tcs 50
Experience energy

DevOps and IT Agility: For Evangelists and Proposal Writers

DevOps CoE

September 2018

Agile Refresher (1/2): What it is About and Not About

Primarily addresses Agile Dev

Gap – What about IT Agility without Agile Ops?

4 Core values of Agile

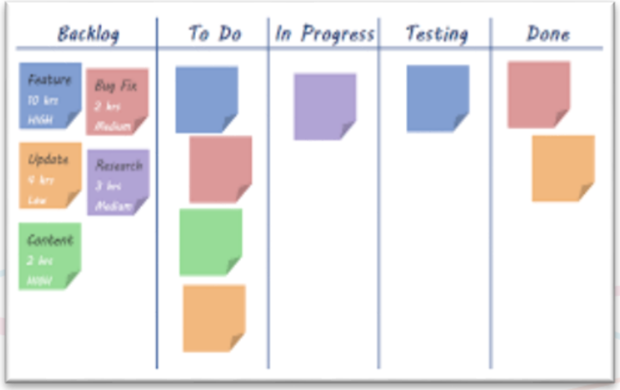
Readings: Agile Manifesto

- ✓ individuals and interactions over processes and tools
- ✓ working software over comprehensive documentation
- ✓ customer collaboration over contract negotiation
- ✓ responding to change over following a plan

Agile Refresher (2/2): What and How To Do

What Should Happen on the Ground?

- ✓ Your daily scrum should have **both Dev (with QA) and Ops teams**
- ✓ Note (w. ref. t. Agile core values)
  - **Interactions (over processes)** does not mean that you delegate frequently and bully your juniors
  - **Documentation** should be auto-generated rather than you doing it (you better spend time coding)
  - **Collaboration (over contract)** does not mean that you always say "yes" to your customer
  - **Responding to change** still has to go by a plan (your backlog with Agile estimates), not ad hoc



Is this Scrum or Kanban Board? Either, based on context …

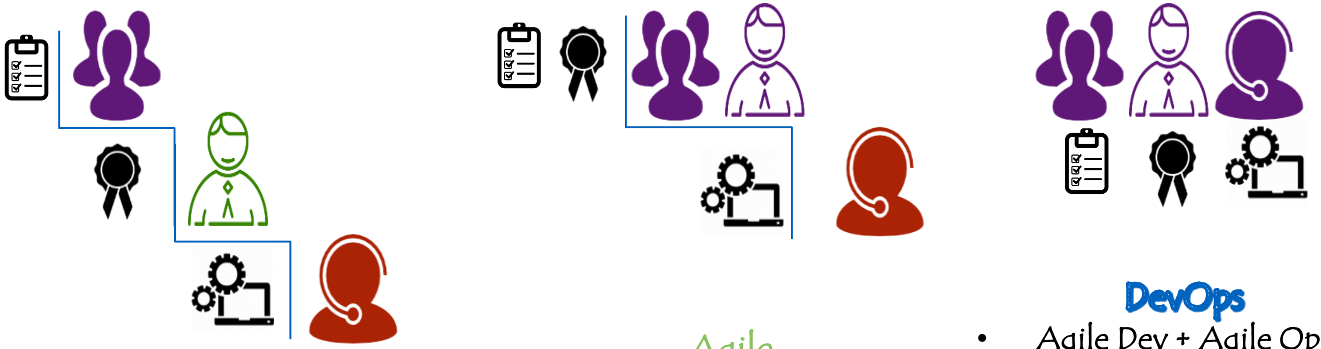
- It is **Scrum board** if the backlog consists of user stories for a prior planned Sprint (say for a development project); so no further addition to the backlog is possible for this Sprint: **Metric is Velocity**
- It is **Kanban board** if the backlog is ever flowing with new items (say for production support); the only constraint can be no. of items that can be accommodated "in progress" at any point of time: **Metric is Lead Time**

DevOps Refresher (1/2): How did it Evolve?

Patrick Debois coined the word in 2008; Collaborated with Andrew Clay Shafer on Agile Infrastructure

Names to know

Patrick Debois
Andrew Clay Shafer
John Allspaw
Paul Hammonds
Gene Kim
Dr. Nicole Forsgren
Jez Humble
John Willis
Alan Shimel
James Turnbull



Waterfall

- Sequential and big bang releases
- Low automation
- Strictly defined span of control and hand-offs

