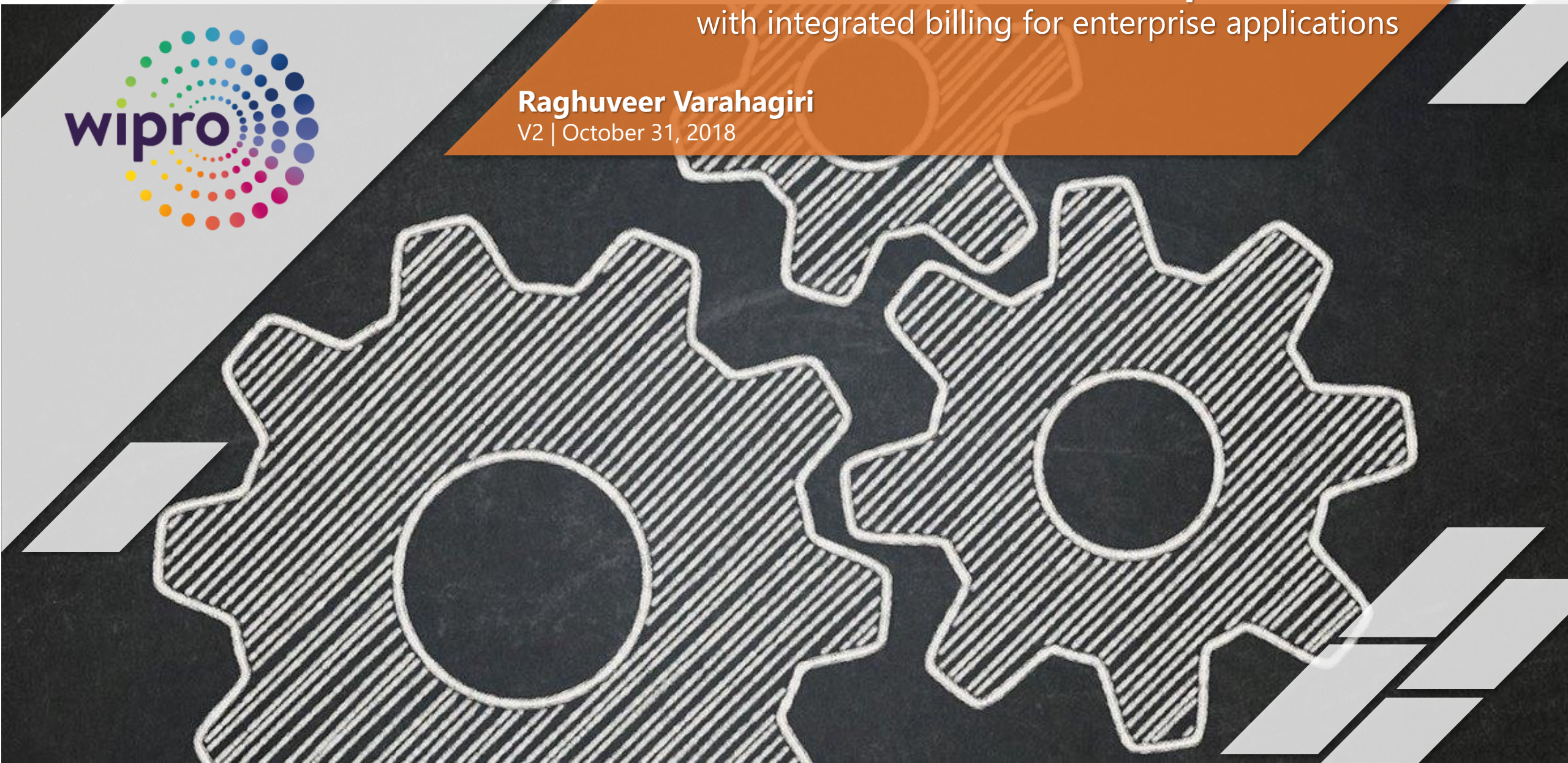
Solution Approach

AWS native DevOps Platform

with integrated billing for enterprise applications

Raghuveer Varahagiri

V2 | October 31, 2018



The Approach

1. Defining the problem
2. Decomposing the problem into components
3. Selecting the right tools for each part of the problem
4. Implement the solution for each part and validate
5. Integrate all the pieces to make them work together
6. Refine the solution / continuous improvement

Problem Definition

Implement an AWS Service Catalog that enables the provisioning and management of DevOps environments at enterprise scale with multiple cloud accounts.

