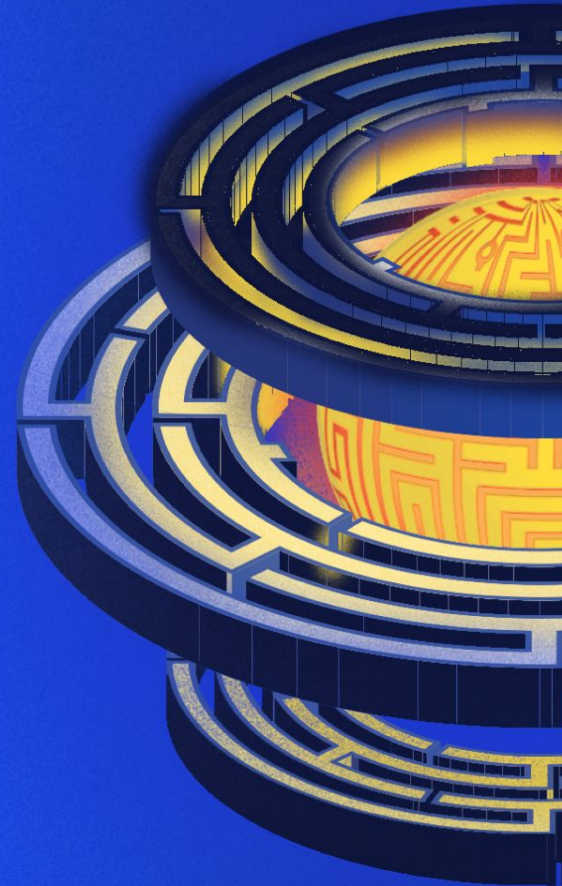


Building a dynamic IDP: A reference architecture for Azure-focused setups



Agenda

“Making off” the ref architecture

01

Problems we aim to solve

02

Design Principles

03

Design

04

End to end walk-through

05



And buzzwords turn into reality - ♥magic♥;

Let's explore some "golden paths"!

Making off the ref architecture

- ◆ Years of bullshit have to end. Platform engineering is becoming a zoo of buzzwords.
- ◆ We had a ton of data but not enough to make this representative.
- ◆ McKinsey took on the task and we contributed.
- ◆ It's started with a ref architecture on AWS and GCP. Today we are discussing the Azure one.



100s

Of setups used as a basis
for this ref architecture

Problems we want to solve

Overwhelmed

- ✗ Long lead times
- ✗ Ticket ops, high cost of maintenance
- ✗ Overwhelmed developers that slow down
- ✗ Waiting times & missing self-service

9/10 operations or DevOps teams are wasting time because of a unstructured tooling setup.

Overdelivered

- ✓ Low lead time
- ✓ High degree of standardization
- ✓ Separation of concern
- ✓ Self-service

Treat your platform as a product, build an Internal Developer Platform.

If you get the fundamentals right, the benefits walk in

Dynamic
Configuration
Management

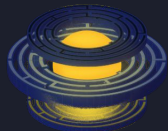
Drives
standardization
by design

Enables:

- Dev self-service
- Elimination of "ticket ops"
- Reduction of config files
- Golden Paths
- Low cognitive load

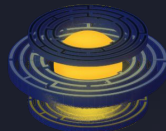
**Slash
your
lead
time!**

Silent legends



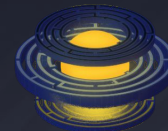
Stephan Schneider

APV at McKinsey focussed on engineering excellence and developer experience.



