KubeCon CloudNativeCon Europe 2025

Platform Abstractions **Asset** or **Liability**?

Let’s Understand The Abstraction Debt Trap

Atulpriya Sharma

Hyderabad, India

Sr. Developer Advocate at InfraCloud Technologies

CNCF Ambassador, Co-Chair Platforms Working Group

Cloud Native, Kubernetes, Platform Engineering

@TheTechMaharaj

https://www.linkedin.com/in/atulpriyasharma

When I’m OOO ;)

atulmaharaj **Follow** **Message**

1,021 posts **4,020** followers **2,299** following

Atulmaharaj
Blogger
🌐 Hyderabad Food, Travel & Lifestyle Blogger
📝 10+ Years Blogging
🎯 6M+ Blog Views
🏆Best Blogger Award
⚠️ Insta Is Not... more
socialmaharaj.com


FRA 🇩🇪'24

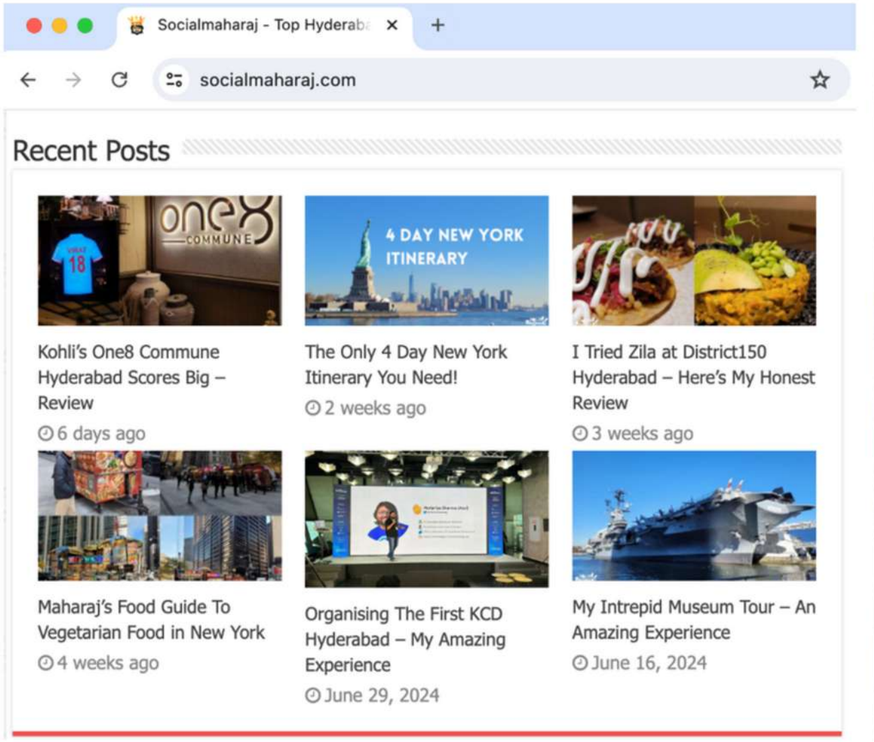

LA 🇺🇸'24

LAS 🇺🇸'24


SLC 🇺🇸'24


QTR 🇶🇦'24


ITA 🇮🇹'24

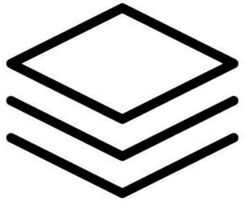


@Atulmaharaj

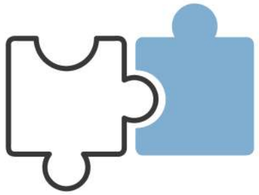
https://socialmaharaj.com


Ingredients:
3/4 cup canola oil
1/4 cup lemon juice
2 garlic cloves, minced
1/2 teaspoon salt
1/2 teaspoon pepper
2 bunches (1 pound each) romaine, torn
2 cups chopped tomatoes
1 cup shredded Swiss cheese
2/3 cup slivered almonds, toasted, optional
1/2 cup grated Parmesan cheese
8 bacon strips, cooked and crumbled
1 cup Caesar salad croutons
1) In a jar with tight-fitting lid, combine the oil, lemon juice, garlic, salt and pepper; cover and shake well. Chill.
2) In a large serving bowl, toss the romaine, tomatoes, Swiss cheese, almonds if desired, Parmesan cheese and bacon.
3)Shake dressing; pour over salad and tos
routons and serve immediately.

