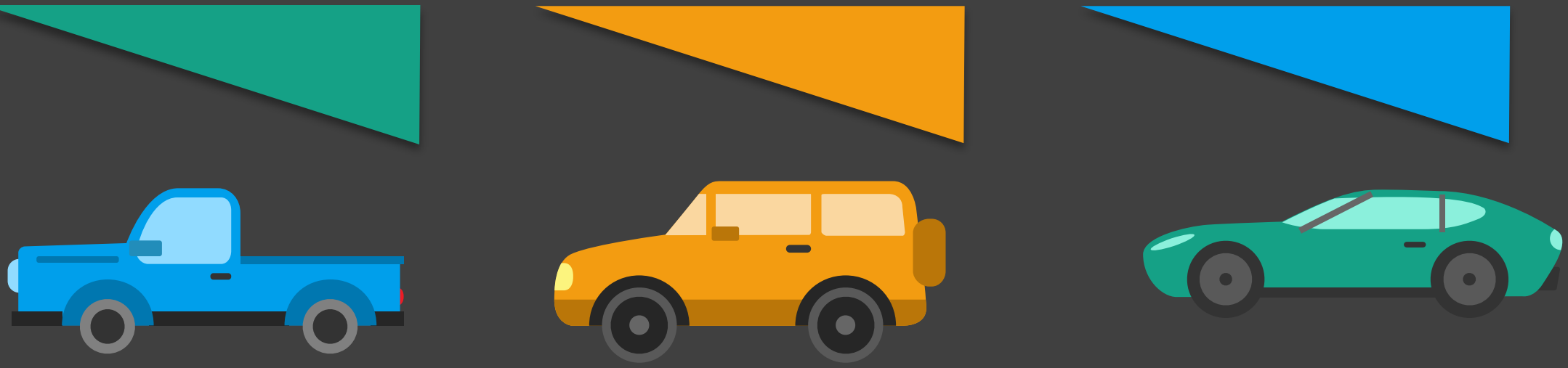
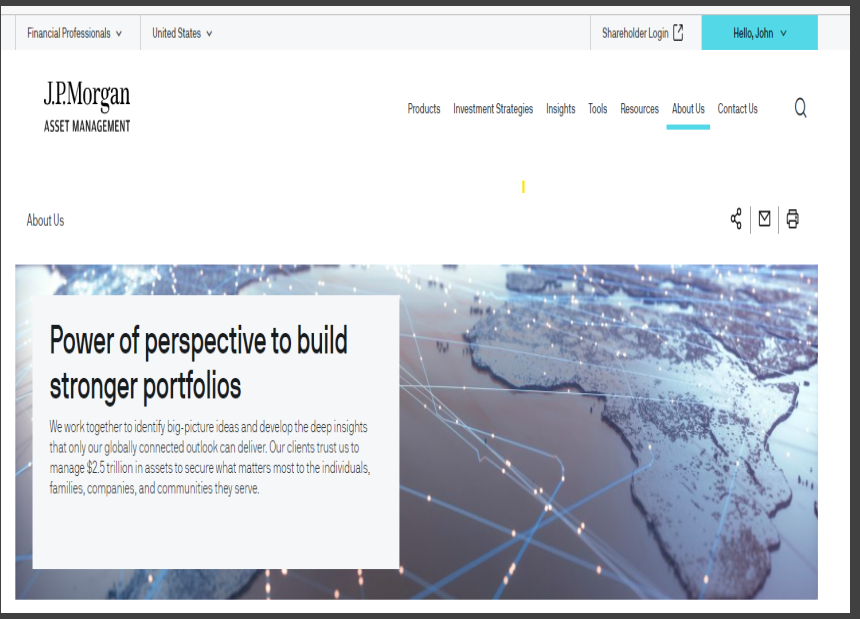
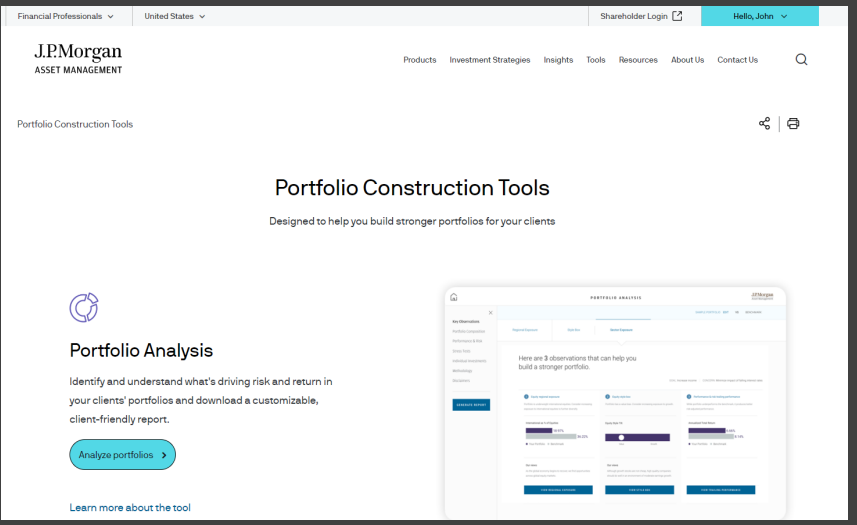