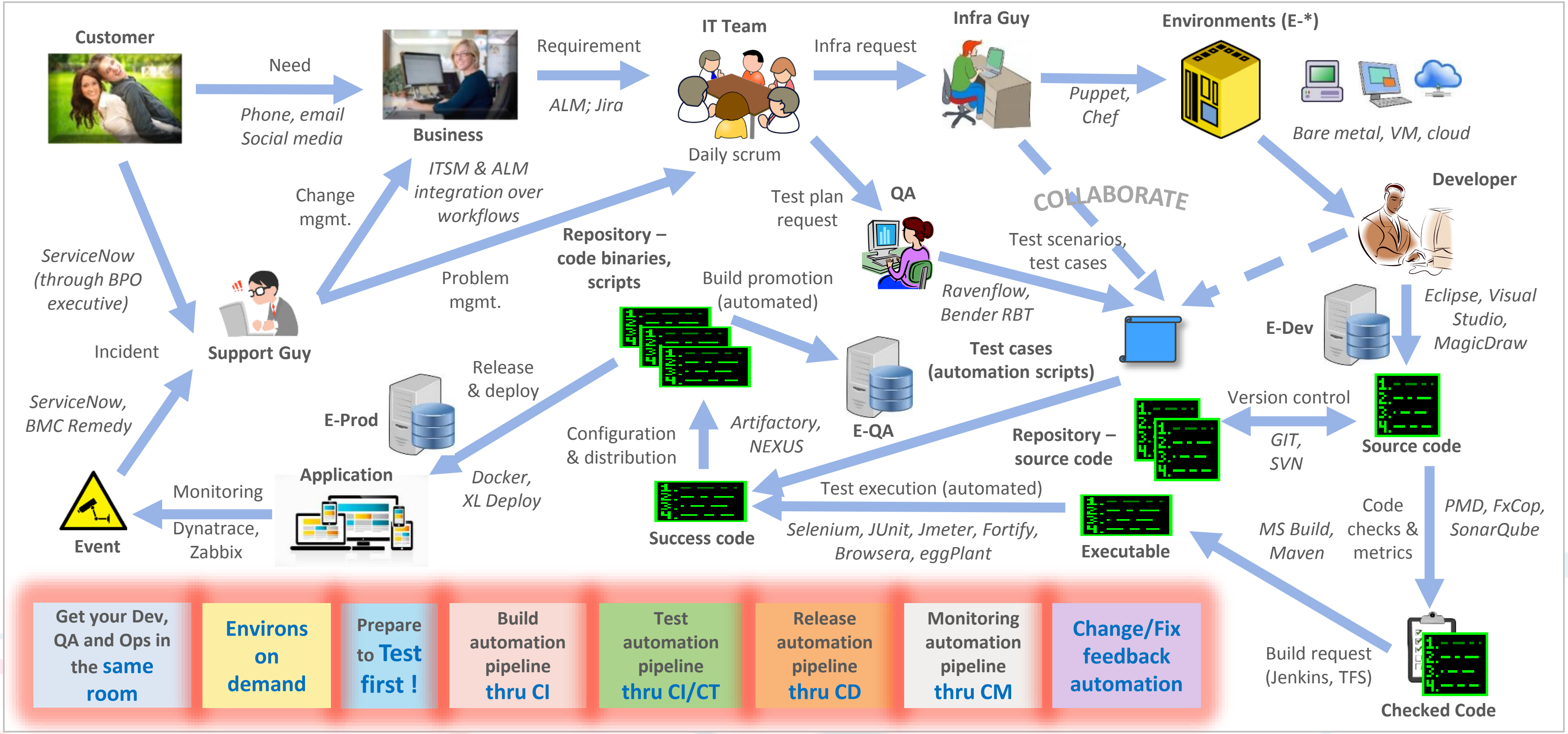
Agile

- Incremental iterative/ parallel processing
- Point automations
- Dev and Ops are treated as separate teams

DevOps

- Agile Dev + Agile Ops
- Continuous processing
- End to end automation
- No strictly defined hand-offs – Dev and Ops are the same

DevOps Refresher (2/2): How To Do



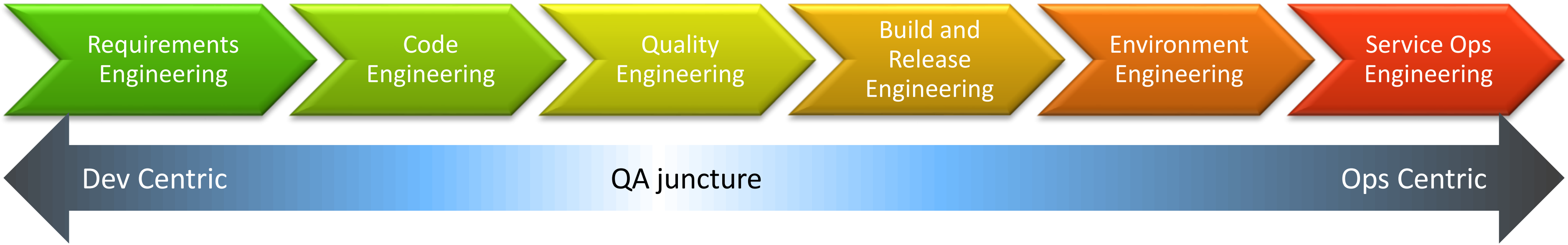
- Get your Dev, QA and Ops in the same room
- Environs on demand
- Prepare to Test first !
- Build automation pipeline thru CI
- Test automation pipeline thru CI/CT
- Release automation pipeline thru CD
- Monitoring automation pipeline thru CM
- Change/Fix feedback automation

CoE's Point of View (PoV)

Why do we need a Point of View?

- ✓ It gives a unique understanding of what DevOps is
- ✓ Helps us map larger IT strategy to on-ground implementation details
- ✓ Aligns all our frameworks and solution assets to a common understanding

What is the PoV?



Requirements Engineering

Code Engineering

Quality Engineering

Build and Release Engineering

Environment Engineering

Service Ops Engineering

Dev Centric

QA juncture

Ops Centric

Six Engineering Principles → Set of Practices spanning …

✓ **People** – How they should be re-structured for better collaboration

✓ **Process** – How IT cycle times can be reduced and quality enhanced

✓ **Technology** – How much can be automated

PoV Explained: Requirements drive Everything!

Requirements Engineering

Rationale: As requirements flow from business to IT teams (Dev-QA-Ops), how easy and quick is it to manage requirements across SDLC, or trace back in case of failures or changes requested?

- ✓ People – Is there a common understanding of requirements across Dev-QA-Ops? To whom does it come first, from business?
- ✓ Process – How is traceability to requirements established? What paths do functional and non-functional requirements take during SDLC or production support processes?
- ✓ Technology – Is there a single tool to capture and manage requirements? Are requirements converted to a test-ready format automatically?

Terms

Functional Requirement

Non-functional Requirement

Translate: Speech → Text → Test Script

Traceability Matrix

Impact Analysis

TDD / BDD

User Story

Tools

HP ALM

Jira

Speech to Text software

Cucumber

JBehave

PoV Explained: What is IT without Code (and Data)?

Code Engineering

Rationale: How is code written today – how are both speed (or writing) and quality of code ensured? How easy is it to manage code and ensure code traceability?

- ✓ People – How do developers work together on different code branches that need to be merged? How do developers collaborate with QA and Ops while writing code for say, non-functional requirements?
- ✓ Process – How is a large code base changed and managed? Are large applications broken into multiple small chunks for ease of maintainability? What about associated database changes?
- ✓ Technology – Are there tools for automated code generation? Tools or IDE plug-ins for code quality, code branching and merging? Does the application architecture facilitate easy maintainability?

Terms

Code Quality

RAD

Microservices Architecture

Model driven Development

Branching and Merging

Feature Toggle

API based Services

Code Metrics

Technical Debt

Tools

SonarQube

SubVersion

IBM Rational Rhapsody

CheckStyle

Eclipse IDE

GIT / GitLab

Acceleo

PMD

PoV Explained: Does it Work as Intended? Every time?

TATA CONSULTANCY SERVICES

Quality Engineering

Rationale: Quality issues caught early on reduces unnecessary rework (and effort) in resolving quality issues every time, thereby lowering cost of quality, reducing time to fix, and having worry-free production runs.

- ✓ People – Do developers code against test cases, or are they kings – they develop, and then push the code to testers? Do developers and testers fix defects together? Do Ops also get involved in testing cycles?
- ✓ Process – When does tests happen – along with coding, or much later? Do tests happen for both application quality and infrastructure reliability? What types of tests are covered? What about Security?
- ✓ Technology – How much of tests are automated – both functional and non-functional? Is testing seamlessly integrated to the overall delivery pipeline, or needs to be manually triggered?

Terms

Unit Testing

SIT

Test Beds

Metric - Change Success Ratio

Test Data Management

Test as Code

Site Reliability Engineering

DevSecOps

TDD / BDD

Integration Tests

Continuous Testing

Technical Debt

Tools

Selenium

HP LoadRunner

Browser Testing

Perfecto

HP ALM

HP Fortify

Jasmine

JMeter

OWASP ZAP

PoV Explained: The Code has to End up Somewhere!

Build and Release Engineering

Rationale: For most, DevOps starts here; Ops helping out Dev in doing seamless build and release of code. What is the extent of manual intervention (hence, delays) in the overall build and release process?

- ✓ People – Do developers frequently blame operations for release failures ("the release environment does not match my development environment!")? Or vice versa ("Your code itself is bad!")?
- ✓ Process –How are multiple release pipelines managed? What about database releases? How easy are rollbacks? How is deployment downtime managed? Are tests adequately integrated to the pipeline?
- ✓ Technology – Is CI / CD practiced? Release as one-click self service for developers? How are build and release on-premise, cloud or hybrid (mix of on-premise and cloud) managed? Containers, anyone?

