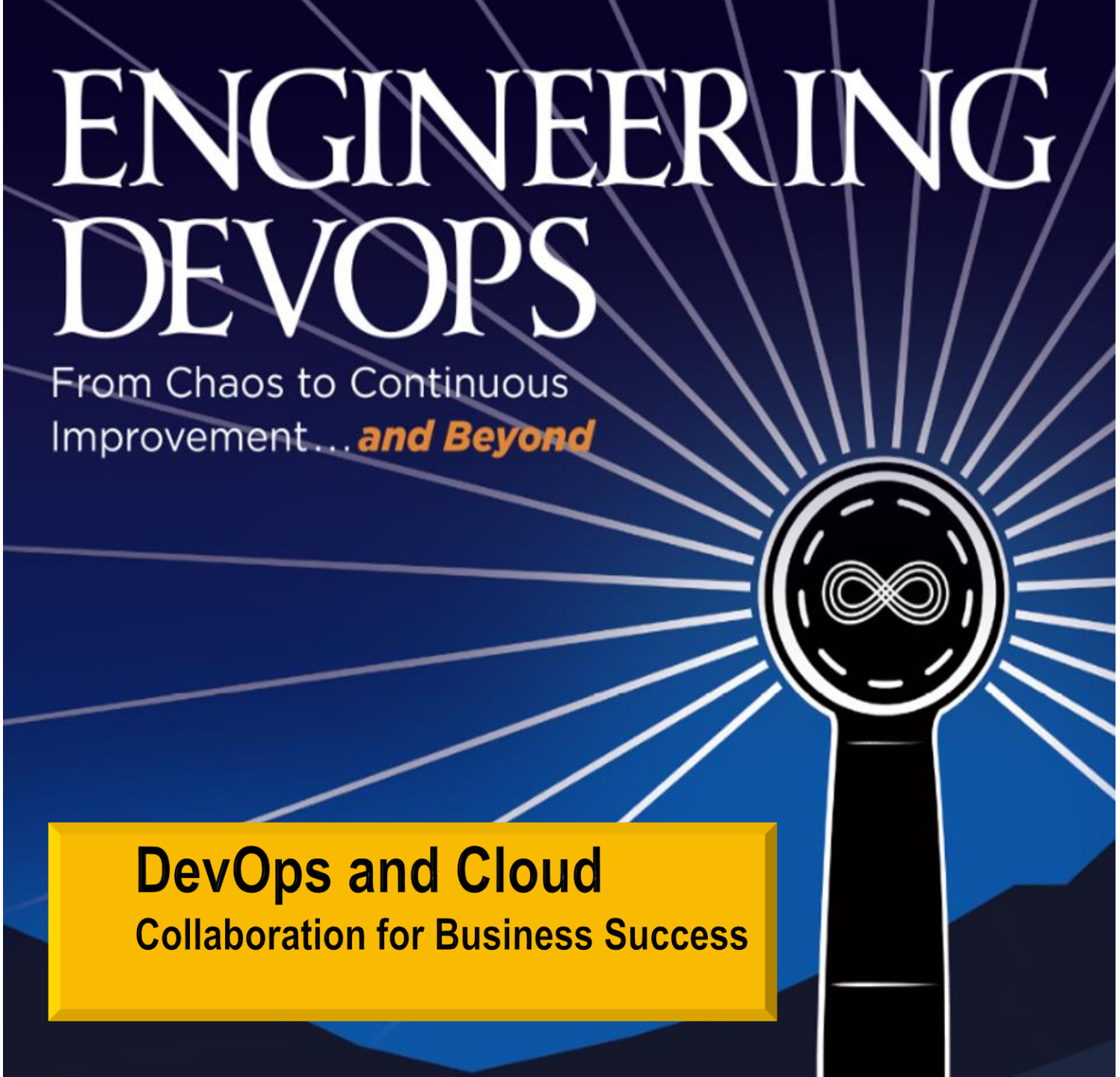
ENGINEERING DEVOPS

From Chaos to Continuous Improvement…*and Beyond*



DevOps and Cloud

Collaboration for Business Success

Marc Hornbeek
Principal Consultant

EngineeringDevOps.com

IEEE

DevOps INSTITUTE