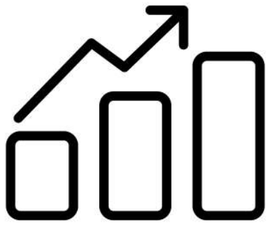
Platforms & **Abstractions**



Structure Through Simplification



Complexity Made Simple



Hidden Details, Visible Value

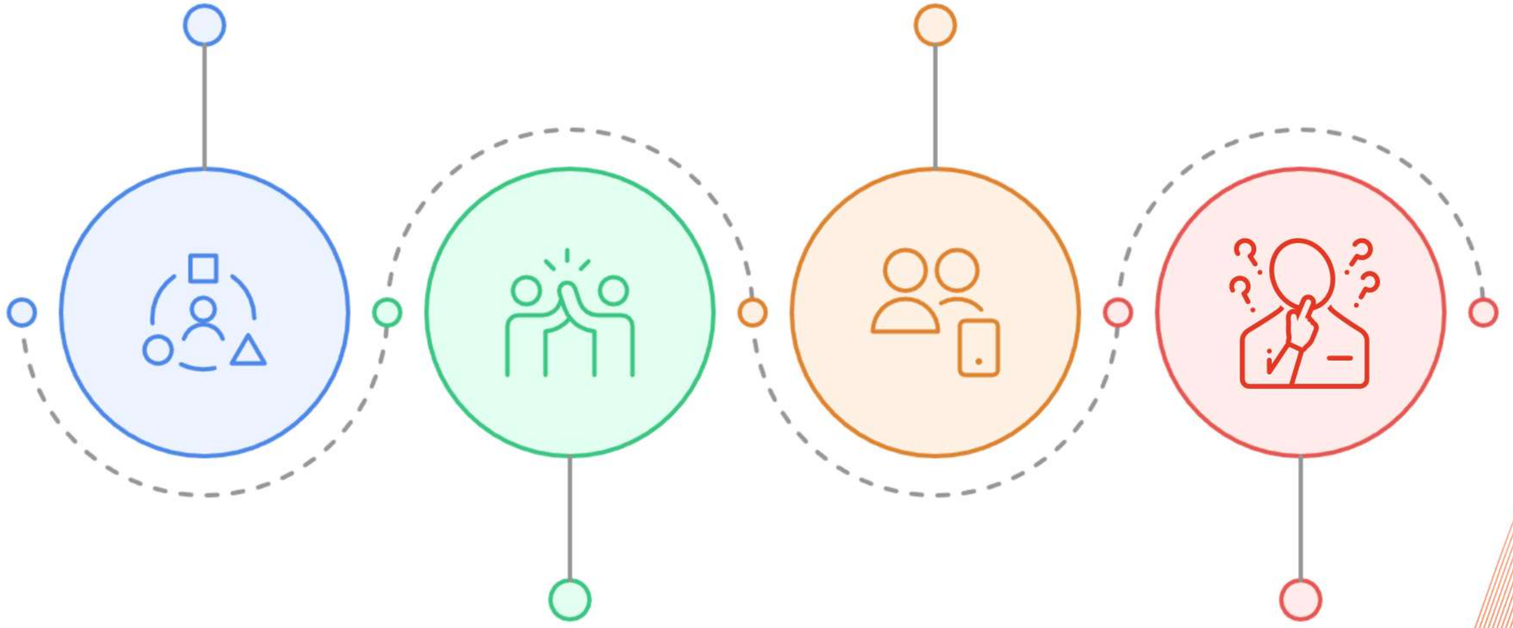
The **Abstraction Dilemma**

Platform Team Creates Abstraction

The platform team develops a database provisioning abstraction for standard use.

Mobile Payments Team Requests Extension

The mobile payments team requests specific PostgreSQL extensions.



Abstraction Works for 10 Teams

The abstraction is successfully implemented by ten application teams.

How do we deal with this?

The platform team must decide how to handle the extension request.

How to handle the mobile payments team's request for database extensions?

Add Configuration Options

Increases complexity of the abstraction, leading to confusion and difficulty for new team members.