Terms

Undeployment

Blue-Green Deployment

Zero Downtime Deployment (ZDD)

Rollback

CI / CD

Continuous Integration (CI)

Containers and Images

Continuous Delivery (CD)

Pipeline as Code

Tools

Maven

Octopus Deploy

Bamboo

Jenkins

IBM UrbanCode Deploy

IBM Rational Team Concert (RTC)

XLDeploy

NEXUS

Artifactory JFrog

PoV Explained: To Think, You Still need a Brain

tcs 50 Experience energy | TATA CONSULTANCY SERVICES

Environment Engineering

Rationale: Only dev agility does not reduce IT cycle times; we need agility in creating and managing infrastructure on-demand. How quickly can environments be churned out and replicated?

- ✓ People – Does Ops makes Dev wait for hours or days to provision an environment? Can Dev do a part of Ops' job in creating his/her own machine when needed? Does testers have to wait as well?
- ✓ Process –How quickly are environments created (and destroyed)? Can environments be easily replicated? Can test data be easily refreshed? Are environments adequately monitored, tested and issues fixed?
- ✓ Technology – How are different types of infrastructure managed – VMs on-premise, on cloud, hybrid? Are containers used? How are server farms or clusters managed? Infrastructure as code, anyone?

Terms

Server Farms

IaaS

PaaS

Hyper Convergence Infrastructure (HCI)

Cloud

Immutable Infrastructure

Virtual Machines (VMs)

Containers and Images

Test Driven Infrastructure

Infrastructure as Code

Tools

Puppet

Amazon Web Services (AWS)

Azure

Chef

Packer

Terraform

Docker

Ansible

Kubernetes

Pivotal

OpenShift

Amazon Machine Image (AMI)

PoV Explained: Living Happily Ever After needs Support

TCS 50 Experience energy | TATA CONSULTANCY SERVICES

Service Ops Engineering

Rationale: In pre-prod, everything works fine … how to recover fast when code crashes in production? Identify and fix recurring issues? And then, how is everything monitored to detect failures fast?

- ✓ People – Do Dev teams collaborate with production support teams? Or they fight out more often on whose failure it is – Dev, QA, Infra, Security, Tom, Dick, Harry? How are failures communicated?
- ✓ Process –Are all code execution touchpoints and infrastructure monitored across pre-prod and prod? How fast are production issues circumvented and fixed? What about recurring issues? Change requests?
- ✓ Technology – Is continuous monitoring established across pre-prod and prod? Are CR workflows automated? Are circumventing and fixing issues automated? Self-healing systems, anyone?

Terms

Self-healing Systems

Event

ITSM

Incident

Bot-based Resolution

Metrics - MTTR

Continuous Monitoring

Change Request (CR)

Metrics - MTBF

DevSecOps

Tools

Zabbix

PagerDuty

Elastic-LogStash-Kibana (ELK)

BluePrism

Dynatrace

AppDynamics

SolarWinds

ServiceNow

BMC Remedy

People: Why Culture is Important

Adopting Agile takes time; sticking to Agile practices is a challenge

```
class AthleteList(list):
    def __init__(self, a_name, a_dob=None, a_times=[]):
        list.__init__([])
        self.name = a_name
        self.dob = a_dob
        self.extend(a_times)

    @staticmethod
    def sanitize(time_string):
        if '-' in time_string:
            splitter = '-'
        elif ':' in time_string:
            splitter = ':'
        else:
            return(time_string)
        (mins, secs) = time_string.split(splitter)
        return(mins + '.' + secs)
```

People either love to code or to configure; mature DevOps is all about loving to code

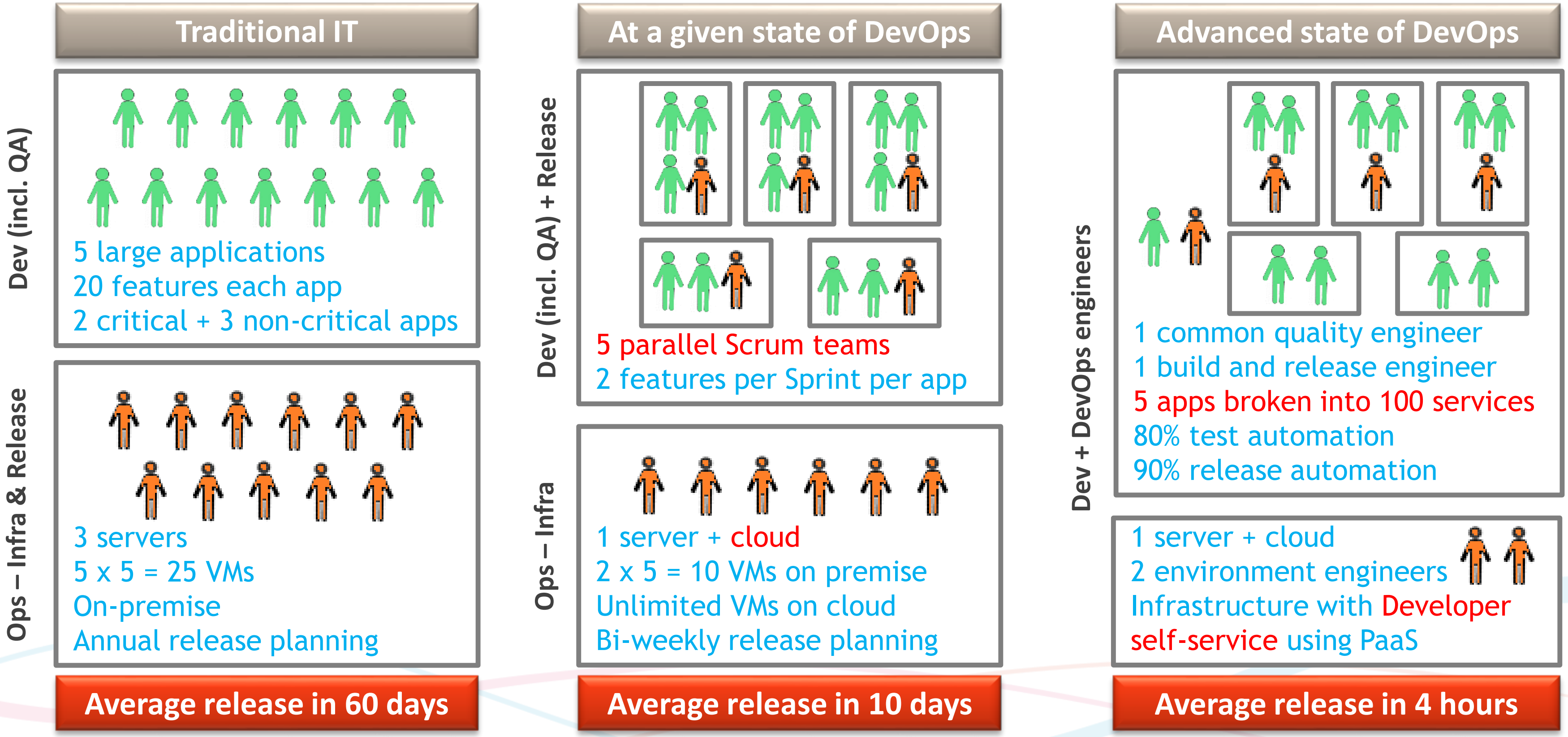
Automation is inversely proportional to doing things manually (losing jobs?!)