ADVANCING THE HUMANS OF DEVOPS

DevOps.com

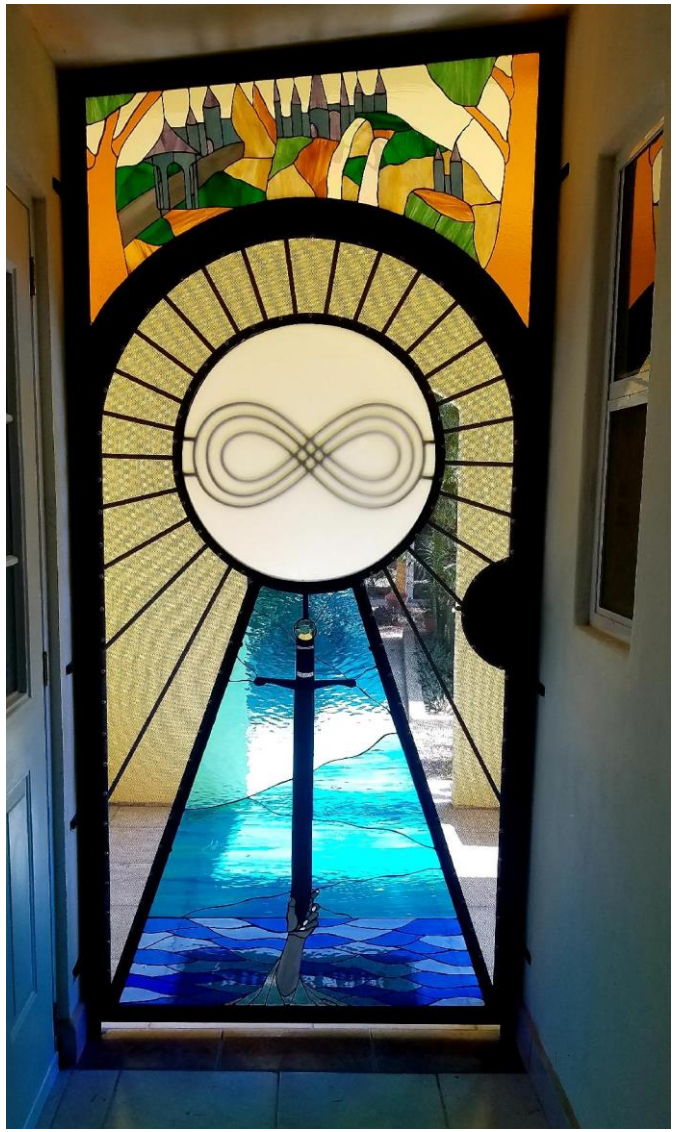
DEVOPS INDIA SUMMIT
An Xellentro Event

August 31st – September 1st 2019
Bengaluru Marriott Hotel Whitefield

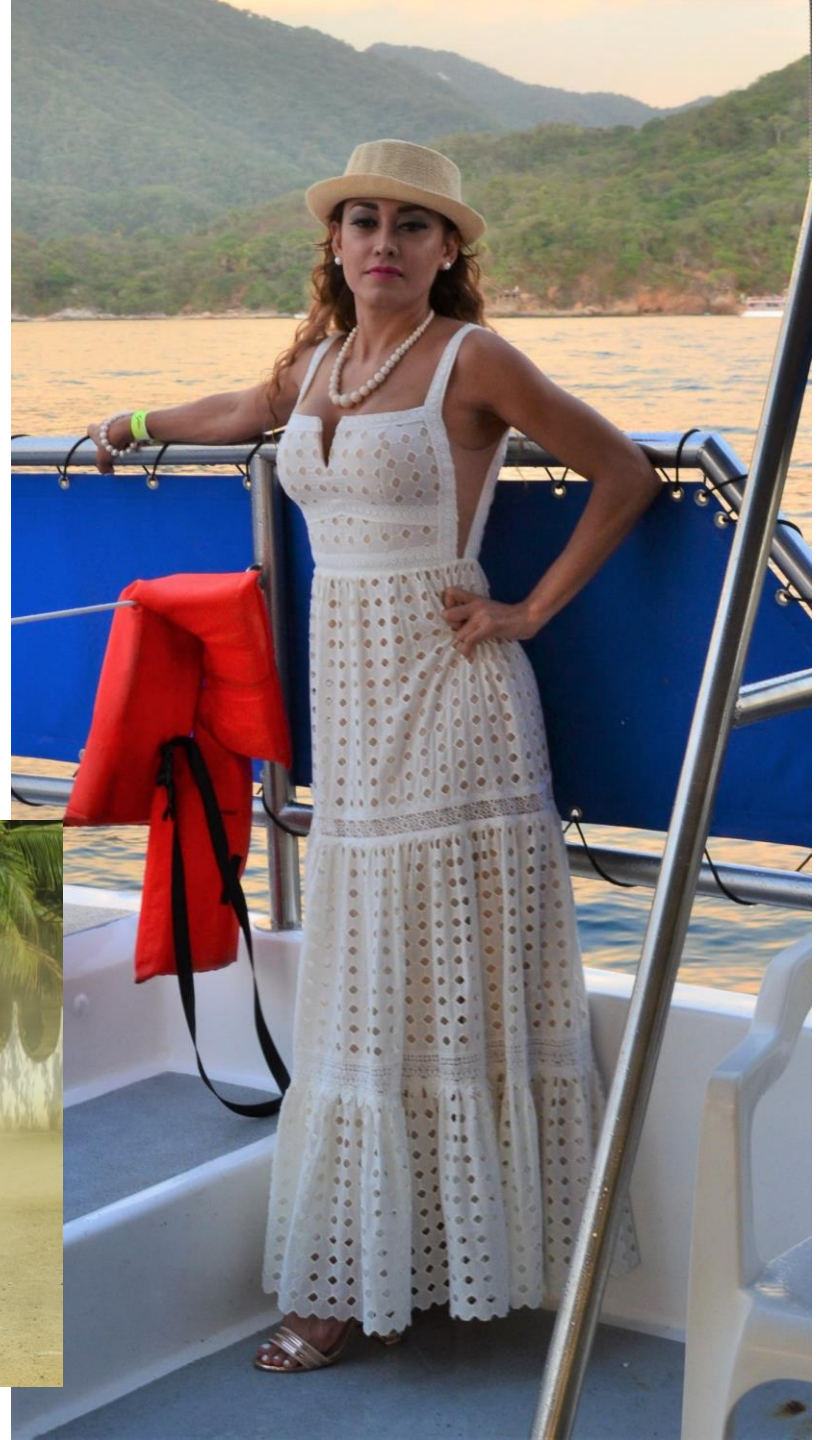