Make One-off Exception

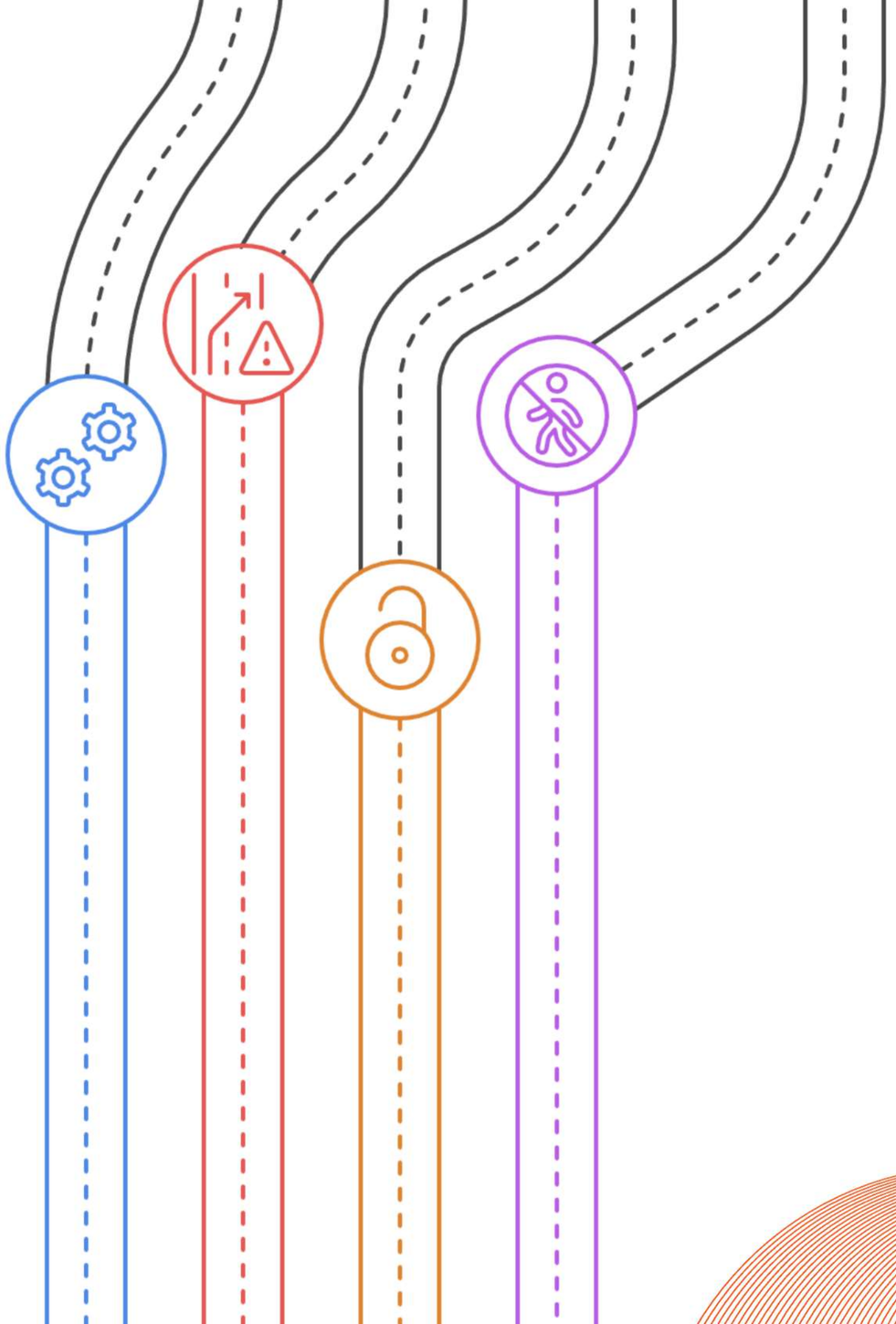
Breaks standardization, potentially leading to inconsistencies across teams.

Require Conformity

Hinders innovation by restricting the mobile team's capabilities.

Allow Bypassing

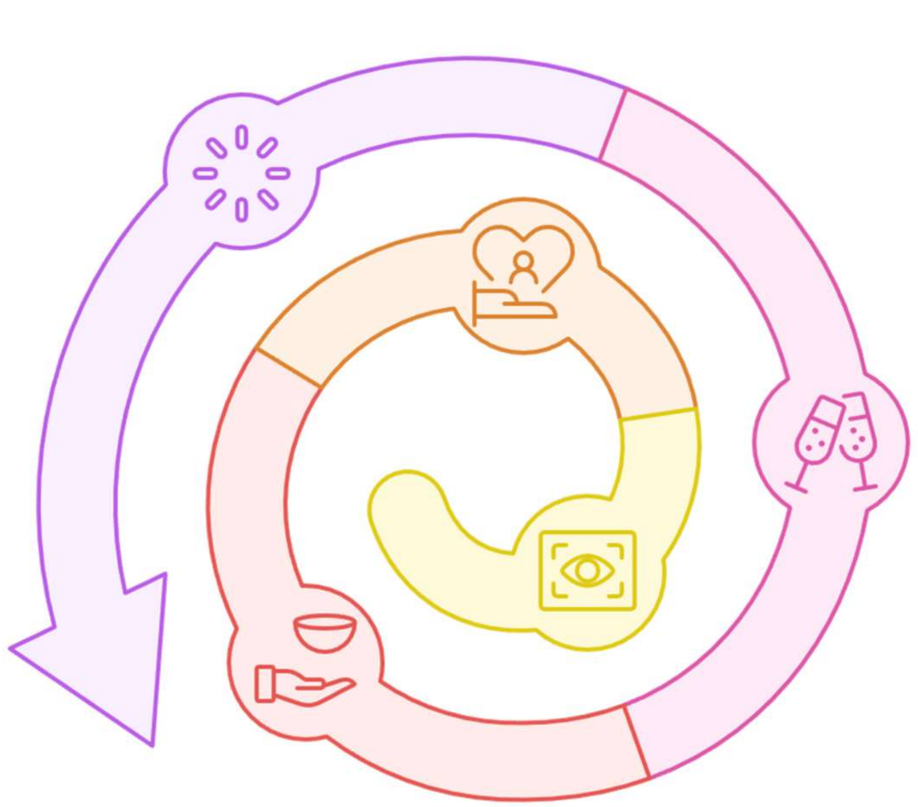
Creates shadow IT, risking security and control over IT resources.



The **Abstraction Debt Trap**

The accumulating cost of maintaining increasingly rigid platform abstractions.