Caution: Customers come to TCS thinking of us as tool implementers for DevOps; however, implementing and integrating tools without understanding the people and process implications result in execution failures, which the customer can then easily attribute to TCS.

What to do: Hence, first talk to your customer on their cultural readiness (for Agile) and IT processes being practiced, before jumping on to tools' discussions. Try to find out:

- ✓ Who is driving DevOps programs – Dev organization, Ops organization, or QA?
- ✓ Is the customer looking at creating a DevOps group, CoE or factory? Has a champion been identified?
- ✓ Are Dev, Ops and QA geographically distributed? Are there multiple vendors?
- ✓ What are the tollgates (read, BARRIERS or WALLS) created where multiple vendors operate?

People Architecture Change Patterns



Traditional IT

Dev (incl. QA)

5 large applications
20 features each app
2 critical + 3 non-critical apps

Ops – Infra & Release

3 servers
5 x 5 = 25 VMs
On-premise
Annual release planning

Average release in 60 days

At a given state of DevOps

Dev (incl. QA) + Release

5 parallel Scrum teams
2 features per Sprint per app

Ops – Infra

1 server + cloud
2 x 5 = 10 VMs on premise
Unlimited VMs on cloud
Bi-weekly release planning

Average release in 10 days

Advanced state of DevOps

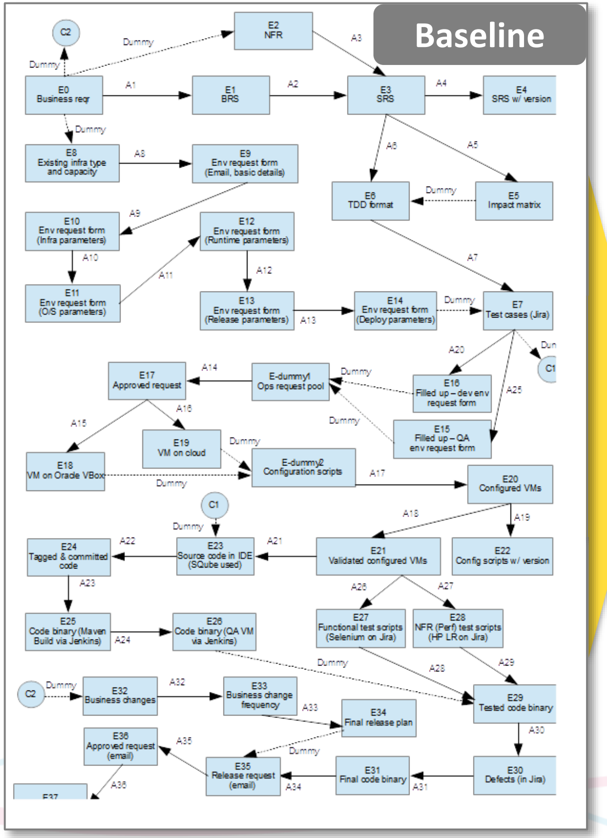
Dev + DevOps engineers

1 common quality engineer
1 build and release engineer
5 apps broken into 100 services
80% test automation
90% release automation

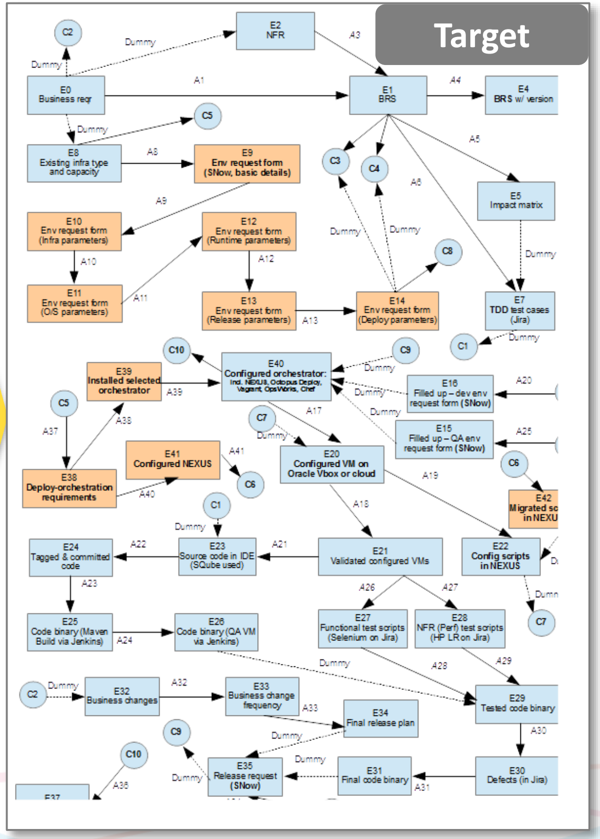
1 server + cloud
2 environment engineers
Infrastructure with Developer self-service using PaaS

Average release in 4 hours

Process Optimization Patterns


Baseline
C2
Dummy
E2 NFR
A3
E0 Business reqr
A1
E1 BRS
A2
E3 SRS
A4
E4 SRS w/ version
Dummy
E8 Existing infra type and capacity
A8
E9 Env request form (Email, basic details)
A6
A5
E6 TDD format
Dummy
E5 Impact matrix
A9
E10 Env request form (Infra parameters)
E12 Env request form (Runtime parameters)
A10
A11
A12
A7
E11 Env request form (O/S parameters)
E13 Env request form (Release parameters)
A13
E14 Env request form (Deploy parameters)
Dummy
E7 Test cases (Jira)
Dun
C1
A20
A14
E17 Approved request
E-dummy1 Ops request pool
Dummy
E16 Filled up – dev env request form
A25
A16
A15
Dummy
E15 Filled up – QA env request form
E19 VM on cloud
Dummy
E18 VM on Oracle VBox
Dummy
E-dummy2 Configuration scripts
A17
E20 Configured VMs
C1
Dummy
A18
A19
E24 Tagged & committed code
A22
E23 Source code in IDE (SQube used)
A21
E21 Validated configured VMs
E22 Config scripts w/ version
A23
A26
A27
E25 Code binary (Maven Build via Jenkins)
A24
E26 Code binary (QA VM via Jenkins)
E27 Functional test scripts (Selenium on Jira)
E28 NFR (Perf) test scripts (HP LR on Jira)
Dummy
A28
A29
C2
Dummy
E32 Business changes
A32
E33 Business change frequency
A33
E34 Final release plan
E29 Tested code binary
A30
E36 Approved request (email)
A35
Dummy
E35 Release request (email)
A34
E31 Final code binary
A31
E30 Defects (in Jira)
A36
E37

