Getting up to speed: Application Deployment Patterns in the Cloud

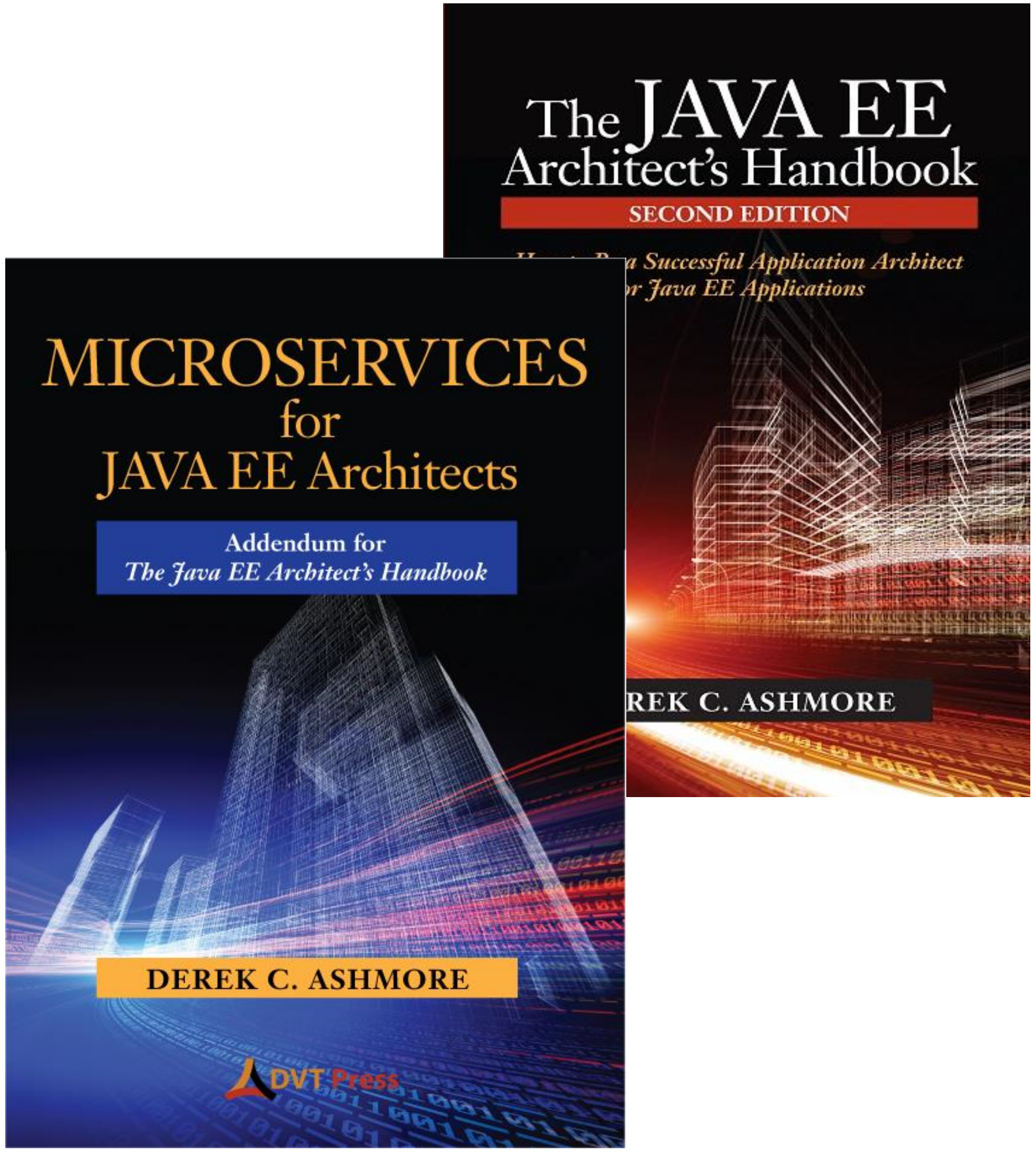
Given by Derek C. Ashmore
Code PaLOUsa 2022
August 19, 2022

Asperitas
CONSULTING

Who am I?

- Professional Geek since 1987
- Java/J2EE/Java EE since 1999
- AWS since 2010
- Azure since 2017
- Specialties
  - Application Transformation
  - Infrastructure Automation
- Yes – I still code!

Asperitas CONSULTING


The JAVA EE Architect's Handbook
SECOND EDITION
a Successful Application Architect
r Java EE Applications
REK C. ASHMORE
MICROSERVICES for JAVA EE Architects
Addendum for The Java EE Architect's Handbook
DEREK C. ASHMORE
DVT Press

Discussion Resources

- This slide deck
  - https://www.slideshare.net/derekashmore/presentations
- Deployment Pattern Article Series
  - http://www.derekashmore.com/2020/05/design-patterns-for-cloud-management.html
- Slide deck has hyper-links!
  - Don’t bother writing down URLs

Agenda

- General Overview
- Application Deployment Patterns
- Application Infrastructure
- Summary / Q&A

Tales from the Field

- National Consumer Product Firm
  - Rebuilds the entire cloud footprint with all applications every two weeks
- National Fast-food Franchise
  - Mobile App Cloud and application footprint
  - Rolling Update pattern
- National Financial Institution
  - Mobile App Cloud and application footprint
    - blue/green capability

A Brief History of Time

- I’ve been writing CI/CD Pipelines for over a decade.
- Before the Cloud
  - Application builds and packaging
    - Automating application build/packaging became standard
    - Deployments sometimes automated / sometimes not
  - OPS handled deployments – often manually
  - Releases required bureaucracy and constrained by schedules
- After the cloud
  - OPS decentralized – DevOps teams formed
  - Everything is (or can be) code now
  - Application developers increasingly responsible for application infrastructure too
    - Instances / Virtual Machines
    - Content Delivery Network (CDN) deployments
    - Serverless function development/deployment
    - Database design/management too
  - CI/CD Pipelines augmented to handle application infrastructure too

