ALL DAY DEVOPS

NOVEMBER 6, 2019

Ashish Jadhav & Shivank Bansal

DevSecOps for Microservices with Resilience

ADDO
ALL DAY DEVOPS

Agenda

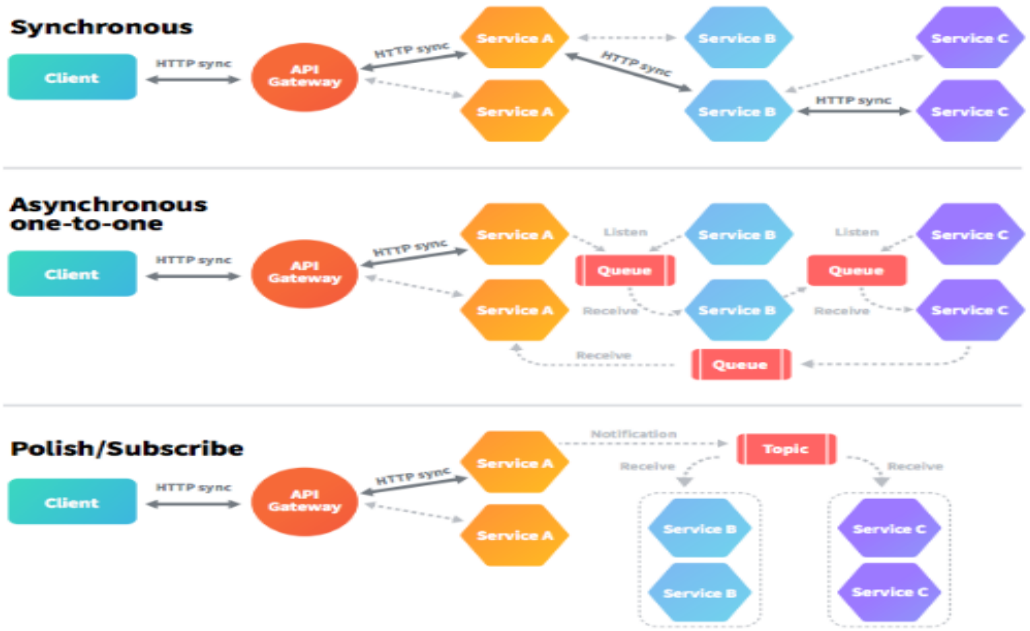
1. What is Microservices architecture
2. Why DevOps for Microservices
3. DevOps pipeline for Microservices
4. DevOps tool chain constitutes
5. Containers for Microservices
6. Microservices DevOps Orchestration
7. CI/CD Pipeline Workflow with Kubernetes
8. Release Management for Microservices