The **Abstraction Debt Cycle**



01 Simplify

02 Adopt

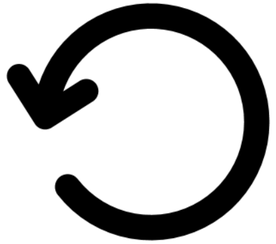
03 New Requirements

04 Add Complexity

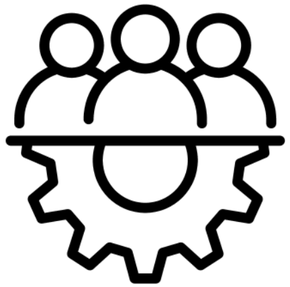
05 Repeat

Warning **Signs**

Rising customization request



Workarounds



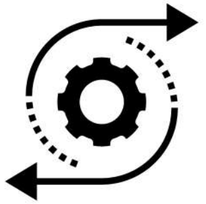
Shadow IT

Impact

Each new feature becomes harder to implement

Teams lose trust in platform capabilities

Platform evolution slows significantly

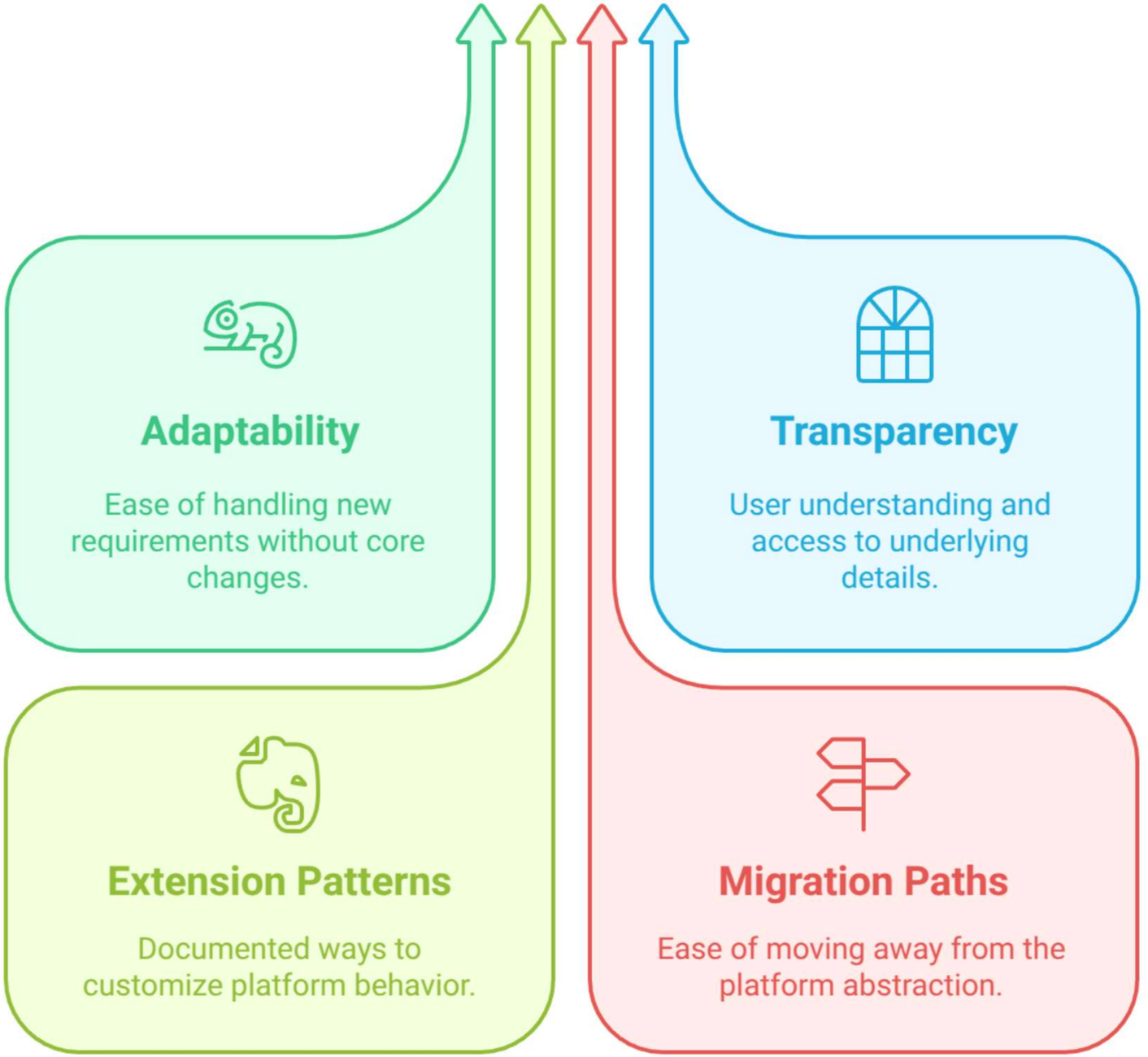


New technologies become difficult to incorporate

Abstraction Elasticity

The ability of abstractions to accommodate change without breaking.

Key **Indicators**



Understanding **Abstraction Elasticity**

- Can teams access lower-level controls when necessary without abandoning the abstraction entirely?
- Do users understand what's happening underneath the abstraction when they need to?
- Are there documented extension points where teams can customize behavior?

