Devops Practices @Myntra for Resiliency in Cloud

About Myntra & Jabong

Over 30 million active users (Myntra & Jabong)

Over 5 million reqs/min during peak traffic.

Targeted $2 Billion run rate in 2018 for Myntra & Jabong

300+ Microservices managed by 470+ engineering team.

Around 50K+ CPU in Hybrid cloud setup between 4 data-centers.

Mix of bare-metal, AWS, FK cloud and Azure in multiple geography, powered by CDN partners like Akamai and Cloudinary.

What DevOps Do?

Everything as code	Application monitoring
Automate everything	Rapid feedback
Continuous Integration/Delivery	Rebuild vs. Repair
Application is always “releaseable”	Delivery pipeline

Devops goals at Myntra?

- Reduce MTTR during outages and have every incident taken to closure with proper RCA and action item.
- Implement a successful monitoring strategy to provide insights and increase operational efficiencies.
- Allow engineering to onboard applications without having to deal with complexities of configuration and operations.
- Address preservation and recovery of business in event of outages
- Bring predictability and simplicity to chaotic build/release processes and pre-prod testing.
- Enable comprehensive security, audit and compliance

Business Demands

- 4 Mega ‘End of Season Sale’ Every Year.
- Attracts 25 Times More Users than BAU.
- 6-8 hours Peak Traffic
- 10-15 minutes Burst Window - 1 Million+ Users

Needs & Challenges

- Bottlenecks for scaling.
  - Compute and Storage
  - Networking
  - Application Deployment
  - Load Balancers
- Quick Scaling- On Demand
  - Impossible with Bare Metal Environment
  - Cloud is the Only Solution
- Challenges with Auto Scale on cloud
  - Mass Deployment (>1000 Servers) in Real Time
  - Propagation of Config Updates
  - Not possible with Central resources like Load Balancers and Databases
  - Software Quality

How we fixed this?

- Containerization- Docker Containers
  - OS level virtualization of the Application
  - Lightweight and Immutable (Images)
- Orchestration
  - Kubernetes and Docker Swarm - Open Source
  - Automated Deployment, Scaling and Container Management
  - Kubernetes still runs on VMs
  - Underlying Hardware should also be Provisioned at the same rate
- Terraform - Infrastructure as Code
  - Automates Hardware Provisioning

How we fixed this? .. cont

- On demand server provisioning and Application deployments solved by Kubernetes and Terraform
- Service config propagation to Load Balancer still an Issue
- Load balancer being a SPoF - Deteriorates Auto Scale Objective
- Service Discovery is the Solution

Service Discovery

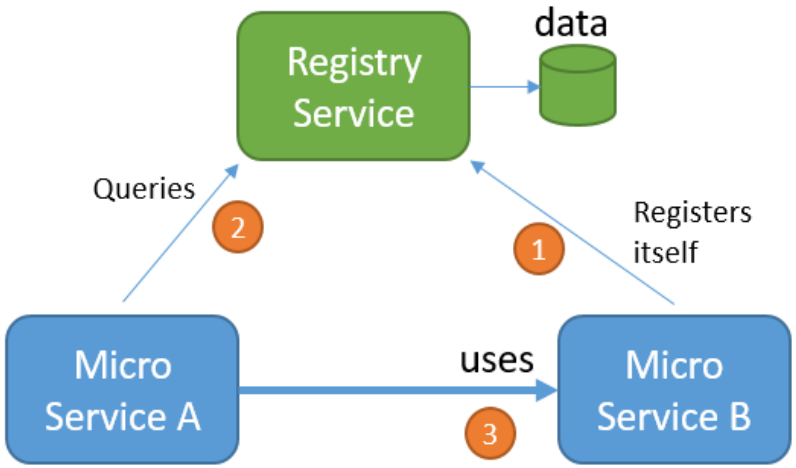
Services should ‘Register’ or ‘Deregister’