https://devopsindiasummit.com/

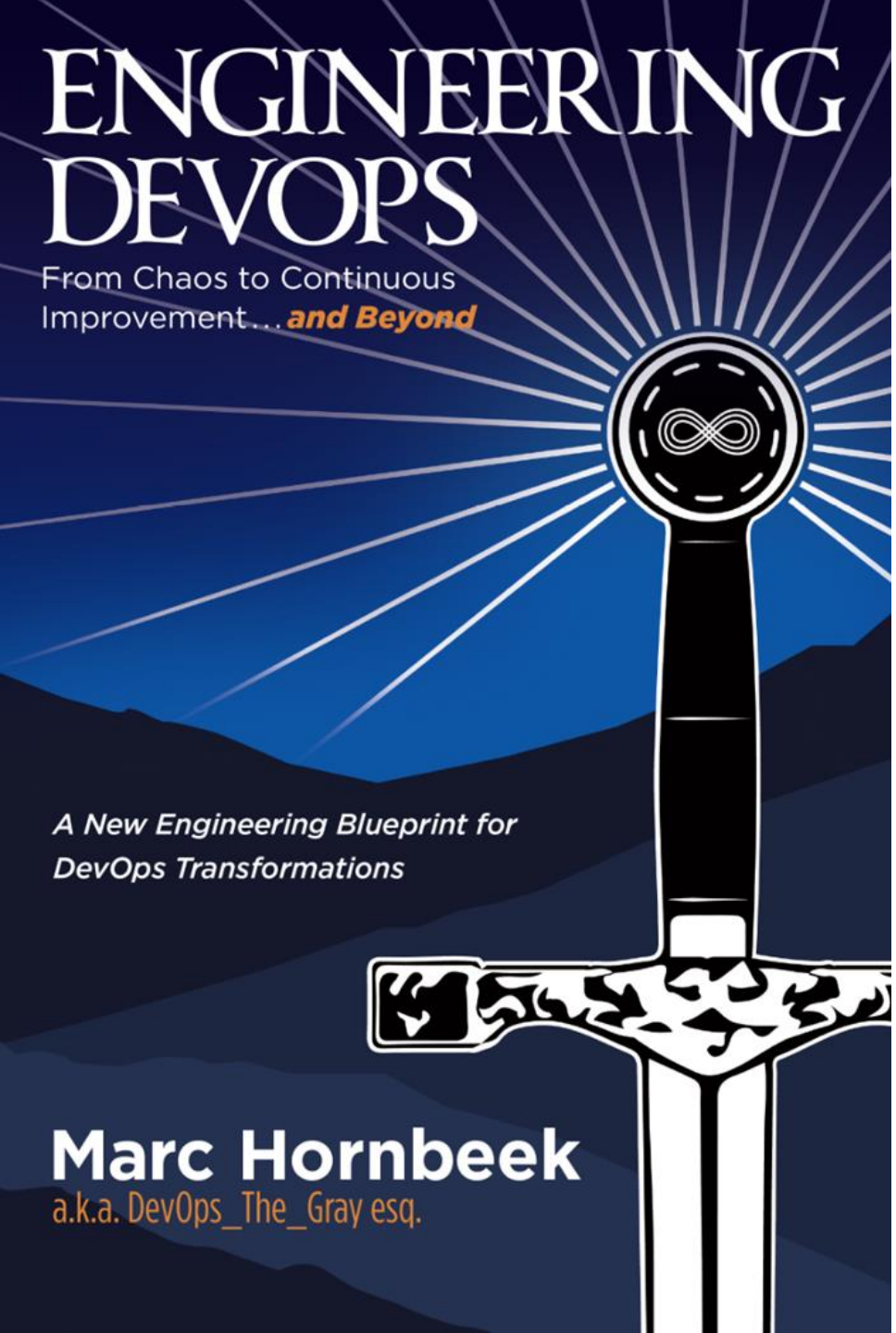
I relocated to Puerto Vallarta Mexico

Mi Casa en Puerto Vallarta, Mexico



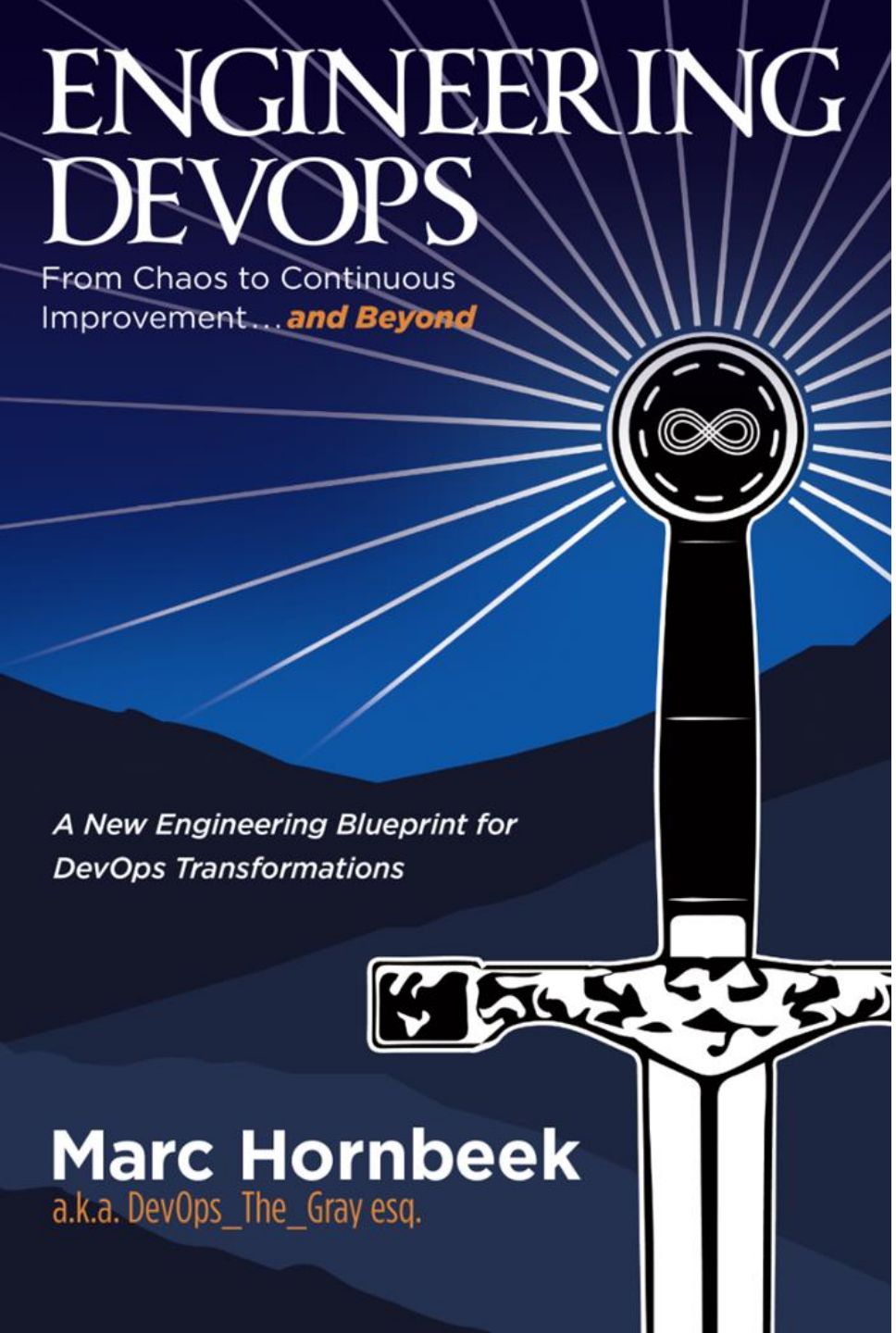
I got engaged to my Spanish Princessa Virginia ☺





