Application

Upon completion of this module, you should be able to:

- Explain the evolution of application platform
- Explain traditional application and its current challenges
- Explain the need for application transformation
- Explain the characteristics of modern application
- Explain the elements of application modernization approach

Accelerating digital business outcomes with application

Application driven business innovation

- What do these companies have in common?
  - They focus on compelling user experiences and responsive design
  - They make strategic use of big data – to improve service and experience through deep instrumentation
  - And they all bring new services to market quicker and more often than entrenched competitors
  - They use cloud-based infrastructure to develop and deploy their applications to meet the business demand

General Electric

Uber

Tesla

Application overview

- Evolution of application platform
- Traditional application and its current challenges
- Traditional application delivery model
- Need for application transformation
- Approaches to application transformation

Application overview

Application is a set of program that is designed to perform a group of coordinated tasks for the benefit of the users and businesses

- Unlock value from the digital world
- Reshape user experiences with new touchpoints
- Critical to achieving business objectives
  - Offer access to massive scale of people, data, devices
  - Increase insight, market reach, and customer satisfaction
- Manages the information and provide it in a form that is useful to the business

Evolution of application platform

Billions of Users

Platform 3

Cloud Big Data Mobile Social

Mobile Devices

Millions of Apps - Modern Applications

Hundreds of Millions of Users

Platform 2

LAN/Internet Client/Server

PC

Tens of Thousands of Apps - Traditional Applications

Millions of Users

Platform 1

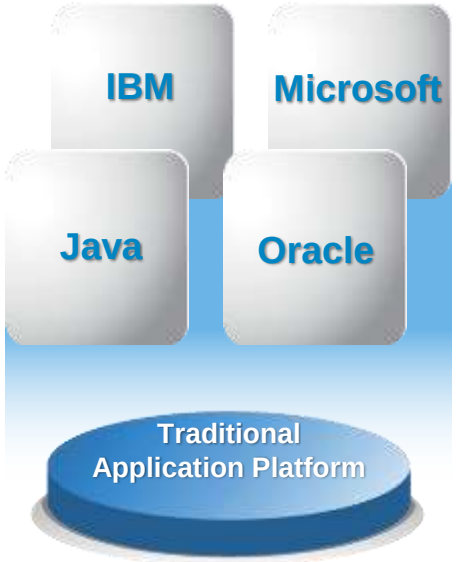
Mainframe, Mini computer

Terminals

Thousands of Apps - Traditional Applications

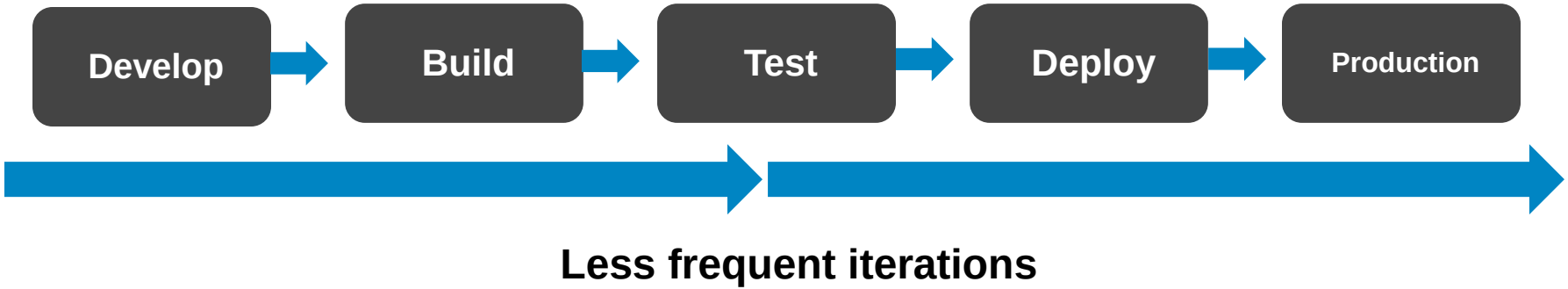
Traditional application

Characteristics	Description
Architecture	Single-tier and Multi-tier Monolithic in nature
User Interaction	Mostly desktop
Scalability	Expand monolithic box – Vertically scalable
Resilience	Infrastructure manages availability
Examples	SAP, Oracle, Microsoft Exchange/SharePoint