Deployment cycle time is **21 days**


Target
C2
Dummy
E2 NFR
A3
E0 Business reqr
A1
E1 BRS
A4
E4 BRS w/ version
C5
Dummy
E8 Existing infra type and capacity
A8
E9 Env request form (SNow, basic details)
C3
C4
A5
A6
E5 Impact matrix
A9
E10 Env request form (Infra parameters)
E12 Env request form (Runtime parameters)
Dummy
Dummy
C8
Dummy
A10
A12
A11
E11 Env request form (O/S parameters)
E13 Env request form (Release parameters)
A13
E14 Env request form (Deploy parameters)
E7 TDD test cases (Jira)
C1
Dummy
C10
C9
Dummy
E39 Installed selected orchestrator
A39
E40 Configured orchestrator: Incl. NEXUS, Octopus Deploy, Vagrant, OpsWorks, Chef
Dummy
E16 Filled up – dev env request form (SNow)
A20
C5
A38
A37
C7
A17
Dummy
E15 Filled up – QA env request form (SNow)
A25
Dummy
E41 Configured NEXUS
A41
E20 Configured VM on Oracle Vbox or cloud
C6
A19
E38 Deploy-orchestration requirements
A40
C6
E42 Migrated sc in NEXU
C1
Dummy
A18
E24 Tagged & committed code
A22
E23 Source code in IDE (SQube used)
A21
E21 Validated configured VMs
E22 Config scripts in NEXUS
D
A23
A26
A27
Dumn
E25 Code binary (Maven Build via Jenkins)
A24
E26 Code binary (QA VM via Jenkins)
E27 Functional test scripts (Selenium on Jira)
E28 NFR (Perf) test scripts (HP LR on Jira)
C7
Dummy
A28
A29
C2
Dummy
E32 Business changes
A32
E33 Business change frequency
A33
E34 Final release plan
E29 Tested code binary
A30
C9
Dummy
C10
Dummy
Dummy
E35 Release request (SNow)
E31 Final code binary
A31
E30 Defects (in Jira)
A36
E37

Deployment cycle time is **12.5 days**

a. First establish DevOps infra (as code) and metrics dashboard [Orange boxes]

b. Shift left tests before dev; dev should code against test cases

c. Establish continuous monitoring and feedback loops for early defect detection

d. Establish multiple release pipelines preferably as code

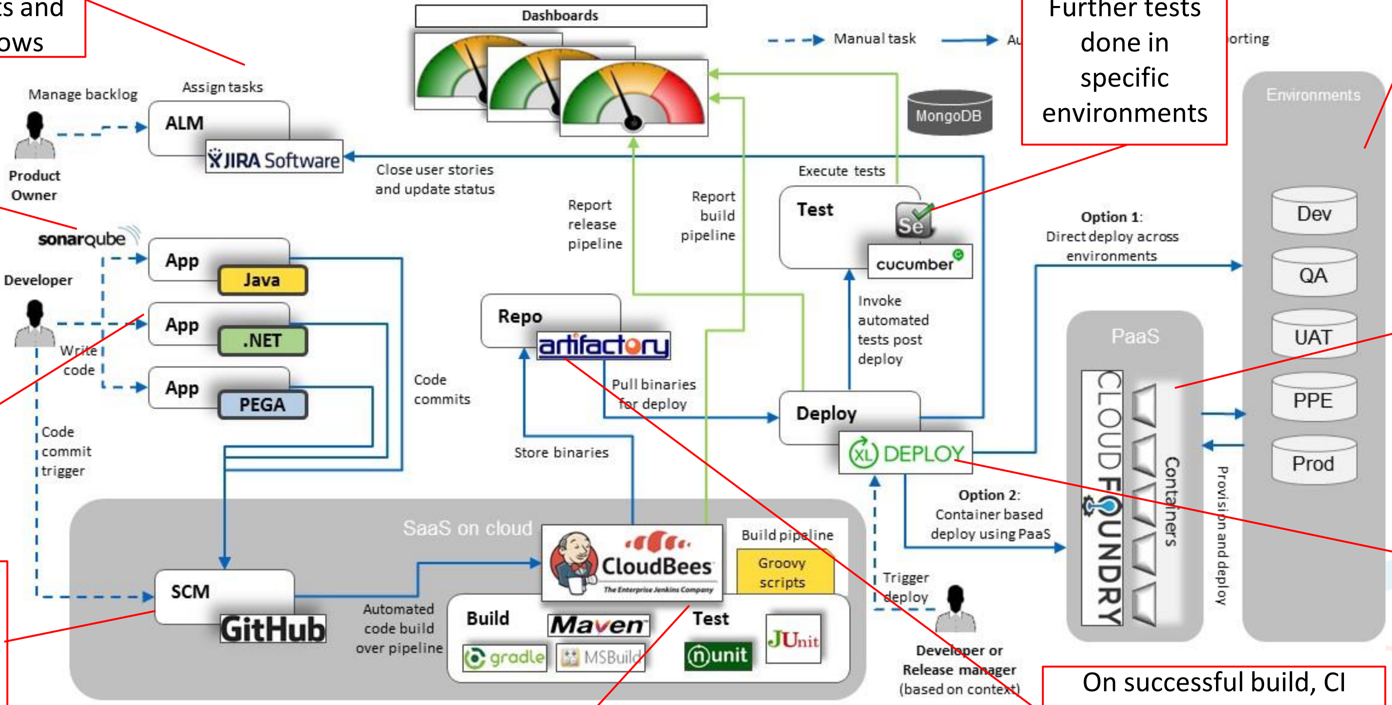
e. Reduce waste (say, too much documentation work)

f. Run Dev, QA and Ops streams in parallel as much as possible

Technology in Context: Deciphering the Tech Flow

tcs 50 Experience energy | TATA CONSULTANCY SERVICES

Lets' Take An Example …



Metrics and monitoring dashboards

ALM tool to track stories, defects and SDLC workflows

Further tests done in specific environments

Environments provisioned on-demand for code deploy/ release (using infra automation tools)

Tool plugin for analyzing code quality and metrics

Cloud or container based environments may be abstracted using PaaS

Dev uses app stack-specific IDEs for coding against test cases

CD tool pulls code from binary repo to deploy in target environment

Source code repo; changes committed by Dev here

CI server automatically pulls code from repo for automated build and tests

On successful build, CI server pushes executable to binary repo

How do You Leverage the CoE

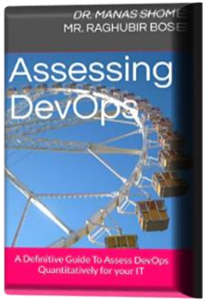
Five Ways …

- ✓ **Consulting** – Reach out to us if you need to do a DevOps assessment; we provide our framework [QODE], mentoring, and guidance during assessment execution; may need specific account investments
- ✓ **Niche customer presentations** – We do customer presentations for select customers on case basis; reach out to us if you want to leverage our PAQ as your GTM tool, get your presentation deck reviewed, or want us to present something to WOW the customer which may include working software demos
- ✓ **Proposal write-ups** – Reach out to us to get our content – we have a wide coverage with multiple flavors of DevOps content, both technical and process based; we can also do a review of your write-up on case basis
- ✓ **Mentoring** – We do specific engagement-based mentoring on case basis in our lab (seats are limited) which may include hands-on; we may need necessary investments by your account for our CoE to do this, say obtaining an instance on our AWS
- ✓ **Solution asset implementations** – Reach out to us to engage us and implement our DevOps technical solutions [Magnum] in your customer's IT to accelerate DevOps led delivery; we provide installation services, mentoring, and guidance during execution