# Journey from PCF to AWS & EKS

# About JP Morgan and our team




JPMorgan Asset Management is a global leader in Investment Management. Our clients trust us to manage $2.5 trillion to secure, protect and strengthen and enjoy hard earned returns.

Morgan Advisor is a suite of financial web applications used by external financial advisors and JPMC client advisors to advise clients on their portfolios and give them insightful decisions to balance risk and reward and select the right retirement funds.

We have data for over 300,000 instruments and across 3 regions, and use quantitative analytics to analyze portfolios and give the right recommendations.

Our team consists of a mix of full stack, dedicated backend and cloud specialists.  Our team follows Agile practices, and is responsible for the cloud infrastructure, architecture, monitoring, and production support of the systems.

# About me – Sheela Shankar

Executive Director - Client Technology, JPMorgan Asset Management

Joined JPMC 7 years ago

Passionate about all things Cloud, SRE, and DevOps

Certified AWS Solutions Architect

Full Stack Developer

# Why did we start Cloud Migration with PCF back in 2015?

Easy one command deploys

PCF had a Low entry barrier and easy to get started

UI console made it easy to start, restart, change configuration variables

Friendly GUI console

No infrastructure to provision, no infra upgrades.

Focus on application logic only

Support for Java, Python, NodeJS, and static websites

Build packs made it easy to support different runtimes

# Areas where we struggled with JPMC Implementation of PCF

01 Running out of hardware and memory space as number of applications increased

02 Limited choice of technology available on JPMC PCF marketplace

03 Pool instability issues needed manual steps to bypass bad pools and to enable pools back

04 No support for serverless applications or applications that need access to file system

05 Application size/Memory could not exceed configured size

06 Need for IP Whitelisting for services and problems caused by IP  Caching of an expired DNS

# Some numbers

| | |
|---|---|
| >14 BN | JPMC Annual Technology Spend |
| >50K | Technologists, 21 Global Tech Centers |

JPMC has the largest PCF deployment in the world !!

| | |
|---|---|
| >1500 | PRODUCTION APPLICATIONS |
| >200K | CONTAINER WORKLOADS |
| >330K | RUNNING MEMORY |
| >37 | POOLS |
| >15K | DAILY DEPLOYMENTS |
| 16 | DATACENTERS across 3 regions |

# Reasons for moving to Public cloud in 2021

01

## Truly elastic

Scalable and reliable infrastructure

02

## Developer Productivity

Freedom to adopt different technology based on the requirement

03

## Lower TCO

Lower operational cost – no need to purchase Infrastructure.