Service Discovery Decentralises the Load Balancing

Service A asks the Registry for Service B

Registry provides Service B’s Availability to Service A

Service A communicates to Service B at destined host



Containerisation

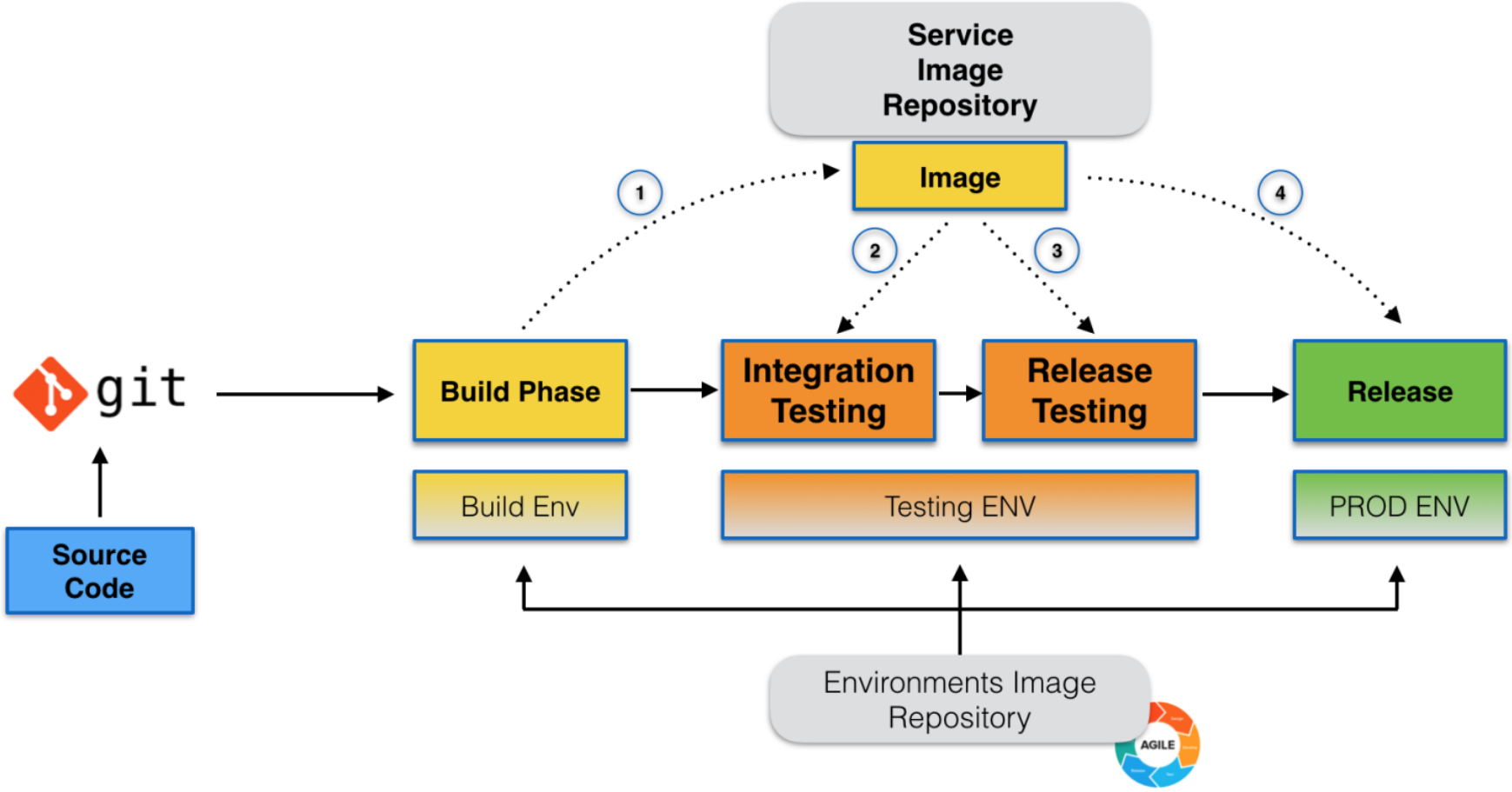
Environment Image Repository

- **Development Environment**
- **Build Environment**
- **Test Environment**
- **Production Environment**

Orchestration

- Application lifecycle Orchestration
  - Dev → Build → Test → Release
- Infrastructure Orchestration
  - On Demand (Sale Time)
  - Disaster Recovery

Application Lifecycle Orchestration



Infrastructure Orchestration

With Terraform we write, plan and create underlying Infrastructure as Code

Ansible configures and attaches the nodes to Kubernetes/Swarm Cluster

In house Controller orchestrates the Deployment and Scaling

Q & A