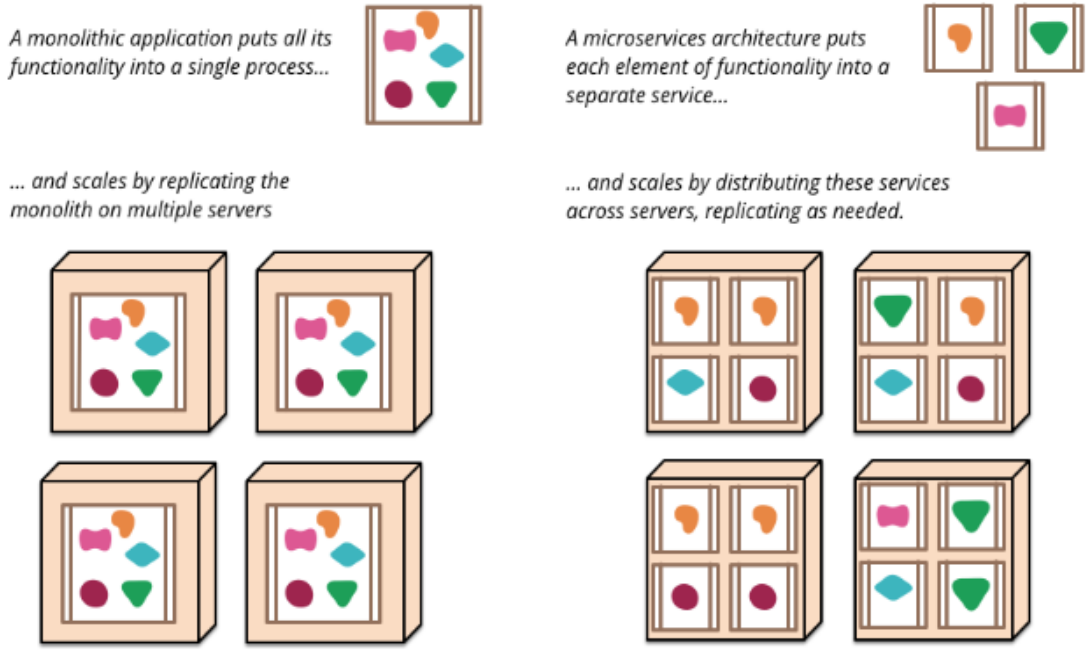
ADDO
ALL DAY DEVOPS

Microservices

What ?

- Microservices architect style is an approach to developing a single application as a suite of small services, each running on its own process and communicating with lightweight mechanism, often an http resource API.
- These services are built around business capabilities and independently deployable by fully automated deployment machinery.



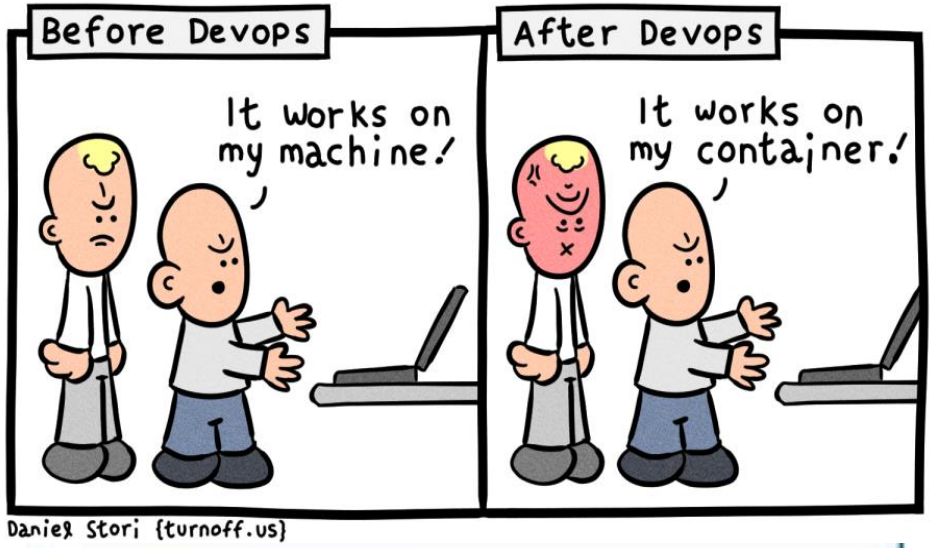


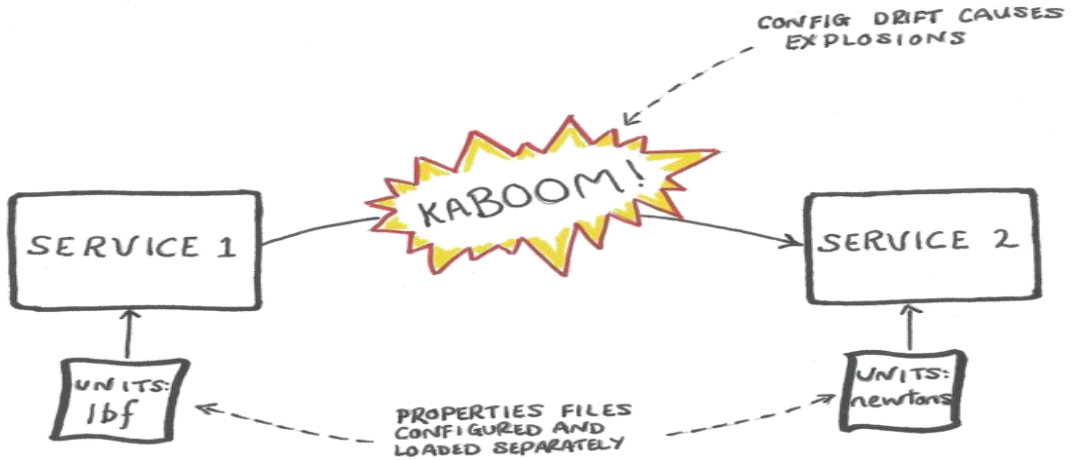
Why ?