Deja Vu

- Cloud deployment pipelines look the same after a while
  - Like build automation before it
- Coding patterns have developed for application infrastructure and releases
  - Just like application code for any programming language

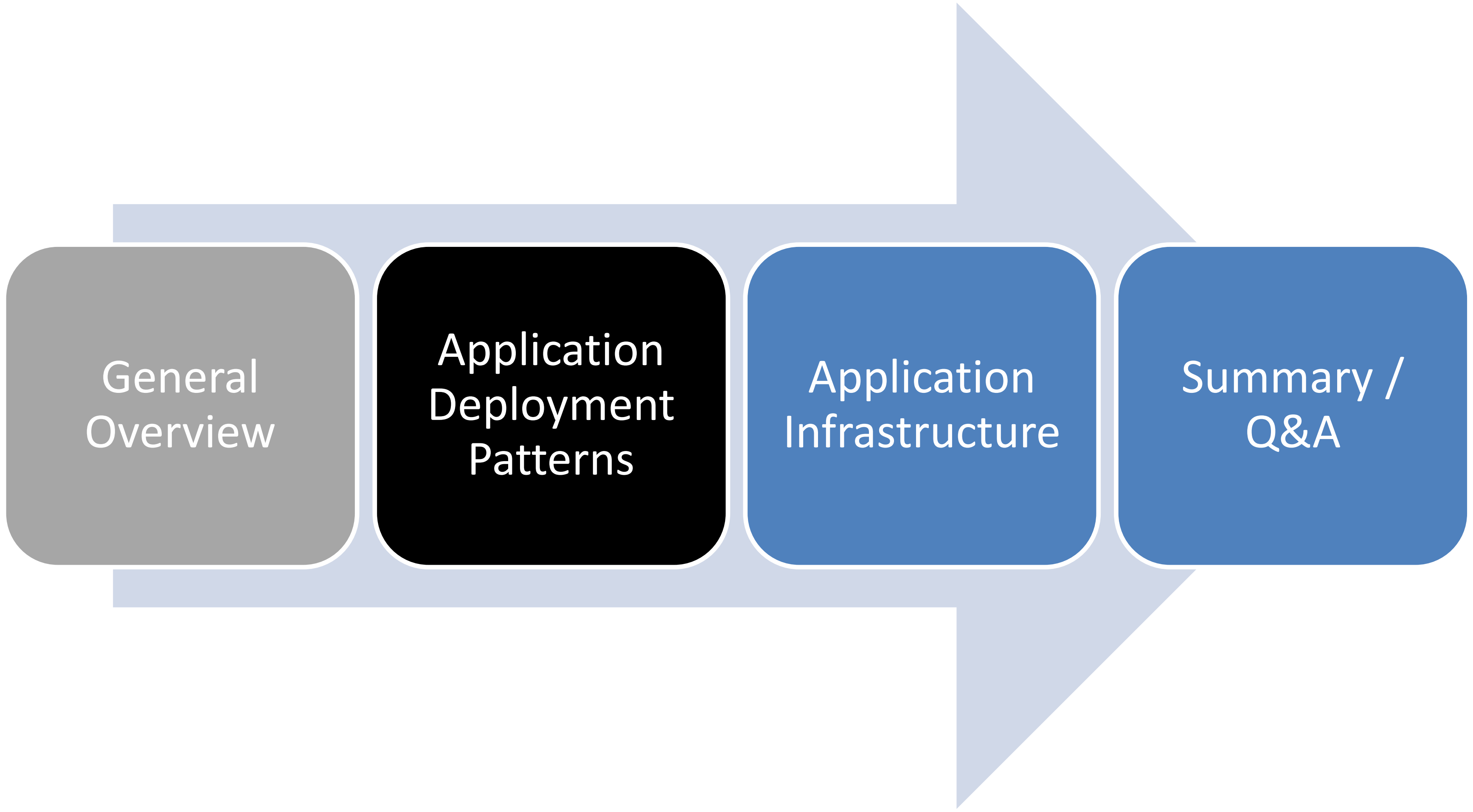
Cloud Pattern Categories

- Build Patterns
  - Creating deployable artifacts from source code
    - Docker Images
    - Java → Wars/Ears
    - Python packages
- Application Release Patterns
  - Making deployable artifacts available to end users
- Infrastructure Patterns
  - Creating/maintaining infrastructure used/needed by applications

What is a Pattern?

- Elements of a Software Pattern
  - Problem Statement
    - Sometimes called “Intent”
  - Example(s) of the problem
  - Applicability Statement
    - When to use this pattern
  - Structure
    - Algorithm used by the pattern
  - Consequences
    - Advantages / Disadvantages
    - Limitations

Agenda


General Overview
Application Deployment Patterns
Application Infrastructure
Summary / Q&A