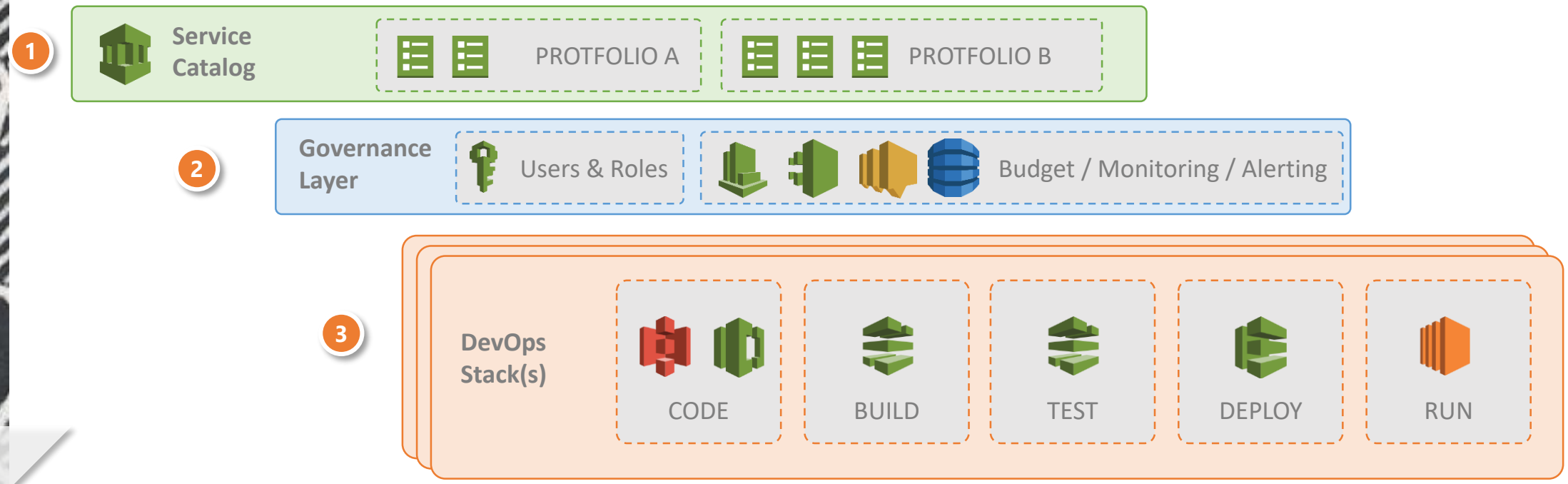
Desired Features:

- Definition of DevOps environment deployment blueprints as templates
- DevOps Lifecycle to be configurable based on use case (Dev/Test/Prod)
- Access restrictions on service catalog
- Budget controls and notifications based on usage
- Testing tools integration

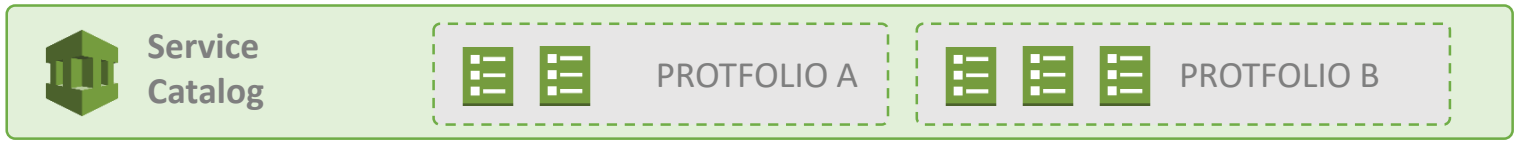
Problem Components

The solution to this problem can be viewed at three levels of abstraction:

1. Design of AWS Service Catalog
2. Design of governance processes using AWS tools
3. Design of DevOps CI/CD flow on AWS platform