A team needs specialized PostgreSQL extensions for geospatial queries that aren't in your standard database abstraction

Add more configuration parameters to the core abstraction for each new extension request, creating an increasingly complex interface that everyone must understand

Create multiple interface options - a simple one for basic needs and more advanced interfaces for specialized requirements.

An ML workload requires specific memory/CPU settings that don't match your platform's predefined resource profiles.

Create predefined "t-shirt size" resource profiles with rigid memory/CPU combinations that teams must use, forcing them to either overprovision resources or work around the platform

Implement flexible resource boundaries with policy based guardrails that allow customization within defined limits based on workload type.

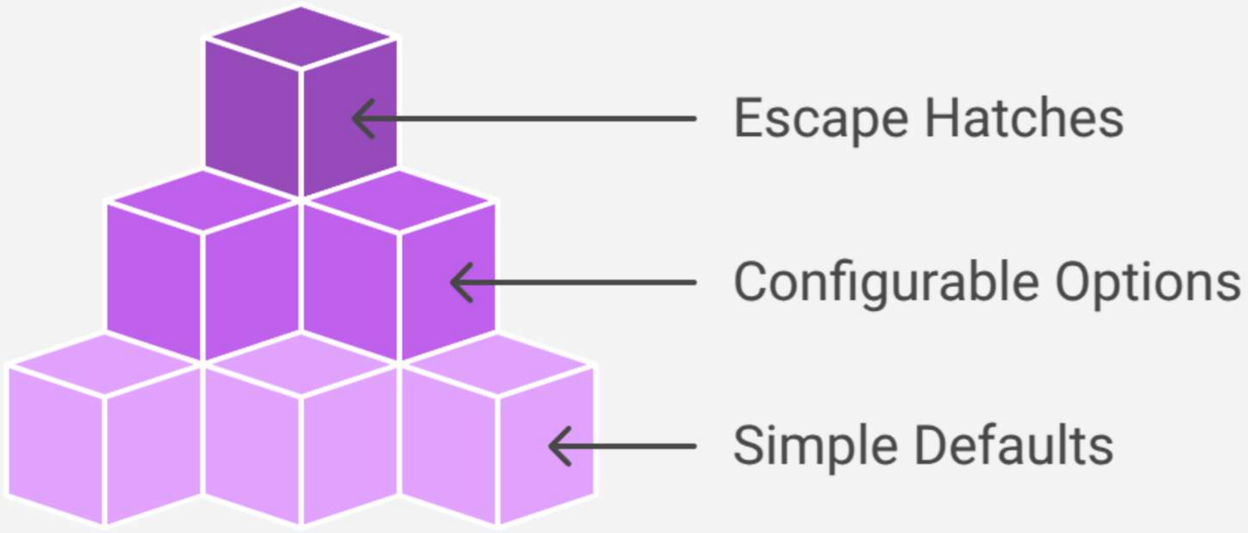
A team needs cloud infrastructure configured in a way that doesn't match your standard templates.

Create infrastructure templates with dozens of parameters trying to account for every possible variation, resulting in overwhelming complexity for users.

Create infrastructure building blocks that can be combined and customized without disrupting standard configurations.

Patterns to **Break** the **Abstraction Debt Trap**

Layered Abstraction Pattern



 # Level 1: Simple Database Abstraction

```yaml
apiVersion: database.example.org/v1alpha1
kind: SimpleDatabase
metadata:
  name: payments-db
  namespace: mobile-payments
spec:
  # Minimal configuration with sensible
defaults
  size: medium
  highAvailability: true
```

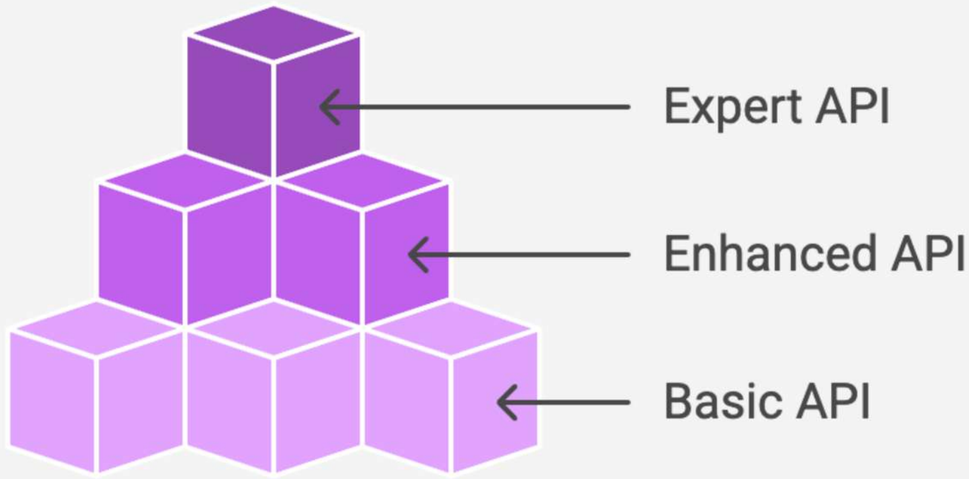
 # Level 2: Extended Database Abstraction

```yaml
apiVersion: database.example.org/v1alpha1
kind: ExtendedDatabase
...
spec:
  # All Level 1 options plus additional
configuration
  size: medium
  highAvailability: true
  version: "13.4"
  # Specific PostgreSQL extensions needed
  extensions:
    - pg_stat_statements
    - pg_repack
    - pgcrypto
  backup:
    schedule: "0 2 * * *"
    retentionPeriod: 14d
```