- Simple design being focused on one business capability
- Independently deployable, Independently scalable & Strongly encapsulated
- Can be developed independently by different teams
- Can be developed using different programming languages and tools
- Decentralized Data Management

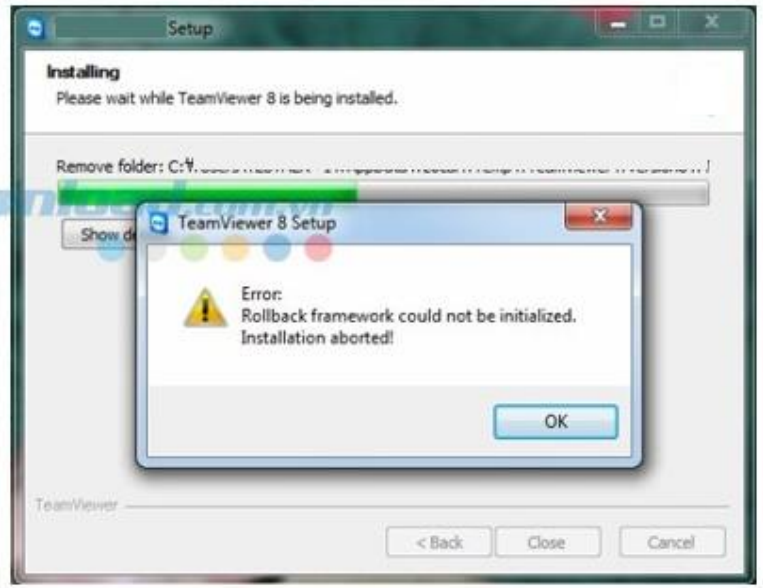
Why DevOps for Microservices (1/3)