Traditional application: Delivery model

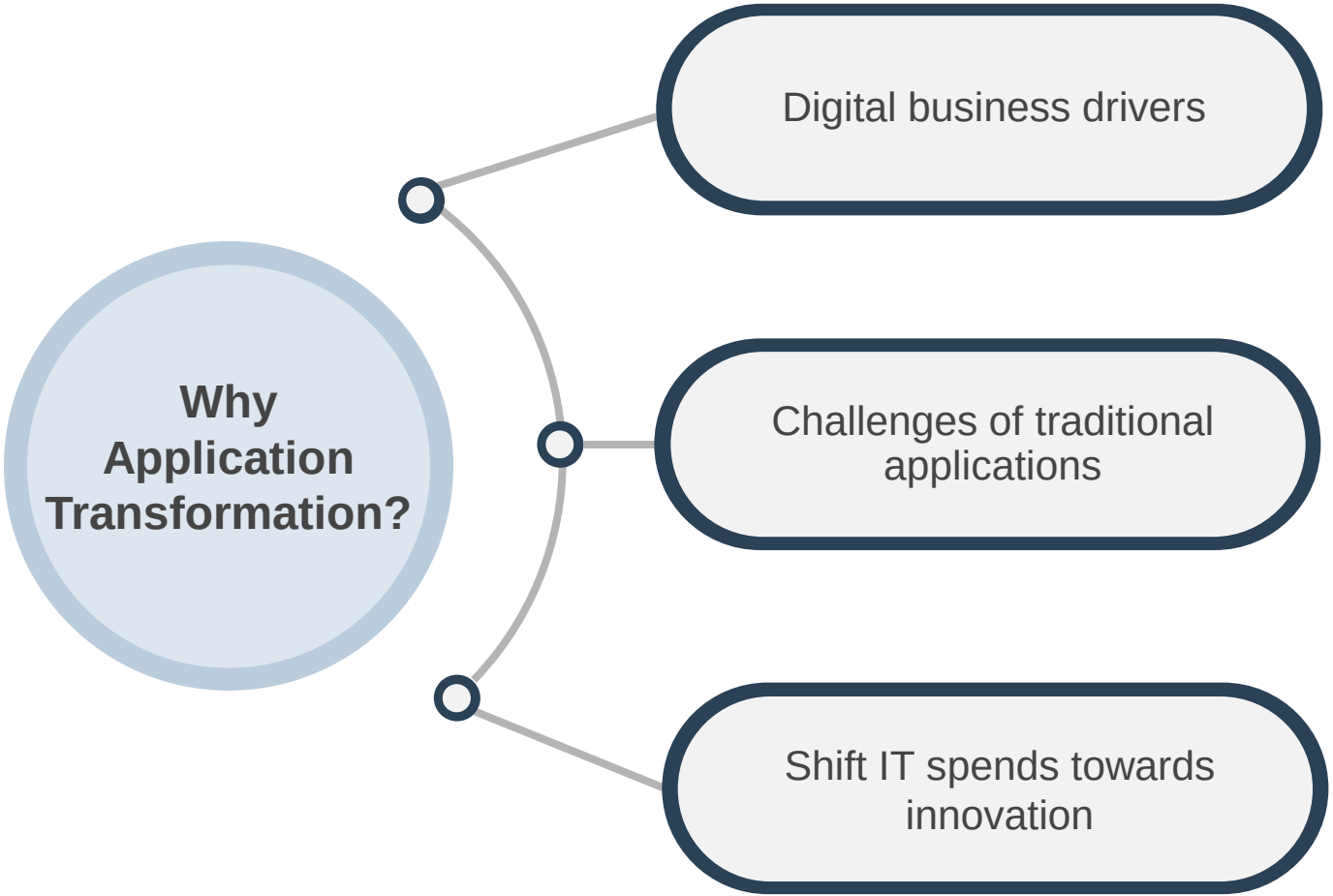
- Typically uses a linear and continuous approach in the development, build, test, deploy, and move to production process flow.
  - This approach offers a way to produce a full featured software product



Traditional application: Challenges

- Tight coupling of components
- Complex traditional software development processes
- Unable to quickly respond to new business opportunities
- Linear software performance growth assumed and planned for
- Competitive solutions are gaining market share

Need for application transformation



Approaches to application transformation

Refactor	Refactor the application using current frameworks and technologies
Revise	To use cloud-based technologies, revise the application
Retain	In the modern data center, retain the existing application to get benefits from the evolving trends
Retire	To optimize the cost, and consolidate infrastructure, retire the application