Serverless lets you run Jobs only when needed

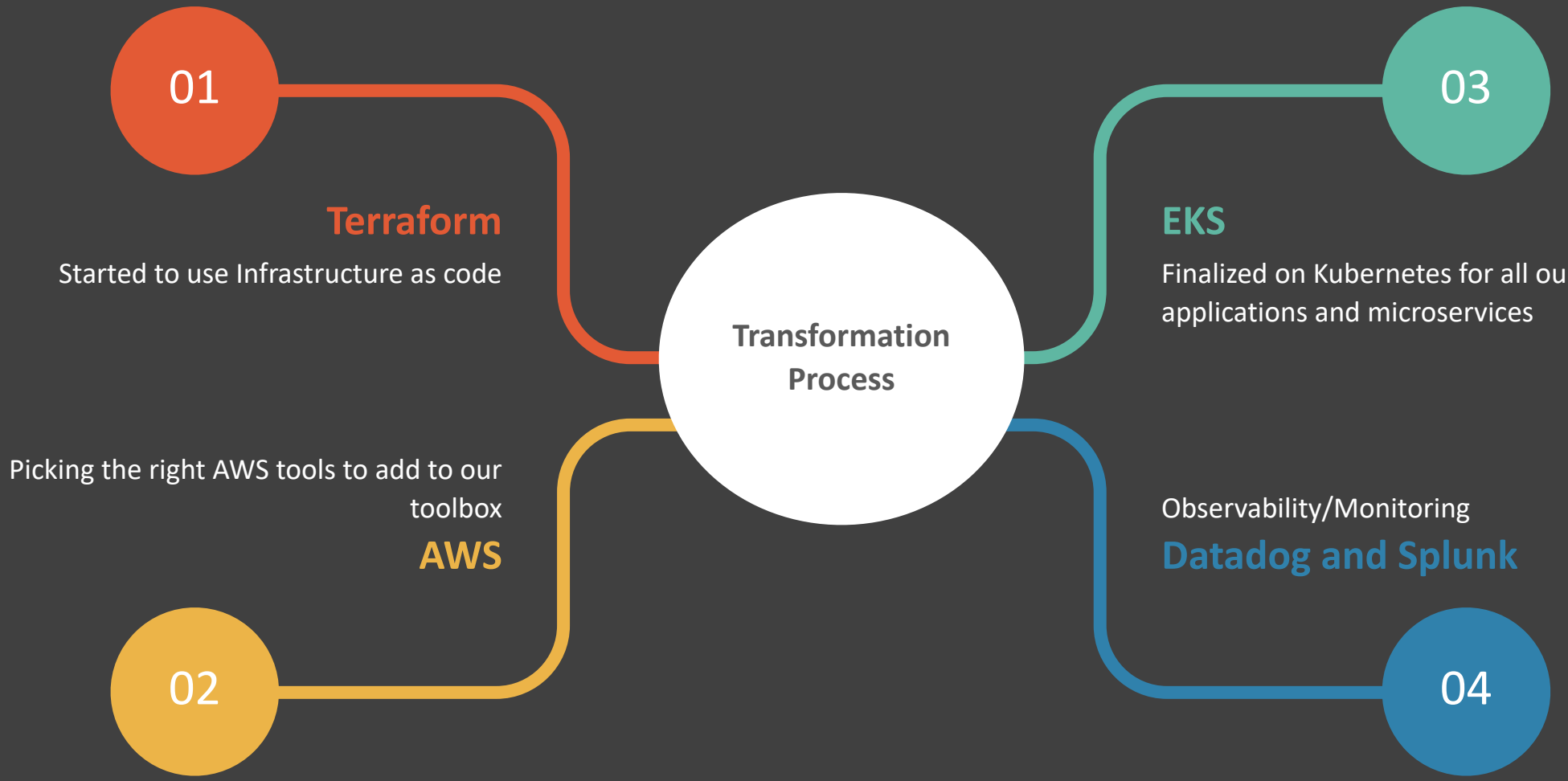
04

## Better Data Solutions

Encryption of data at rest, Ease of data replication, storage, backups.