Faster and Frequent releases

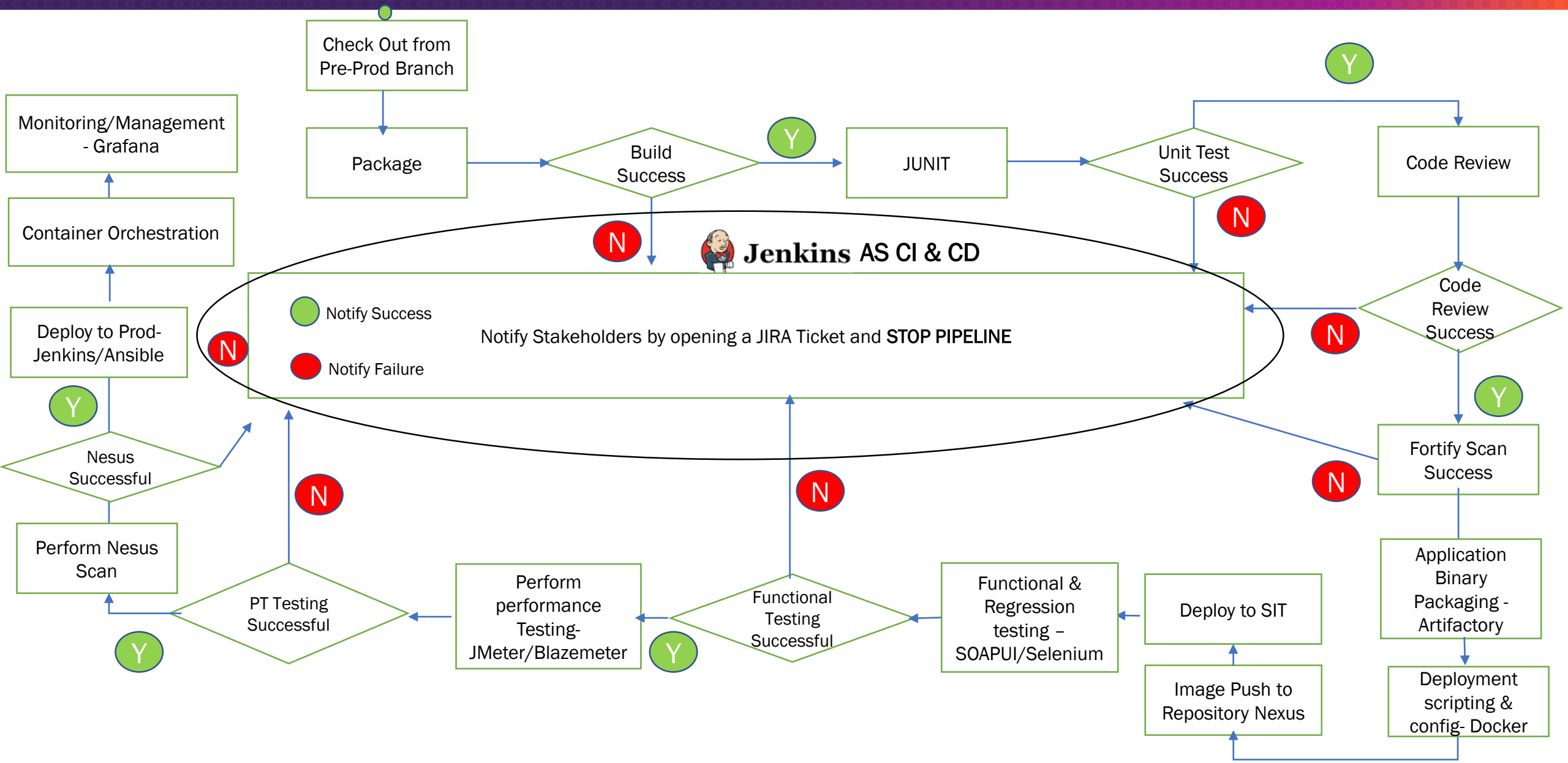




Why DevOps for Microservices (2/3)



DevOps Pipeline for Microservice



ADDO
ALL DAY DEVOPS

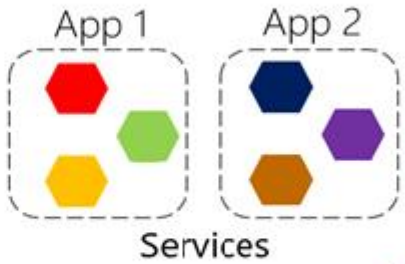
DevOps tool chain constitutes:

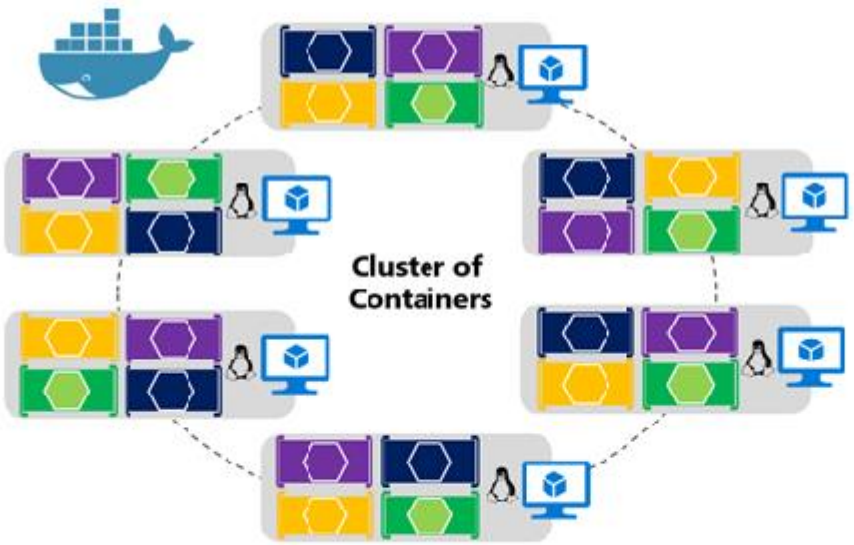
- Code — code development and review, source code management tools, code merging
- Build — continuous integration tools, build status
- Test — continuous testing tools that provide feedback on business risks
- Package — artifact repository, application pre-deployment staging
- Release — change management, release approvals, release automation
- Configure — infrastructure configuration and management, Infrastructure as Code tools like Ansible, Terraform etc.
- Monitor — applications performance monitoring, end–user experience