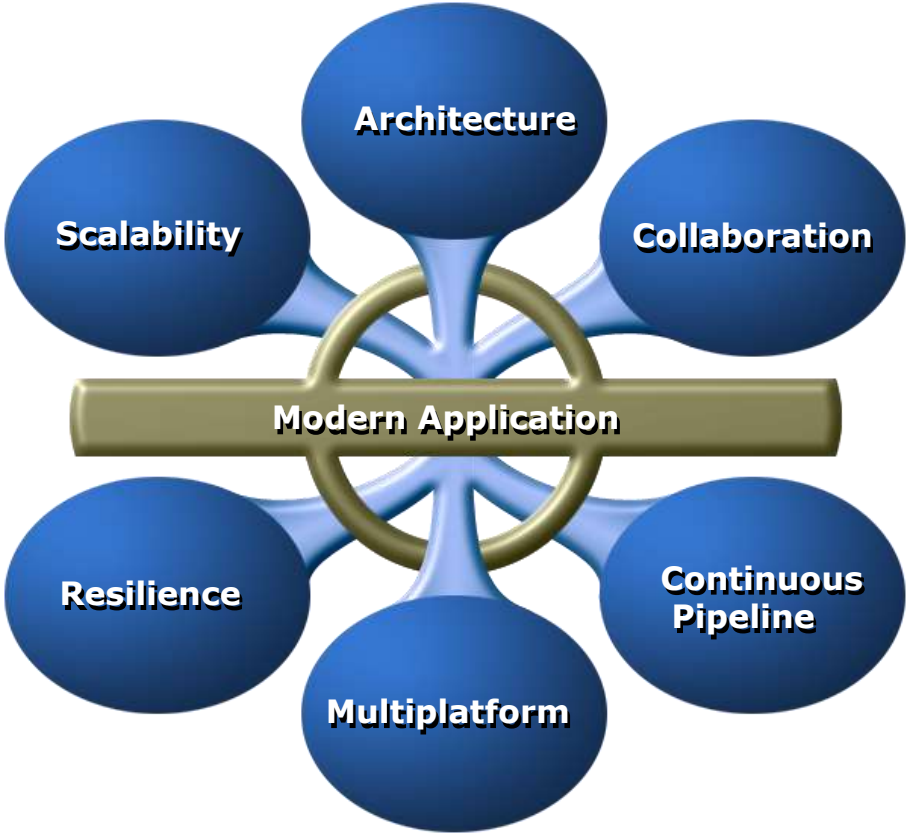
Modern application

- Key characteristics of modern application
- Comparison between traditional and modern application
- Application modernization approach

Modern applications

- Consists of a set of business-related functional parts that are assembled with specific rules and best practices
  - Things are loosely coupled making updates much easier and seamless from the end user perspective
- Modern application uses dynamic modern infrastructure platform.
- Enables services to be delivered in hours or days, not weeks
- The platform scales horizontally, instantly, and is designed for fault tolerance
- Long-term technology commitments are reduced because it is easier to replace particular modules

Modern application: Key characteristics



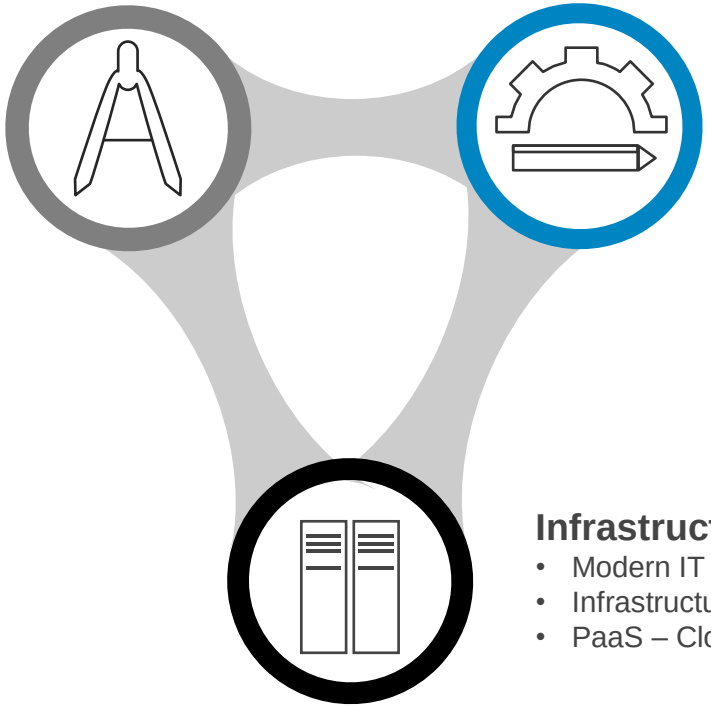
Traditional application vs. Modern application

Traditional Application	Modern Application
Monolith	Distributed
Common programming language	Multiple programming languages
Closed source	Open source
Resiliency and scale are infrastructure managed	Resiliency and scale are application managed
Infrastructure is application-specific	Infrastructure is application-agnostic
PC-based devices	Large variety of devices (BYOD)
Separate Build/Test/Run	Continuous development and deployment
Example: CRM, ERP, and Email – MS Outlook	Example: Facebook, Uber, and Netflix