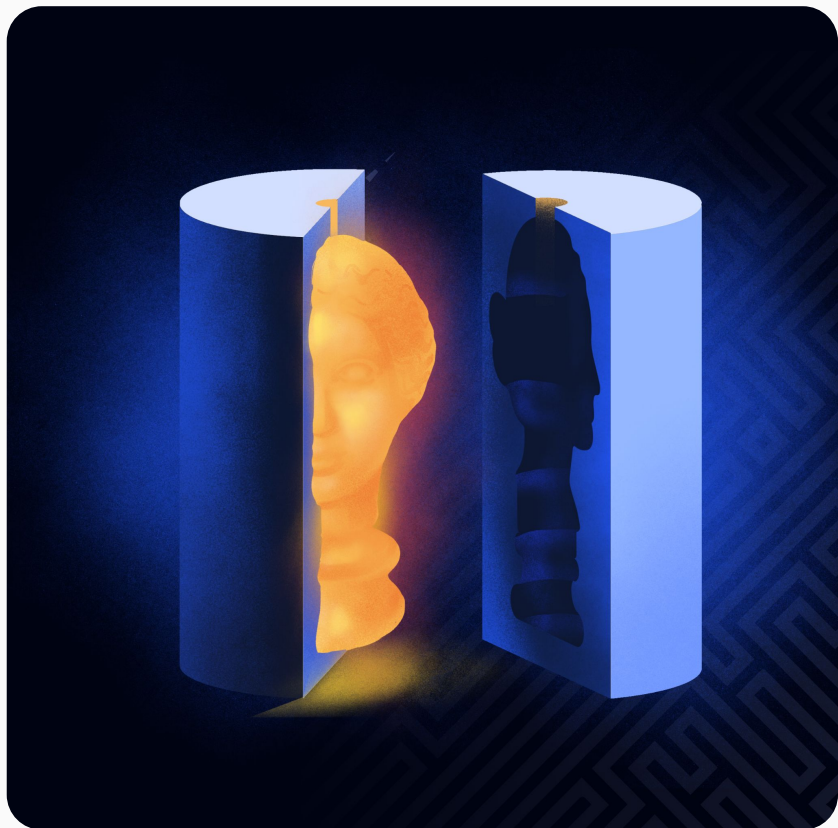
Mike Gatto

Senior Platform Engineer focussed on the AWS ecosystem.



Marco Marulli

Senior Platform Engineer focussed on the GCP ecosystem.



What is a ref architecture?

A standard pattern of most commonly used architectural designs of different tools to deliver software. Combined by platform engineers into Internal Developer Platform. Ref architecture comes as:

- ◆ Visual flow diagrams
- ◆ Packaged as code
- ◆ Whitepapers
- ◆ Tutorials

Design principles



Golden paths over cages

Pull developers, do not push them. If you abstract, never take context.



Standardization by design

By using the platform, the degree of standardization stays constant or increases.



Dynamic over static configs

The platform should be able to dynamically create configs with every deployment.