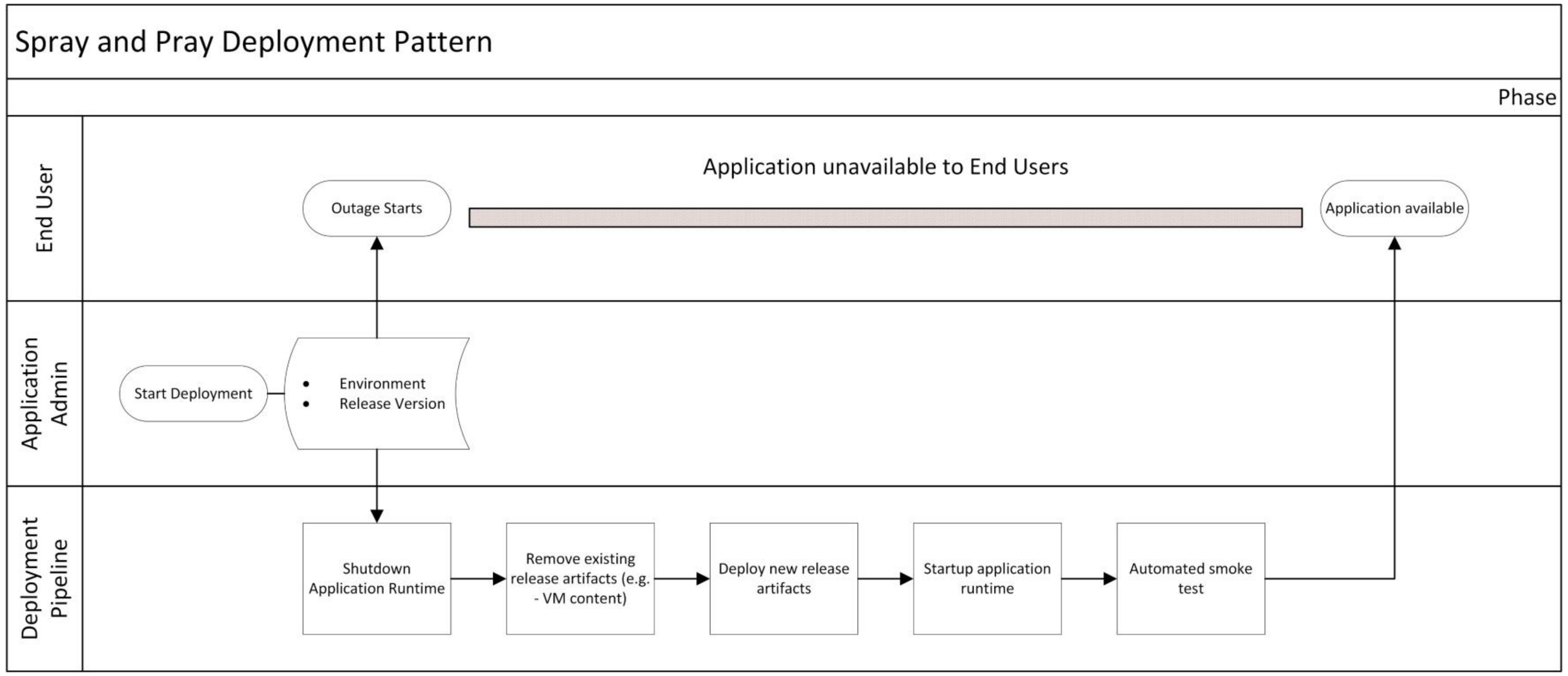
Application Deployment Pipeline Patterns

- Application Deployment Pipeline Patterns
  - Make build pipeline output available to end users
    - Docker images, Java Wars/Ears, Python Packages, etc.
  - Many require cloud (any vendor)
  - Can be implemented using any CI/CD deployment software
    - Jenkins, Bamboo, Azure DevOps, AWS Code* products
  - Just like GoF patterns that can be implemented in any programming language
- The Pattern List
  - Spray and Pray (All at Once)
  - Rolling Updates
  - Blue/Green
  - Canary

Spray and Pray (All at Once)

- ***Problem Statement***
  - An application release needs to be made available to users

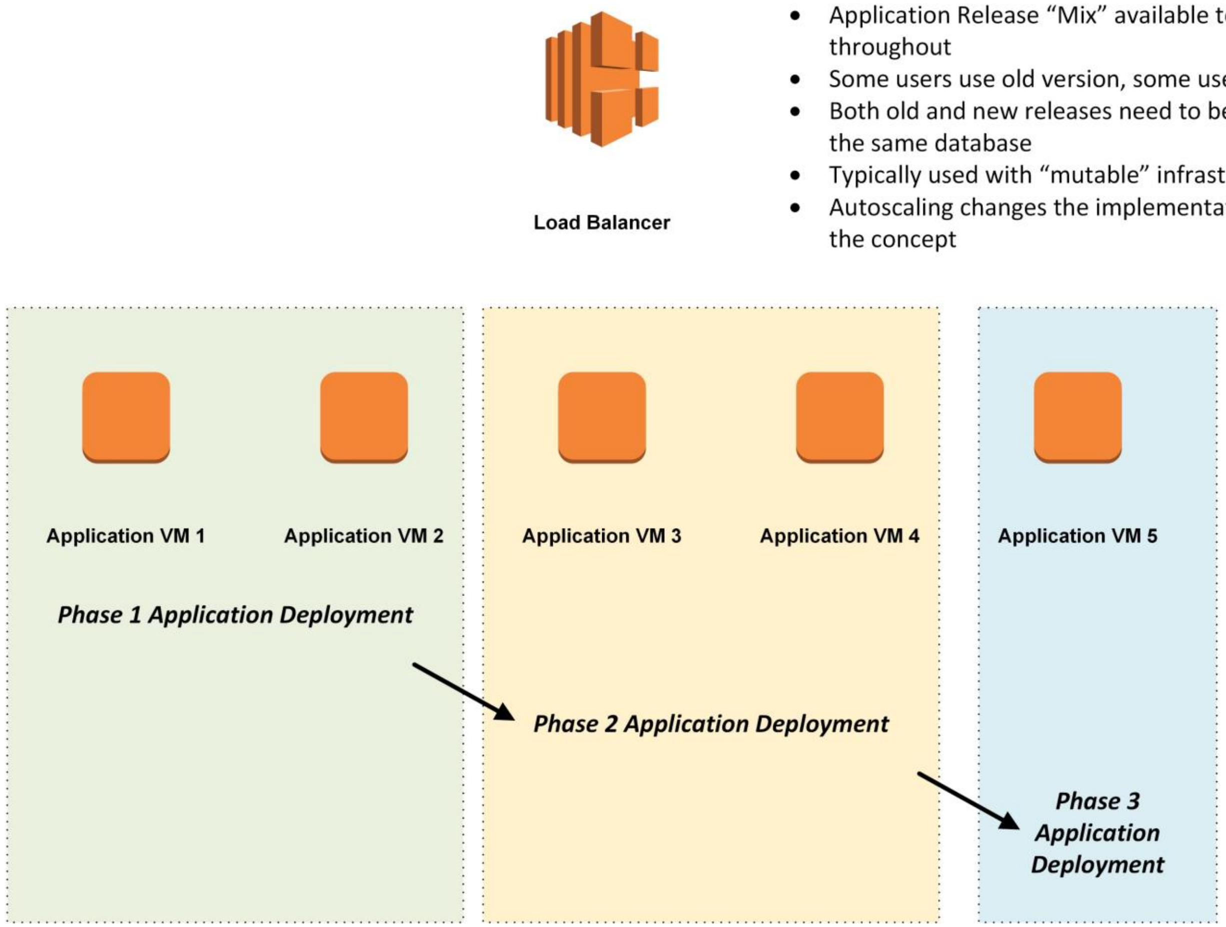
- ***An Example***
  - An internal web application used for company profitability analysis



