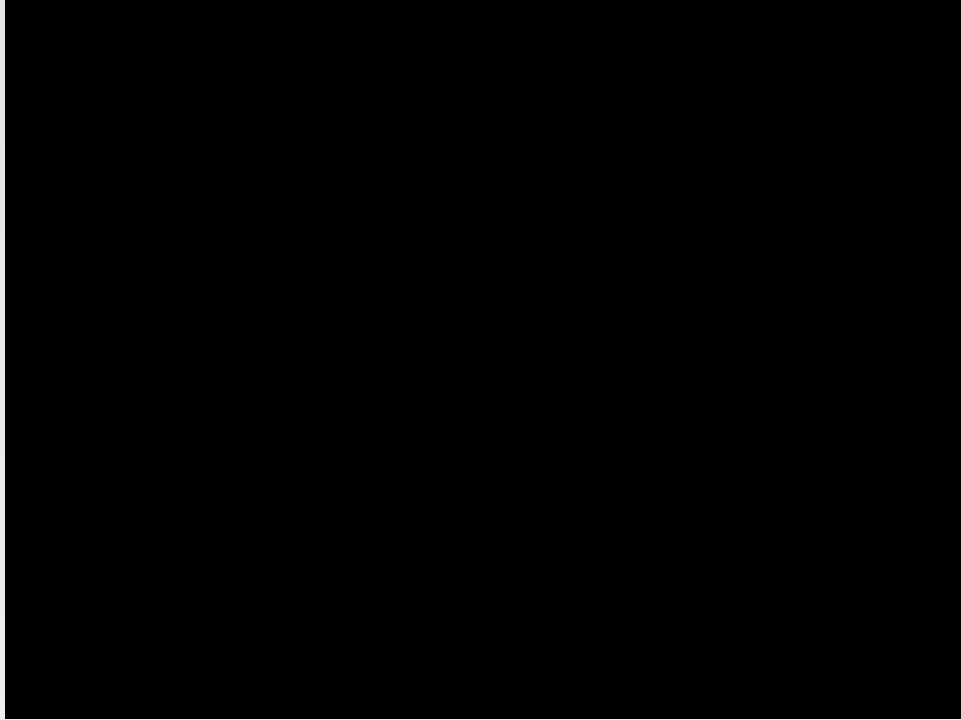
Where does Dracon fit in?

- Could run after your CI builds?
- Should we block code merges?
- Or what if we prevent releases?

Pipeline Fit



DeMo



Advantages

- Mix and match tasks
- Wrapping tools doable
- Runs anywhere*
- CI friendly

Future

- ++Images
- Active scans, Zap
- Extensible message format

Standing on the shoulders of giants

Thanks to all the work to get us here

- Tekton
- Kubernetes
- Postgres
- Elasticsearch
- Gosec, Bandit, Find Sec Bugs
- OWASP community
- and many more...



Open-sourcing in progress

Watch this space..

Q&A

github.com/thought-machine/dracon

@0xfde

@steakunderscore

Thank you