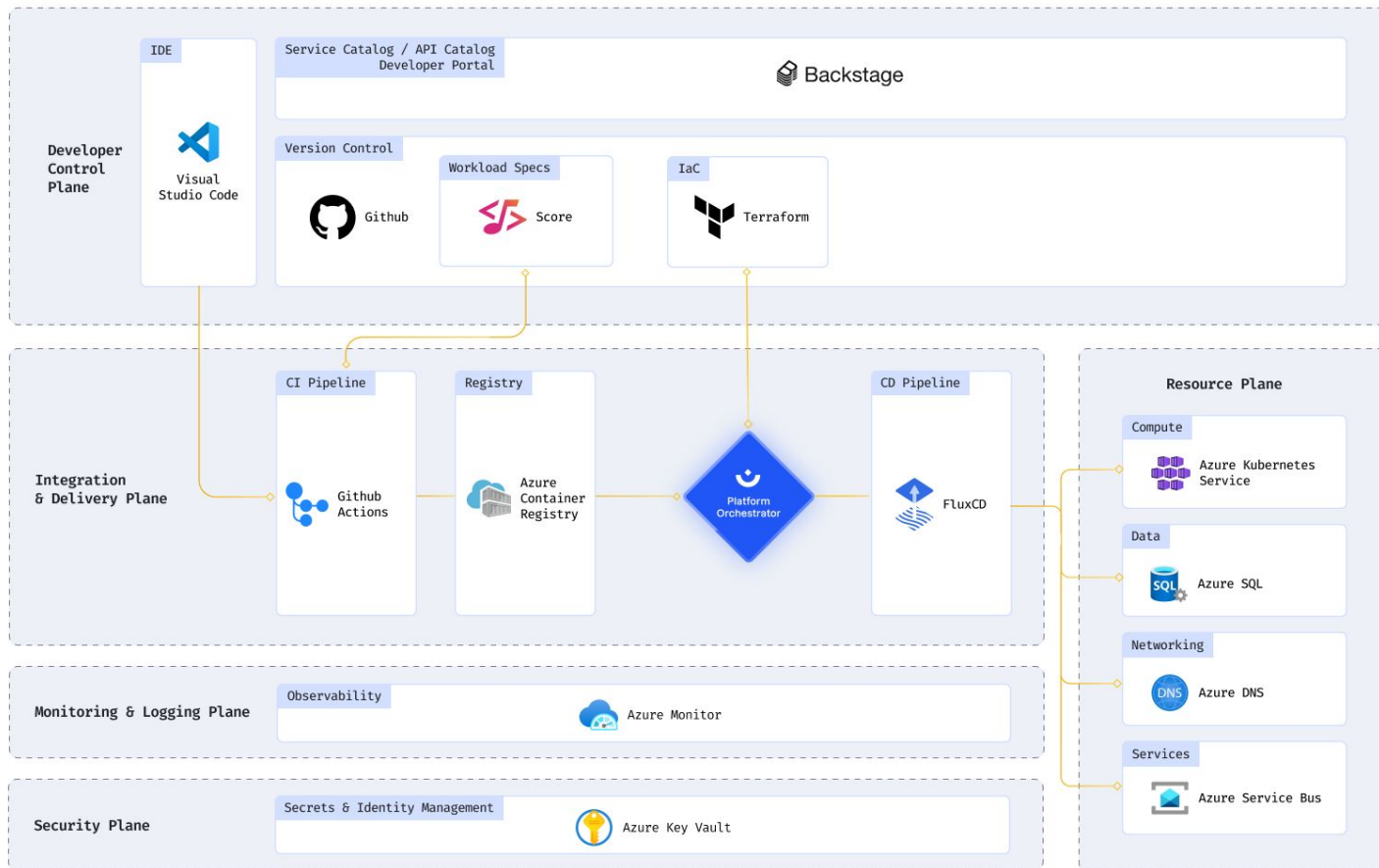


Code first / interface choice

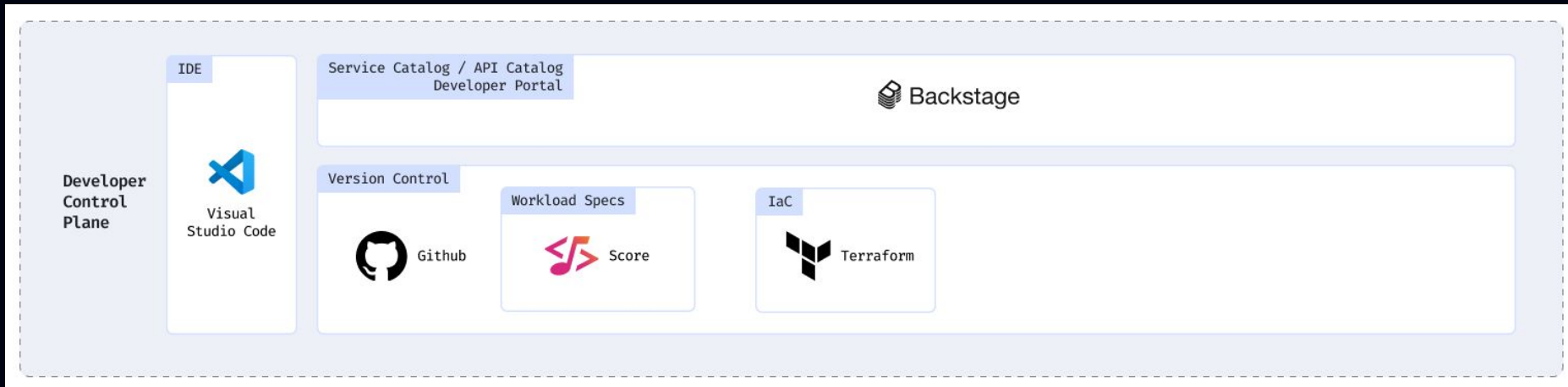
Code should be the single source of truth. Users should have interface choice.



Internal Developer Platform on Azure Cloud



Developer Control Plane



Interface choice - and it depends!