Agenda 9:30 TO 1PM

- What is Engineering DevOps and Why is it important?
- How do you Engineering People, Process and Technology for DevOps?
- How do you Engineer Applications, Pipelines and Infrastructures for DevOps?
- Seven-Step DevOps Engineering Transformation Blueprint
- What is the future of Engineering DevOps – beyond Continuous Improvement?
- How can you learn more about Engineering DevOps?



What is Engineering DevOps?
and
Why is it important?

Why Engineering is needed for DevOps!

"How to do DevOps"?

"Doing DevOps" but not satisfied with the results?

Why?

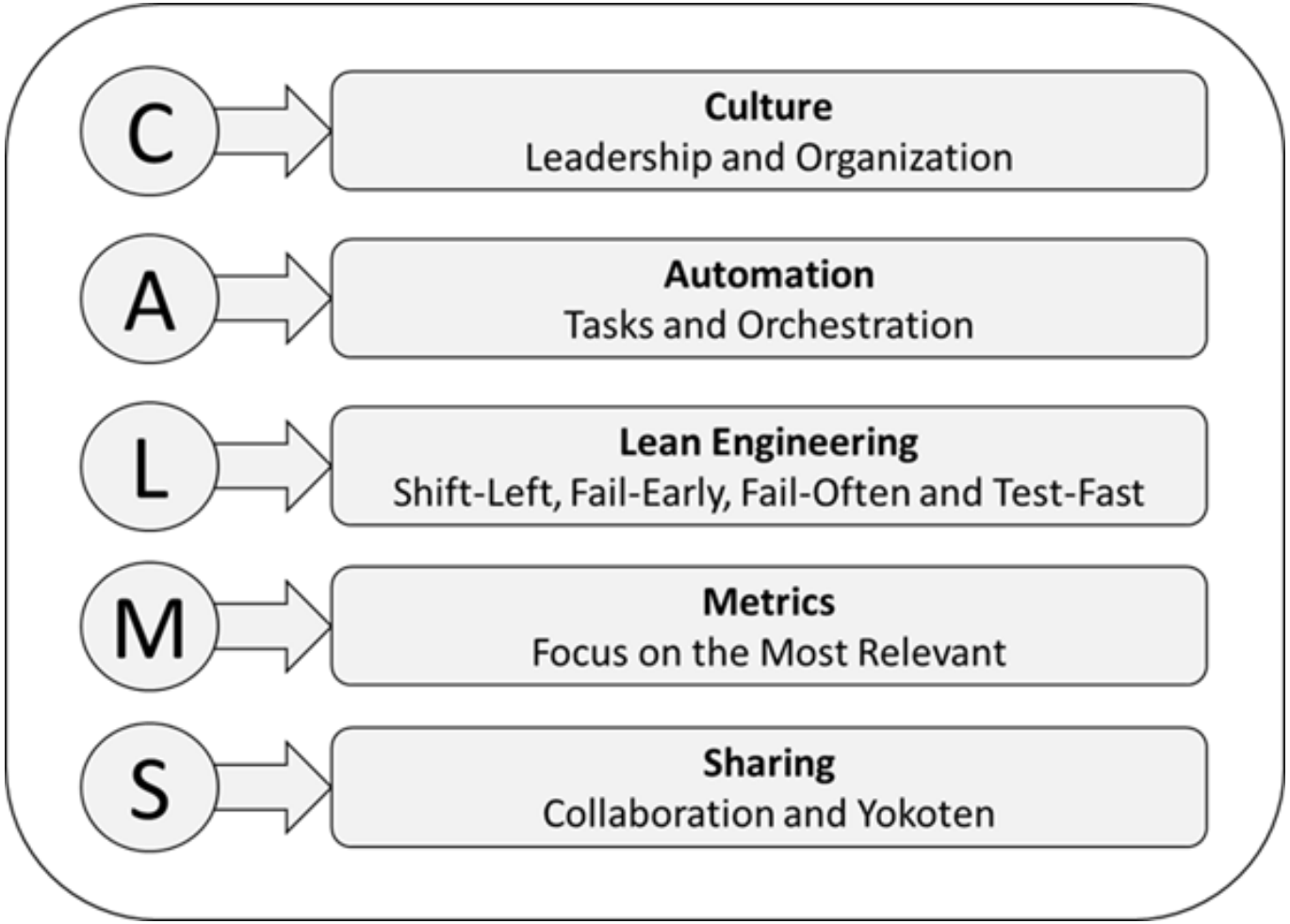
- DevOps is complex
- Three axis of DevOps variation : Applications, pipelines, infrastructures
- Multiple levels of DevOps maturity: Chaos, Continuous Integration, Continuous Flow, Continuous Feedback, Continuous Improvement
- No prescription, installation guide, user's manual or maintenance manual
- No standard definition!

How can anyone expect to do DevOps and get good results with DevOps without clear and definitive guidance?