Level 3: Advanced Database Abstraction

```yaml
apiVersion: database.example.org/v1alpha1
kind: AdvancedDatabase
...
spec:
  # All Level 1 & 2 options plus escape hatches
  size: medium
  highAvailability: true
  version: "13.4"
  extensions:
    - pg_stat_statements
    - pg_repack
    - pgcrypto
  backup:
    schedule: "0 2 * * *"
    retentionPeriod: 14d
  # Direct PostgreSQL parameter configurations
  parameters:
    shared_buffers: "8GB"
    max_connections: 200
    work_mem: "64MB"
    maintenance_work_mem: "512MB"
    effective_cache_size: "24GB"
    max_parallel_workers: 8
```

Progressive Enhancement Pattern



Basic API (For Developers)

```yaml

# Deploy a web application
POST /api/v1/apps
{
  "name": "my-app",
  "image": "my-repo/app:v1",
  "replicas": 2
}
```

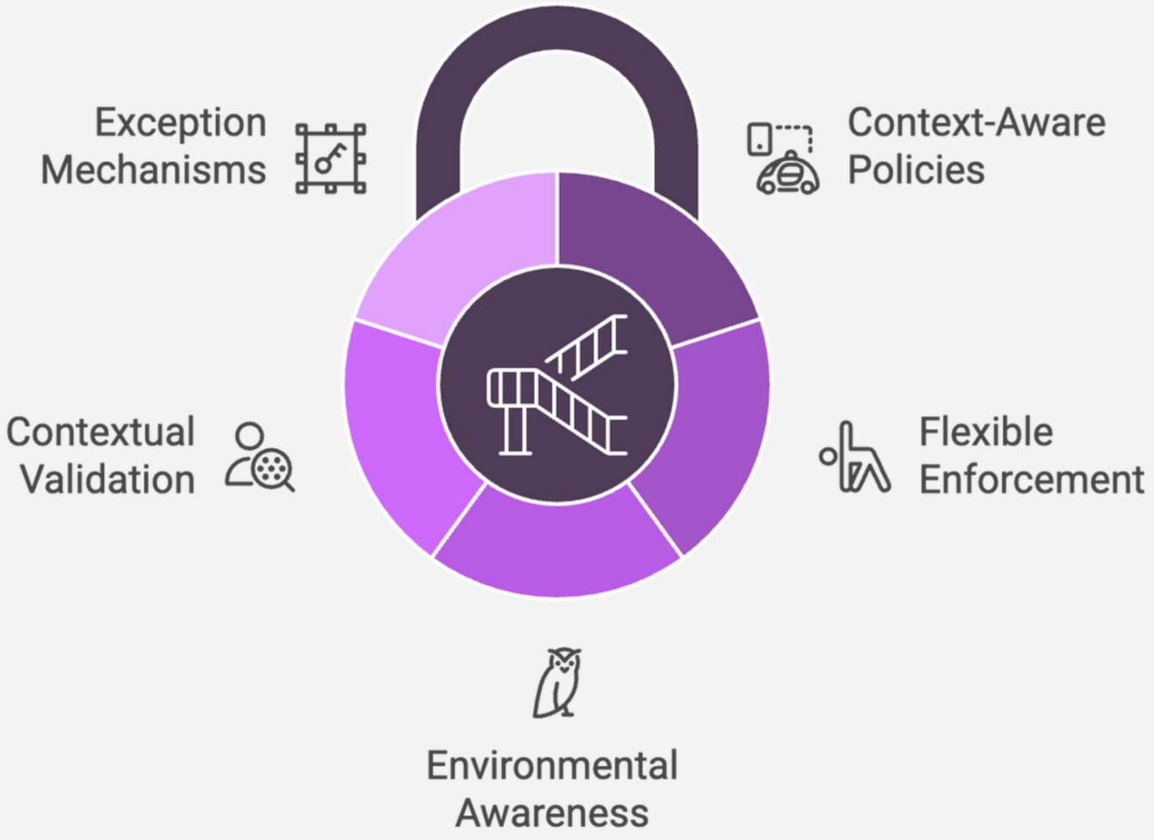
Enhanced API (For DevOps)

```yaml

# Deploy with additional capabilities
POST /api/v1/apps
{
  "name": "my-app",
  "image": "my-repo/app:v1",
  "replicas": 2,
  # Enhanced capabilities
  "resources": {
    "cpu": "1",
    "memory": "2Gi"
  },
  "autoscaling": {
    "min": 2,
    "max": 10
  }
}
```