What the CoE Has for You

Space and Literary

- ✓ We have our own lab in Kolkata, India (TCS Gitanjali Park, 4th Floor Block L)
- ✓ We have written a book "Assessing DevOps" that you can get on Amazon


DR. MANAS SHOM
MR. RAGHUBIR BOS
Assessing DevOps
A Definitive Guide To Assess DevOps Quantitatively for your IT

Framework and Solution Assets

- ✓ QODE – "Quantitatively Optimized DevOps Evaluation" is our assessment framework that you can leverage for consulting engagements; PAQ (Pre-Assessment Questionnaire) comes as part of it, which you can use to get a quick one-pager report on customer's DevOps readiness to start with

- ✓ Magnum– A set of technical solutions – working software sitting on our AWS – that can accelerate creating a given CI/CD infrastructure (configurable based on customer requirements, existing tools, and specific application stacks) in a matter of minutes to a few hours, rather than weeks or months
  - o Readily available for java (and java based stacks) and PEGA
  - o Configurations available for architectures incorporating multi-stack deploys (using XL Deploy), Puppet and Chef based release automation, and CI/CD for SalesForce
  - o Reference architecture available for .NET; reference tech flows available for Guidewire, IBM Z-series (mainframe) and SAP

QODE: Assess DevOps Quantitatively

tcs 50 Experience energy | TATA CONSULTANCY SERVICES

Data Capture on As-Is IT Landscape

Customer Objectives

Customer Pain Areas

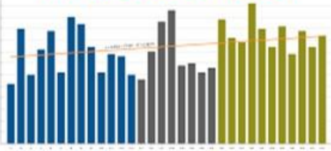
- Challenges for Dev
- Challenges for QA
- Challenges for Ops
  - Infra and release
  - Services support

IT Prerogatives

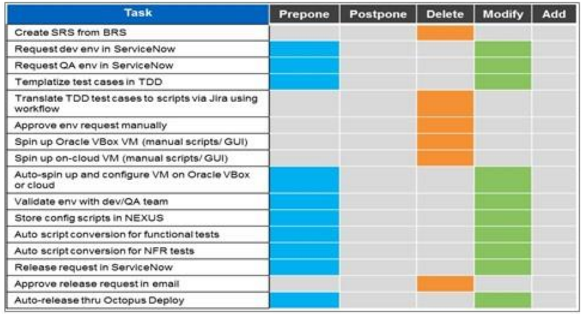
- Support business changes faster
- Standardize IT portfolio
- IT agility
- Reduce IT costs

Key Benefits Prediction

Metrics Driven

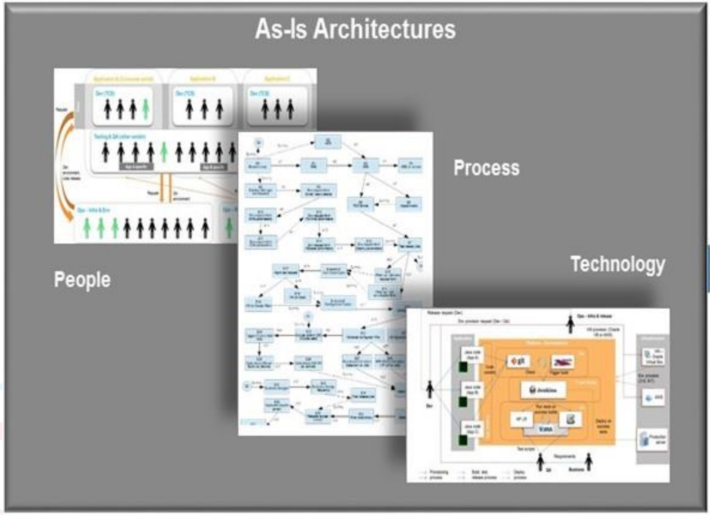


Detailed Implementation Stories with Timelines

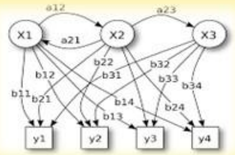


Task	Prepone	Postpone	Delete	Modify	Add
Create SRS from BRS			🟧		
Request dev env in ServiceNow	🟦			🟩	
Request QA env in ServiceNow	🟦			🟩	
Templatize test cases in TDD	🟦			🟩	
Translate TDD test cases to scripts via Jira using workflow			🟧		
Approve env request manually			🟧		
Spin up Oracle VBox VM (manual scripts/ GUI)			🟧		
Spin up on-cloud VM (manual scripts/ GUI)			🟧		
Auto-spin up and configure VM on Oracle VBox or cloud	🟦			🟩	
Validate env with dev/QA team	🟦			🟩	
Store config scripts in NEXUS	🟦			🟩	
Auto script conversion for functional tests	🟦			🟩	
Auto script conversion for NFR tests	🟦			🟩	
Release request in ServiceNow	🟦			🟩	
Approve release request in email			🟧		
Auto-release thru Octopus Deploy	🟦			🟩	

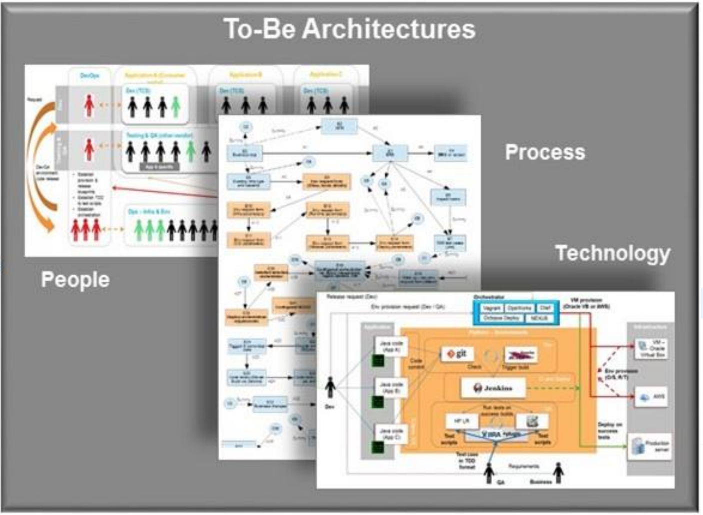
As-Is Architectures



Quantitative Algorithms Sandbox



To-Be Architectures



QODE Explained

Use PAQ to get to get started on a customer discussion; detailed QODE framework is for serious consulting engagements thereon

✓ **Data capture** – PAQ has 30 questions based on which you can this type of report
QODE has a detailed questionnaire that captures various aspects of a given IT process