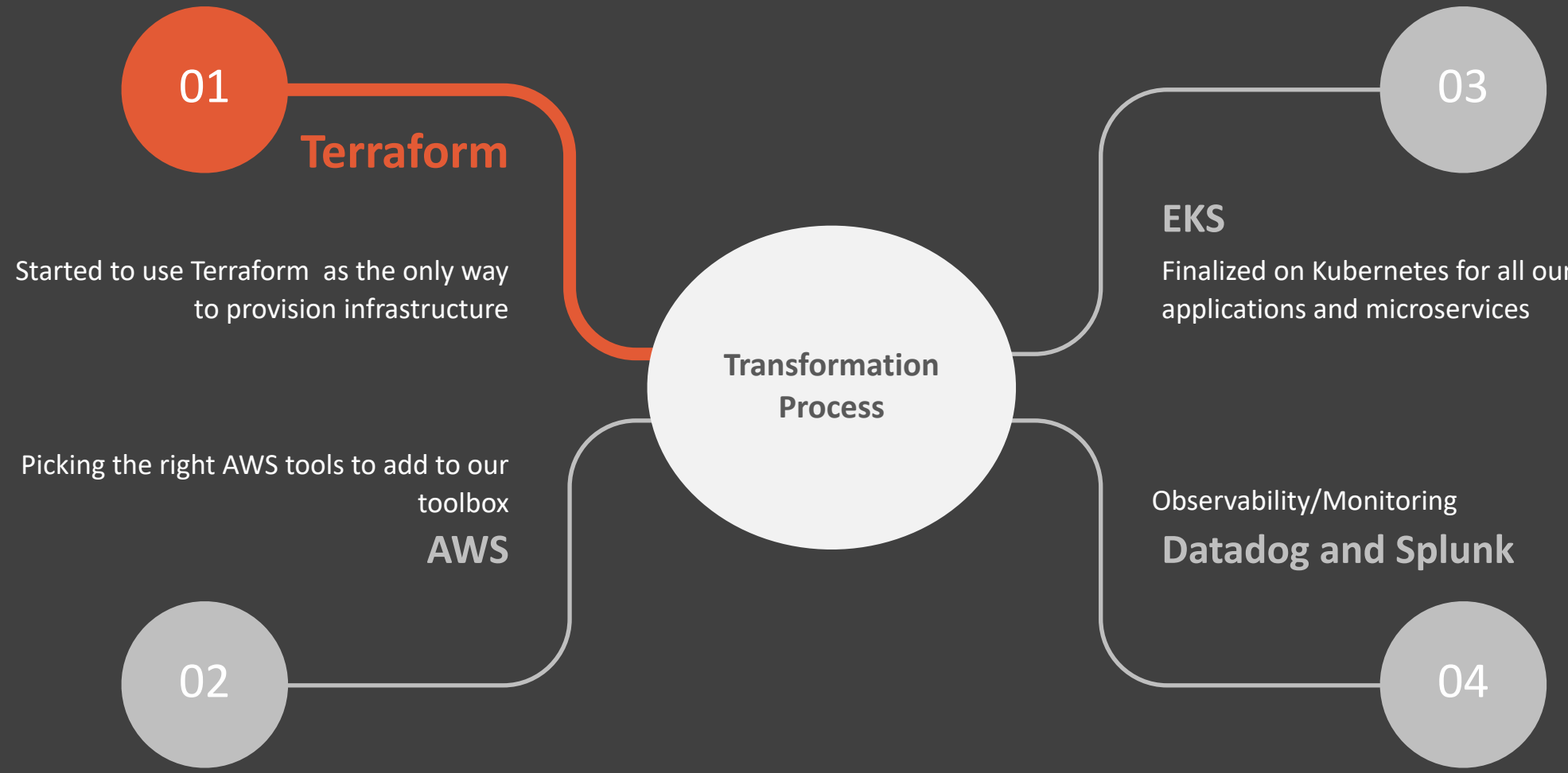
# 4 step process to AWS/EKS

Cloud modernization journey



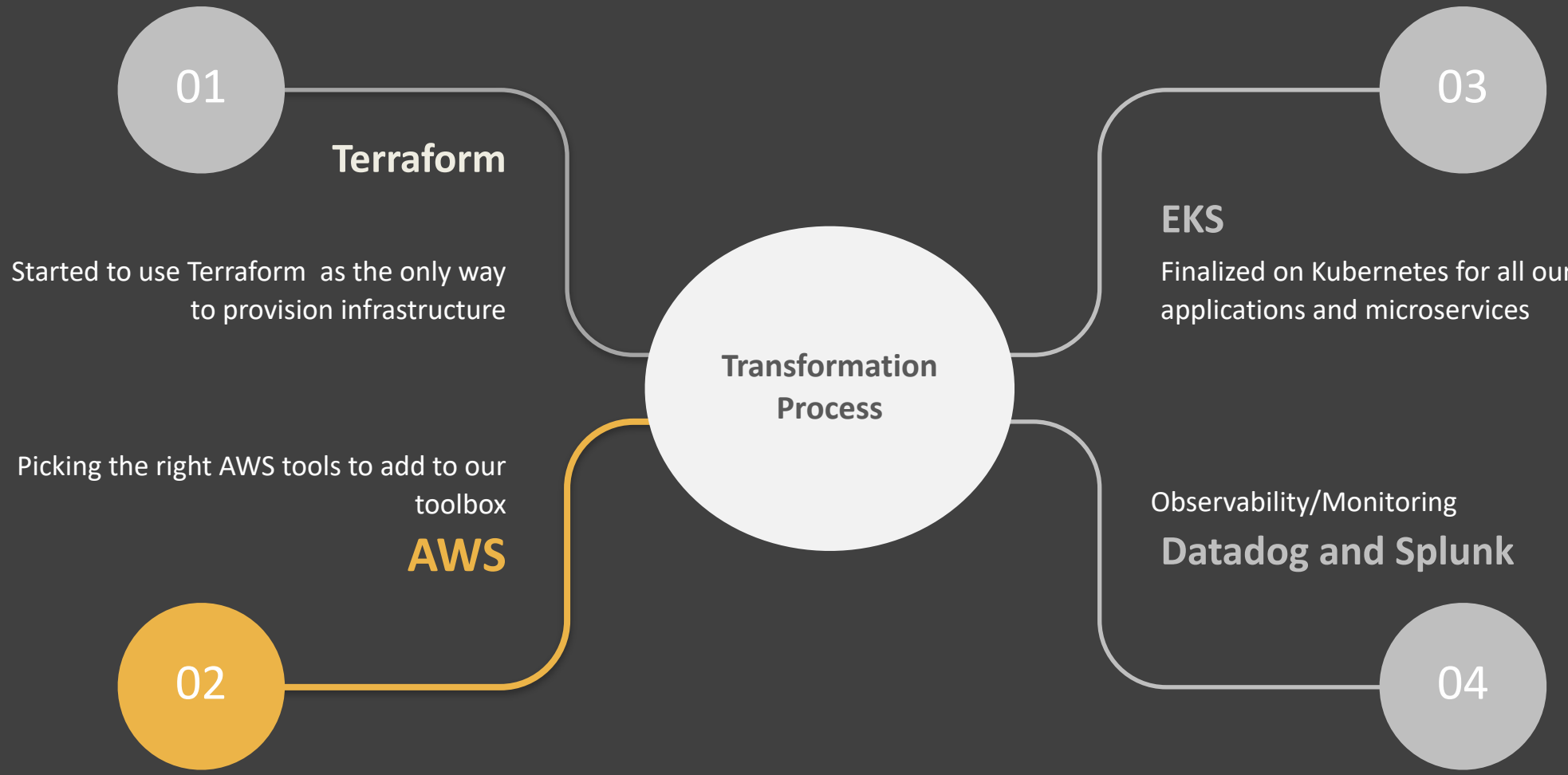
# 4 step process to AWS/EKS

Cloud modernization journey



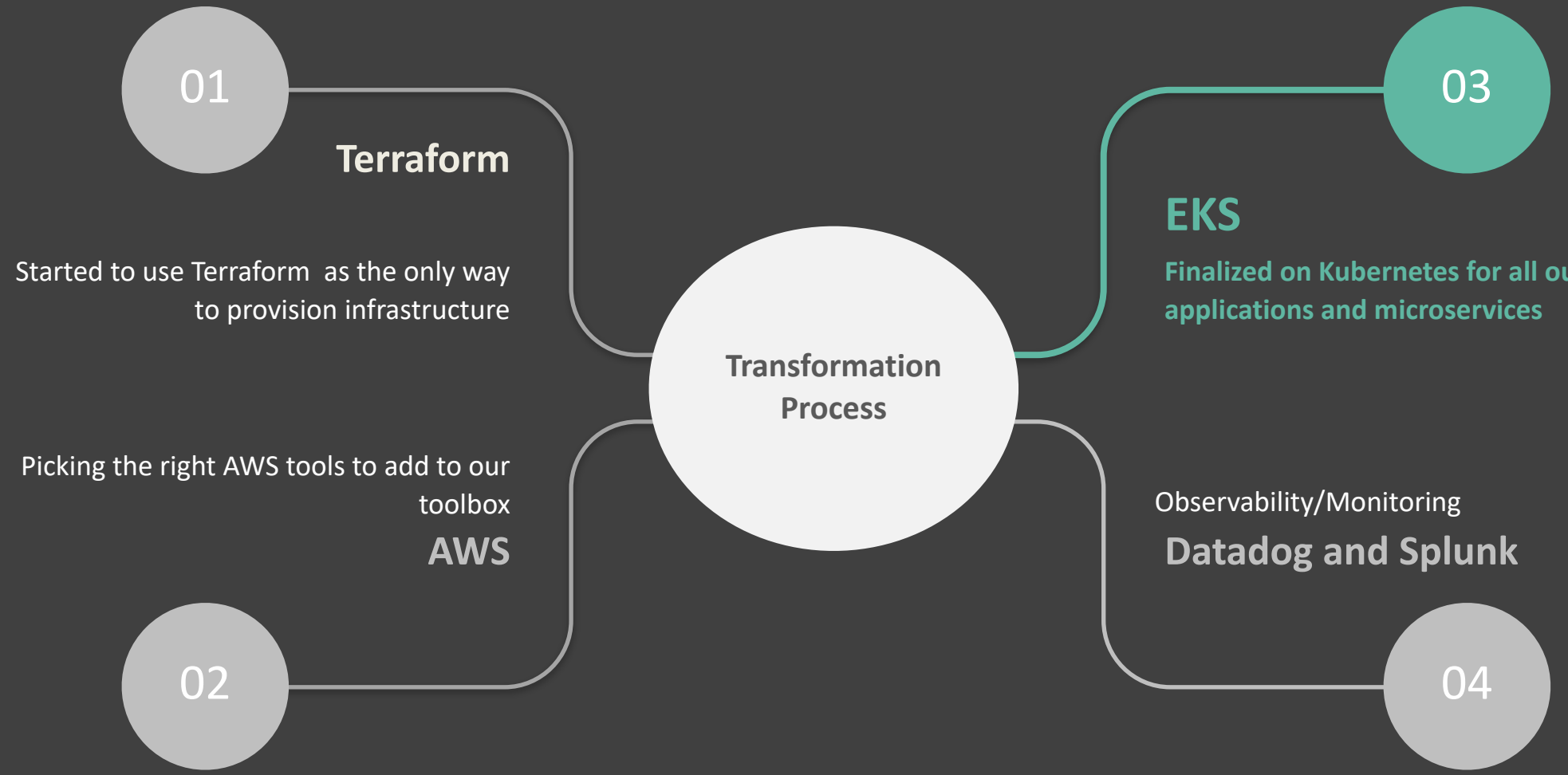
# 4 step process to AWS/EKS

Cloud modernization journey



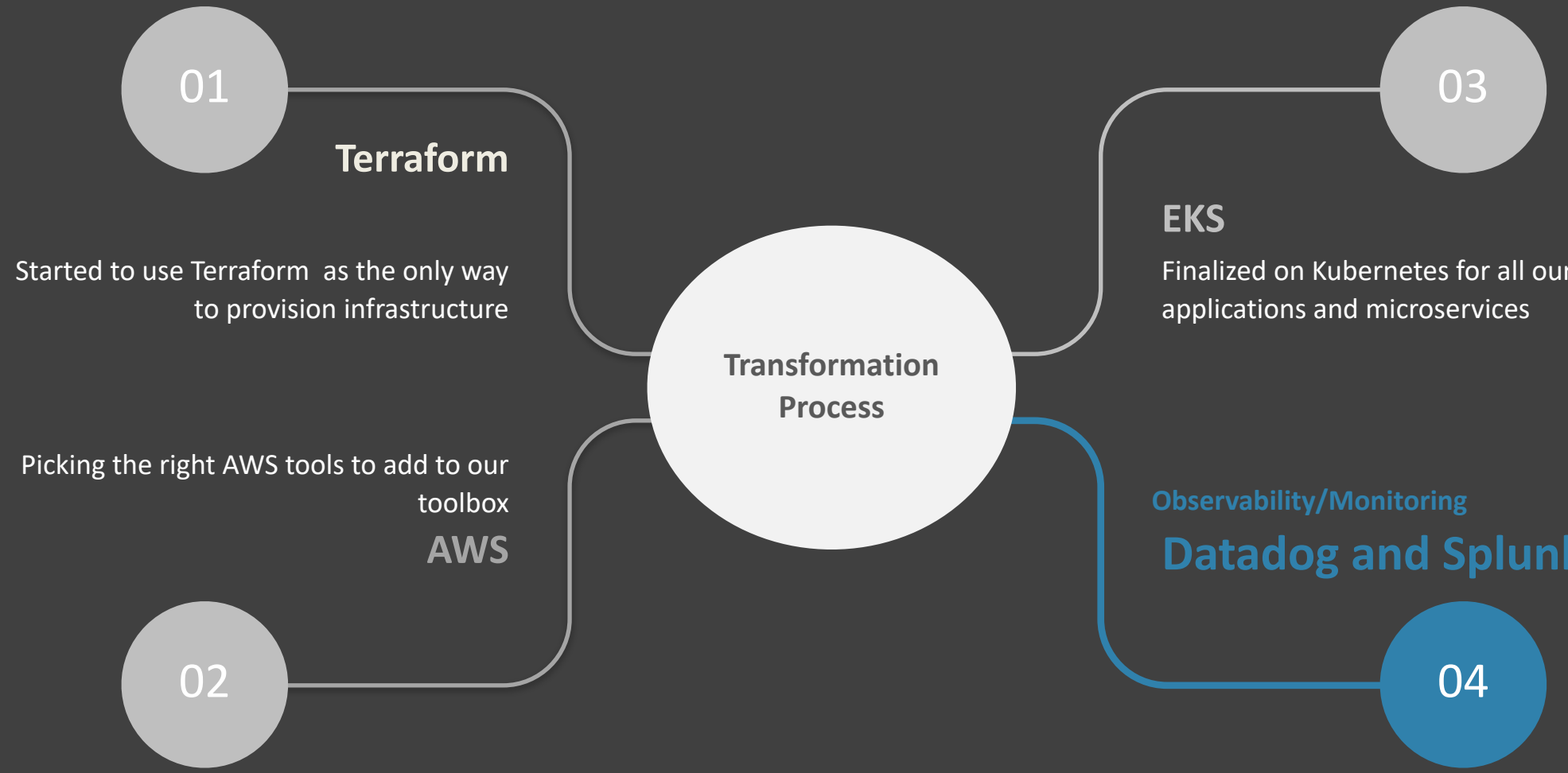
# 4 step process to AWS/EKS

Cloud modernization journey



# 4 step process to AWS/EKS

Cloud modernization journey

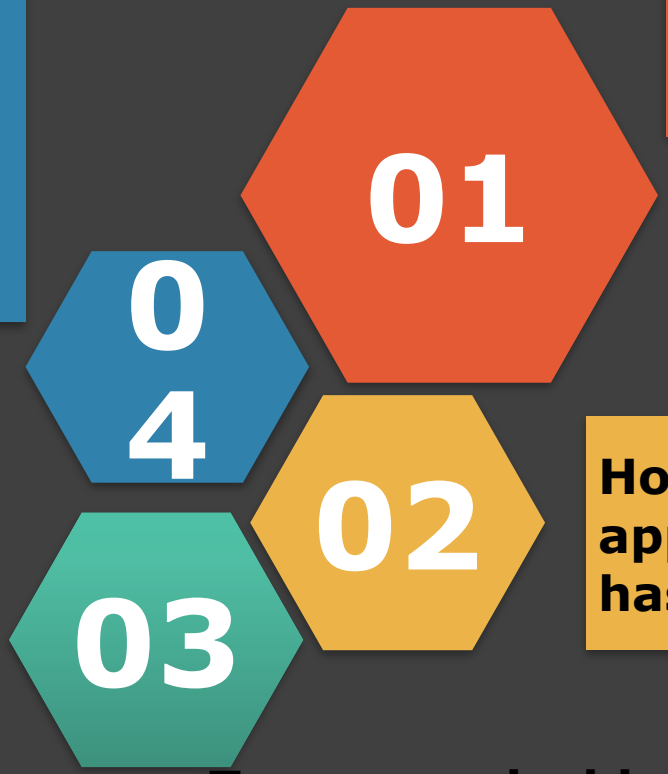


# Team frustrations with AWS/EKS

**Why do I need to care about where my code runs?**

**I just want to write UI code.**

**Where is my UI console to manage my app? You mean I have to use kubectl?**




**How do I know which applications are in PCF and what has been migrated to AWS?