DevOps CALMS Model

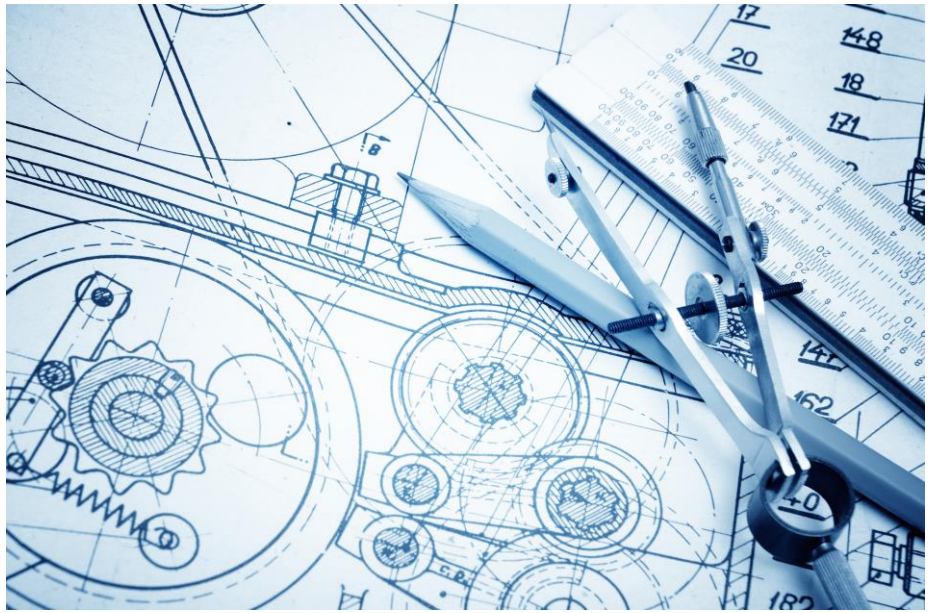
CALMS helps understand DevOps principles … but …

… is not prescriptive enough to “Do DevOps” or to “Assess and Improve DevOps performance”.



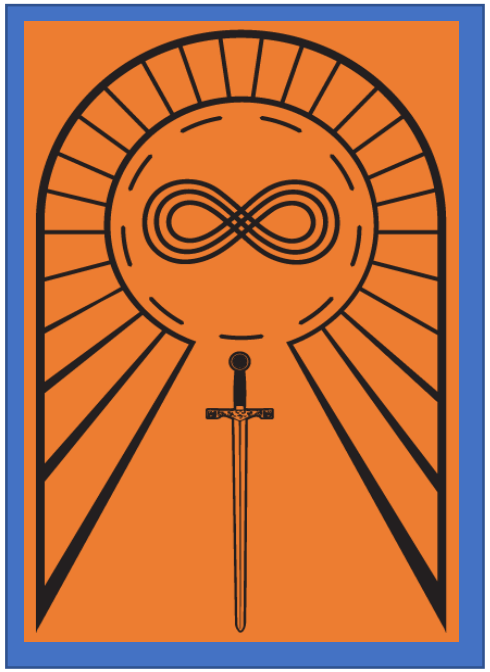
DevOps is Engineering

- Leadership and culture
  - Teamwork, Governance
- Lean Manufacturing
  - Value stream management, CI/CD Pipelines, Release Orchestration, Release Management, Deployments
- Simulations and Prototypes
  - Agile, Dark launches
- Testing, Measurement and Statistical Quality control
  - Continuous Testing, Monitoring, Security
- Tools
  - Infrastructure, Tool chains



DevOps Engineering Tools

www.EngineeringDevOps.com

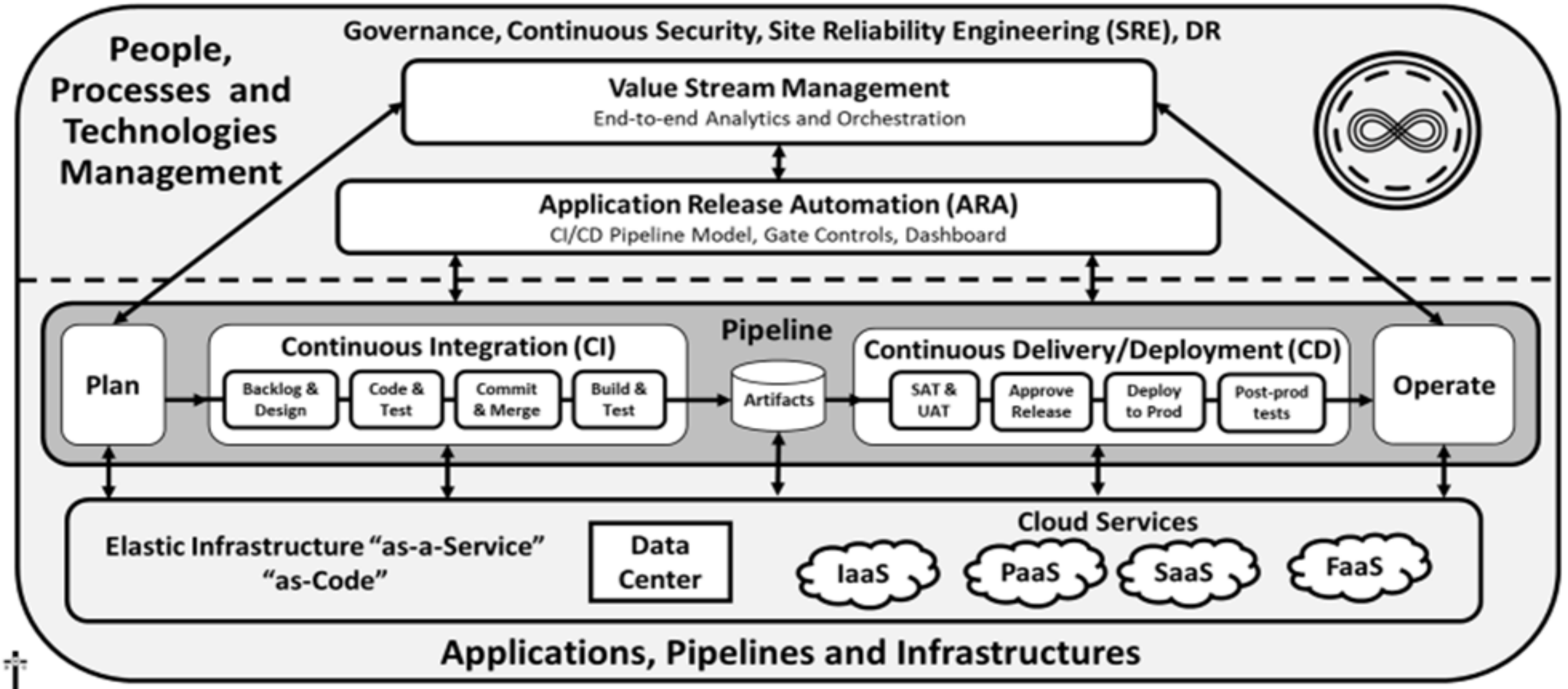


- DevOps engineering tools
  - Blueprints for major configurations
  - Recommended engineering practices
  - Measurements
  - Disciplined construction and evolution
    - ***Seven-step engineering transformation process***