Application modernization approach

Modern Architectures – Cloud-native Architecture

- 12 Factor Apps
- Microservices
- Containers



Lean Practices

- Agile development
- DevOps
- Continuous delivery

Infrastructure/Platform

- Modern IT infrastructure
- Infrastructure-As-Code
- PaaS – Cloud-native platform

Modern architecture: 12-factor apps

Codebase
One codebase tracked in revision control, many deploys

Dependencies
Explicitly declare and isolate dependencies

Config
Store config in the environment

Backing Services
Treat backing services as attached resources

Build, Release, and Run
Strictly separate build and run stages

Processes
Execute the application as one or more stateless processes

Port Binding
Export services via port binding

Concurrency
Scale out via the process model

Disposability
Maximize robustness with fast startup and graceful shutdown

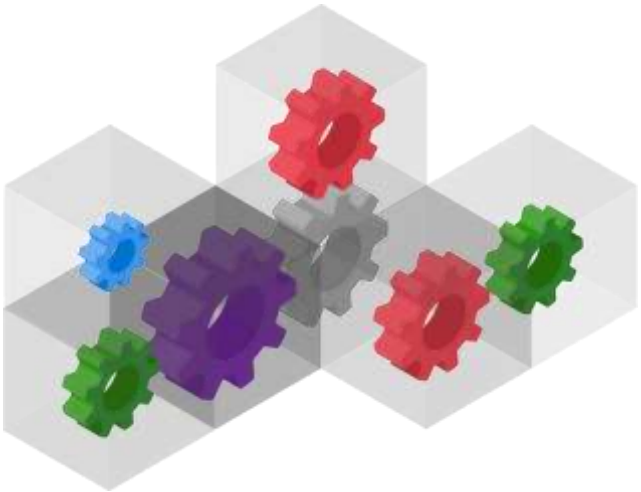
Dev/Prod Parity
Keep development, staging, and production as similar as possible

Logs
Treat logs as event streams

Admin Processes
Run admin/management tasks as 1-off processes

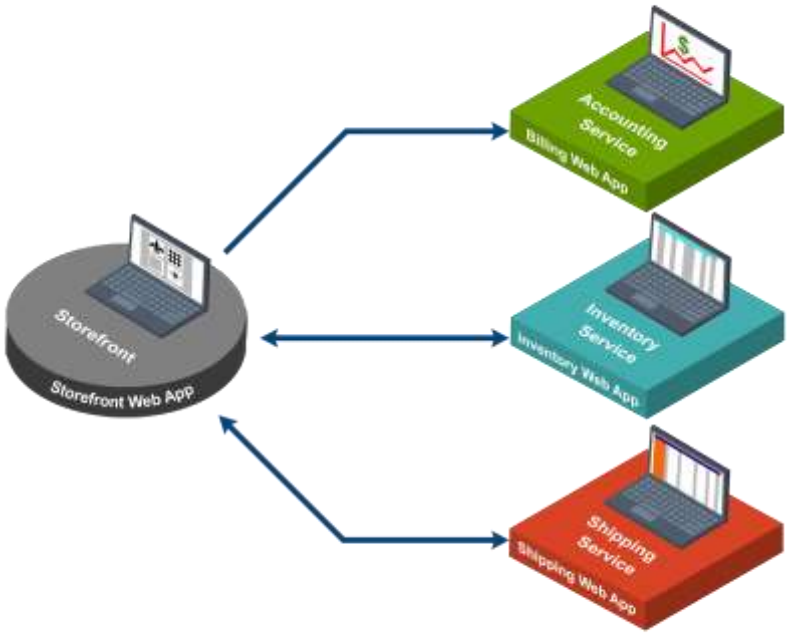
Modern architecture: Microservices

- Application functionality is segmented into independent and loosely coupled services
- Microservice runs in its own process and communicates to other services via REST APIs
- Every microservice can be deployed, upgraded, scaled, and restarted independent of other services in the application
- Teams can frequently update live applications without negatively impacting users



Microservices

Microservices design style: Example



- Benefits
  - Modularity
  - Small and independent
  - Enhances CI/CD
  - Increased fault isolation