Activity type	Predominant interface choice
Deploy	Terminal/IDE
Change configuration	Code: Workload specification (Score)
Add/remove resource	Code: Workload specification (Score)
Roll back/Diff	Orchestrator API/CLI/UI
Configure resource in detail	Code: IaC
Spin up a new environment	Orchestrator API/CLI/UI
See logs/error messages	Orchestrator API/CLI/UI
Search service catalog	Portal/Service Catalog
Inner source use case	Portal/Service Catalog
Scaffolding Use case	Portal/Service Catalog or templating in VCS

Workload specification - a centerpiece

```
score.yaml

apiVersion: score.dev/v1b1

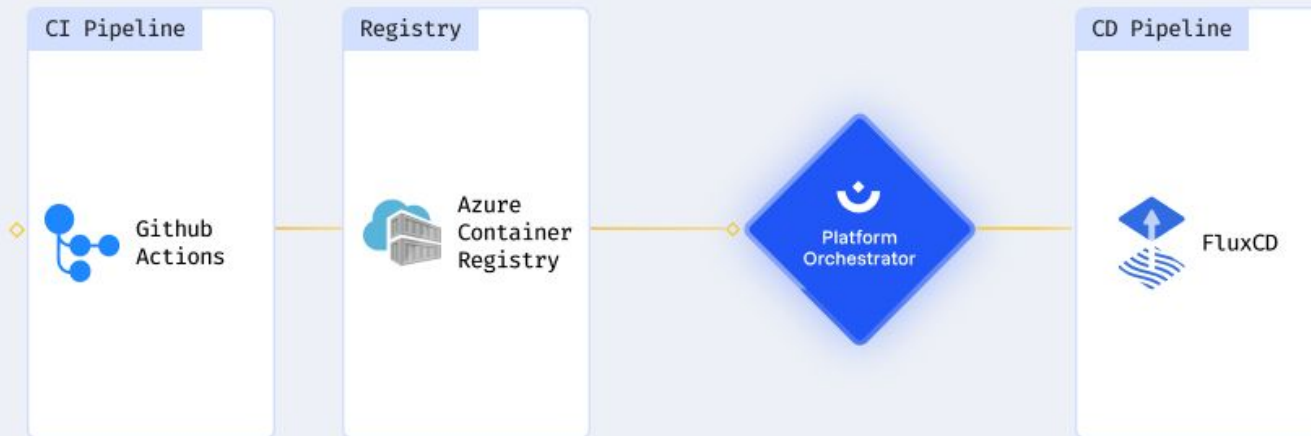
metadata:
  name: python-service

containers:
  python-service:
    image: python
    variables:
      CONNECTION_STRING: postgresql://${resources.db.user}

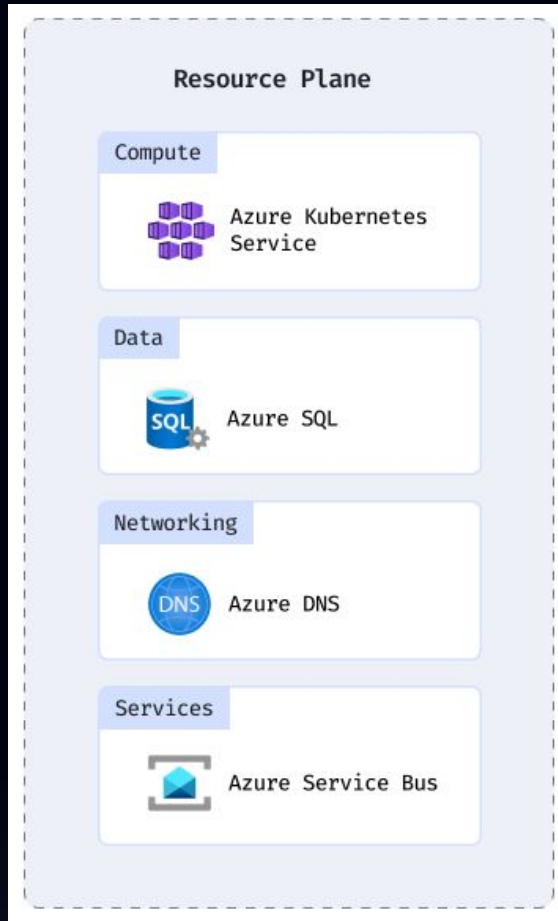
resources:
  db:
    type: postgres
  storage:
    type: s3
  dns:
    type: dns
```