Spray and Pray Deployment Pattern
Phase
End User
Application Admin
Deployment Pipeline
Application unavailable to End Users
Outage Starts
Application available
Start Deployment
• Environment
• Release Version
Shutdown Application Runtime
Remove existing release artifacts (e.g. - VM content)
Deploy new release artifacts
Startup application runtime
Automated smoke test

Rolling Updates

- ***Problem Statement***
  - An application release needs to be made available to users without downtime to users
- ***An Example***
  - multiple (highly available) node deployment -- any technical stack
  - Mobile application for Fast Food franchiser
    - AWS Elastic Beanstalk

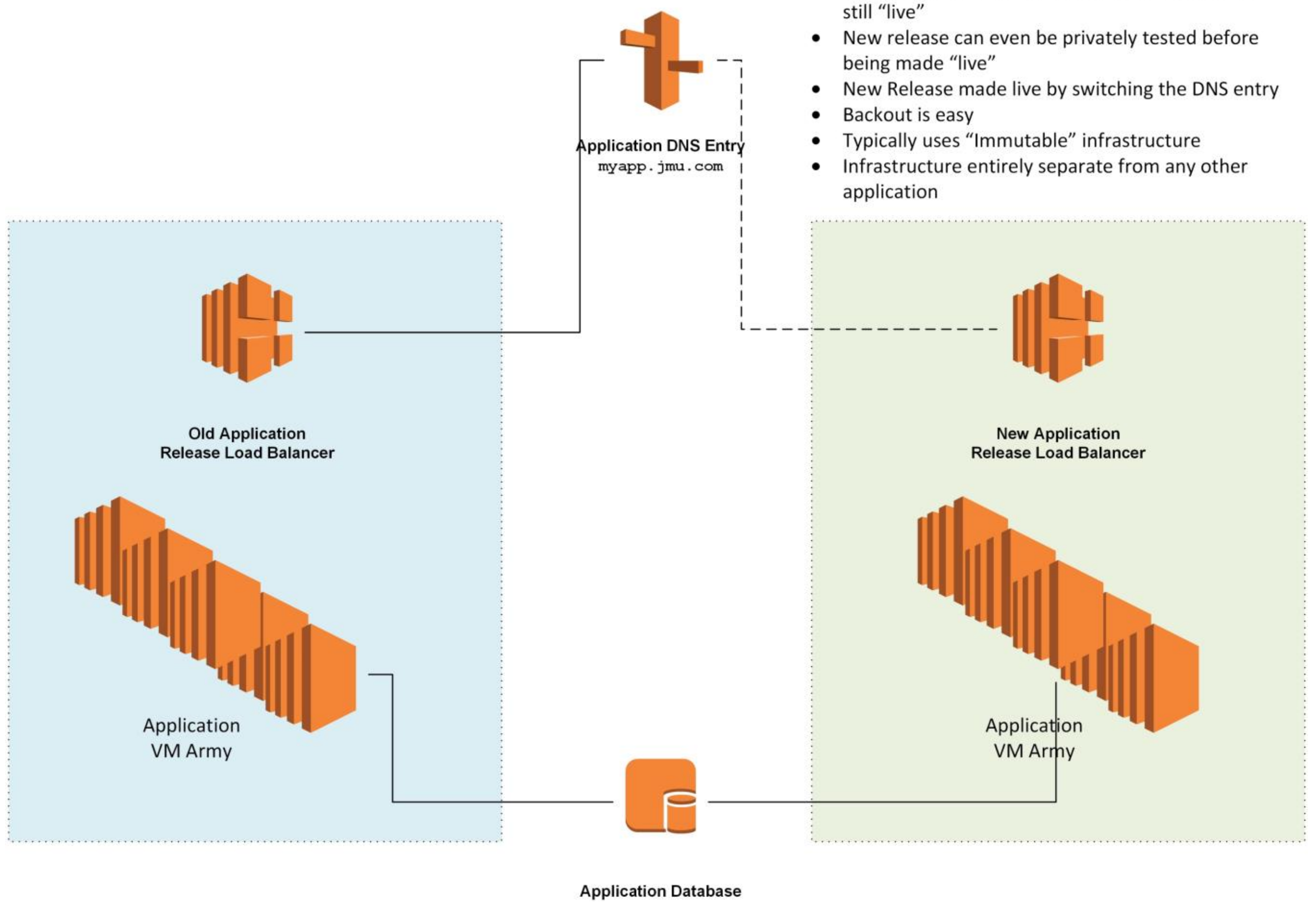


- Application Release "Mix" available to users throughout
- Some users use old version, some use new version
- Both old and new releases need to be able to use the same database
- Typically used with "mutable" infrastructures
- Autoscaling changes the implementation, but not the concept