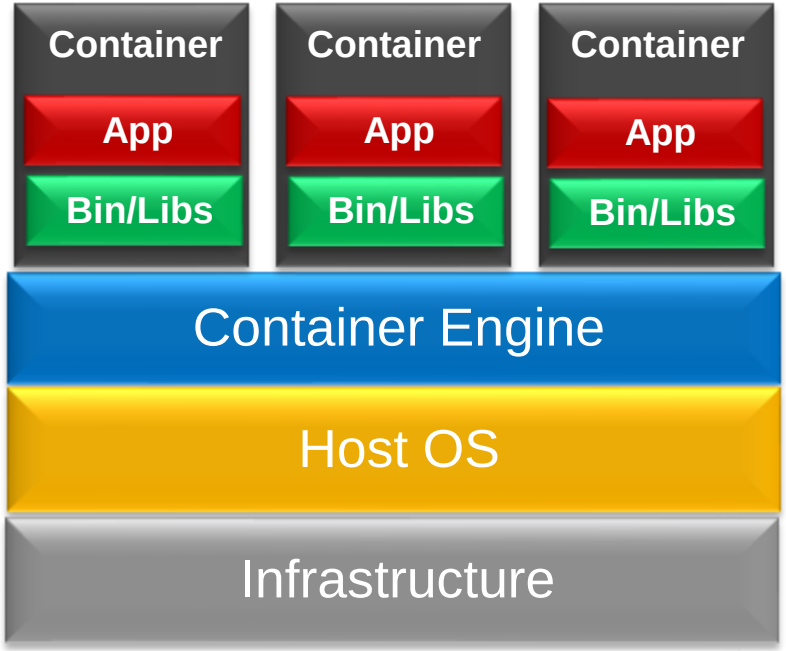
Microservices use case: A large entertainment company

- **Challenges and Requirements**
  - Used an entirely on-premises traditional infrastructure and struggling with expanding their infrastructure to handle an increasing customer demand
  - Even hiring more operations staff could not keep up
  - Faced challenges with availability, scalability, and performance of the application
  - Needed an architecture that allowed the company to be up and running 24/7, scale to the next order of magnitude, and be optimized for speed
  - Thousands of server instances need to be commissioned simultaneously if needed to meet increased demand for services
- **Solution**
  - The company transitioned from a traditional development model producing a monolithic DVD-rental application to a microservices architecture
    - Many small teams responsible for the development of hundreds of microservices
    - These microservices work together to stream digital entertainment to millions of customers every day

Container

Container is a lightweight, stand-alone, executable package. It consists of whole runtime environment such as an application, and its libraries, binaries, and configuration files needed to execute the application

- Containers are run on a control host system and share OS kernel of the host system
- Provide better portability without requiring code change
- Enable each microservice to be instantiated only when they are needed and are almost available immediately



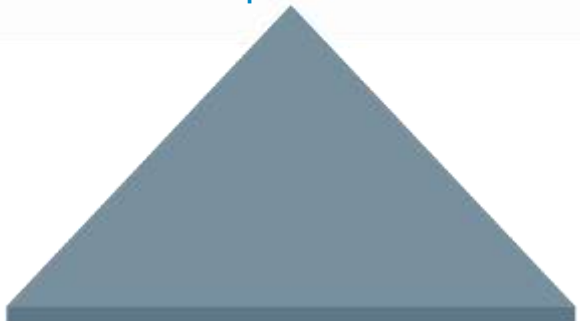
Lean practices: Agile

WATERFALL

Predictive Process

Fixed

Requirements



Estimated

Resources **Time**

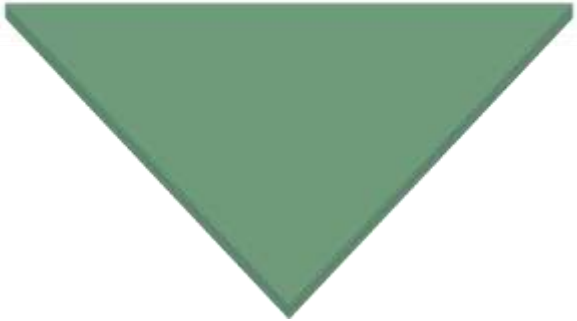
The plan creates cost and schedule estimates

- Linear
- Continuous
- All features added

AGILE

Adaptive Process

Resources **Time**



Features

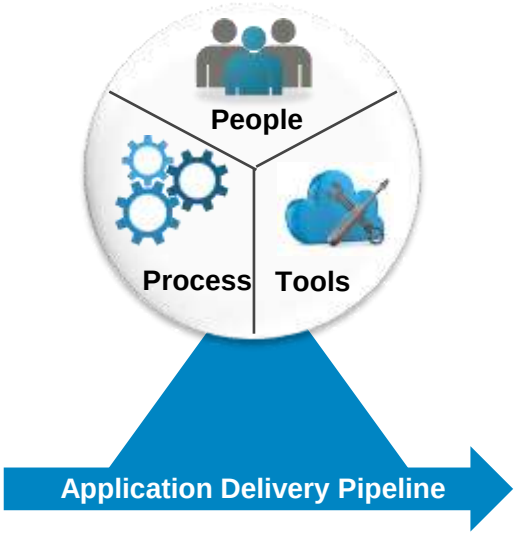
The vision creates feature estimates

- Iterative
- Focused on Minimal Viable Product (MVP)