Integration & Delivery Plane

Integration
& Delivery Plane



Resource Plane



Monitoring & Logging Plane

Monitoring & Logging Plane

Observability



Azure Monitor

Security Plane

Security Plane

Secrets & Identity Management



Azure Key Vault

How this works

Git-push

Developers

```
scoreyaml
apiVersion: score.dev/v1b1
metadata:
  name: python-service
containers:
  python-service:
    image: python
  variables:
    CONNECTION_STRING: postgres://${resources.db.username}
resources:
  db:
    type: postgres
  storage:
    type: s3
  dns:
    type: dns
```

Score file and workload name

Platform Engineers

```
resource "humanitec_resource_definition" "postgres" {
  id      = "db-dev"
  name    = "db-dev"
  type    = "postgres"
  driver_type = "humanitec/postgres-cloudsql-static"
  driver_inputs = {
    values = {
      "instance" = "test:test:test"
      "name"     = "db-dev"
      "host"     = "127.0.0.1"
      "port"    = "5432"
    }
  }
  secrets = {
    "username" = "test"
    "password" = "test"
  }
}
criteria = [
  {
    app_id = "test-app"
  }
]
```

Baseline Configurations

Deploy
Read
Match: env type = staging
Create

CI



CD

Application deployed

Golden path: deploy to dev

(dev perspective)

Dev
Request

```
score.yaml
apiVersion: score.dev/v1b1
metadata:
  name: python-service
containers:
  python-service:
    image: python
    variables:
      CONNECTION_STRING: postgresql://${resources.db.user}
resources:
  db:
    type: postgres
  storage:
    type: s3
  dns:
    type: dns
```

context:
env=development



Platform
response

- ✓ **Read** workload specification
- ✓ **Match** resource definitions
- ✓ **Create** app configs, configure resources
- ✓ **Deploy**

- EKS cluster configured
- RDS credentials injected
- S3 credentials injected
- Route 53 DNS configured

Golden path: create new environment (dev perspective)

Dev
Request

```
score.yaml
apiVersion: score.dev/v1b1
metadata:
  name: python-service
containers:
  python-service:
    image: python
    variables:
      CONNECTION_STRING: postgresql://${resources.db.user}
resources:
  db:
    type: postgres
  storage:
    type: s3
  dns:
    type: dns
```

context:
env=ephemeral



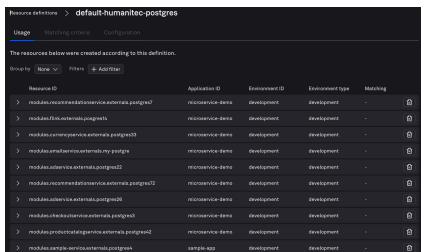
Platform
response

- ✓ **Read** workload specification
- ✓ **Match** resource definitions
- ✓ **Create** app configs, configure resources
- ✓ **Deploy**

- Create new namespace
- Create RDS
- Create S3
- Create DNS entry

Golden path: Update Postgres from V 14 -> 15 (Platform Engineer)