✓ **Analysis** – Send us the filled-up PAQ for us to generate the one-pager readiness report for you
QODE analysis has the following steps:

- Identifies the critical path in the IT process that needs to be optimized
- Applies quantitative methods, taking care of people and technology constraints captured, to arrive at a target IT process flow
- Derives the corresponding people structure and technology flow (with landscape)
- Derives DevOps implementation stories with estimates on a timeline
- Derives set of key metrics (some of them with target values) to be established

✓ **Report** – We help you create an assessment report that can be directly taken to implementation; you may however include formal technology architectures to the report (with help of your account's or our architects) for completeness, based on the analysis outcomes

PDF Sample PAQ Report

X PAQ - 30 questions

PDF ToC - QODE Report

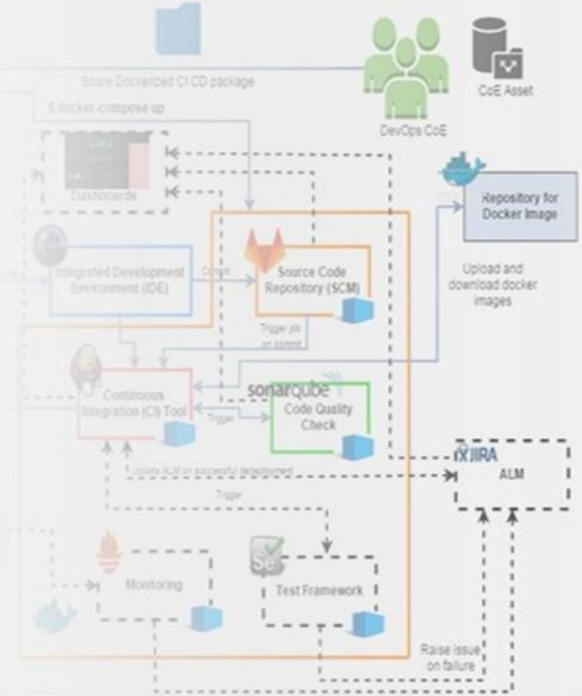
Magnum: Accelerate building CI/CD Infra On-Demand

tcs 50 Experience energy | TATA CONSULTANCY SERVICES

Technology Support: Java/J2EE* Microsoft .NET Guidewire PEGA* Mainframe | AS/400 Mobile and IoT

What is Magnum?

- Engineering offering of TCS' DevOps CoE
- Maintained code to integrate Dev and Ops to quickly build DevOps led infrastructure frameworks
- Extensible by design – across technology stacks, across DevOps engineering practices



Share Dockerized CI CD package
$ docker-compose up
DevOps CoE
CoE Asset
Dashboards
Repository for Docker Image
Upload and download docker images
Integrated Development Environment (IDE)
Commit
Source Code Repository (SCM)
Trigger job on commit
Continuous Integration (CI) Tool
sonarqube
Code Quality Check
Trigger
JIRA
ALM
Update ALM on successful deployment
Trigger
Monitoring
Test Framework
Raise issue on failure

30 min. **to generate a base development infrastructure**

80% reduction in operations cost

70% reduction in developer rework time

Zero downtime for deployments

In-built integration to ALM & monitoring tools

Reduced CoQ; embedded quality engineering

Platform Support: AWS* Azure GCP PCF OpenShift OpenStack Bare-metal VMWare

* Current implementation

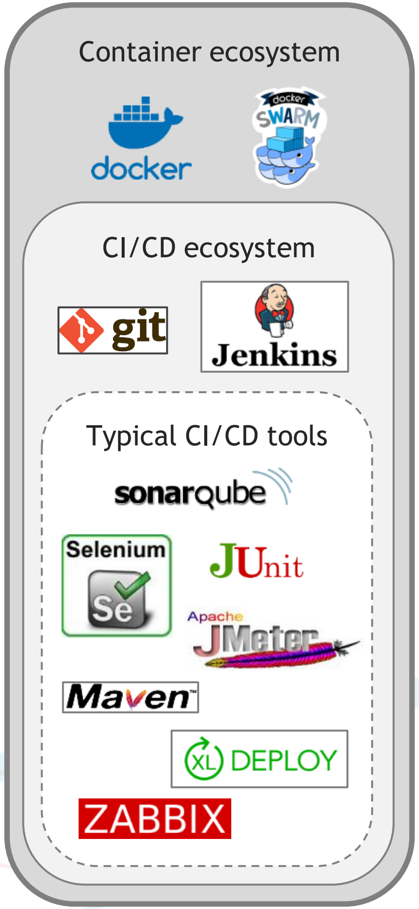
Magnum Explained

This is about Magnum for java stack (as an example)

- ✓ **Background** – Consider at least two servers (say, AWS machines); one to host Docker containers with CI/CD tools, and another with Docker Swarm (container cluster)
  - o The tools in containers in the first server act as a CI/CD tool chain
  - o The application itself gets deployed on a container in Docker Swarm using the CI/CD pipeline (configured in Jenkins installed in one of the containers)

- ✓ **What gets installed** – Magnum is a set of maintained code that does this:
  - o In the first server, Magnum installs Git, Docker and Docker-compose
  - o In the second server(s), the above + Docker Swarm is installed manually
  - o Another set of code in Magnum creates the necessary CI/CD tool chain (on containers) inside the first server; the pipeline is configured / coded out-of-box in Jenkins; the pipeline would have different build, test and deploy tools to execute CI/CD for applications

- ✓ **What the developer gets** – Ample time to code applications in his/her development machine; rest all is taken care by the CI/CD pipeline created by Magnum once developer commits the code in Git


Container ecosystem

CI/CD ecosystem

Typical CI/CD tools

How to Present DevOps to A Customer

Before you present

- Get the **PAQ** filled up and report generated; this will help you set the context of discussion
- Use a mix of **customer specific context** (including PAQ report) and generic slides to create your deck
- Preferably have a **working software demo** ready with you, if it fits the context
- DevOps presentations start being exploratory; have an **end objective** in mind on next steps – either getting into an assessment, or getting into an implementation (technical and/or people mentoring)

While presenting

- First probe the customer, to understand expectations and pain areas; DevOps is about a set of practices and culture, so DO NOT talk outright on tools
- There are now three routes:
  - **Customer wants to know TCS' capabilities** – cover PoV, TCS' assets, case studies, challenges
  - **Customer wants to get an assessment done** (possibly they got one done earlier by some vendor?) – Start with PAQ report; talk about QODE and how it leads to downstream implementations
  - **Customer wants to implement DevOps tools** – ON YOUR GUARDS! ASK – Have they done an assessment? Is there an IT roadmap in place? Do they know what metrics to check for? Have they piloted a few tools in pockets – if yes, did they evaluate if they can scale to enterprise? Propose Magnum if it fits