Expert API (For Platform Engineers)

```yaml
# Deploy with full control
POST /api/v1/apps
{
  "name": "my-app",
  "image": "my-repo/app:v1",
  "replicas": 2,

  # Enhanced capabilities
  "resources": {...},
  "autoscaling": {...},

  # Expert capabilities
  "security": {
    "nonRoot": true,
    "readOnlyFs": true
  },
  "networking": {
    "policy": "restricted",
    "serviceMesh": true
  }
}
```

Policy-based Guardrails



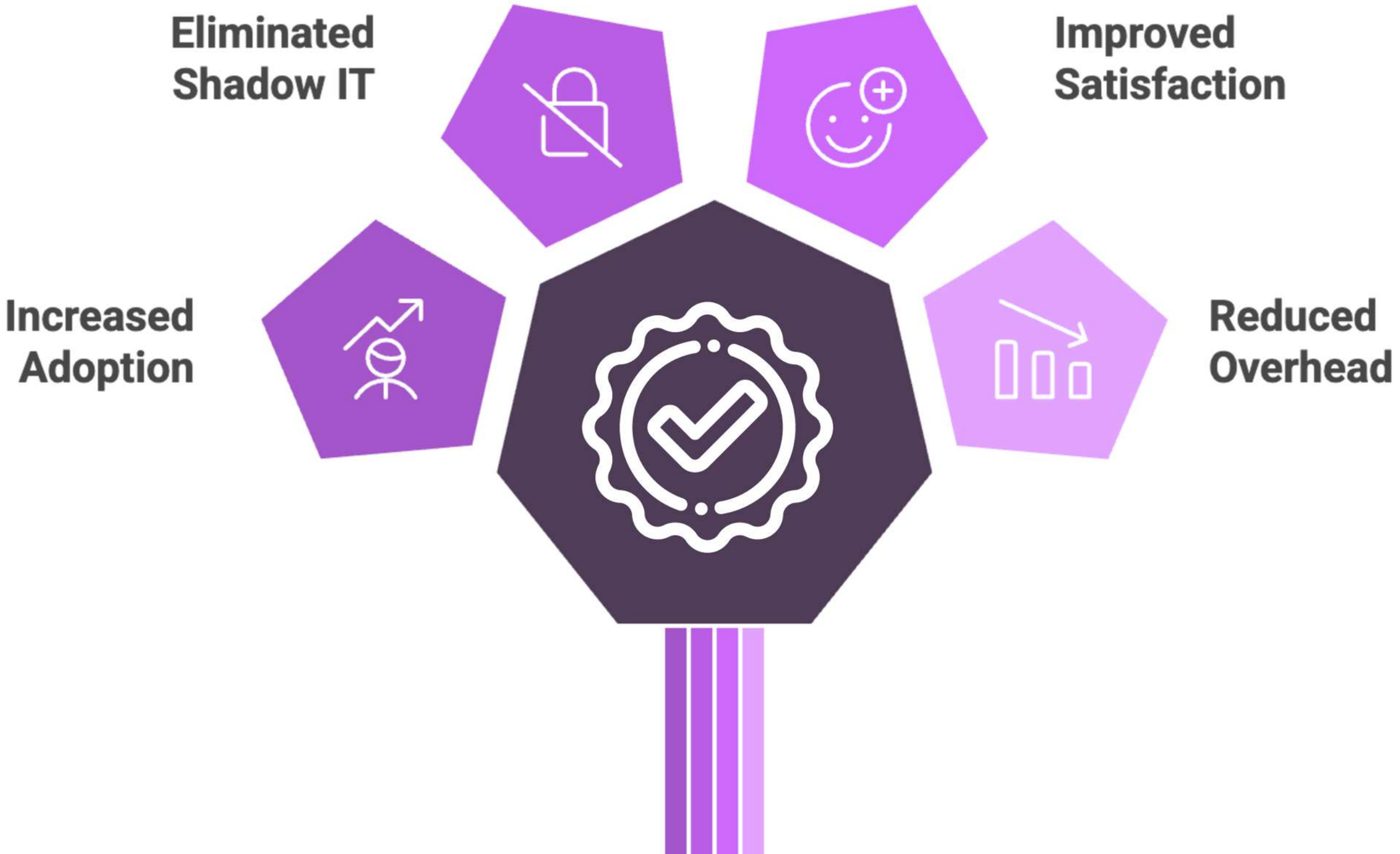
Environment-Aware Database Policies

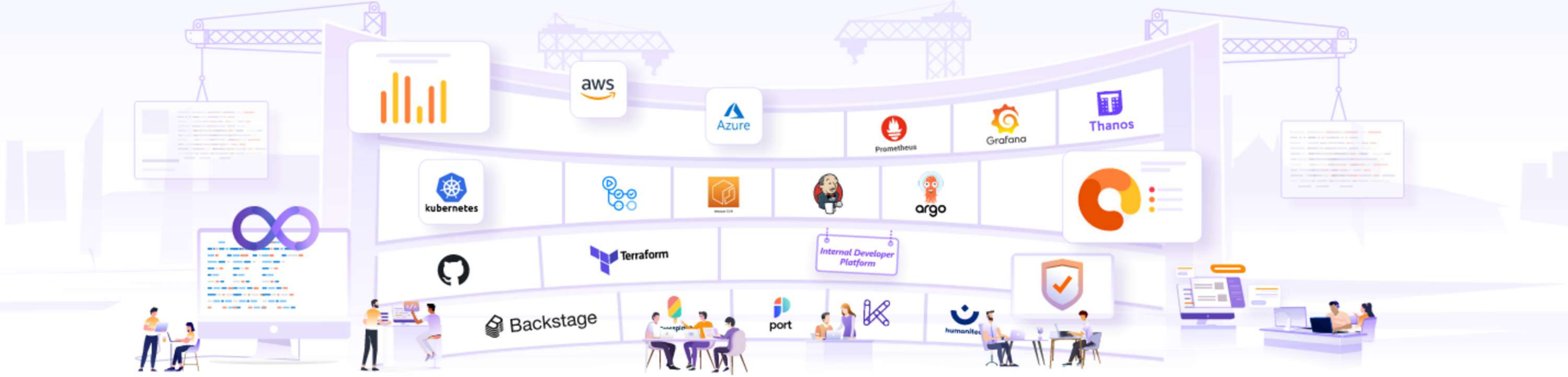
```
# OPA policy that adapts to environment
context
allow {
  # Encryption required in ALL environments
  input.database.encryption == true

  # Environment-specific requirements
  environment_requirements[input.environment]
}

# Different requirements by environment
environment_requirements = {
  "production": {
    "backup": {"frequency": "daily",
"retention": "30days"},
    "highAvailability": true
  },

  "development": {
    "backup": {"frequency": "weekly"},
    "highAvailability": false
  }
}
```

PostgreSQL Extension Policy

```
# Approved PostgreSQL extensions policy
approved_extensions = {