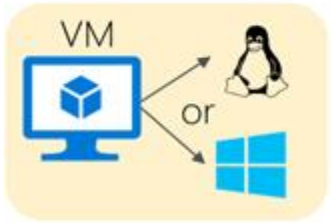
Containers for Microservices

Docker Containers for Microservices

- For each service instance you use one container
- Docker images/containers are "units of deployment"
- A container is an instance of a Docker Image
- A host (VM/server) handles many containers

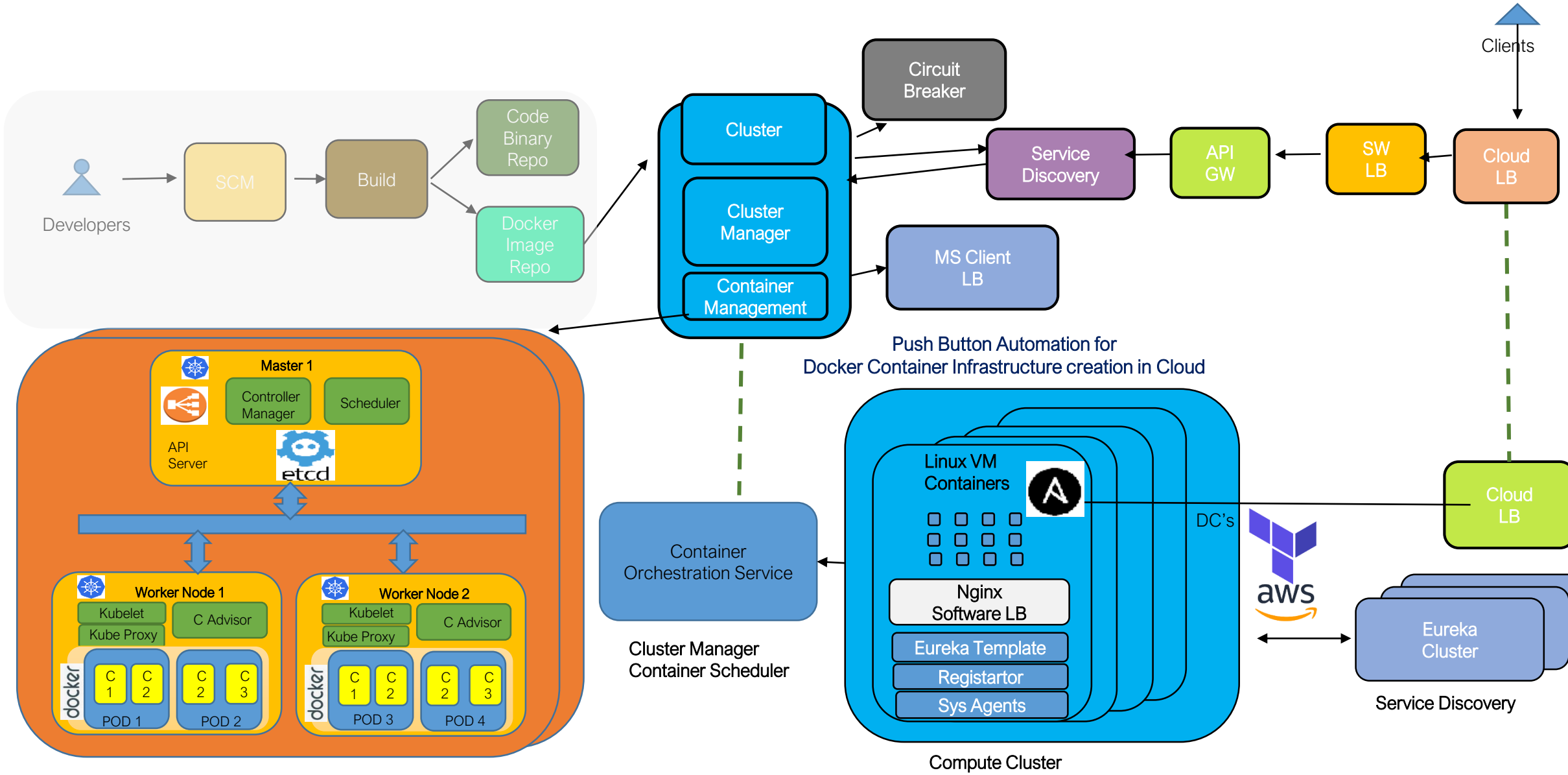




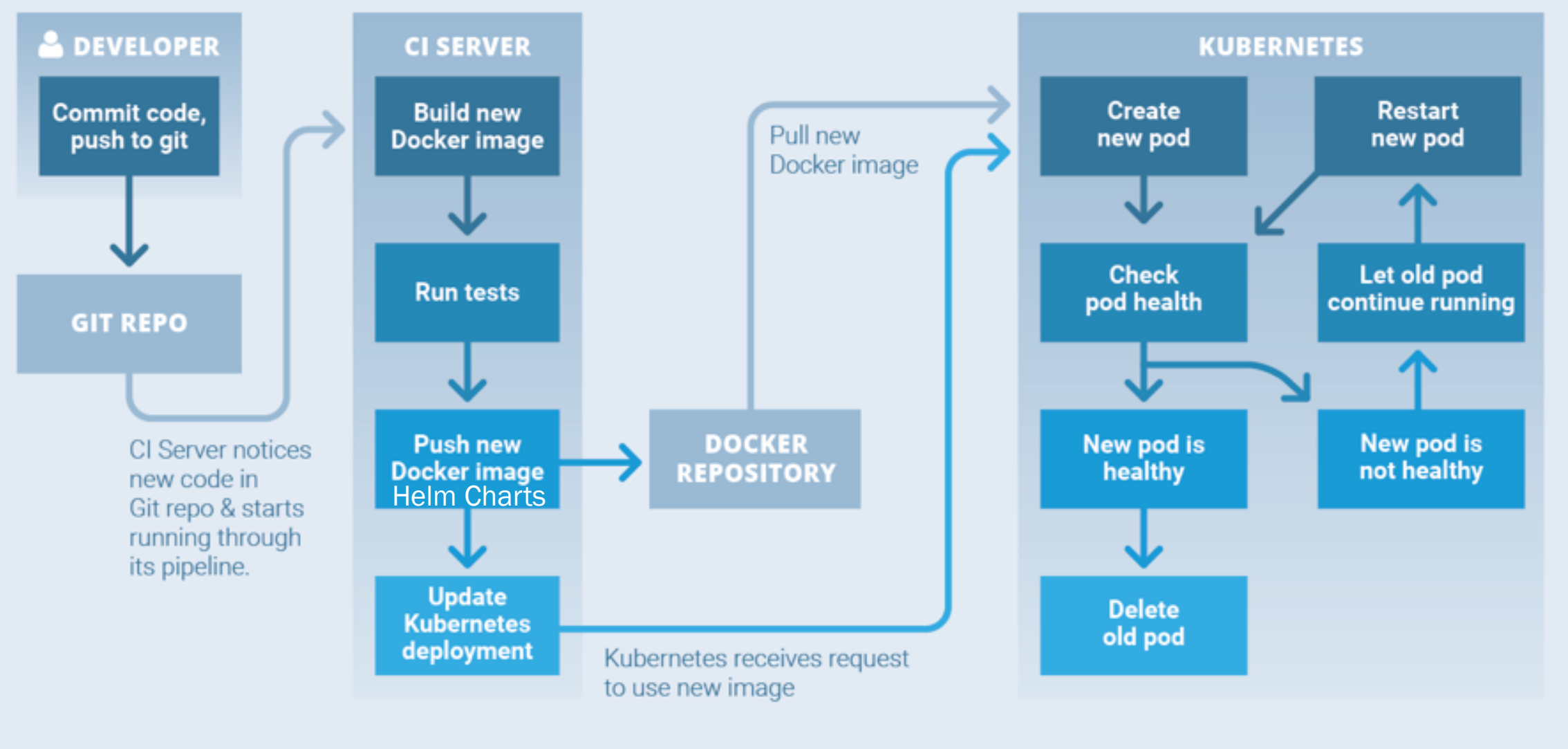



```
sudo docker run -d -p 3000:80 hello-world
sudo docker run -d -p 3001:80 hello-world
sudo docker run -d -p 3002:80 hello-world
http://localhost:3000
http://localhost:3001
http://localhost:3002
```

