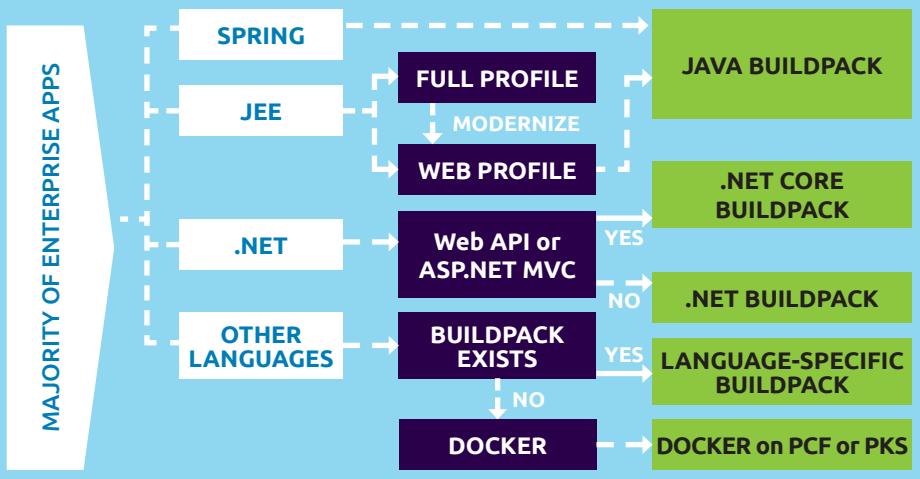
Capgemini

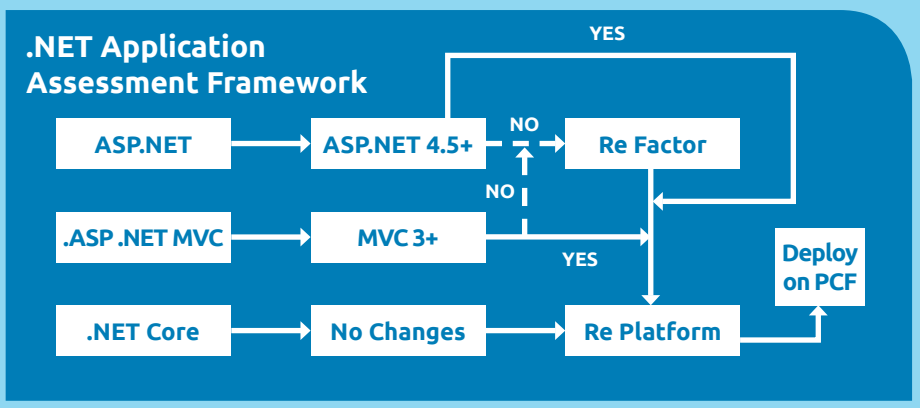
Challenges

- Lack of Cloud Provider Agnostic Deployment model for .NET applications
- .NET applications tightly coupled and limited with Windows offered resources
- .NET applications traditionally monolithic in nature

Solution

CAPGEMINI .NET TO PCF MIGRATION FRAMEWORK





.NET MICRO-SERVICES AND STEELTOE

Cloud Services

Service Registry

Config Server

Circuit Breaker

steeltoe by Pivotal

Parameters to be considered for various approaches

RE-PLATFORM:

.NET Core Build pack

- Modify MAIN method to use HTTP port
- Configure other required configuration files in copyToOutput
- Specify minor version of .NET framework for app in .csproj or project.json
- Use .deployment file for multi-projects
- Add external shared libraries to the library path of the app root
- Create manifest.yml file for deployment

.NET Build pack

- Recompile app using Mono
- Check for multiple copies of .exe.config
- For Console apps, configure with no route for http requests
- Create manifest.yml file for deployment

RE-FACTOR:

- Interfaces – modify to use only HTTP
- Global Assembly Cache – remove dependency to it
- Older version of .NET – upgrade to those supported by PCF
- Self-hosted apps – convert to IIS hosted
- Reading/writing to the registry – use local or centralized configuration
- Windows authentication – use OAUTH2 with SSO
- Job controller invoked – convert to PCF one-off tasks
- Native DLL usage – replace with non-native
- Local disk use – write to S3 or other PCF compatible file system
- Dependencies – deploy in bin with application
- Logging – change to console logging

MODERNIZE:

- Adopt an API as Product mindset and organize code around it by decomposing into smaller microservices
- Leverage Pivotal's Steeltoe framework to enable rapid adoption of cloud-native reference architectures
- Service discovery – use Netflix Eureka service registry via .NET client
- Centralized configuration - use Spring Cloud Config Server to increase environment portability
- Stateless services – Use Redis to cache data in a distributed architecture
- Backing resources – Replace MS-centric back services such as MSMQ and SQLServer with RabbitMQ and MySQL with easy binding