Lean practices: DevOps

A methodology where the operations and development staff collaborate in the entire software lifecycle, from the design phase through the development and production phases

- DevOps is a framework for rapid application delivery
  - It uses the Agile and Lean methodologies.
- Helps organizations to improve their software release cycles, software quality, and ability to get rapid feedback on product development
- DevOps focuses heavily on building an integrated, automated tool chain minimizing issues and defects



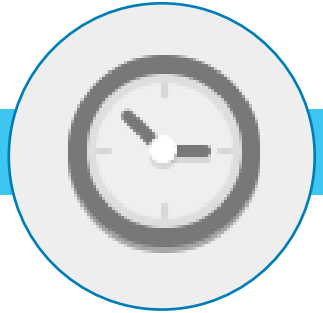
Lean practices: DevOps (Cont'd)

24x

Faster recovery from failures than peers

200X

More deployments than peers



49%

People time is spent on new work

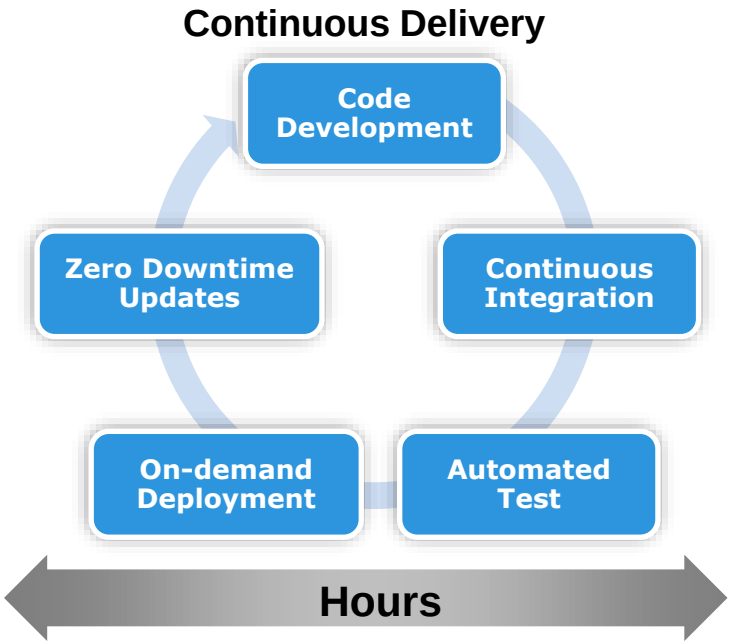
DevOps supports businesses to achieve agility to meet business goals faster

Lean practices: Continuous Integration and Continuous Delivery

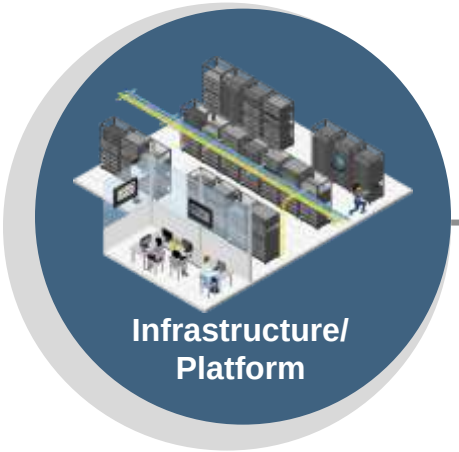
Continuous Integration (CI) enables software changes to be pushed to production repeatedly and reliably.

Continuous Delivery (CD) enables teams to produce software in short cycles, ensuring the rapid and reliable delivery of software at any time in a low-risk manner

- Continuous delivery processes are automated, and supporting tools are integrated to minimize manual steps and human intervention
- Allows developers to spend more time on developing software features instead of being concerned about internal meetings and approval cycles
- Automated delivery pipeline accelerates time-to-market and reduces bugs