Cracking the 3.5 Hours DevOps Workshop in 3 Steps

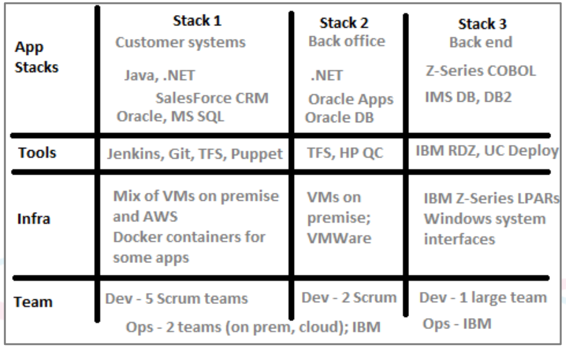
Prepare

- ➢ Go through the book "Assessing DevOps"
- ➢ Have the PAQ report handy
- ➢ Get white board, markers, art papers, post-its
- ➢ Key notes on prior customer discussions

Why 3.5 hours?
We see that most DevOps get-started workshops are for 4 hours, with 30 min. reserved for breaks and lunch

1 hr — Probe the Customer

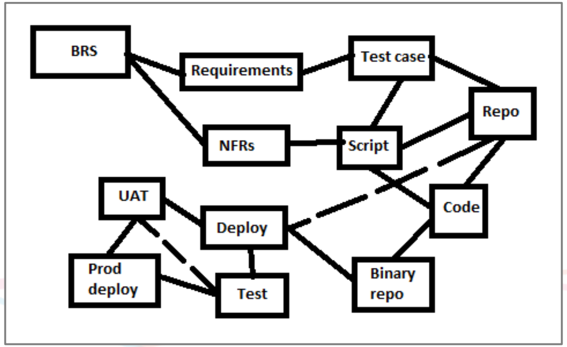
- o How does customer's IT look like – app stacks, tools, infra, teams
- o Draw this



	Stack 1	Stack 2	Stack 3
App Stacks	Customer systems Java, .NET SalesForce CRM Oracle, MS SQL	Back office .NET Oracle Apps Oracle DB	Back end Z-Series COBOL IMS DB, DB2
Tools	Jenkins, Git, TFS, Puppet	TFS, HP QC	IBM RDZ, UC Deploy
Infra	Mix of VMs on premise and AWS Docker containers for some apps	VMs on premise; VMWare	IBM Z-Series LPARs Windows system interfaces
Team	Dev - 5 Scrum teams Ops - 2 teams (on prem, cloud); IBM	Dev - 2 Scrum	Dev - 1 large team Ops - IBM

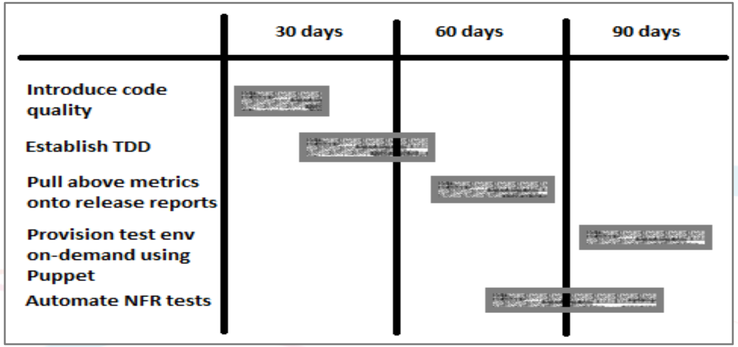
1.5 hrs — Probe the Customer Again

- o Pick up one stack and draw the full IT process network
- o Identify gaps with total cycle time



1 hr — Chart out the Plan

- o Prioritize gaps and create a plan with timelines (30-60-90 days), mapping with other similar stacks and PAQ report observations
- o Articulate key benefits incl. cycle time reduction



What to Write in a DevOps Proposal

Is it a capability ask, or specific ask for a solution?

- If it is an **existing customer** (possibly with TCS in either Dev or Ops, or both), try to get a **PAQ report** done – this will help identify potential gaps
  - Also try to obtain any **IT strategy/ roadmap document** if they already have; this will help aligning the response to what the customer expects
- RFIs and PQQs typically ask for capabilities, possibly through specific questions; leverage content from multiple sources such as Knowmax, and materials from CoE
- RFPs typically ask for solutions to do DevOps for specific stacks, or for enterprise; note the following:
  - Since DevOps is culture driven, it is safer to **propose a formal assessment** for the stacks in scope (or enterprise, if applicable) prior to any implementation
  - If you are providing any **technology solution**, provide a disclaimer that the validity would be further determined during such assessment
  - Response, in addition to any technology solution (if provided), should also **include people and process** transformation plans/ views – for example, training required for DevOps, and shift left in development practices to incorporate TDD/BDD
  - **Estimates** provided should ideally be in Agile (plan to have Sprints defined)
  - For DevOps **support engagements** (not actually DevOps, but automation support), the typical ITSM structure may be followed
  - It is good to propose a **DevOps CoE/factory model** to drive DevOps in customer organization where customer FTEs, TCS (and other vendors) would participate

Learning Resources

TCS Sources: Content	Type	Source
DevOps – Beginner – ILT	Course (49641)	iEvolve
Digital : DevOps Fundamentals_Awareness_iON LX_SP_Assessment	Course (51685)	iEvolve
DevOps Foundation Certificate	Course (49991)	iEvolve
https://play.fresco.me/category/499	Course	Fresco Play
https://www.safaribooksonline.com/library/view/the-phoenix-project/9781457191350/	Book	Safari Online
https://www.safaribooksonline.com/library/view/the-devops-handbook/9781457191381/	Book	Safari Online

Getting Started	http://agilemanifesto.org/ https://en.wikipedia.org/wiki/DevOps http://www.jedi.be/blog/2010/02/12/what-is-this-devops-thing-anyway/ ..... And more of Patrick's blogs here https://www.atlassian.com/devops, https://aws.amazon.com/devops/what-is-devops/, https://xebialabs.com/periodic-table-of-devops-tools/
Technology: Build, test, release and infrastructure as code	https://programmaticponderings.com/2013/11/04/continuous-integration-and-deployment-using-git-maven-jenkins-and-glassfish/ https://programmaticponderings.com/2014/12/14/installing-puppet-master-and-agents-on-multiple-vm-using-vagrant-and-virtualbox/ https://www.thoughtworks.com/insights/blog/infrastructure-code-reason-smile https://www.blazemeter.com/blog/how-automate-jmeter-and-selenium-testing https://www.blazemeter.com/blog/how-automate-testing-using-selenium-webdriver-jenkins-and-allure
Technology: Requirements to code engineering	https://www.tracecloud.com/GloreeJava2/jsp/WebSite/TCJiraIntegration.jsp https://dzone.com/articles/micro-services-simple-example https://dzone.com/articles/java-microservices-code-examples-tutorials-and-more
More aspects	https://devops.com/metrics-devops/ http://web.devopstopologies.com/ http://blogs.gartner.com/cameron_haight/2014/10/13/getting-your-arms-around-devops-devops-patterns-and-practices/ https://devops.com/self-healing-service-management-future-devops/ https://devops.com/continuous-monitoring-role-devops-and-apm/ https://insights.sei.cmu.edu/devops/2015/12/monitoring-in-the-devops-pipeline.html

tcs 50
Experience energy

TATA
CONSULTANCY
SERVICES

Thank You