Microservices Orchestrations for DevOps



CI/CD Pipeline WF with Kubernetes



ADDO
ALL DAY DEVOPS

CI/CD with Kubernetes

The goal is to automate the following process:

- Checkout code.
- Compile code.
- Run test cases.
- Build docker images.
- Push images to docker registry.
- Pull new images from registry.
- Deploy the app on Kubernetes.
- Kubernetes' zero-downtime deployment
- As part of a rolling update, Kubernetes spins up separate new pods running your application while the old ones are still running. When the new pods are healthy, Kubernetes gets rid of the old ones.

ADDO
ALL DAY DEVOPS

Kubernetes Deployments

Setting up applications On Kubernetes Cluster, we will do the following.

- Create a Namespace
- Create a deployment yaml and deploy it.
- Create a service yaml and deploy it.
- Access the application outside the cluster on a Node Port Or
- Create a Ingress yaml for the servicename and deploy it to access the services outside the cluster.

➢Deployments manage the deployment of replica sets and are also capable of rolling back to the previous version.
➢We have got a controller in the Kubernetes master called the deployment controller which makes it happen. It has the capability to change the deployment midway.

```yaml
# hello-kubernetes.yaml

apiVersion: v1
kind: Service
metadata:
  name: hello-kubernetes
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: hello-kubernetes
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-kubernetes
spec:
  replicas: 3
  selector:
    matchLabels:
      app: hello-kubernetes
  template:
    metadata:
      labels:
        app: hello-kubernetes
    spec:
      containers:
      - name: hello-kubernetes
        image: paulbouwer/hello-kubernetes:1.5
        ports:
        - containerPort: 8080
```

Release Management for Microservices

- The expectation is to do seamless deployment with least/no impact to business. There are two ways to achieve this:
- Blue-Green Deployment: Container orchestration tools like Openshift, Kubernates provides support for blue green deployment with no downtime on production environment.