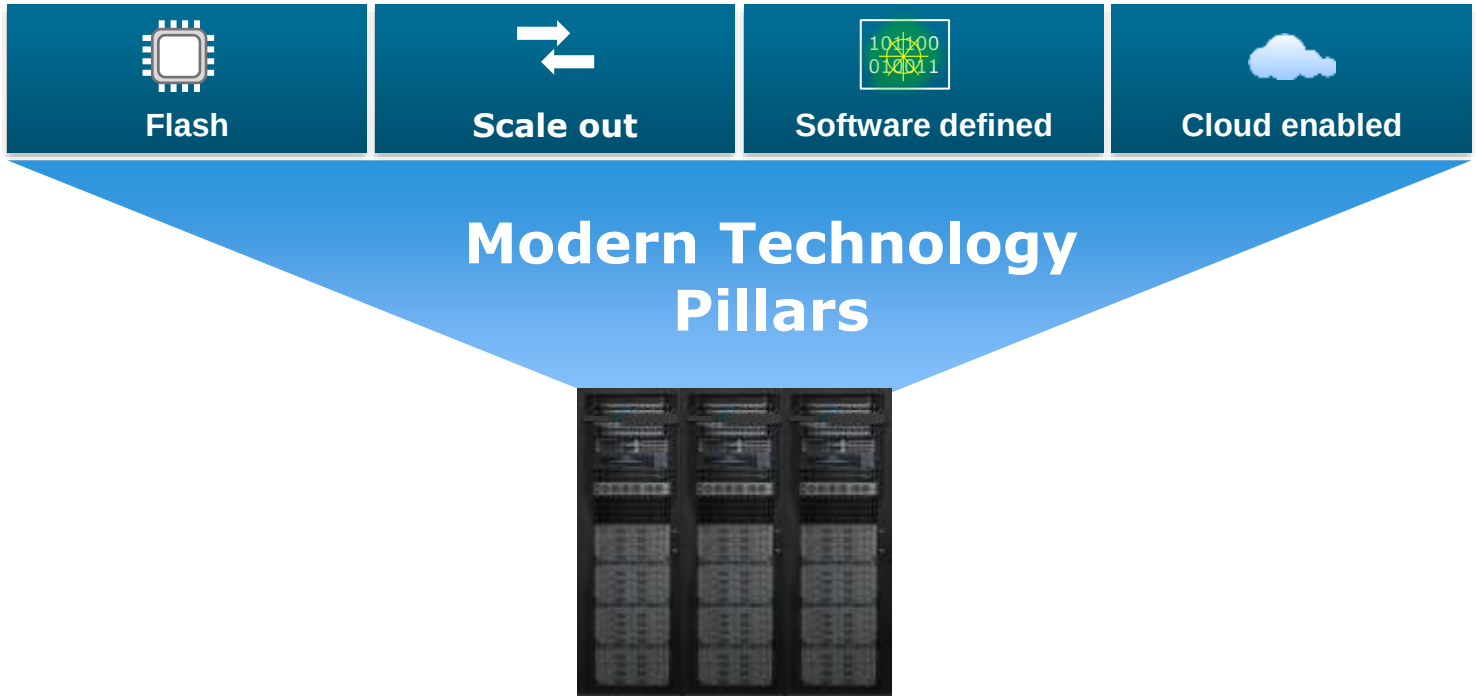


Infrastructure/Platform to support modern application



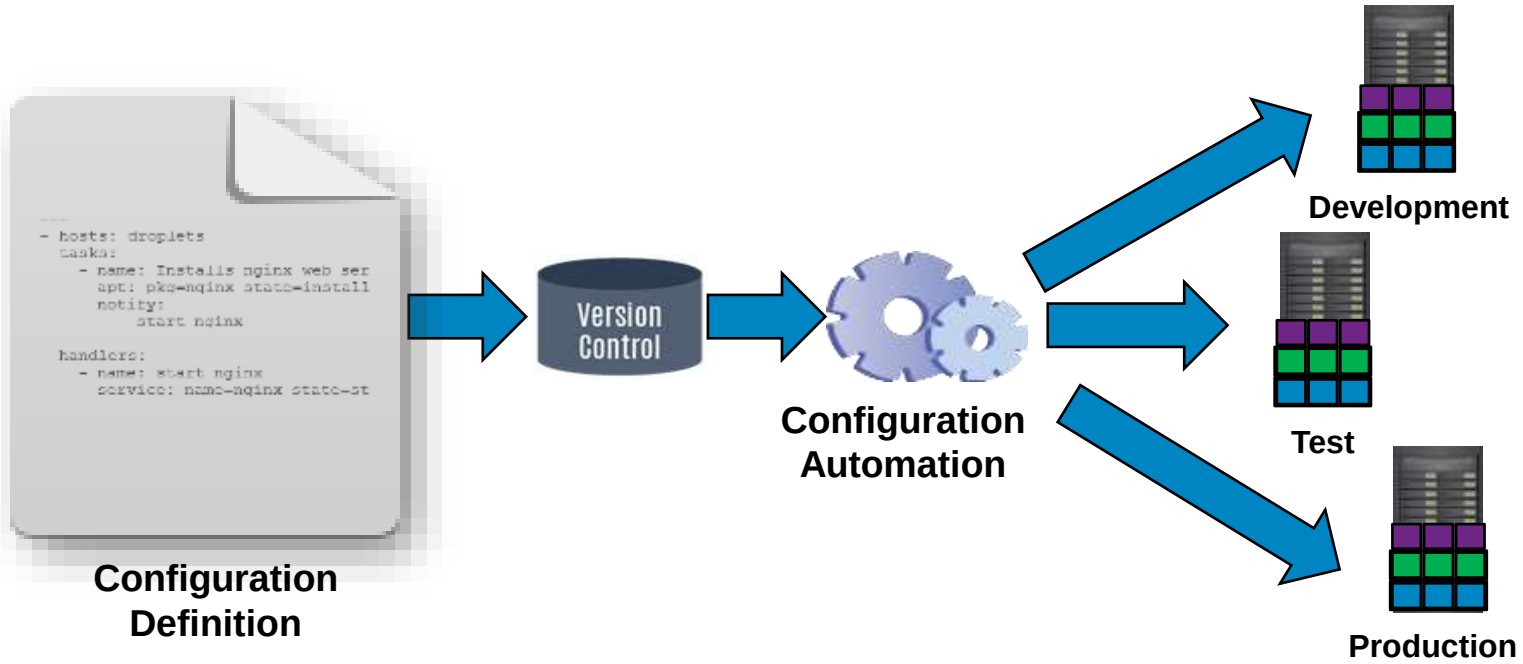
- Modern IT infrastructure
  - ✓ Flash, Scale out, Software-defined, Cloud-enabled
- Infrastructure as Code
  - ✓ Key enabler for continuous delivery
- Platform as a Service – Cloud-native Platform