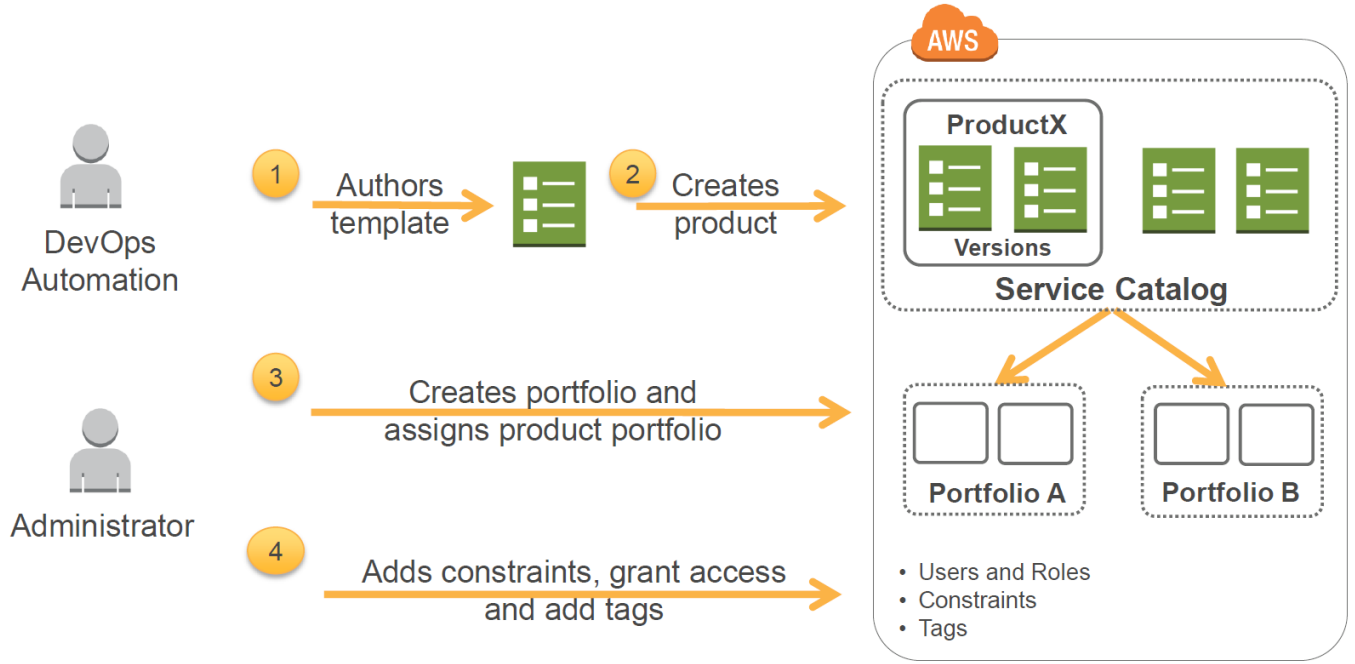


(1) AWS Service Catalog



The administrator defines the desired infrastructure in the form of AWS Cloud Formation templates and deploys them to the AWS Service Catalog, grouped by various portfolios.

The users and roles are mapped to constraints defined in the catalog – and control who can request provisioning of what configurations. For example: Dev team can only request dev configurations.



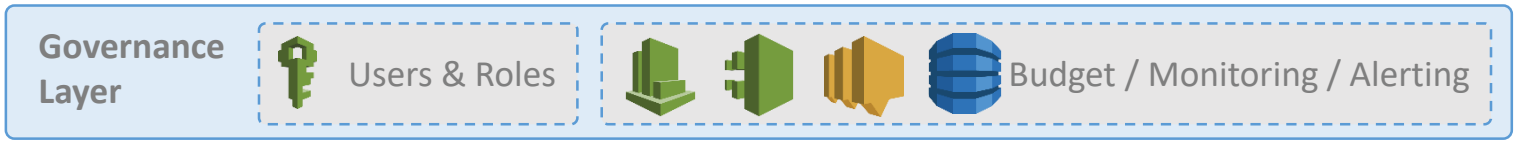
(1) AWS Service Catalog [Detailed Design]

Design Components of the AWS Service Catalog:

- CloudFormation Templates –
  - One each for the four environments DEV, TEST, STAGING, PROD
  - For the purpose of this exercise we use the following AWS reference architecture:

    Web Application Hosting: https://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf
  - Each CloudFormation template (“product” in the catalog) shall create the following resources:
    - Static Website Content : S3 Bucket
    - DNS resolution : Route 53 entries
    - Database Layer: Amazon RDS or DynamoDB instance backend with optional replication
    - Application Server Layer : EC2 instances optionally in an Auto Scaling Group fronted by an internal Elastic Load Balancer
    - Web Server Layer : EC2 instances optionally in an Auto Scaling Group fronted by an external Elastic Load Balancer
    - Optional Cloudfront distribution for content delivery
    - Each resource is tagged appropriately
    - CloudWatch metrics and other hooks for governance and budgeting processes
  - Depending on the environment selected, the corresponding CF template shall deploy a variation of this above stack with differences for instance: Only production environment needs an ELB and CloudFront configuration
  - Which user groups will be granted what access to the various resources created by the CF template will be driven by the CF template with some minor customizability through parameters when launching the product stack
  - Metadata section of the CF template can be used to provide rich context and prompts to the user when selecting the various parameters
- Service Catalog and Portfolios
  - Service Catalog, Portfolios, and Products need to be defined using the CF templates above
  - User permissions to access the Service Catalog and the constraints are defined and mapped to IAM User Roles and Groups

(2) Governance Processes



The governance layer consists of a number of AWS services and tools to tie together the AWS Service Catalog and the DevOps stacks that will be provisioned.