Blue / Green

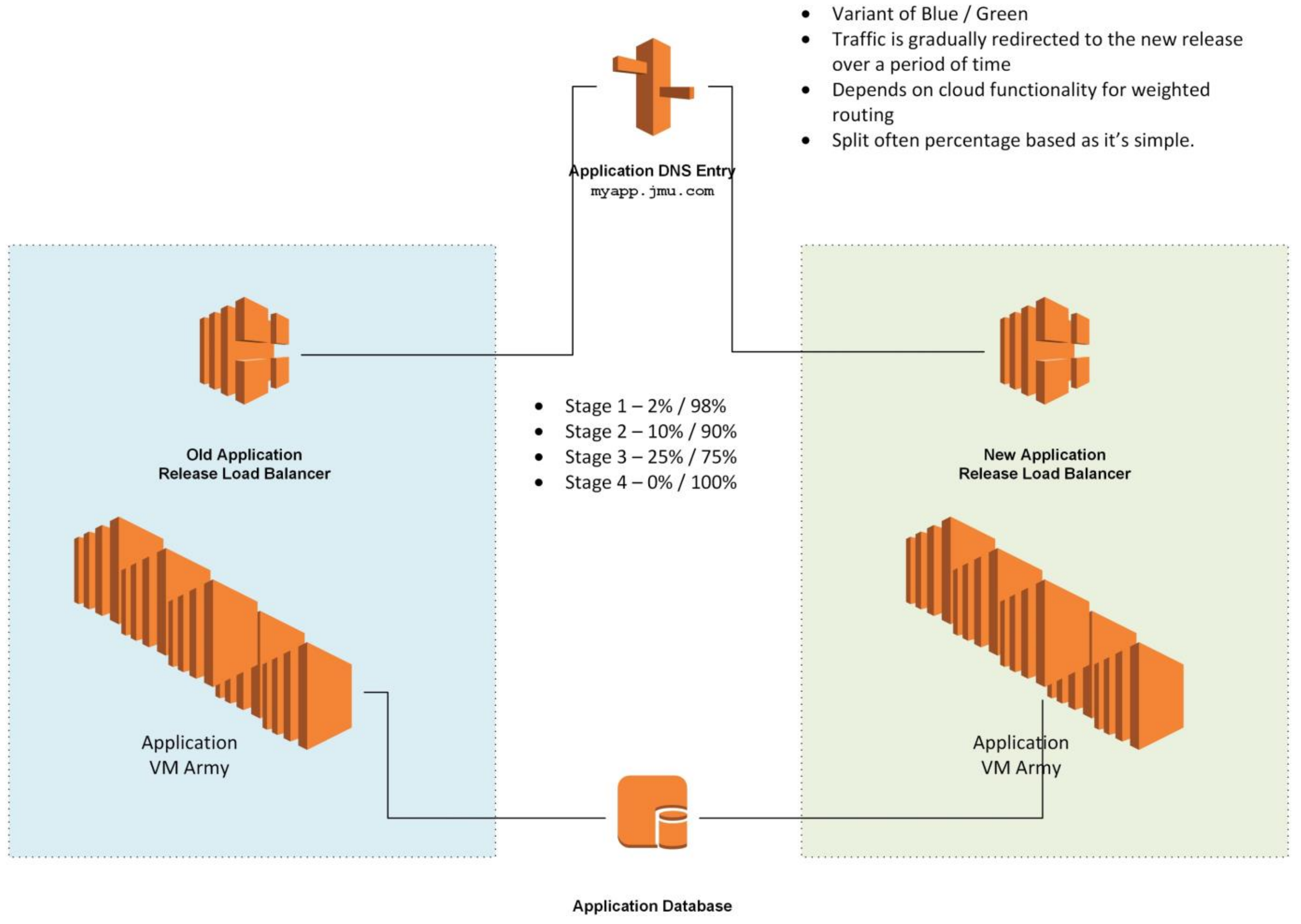
- ***Problem Statement***
  - An application release needs to be made available to users without downtime to users and minimizing rollback time
- ***An Example***
  - Mobile consumer banking application suite
    - Large bank with US presence
    - High Availability a must
    - Customer outages must be eliminated at all costs
    - Ability to quickly rollback at the first sign of trouble

- New Release set up separately while old release still "live"
- New release can even be privately tested before being made "live"
- New Release made live by switching the DNS entry
- Backout is easy
- Typically uses "Immutable" infrastructure
- Infrastructure entirely separate from any other application



Canary

- Problem Statement
  - An application release is "tested" in production by releasing it at first to a smaller percentage of users
- An Example
  - Mobile consumer banking application suite



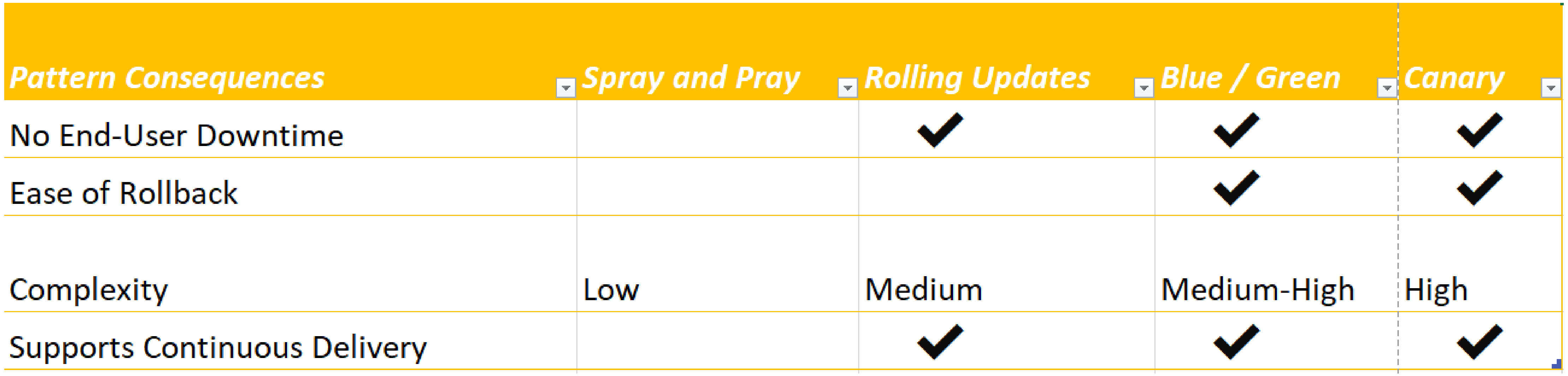
- Variant of Blue / Green
- Traffic is gradually redirected to the new release over a period of time
- Depends on cloud functionality for weighted routing
- Split often percentage based as it's simple.