CAPGEMINI .NET TO PCF MIGRATION FRAMEWORK

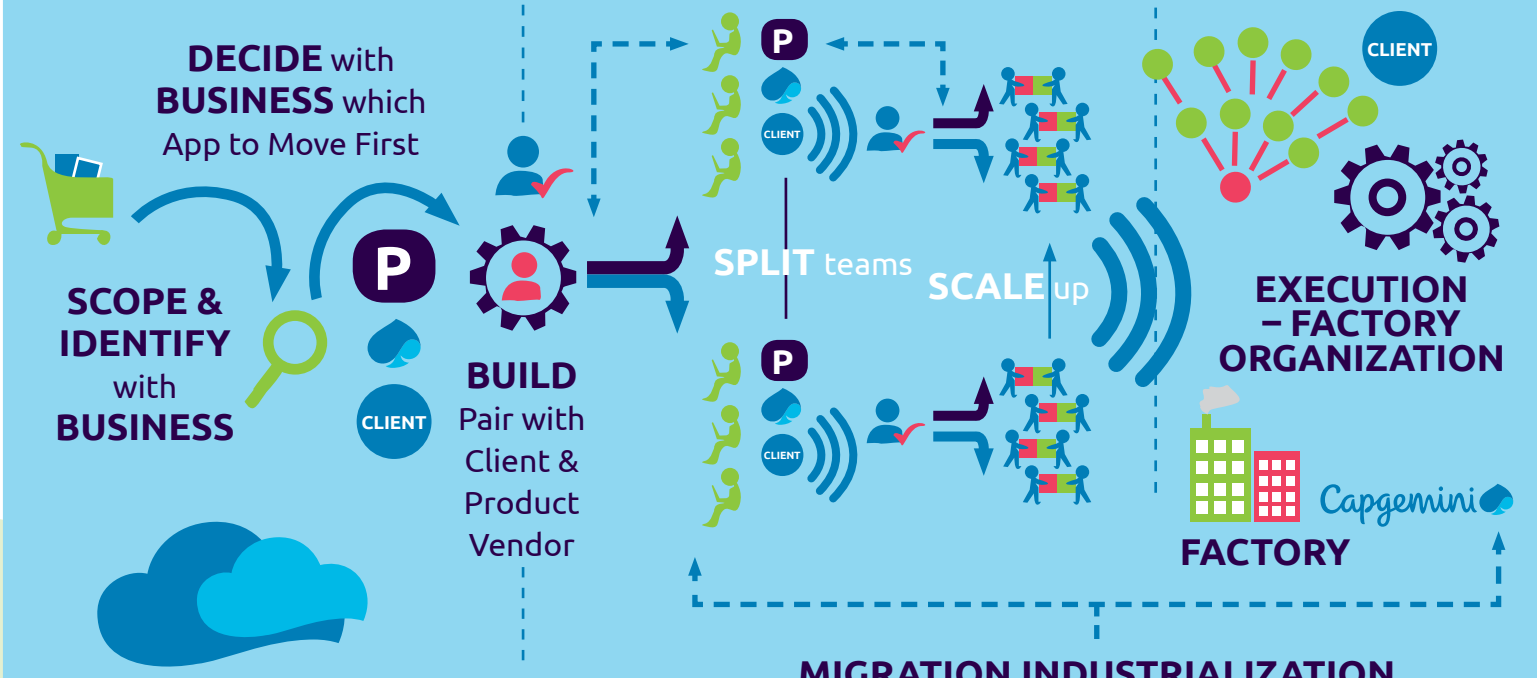
3 Phased Unified Approach (PaaS COE)

MODEL

1
Unified Cap & Client & PaaS Vendor Delivery Path

2
Paired Pilot App Re-platform & Modernization

3
Split & Scale & Factory



Benefits

- Instant self-service provisioning
- Concept to code in minutes
- Continuous integration and delivery
- Automation and failure response
- Self healing and recovery of apps & platform
- Platform is independently upgradable from Apps (No downtime)
- No undifferentiated heavy lifting (upgrade and patch)
- No ad hoc automation
- Push button control of entire application stack
- Manage services, not servers

Contact us:

Charlie Li
EVP, NA Chief Cloud Officer
Charlie.Li@capgemini.com

Subrata Pal
VP, NA Cloud Foundations Leader
Subrata.Pal@capgemini.com

Pankaj Sehgal
Principal, NA Cloud Native & PaaS Leader
Pankaj.Sehgal@capgemini.com

Kaushik De
Principal, NA Go To Market Leader
Kaushik.De@capgemini.com

Get ahead.
Stay ahead.
Shift to a cloud-first core.