Modern IT infrastructure



Infrastructure as Code

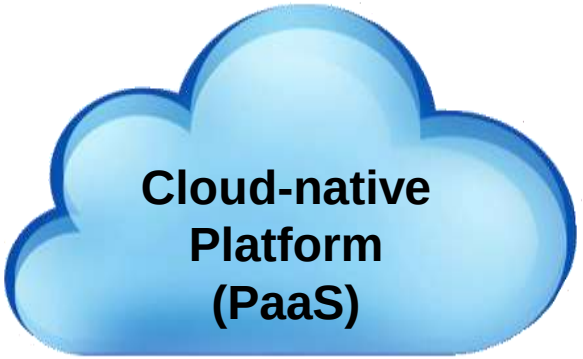
Infrastructure as code, IaC, is the process of managing and provisioning IT resources using script files rather than manually configuring resources.



https://searchitoperations.techtarget.com/definition/Infrastructure-as-Code-IAC

Cloud-native platform for modern applications

- Cloud-native platform is an integrated stack – from infrastructure through application development framework that is optimized for rapidly scaling modern application.
- Cloud-native platform supports application developers with multiple languages, frameworks, and middleware.
- Few important capabilities of Cloud-native platforms
  - Application auto scaling
  - Application portability
  - Dynamic load balancing and routing
  - Resilience and fault tolerance
  - Application health management through monitoring
  - Role-based access for deployed applications



Customer success stories: Pivotal Cloud Foundry – Cloud-native platform

To access the video, please click the below link

https://edutube.emc.com/Player.aspx?vno=iWYOZZ/p70rYOdkKlZBaAg==&autoplay=true&t=0h0m0s

Case Study

Case study: A Leading Broadcast Satellite Provider

- **Organization Profile**
  - The company began as a tech startup and installing large satellite dishes and connecting people to a new world of entertainment
  - The company grew, took risks to succeed, and launched a small dish satellite service
  - It has pioneered technology in their domain and the company has approximately millions of subscribers

Case study: A Leading Broadcast Satellite Provider (Cont'd)

- **Challenges and Requirements**
  - The company operates in the confluence of several highly competitive industries:
    - Technology, Entertainment, and Telecommunications
  - Want to continually deliver new technology faster to customers
  - Need to continue to innovate for customers by enabling availability of applications on multiple streaming and mobile devices
  - Demand to provide a reliable and positive customer experience for consumers
  - Must attract the right talent to deliver innovation
  - Company executives began to look at different cloud platforms and software frameworks that would improve speed-to-market.

Case study: A Leading Broadcast Satellite Provider (Cont'd)

- **Solution**
  - Pivotal's Cloud Foundry and Pivotal Labs native cloud platform and software frameworks improves the speed-to-market
  - The company adopted Pivotal's agile approach and Pivotal Cloud Foundry as the platform to enable strategic business initiatives
  - Developers are working in pair programming stations to deliver applications for consumers and employees
- **Business Benefits**
  - Save hours and days by quickly prototyping and innovating ideas
  - Introduce biweekly software launches instead of quarterly so customer experience is constantly evolving
  - Create a thriving, agile culture that appeals to top talent

Summary

Key points covered in this module:

- Evolution of application platform
- Traditional application and its current challenges
- Need for application transformation
- Characteristics of modern application
- Elements of application modernization approach