- Stage 1 – 2% / 98%
- Stage 2 – 10% / 90%
- Stage 3 – 25% / 75%
- Stage 4 – 0% / 100%

Pattern Requirements

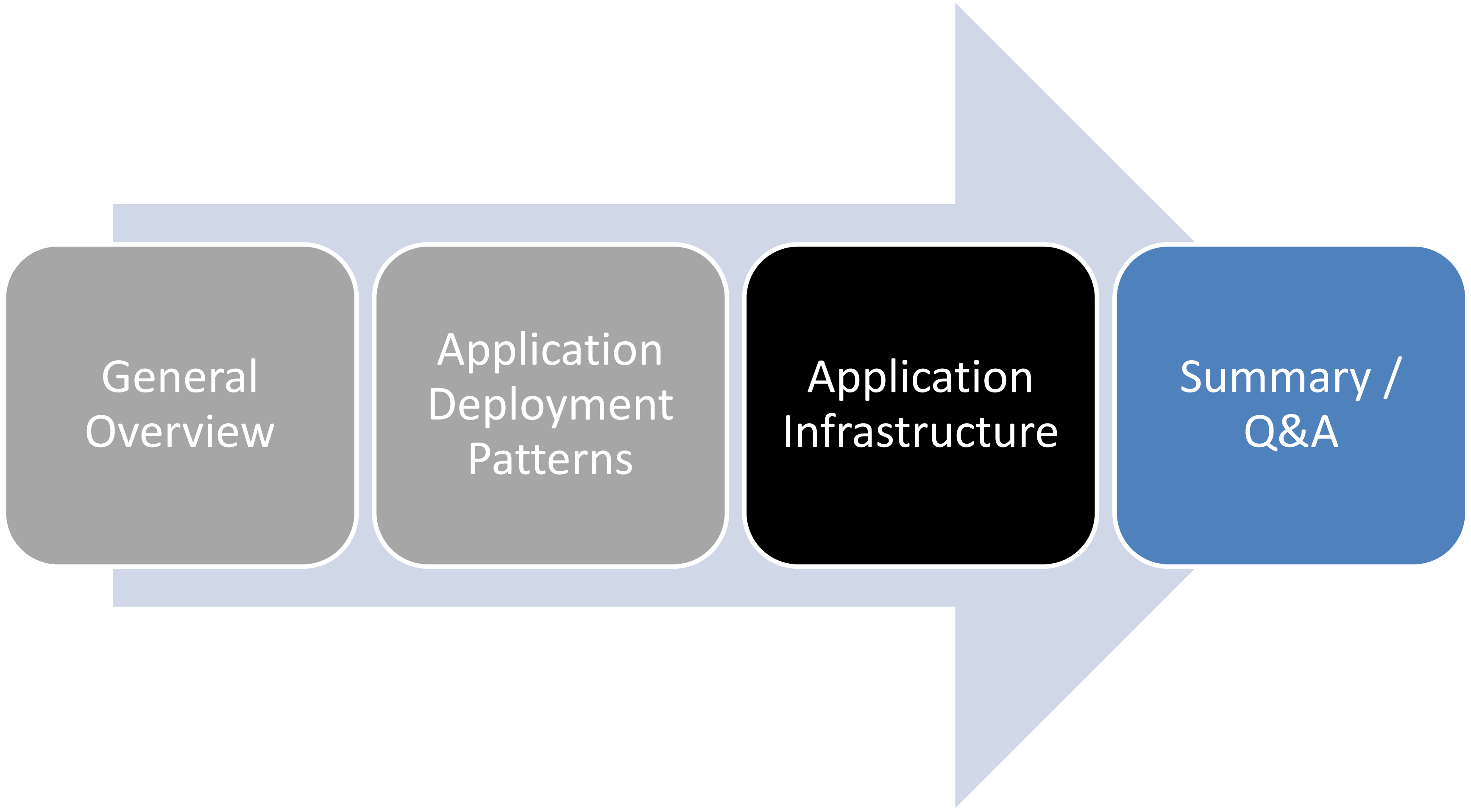
Requirements	*Spray and Pray*	*Rolling Updates*	*Blue / Green*	*Canary*
End-User Outage	✓			
Cloud Required			✓	✓
Infrastructure as Code Required			✓	✓
Immutable Infrastructure Required			✓	✓
Release Agnostic Database Required		✓	✓	✓

Pattern Consequences



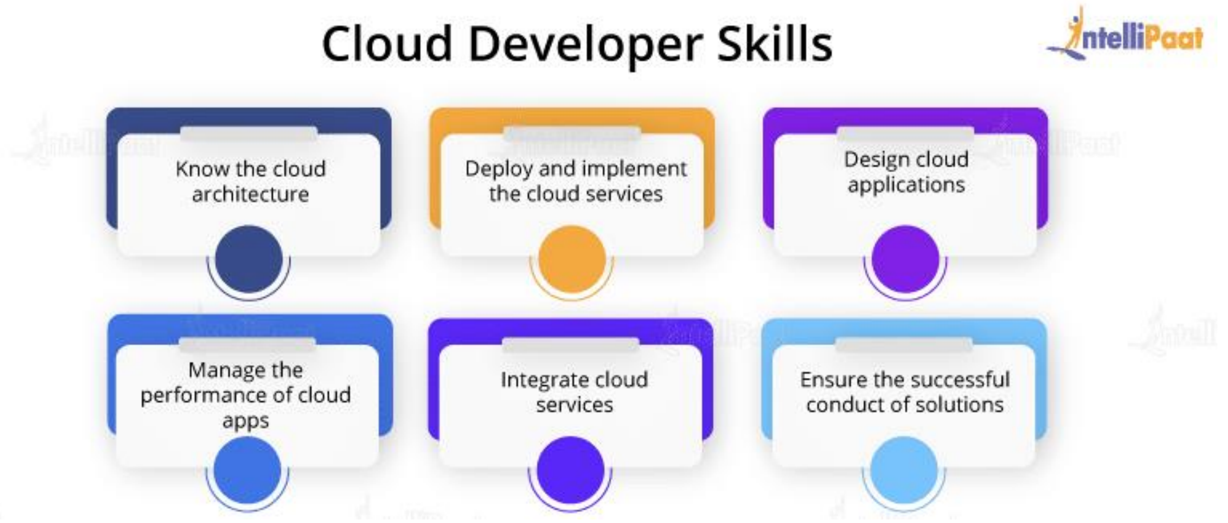
Pattern Consequences	Spray and Pray	Rolling Updates	Blue / Green	Canary
No End-User Downtime		✓	✓	✓
Ease of Rollback			✓	✓
Complexity	Low	Medium	Medium-High	High
Supports Continuous Delivery		✓	✓	✓