Assist DevOps leaders and practitioners to understand, assess, define, implement, operationalize and evolve DevOps specific to and organization

Applicable to the widest possible range of implementations

DEVOPS / CLOUD ENGINEERING BLUEPRINT

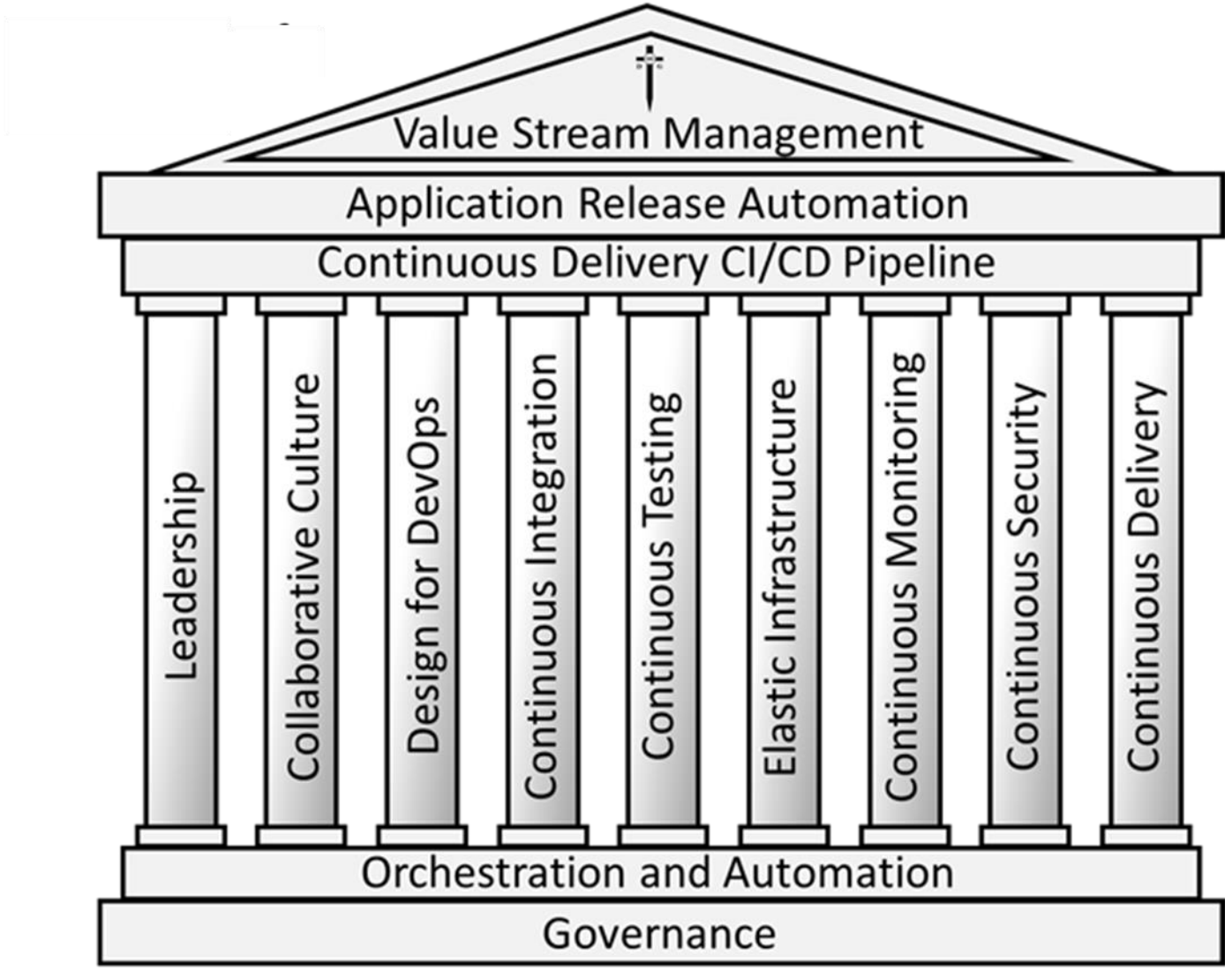


Nine Pillars of Engineering DevOps

- 9 DevOps Pillars with:
  - Concrete engineering blueprints
  - Recommended engineering practices