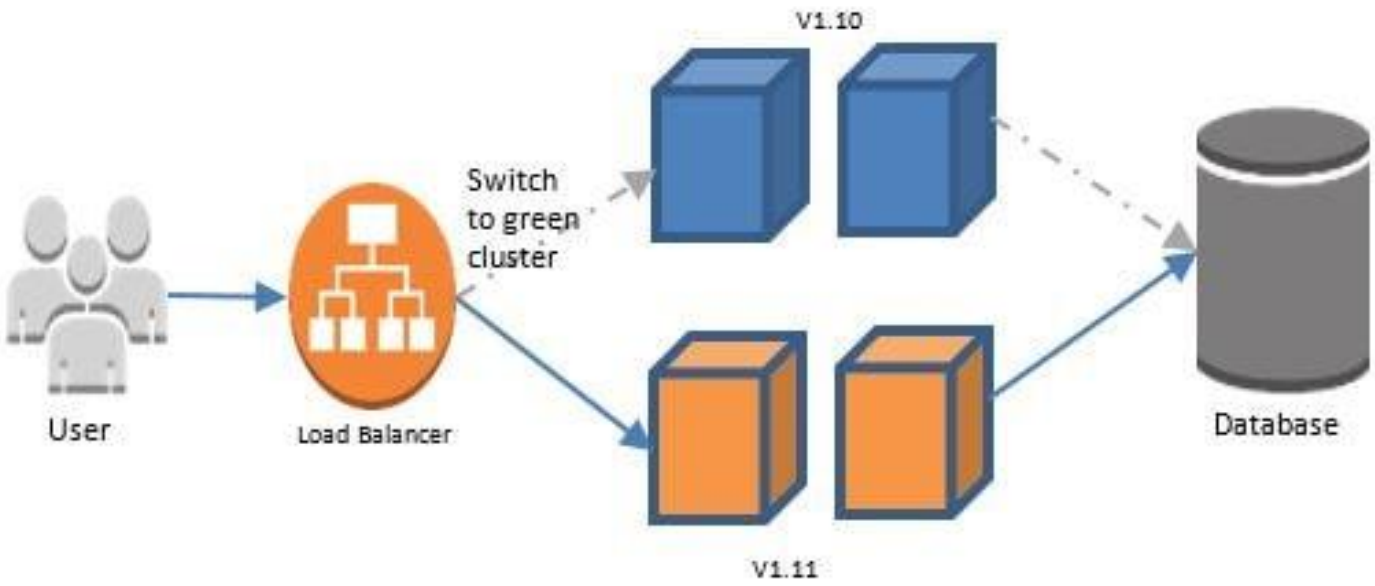


Figure 5: Blue - Green deployment model

Release Management for Microservices

- Canary deployment model is about deploying small, incremental replications sets of the new version and by controlling the traffic exposure to minimal percentage. There are multiple toggles that one can adopt apart from traffic exposure percentage to specific functionalities or to specific consumers.

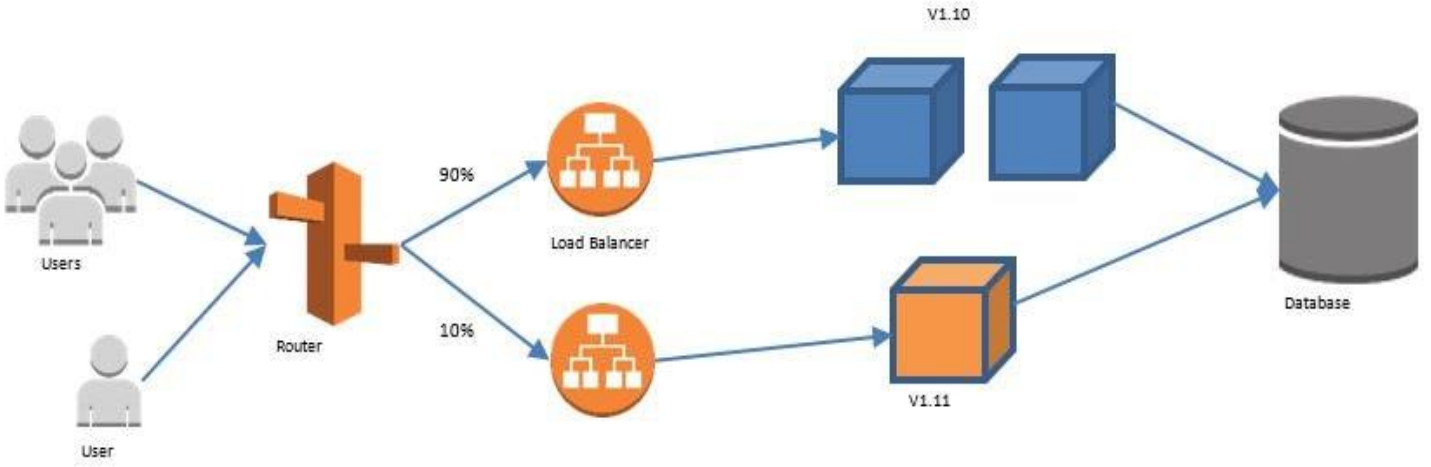


Figure 6: Canary deployment model

Thank you

Ashish.Jadhav@ril.com/Ashish_Jadhav@Hotmail.com

Shivank.Bansal@ril.com/Shivank.Bansal@gmail.com

https://ashishjadhavtechnologyforum.tumblr.com/

https://www.linkedin.com/in/ashish-jadhav-2364082/

Backup Slides

Microservices Devops Pipeline Component