Agenda


General Overview
Application Deployment Patterns
Application Infrastructure
Summary / Q&A

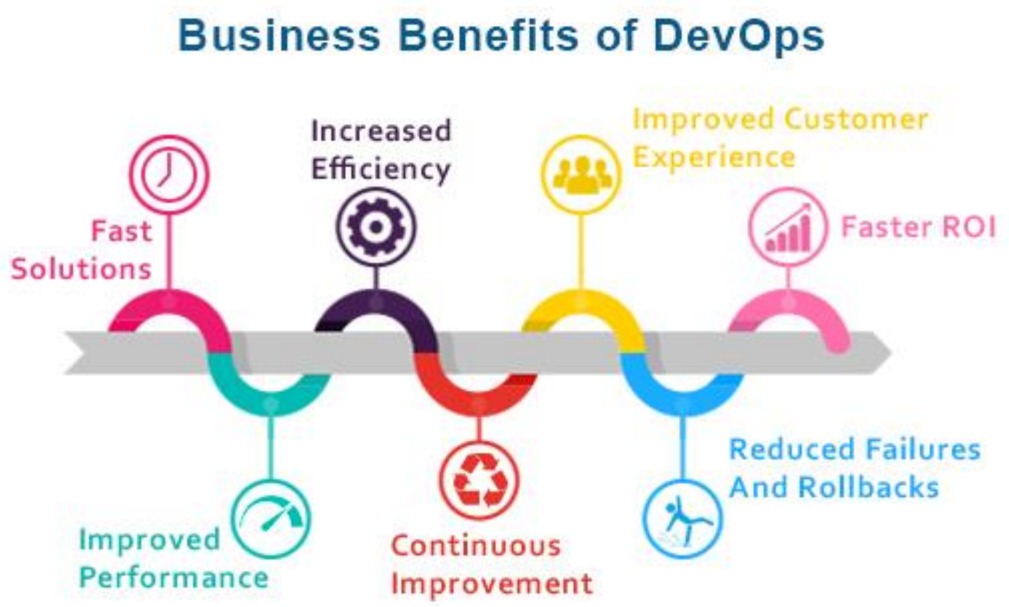
Added Skills for App Developers

- Cloud Responsibilities are Common Now
  - Application Infrastructure in the cloud
  - Infrastructure coding
- App Infrastructure Pipelines
  - Different Coding Languages
    - Terraform / ARM / CloudFormation


Cloud Developer Skills
IntelliPaat
Know the cloud architecture
Deploy and implement the cloud services
Design cloud applications
Manage the performance of cloud apps
Integrate cloud services
Ensure the successful conduct of solutions

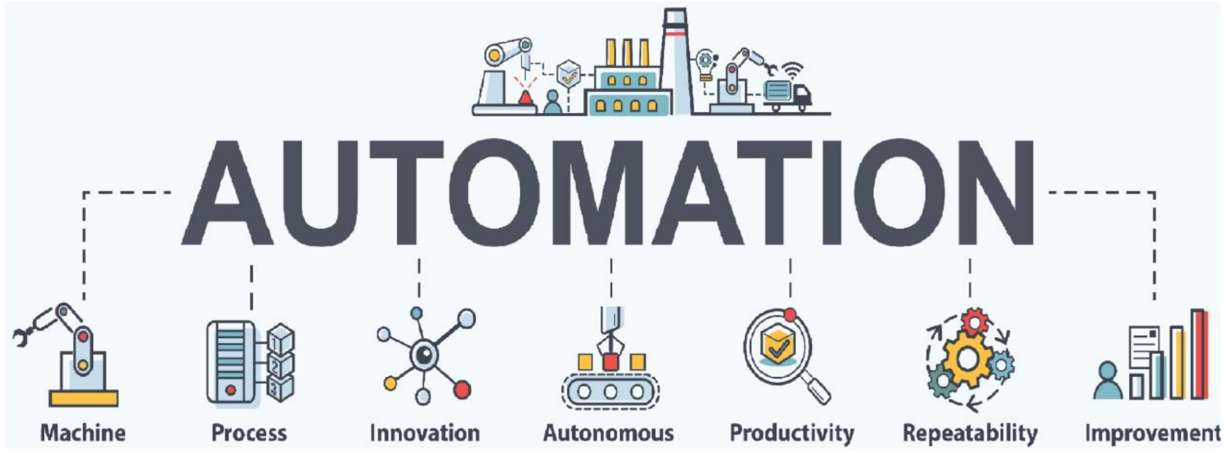
DevOps Automation Defined

- Manage entirely through code
  - No more manual changes
  - 100% Infrastructure as Code
- Automation Types
  - Cloud Infrastructure Code
  - Application Infrastructure
    - CI/CD Pipelines
  - Image factories
    - VMs and Docker
  - Security Enforcement