Prescriptive engineering foundations

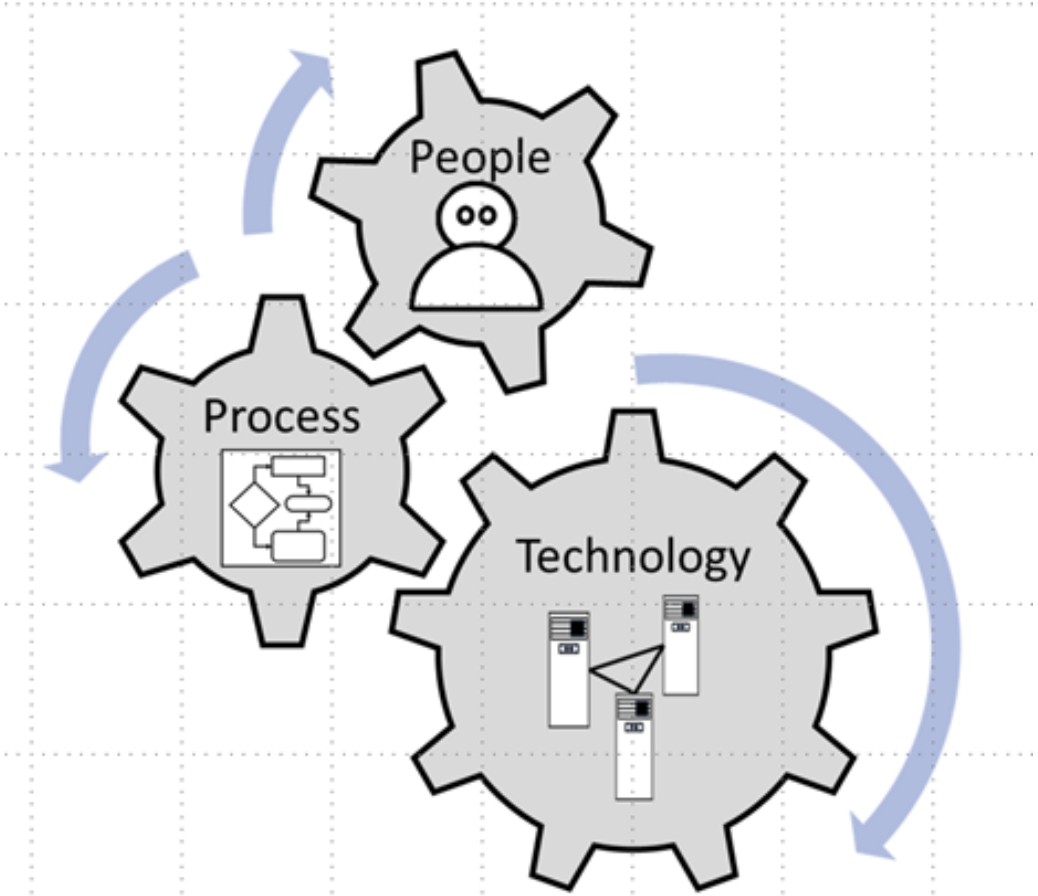
- how to “Do DevOps”
- How to Assess / Improve DevOps Performance



Three Dimensions of DevOps

Each DevOps Pillar has Three Dimensions that are relevant engineer solutions:

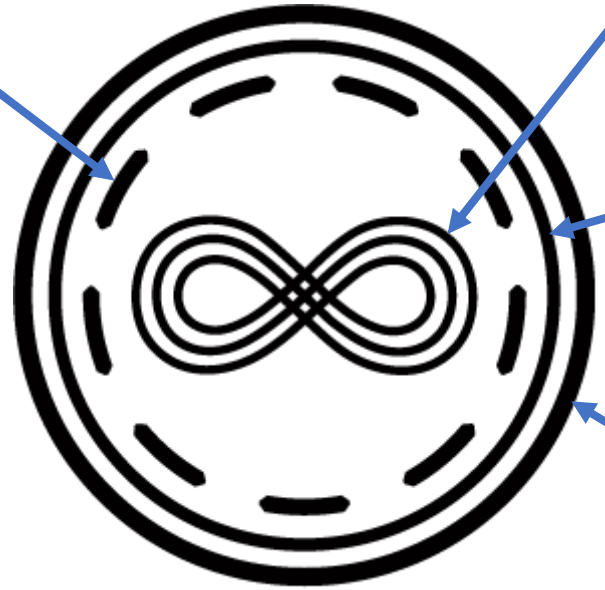
1. People
2. Process
3. Technology



Engineering DevOps Symbol

Nine Pillars of DevOps

1. Leadership
2. Collaborative Culture
3. Design for DevOps
4. Continuous Integration
5. Continuous Testing
6. Continuous Security
7. Continuous Monitoring
8. Elastic Infrastructure
9. Continuous Delivery



Three Dimensions of DevOps

1. People
2. Process
3. Technology

Three Axis of DevOps Variations

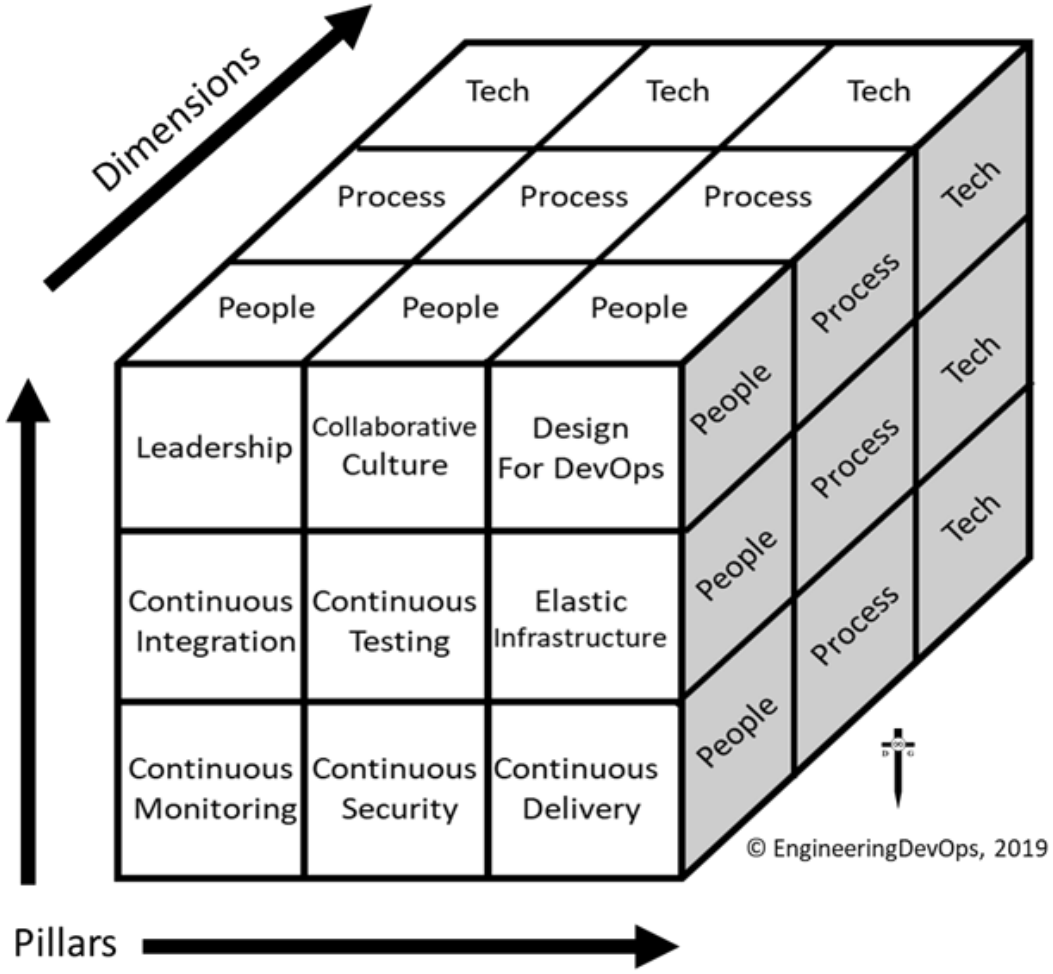
1. Applications
2. Pipelines
3. Infrastructure