**

**Too many dashboards…. Why do I need to look at Datadog, Splunk, Grafana and AWS Console?**

# What helped in driving AWS adoption in the team

**Blue prints & Tech Primers** – for established patterns so that teams can adopt those patterns – these patterns should illustrate good practices.

**Cloud Parties** – 2-3 day events to deploy apps to Production and which bring developers, specialists, cyber, network engineers in 1 room

**Self Service Everything** – make it easy for developers to provision software

**Build Culture/Community** – Discuss and share issues, resolutions on Symphony chats, Email forums

**Treat Developers as your biggest clients** – listen to their pain points

# Devops Learnings after moving to AWS

## REFACTOR

Take the opportunity to re-architect instead of lift and shift. There is never time to refactor later, so do it right the first time.

## WATCH FOR HIGH COSTS

Monitor cost constantly – especially in DEV due to multiple POCs and test massive data migration dress rehearsals

## DATA MIGRATION

Plan for Data migration and schema migration. Get business users to test the application after data migration

## INFRASTRUCTURE

Unlike PCF, in AWS/EKS world, developers are responsible for managing the infrastructure and to apply needed patches e.g. log4j remediation

## KNOW YOUR DEPENDENCIES

Check for dependencies in On-Prem. We faced complexities when calling services in PCF from AWS. Required firewalls to be opened and causes increased latency