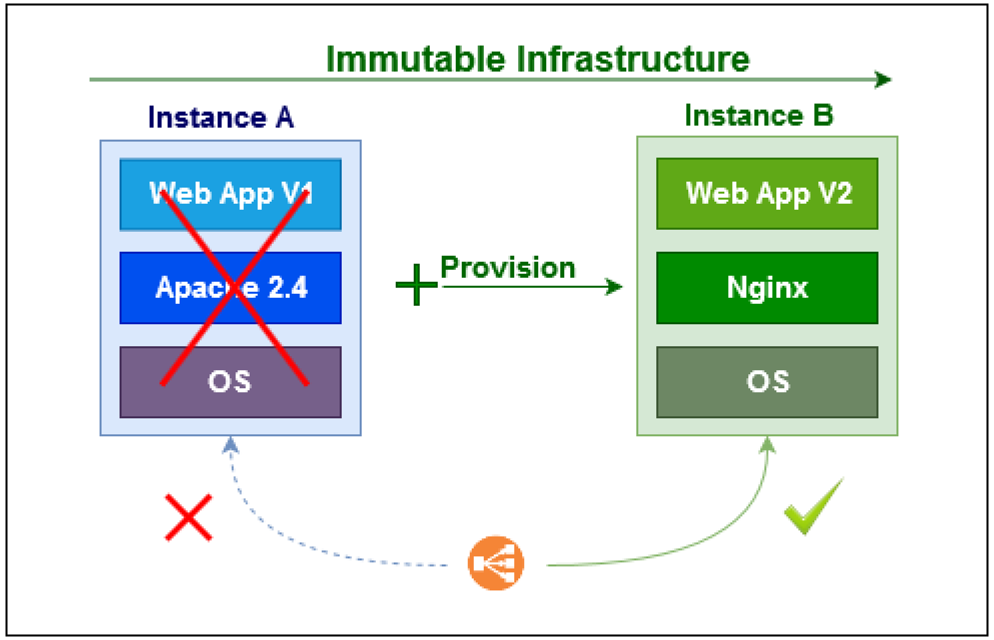
Automate Once – Deploy Many Times

- Same Automation for all environments
  - New environments require "pushing a button" and providing a few inputs
- Supports cloud computing benefits
  - Provides Environment Consistency
  - New environments work because they were tested.
  - Speed to Market – New environments are quick to set up
  - All environments are consistent
  - All environments have the same security posture
  - Safety for Change
    - All changes can follow SDLC lifecycle
  - Cost Effectiveness
    - Environments easily destroyed when no longer needed



Head toward Immutable Infrastructure

- Most Legacy Apps use Mutable Infrastructure
  - Servers/VMs exist – apps deployed to them
  - Content Update
- Go Immutable!
  - App Infrastructure created with each deployment
  - Allows Blue/Green and Canary Patterns
  - Treats your infrastructure like your app code



More Tales from the Field

- Financial Data Provider
  - Team of 4 administrates global network infrastructure
    - 6 regions
    - 3 environments per region
- Financial Institution
  - Team of 3 maintains CI/CD pipelines across the enterprise
    - 12 applications and growing
    - Blue/green deployments

Current Application Trends

- Greenfield trends toward serverless
  - AWS Lambda
  - Azure App Services/Functions
- Old model with VMs heavier lift for App Devs

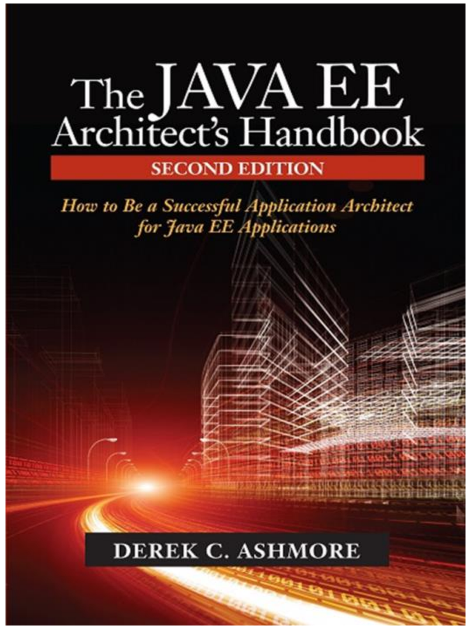
Further Reading

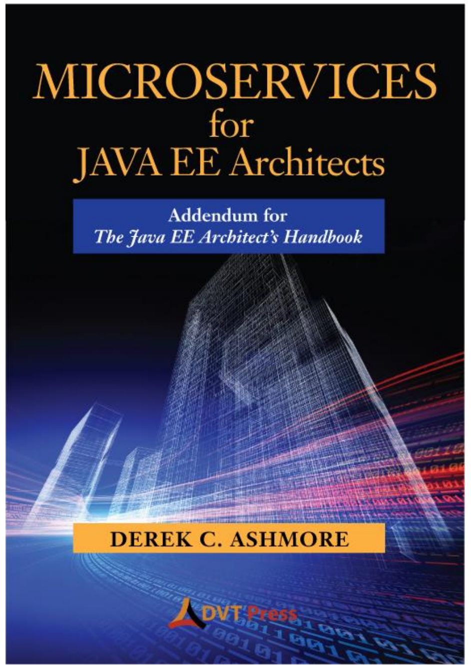
- This slide deck
  - https://www.slideshare.net/derekashmore/presentations
- Deployment Pattern Article Series
  - http://www.derekashmore.com/2020/05/design-patterns-for-cloud-management.html
- Written Material for Deployment Patterns
  - https://learningactors.com/intro-to-deployment-strategies-blue-green-canary-and-more/

Questions?

- Derek Ashmore:
  - Blog: www.derekashmore.com
  - LinkedIn: www.linkedin.com/in/derekashmore
    - Connect Invites from attendees welcome
  - Twitter: https://twitter.com/Derek_Ashmore
  - GitHub: https://github.com/Derek-Ashmore
  - Book: http://dvtpress.com/

Asperitas CONSULTING


The JAVA EE Architect's Handbook
SECOND EDITION
How to Be a Successful Application Architect for Java EE Applications
DEREK C. ASHMORE


MICROSERVICES for JAVA EE Architects
Addendum for The Java EE Architect's Handbook
DEREK C. ASHMORE
DVT Press