Platform Request



Resource ID	Application ID	Environment ID	Environment type	Marking
resource:postgres:external:postgres1	resource:deva	development	development	-
resource:postgres:external:postgres2	resource:deva	development	development	-
resource:postgres:external:postgres3	resource:deva	development	development	-
resource:postgres:external:postgres4	resource:deva	development	development	-
resource:postgres:external:postgres5	resource:deva	development	development	-
resource:postgres:external:postgres6	resource:deva	development	development	-
resource:postgres:external:postgres7	resource:deva	development	development	-
resource:postgres:external:postgres8	resource:deva	development	development	-
resource:postgres:external:postgres9	resource:deva	development	development	-
resource:postgres:external:postgres10	resource:deva	development	development	-
resource:postgres:external:postgres11	resource:deva	development	development	-
resource:postgres:external:postgres12	resource:deva	development	development	-
resource:postgres:external:postgres13	resource:deva	development	development	-
resource:postgres:external:postgres14	resource:deva	development	development	-
resource:postgres:external:postgres15	resource:deva	development	development	-
resource:postgres:external:postgres16	resource:deva	development	development	-
resource:postgres:external:postgres17	resource:deva	development	development	-
resource:postgres:external:postgres18	resource:deva	development	development	-
resource:postgres:external:postgres19	resource:deva	development	development	-
resource:postgres:external:postgres20	resource:deva	development	development	-
resource:postgres:external:postgres21	resource:deva	development	development	-
resource:postgres:external:postgres22	resource:deva	development	development	-
resource:postgres:external:postgres23	resource:deva	development	development	-
resource:postgres:external:postgres24	resource:deva	development	development	-
resource:postgres:external:postgres25	resource:deva	development	development	-
resource:postgres:external:postgres26	resource:deva	development	development	-
resource:postgres:external:postgres27	resource:deva	development	development	-
resource:postgres:external:postgres28	resource:deva	development	development	-
resource:postgres:external:postgres29	resource:deva	development	development	-
resource:postgres:external:postgres30	resource:deva	development	development	-
resource:postgres:external:postgres31	resource:deva	development	development	-
resource:postgres:external:postgres32	resource:deva	development	development	-
resource:postgres:external:postgres33	resource:deva	development	development	-
resource:postgres:external:postgres34	resource:deva	development	development	-
resource:postgres:external:postgres35	resource:deva	development	development	-
resource:postgres:external:postgres36	resource:deva	development	development	-
resource:postgres:external:postgres37	resource:deva	development	development	-
resource:postgres:external:postgres38	resource:deva	development	development	-
resource:postgres:external:postgres39	resource:deva	development	development	-
resource:postgres:external:postgres40	resource:deva	development	development	-
resource:postgres:external:postgres41	resource:deva	development	development	-
resource:postgres:external:postgres42	resource:deva	development	development	-
resource:postgres:external:postgres43	resource:deva	development	development	-
resource:postgres:external:postgres44	resource:deva	development	development	-
resource:postgres:external:postgres45	resource:deva	development	development	-
resource:postgres:external:postgres46	resource:deva	development	development	-
resource:postgres:external:postgres47	resource:deva	development	development	-
resource:postgres:external:postgres48	resource:deva	development	development	-
resource:postgres:external:postgres49	resource:deva	development	development	-
resource:postgres:external:postgres50	resource:deva	development	development	-

What services depend on the resource definition?

```
resource "humanitec_resource_definition" "postgres" {
  id      = "db-dev"
  name    = "db-dev"
  type    = "postgres"
  driver_type = "humanitec/postgres-cloudsql-static"

  driver_inputs = {
    values = {
      "instance" = "test:test1"
      "name"     = "db-dev"
      "host"     = "127.0.0.1"
      "port"     = "5432"
    }
  }

  secrets = {
    "username" = "test"
    "password" = "test"
  }

  criteria = [
    {
      app_id = "test-app"
    }
  ]
}
```

Update resource definition



→ Postgres version update rolled out across all dependent services.

Platform response

- ✓ **Read** workload specification
- ✓ **Match** resource definitions
- ✓ **Create** app configs, configure resources
- ✓ **Deploy**

Off the golden path: add ArangoDB

Dev
Request

I need ArangoDB for my workload but there is no default.

I add a resource definition



Platform
response

- ✓ **Read** workload specification
- ✓ **Match** resource definitions
- ✓ **Create** app configs, configure resources
- ✓ **Deploy**

ArangoDB is available for reuse by the next user. Standardization by design!