## PERFORMANCE TESTING

Test performance early – to avoid nasty surprises later

# Production Issues to watch out for

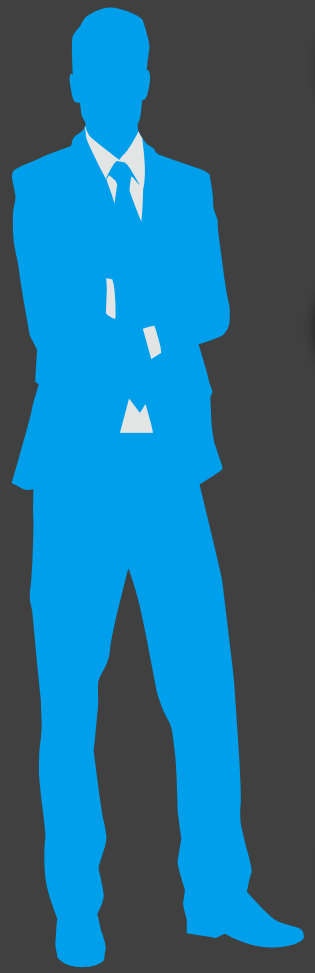
| Configuration | Data migration | Continuous Testing | Test All Consumers | Test edge cases |
|---|---|---|---|---|
|  |  |  |  |  |
| Check for Prod Configuration<br>and check for any missing keys | Dry run migrations and watch out for duplicate identity keys, and unique constraints | Don't get your guard down after a few successful migrations | Don't forget to inform, test and get sign-off from consuming applications about your migrations | Connection timeouts, Large file uploads, ensure that auto scaling policies are kicking in |

# PCF vs EKS Differences

PCF

1. Application PaaS
2. Higher level abstraction- Good for POCs and simple web applications and services
3. Offers vertical and horizontal scaling
4. Developers need not manage any infrastructure

EKS



1. Container PaaS
2. Less abstraction hence good when additional customization is needed.
3. EKS supports auto scaling of containers and volumes
4. Control plane managed by AWS, developers must manage worker nodes

# Here's where we need help …

1. Considerable Toil spent in Terraform, EKS Infrastructure Upgrades
2. How do we prevent CICD Pipeline getting slower as we keep adding different layers of automated testing?
3. How to be on top of security vulnerabilities at both layers- infrastructure and application?

THANK YOU!

Do you have any questions?