And indicative list of services and their purpose:

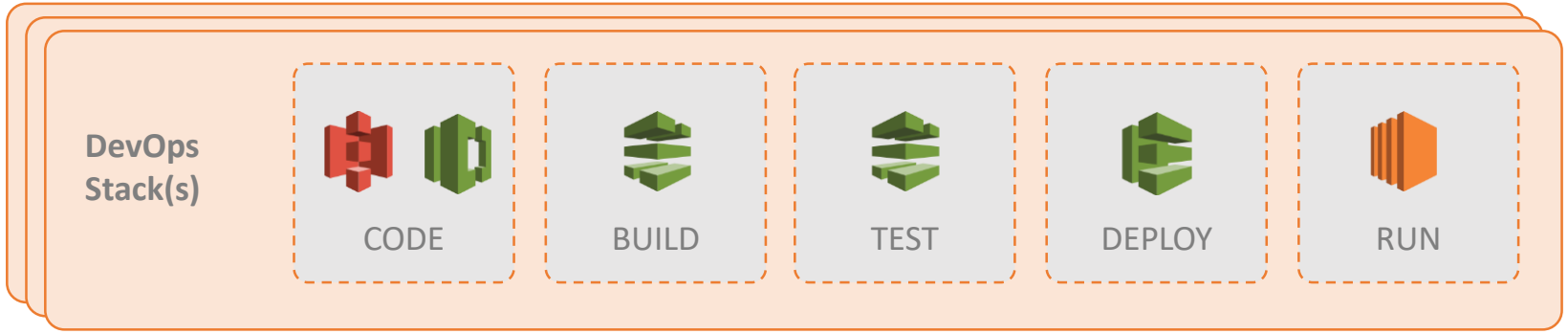
- IAM – Definition of roles, groups and users
- Cloudwatch – Setup monitoring and alerts based on budgets and utilization
- Dynamo DB – could be used to store budgets, configuration
- Cloudtrail – operational and risk audit trails
- S3 – Storing of logs for future audits
- SNS – Notifications on budget usage

(2) Governance Processes [Detailed Design]

Design components of the Governance layer:

- Project Roles
  - User Groups are defined with permissions on the AWS Service Catalog
  - For example “PRODUCTION” environment can be requested only by users who belong to “Production Deployment” group
  - These groups are mapped to the constraints on the Service Catalog product definition
- Budget Control
  - For each resource that is covered by budget controls, the CF template ensures that (1) the resource is tagged appropriately and (2) cloudwatch metrics are defined and tied to the resources so that actual utilization is monitored and logged
  - A DynamoDB table would maintain the budge limits – per resource type, per stack, per environment type, overall limits, etc.
  - A lambda function triggered by cloudwatch compares actual utilization versus the budget and if it meets or exceeds a threshold (say 50%, 75%, 100% of budget) then it generates an SNS notification to the budgetary control group

(3) DevOps Stack : CI/CD Pipeline



The DevOps stacks for Dev / Test / Prod environments are defined in the form of Cloud Formation templates. These need to be designed and built as an project of its own and tested thoroughly before being incorporated into the service catalog.

Permissions/roles that are required by the end users of these stacks will need to be automatically provisioned and granted.

The infrastructure provisioned shall be tagged appropriately to allow monitoring of usage against corresponding budgets.

And indicative list of services and their purpose:

- Code – S3 / CodeCommit
- Build and Test – Code Pipeline
- Deploy – Code Deploy
- Run – EC2 / ECS
- Monitoring - CloudWatch

(3) DevOps Stack : CI/CD Pipeline [Detailed Design]

AWS CodePipeline is a service that automatically detects changes to code (in an S3 bucket, Github or AWS CodeCommit repository), and automatically builds, tests and deploys those changes.

For the purpose of this project we could limit the source repository to say S3 bucket and use Elastic Beanstalk or OpsWorks for deployment of the changes detected.