Platforming is about structuring repos (more than anything). If your setup is well platformed (following this ref architecture), this is how your repo structure looks:

Developer owned

Workload

- Workload source code
- Workload spec (Score)
- Docker file
- Pipeline YAML

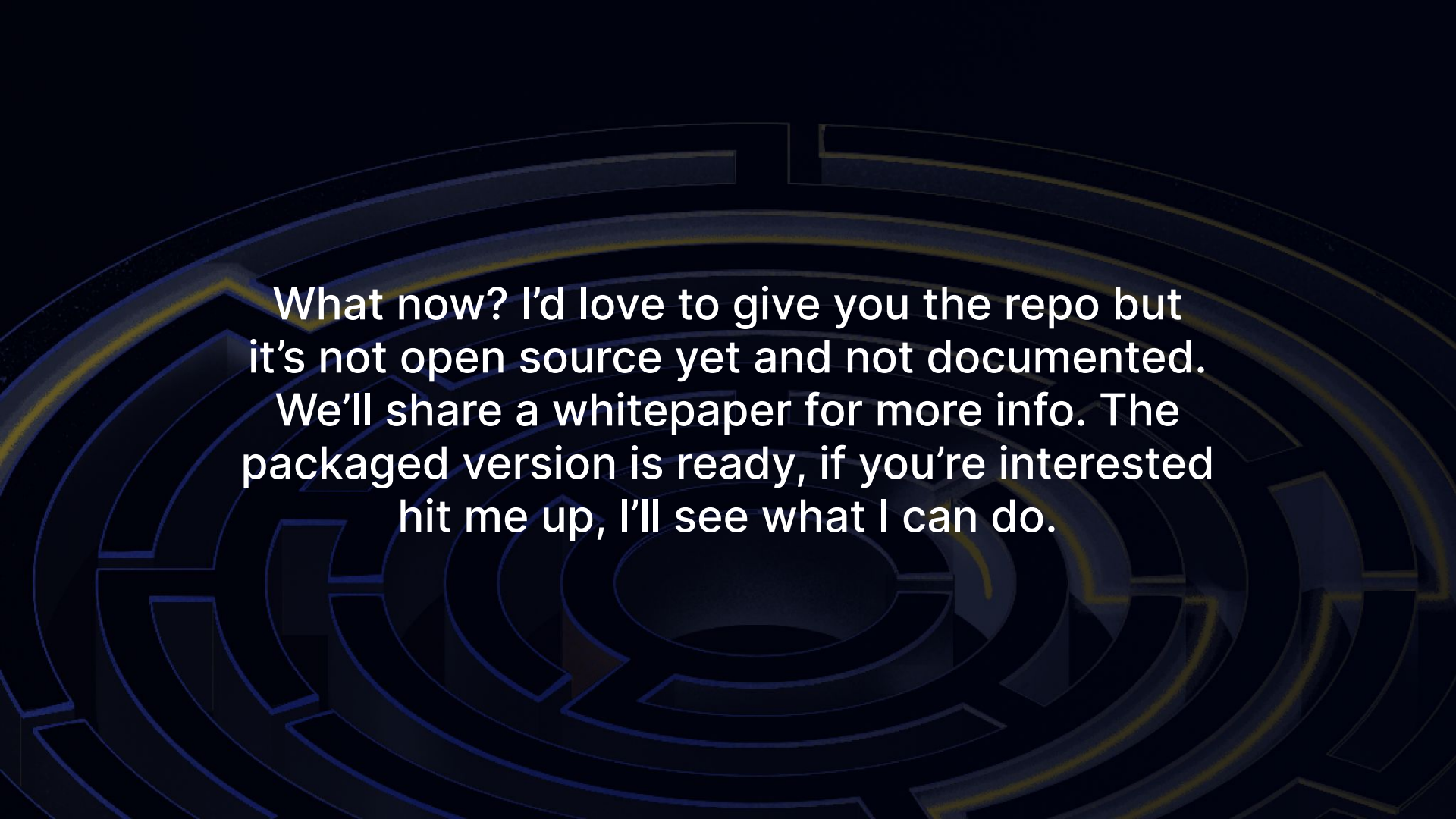
Platform Engineering owned

Resource
Definitions

Resource
Drivers/IaC
(static and
dynamic)

Workload
Profiles

Automations/
compliance

The background features a complex, glowing yellow maze pattern on a dark blue gradient. The maze consists of multiple concentric and interlocking paths that create a sense of depth and complexity. The text is centered within the maze.

What now? I'd love to give you the repo but it's not open source yet and not documented. We'll share a whitepaper for more info. The packaged version is ready, if you're interested hit me up, I'll see what I can do.