  # Extensions allowed in all environments
  "core": ["pgcrypto", "pg_stat_statements"],

  # Extensions allowed only in development
  "development": ["postgis", "fuzzystrmatch"],

  # Extensions requiring approval
  "restricted": ["pg_partman", "pg_repack"]
}

# Extension request process
kind: ExtensionRequest
metadata:
  name: "request-pg-partman"
  team: "mobile-payments"
spec:
  extension: "pg_partman"
  reason: "Time-based partitioning for
transactions"
  security_review: "Approved"
```

Results

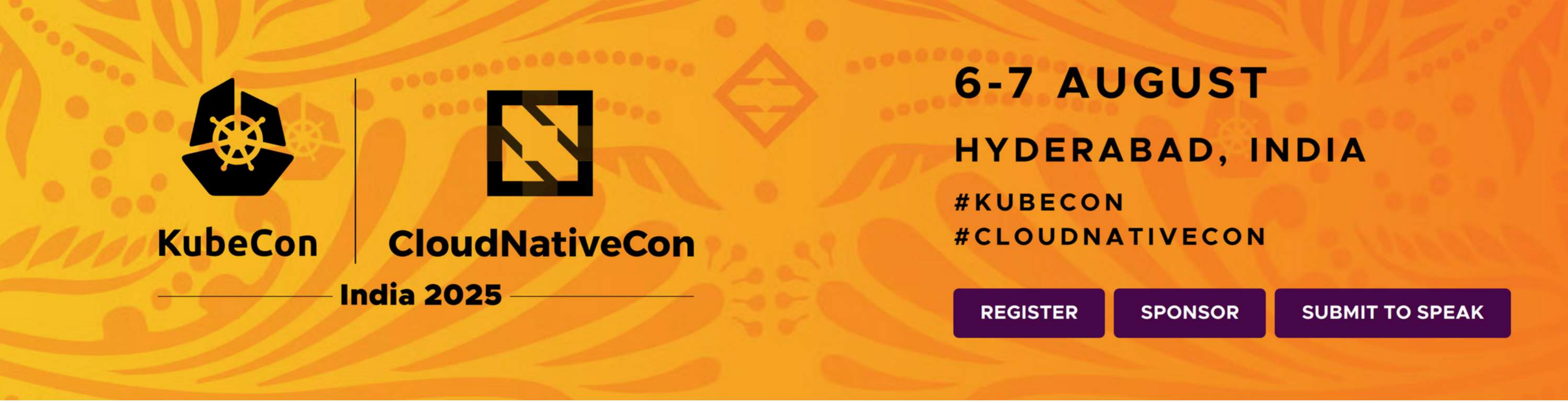

Eliminated
Shadow IT
Improved
Satisfaction
Increased
Adoption
Reduced
Overhead



Platform Engineering eBook

- Reference Architectures
- Blueprints
- Capability Maps
- Self-check Questionnaire

Get The eBook



Event Date: August 6 - 7 2025
Location: Novotel, HICC
Registration:

- Group discounts 10% if more than 5 people
- Scholarship available for tickets and travel

Thank **You**

Leave
Session
Feedback

Let’s Get In **Touch**

Email

atulpriya.sharma@infracloud.io

Social Media

@TheTechMaharaj | Atulpriya Sharma

Website

https://infracloud.io