5 Levels of DevOps Maturity

1. Chaos
2. Continuous Integration
3. ***Continuous Flow***
4. ***Continuous Feedback***
5. ***Continuous Improvement***

27 Critical Success Factors For Engineering DevOps

- 9 Pillars x 3 Dimensions = 27
- Complex ! - DEAL WITH IT!
- Engineering methods and tools needed to craft well-engineered DevOps solutions



Well-Engineered DevOps Requires Balance

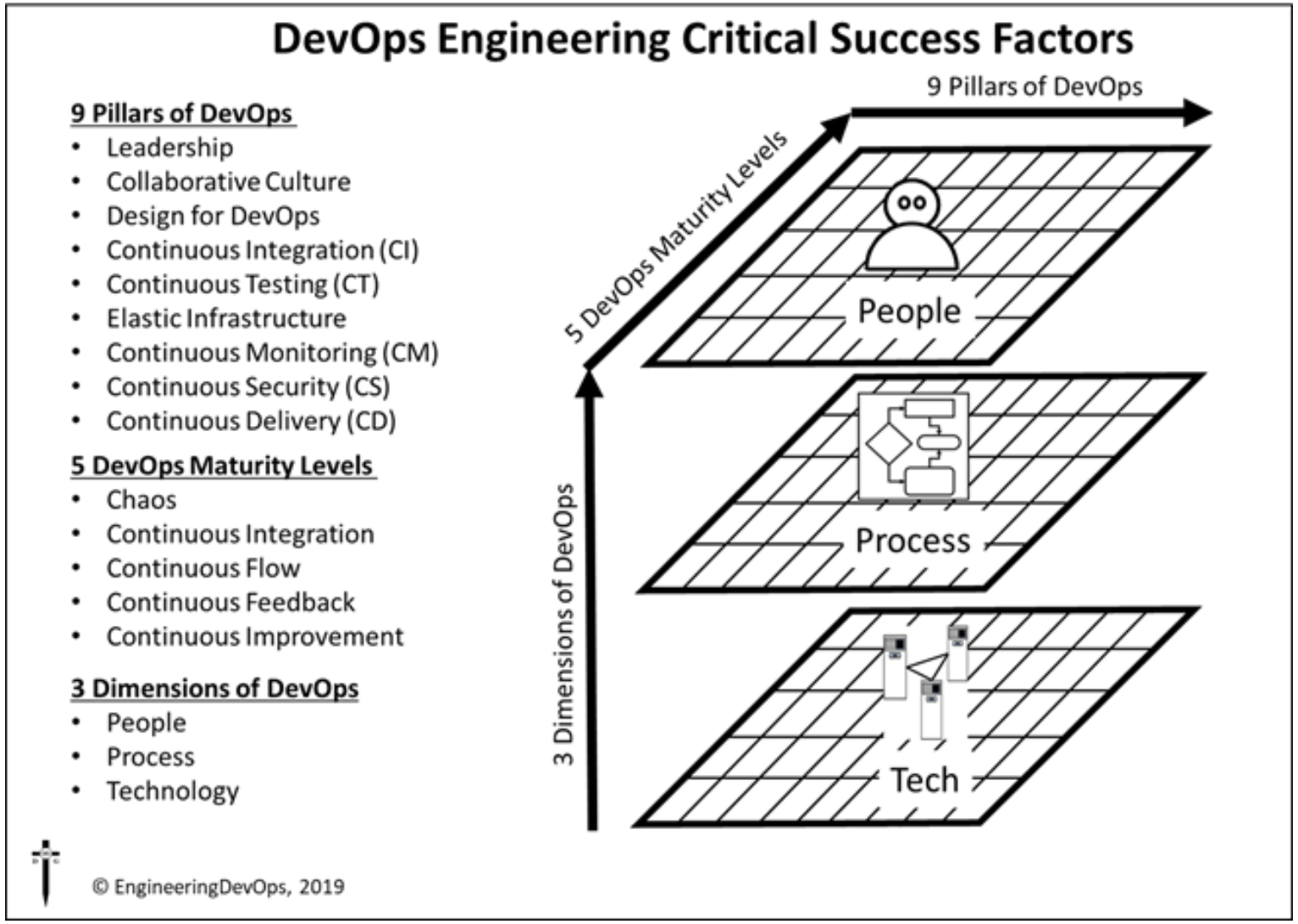
Engineering DevOps Solution approach is to pick solution components which

Align:

- Nine Pillars of DevOps and
- Three Dimensions of DevOps

To:

DevOps Maturity Level



Benefits of Well-Engineered DevOps

Tactical Benefits

- Agility
- Stability
- Efficiency
- Security
- Quality
- Satisfaction

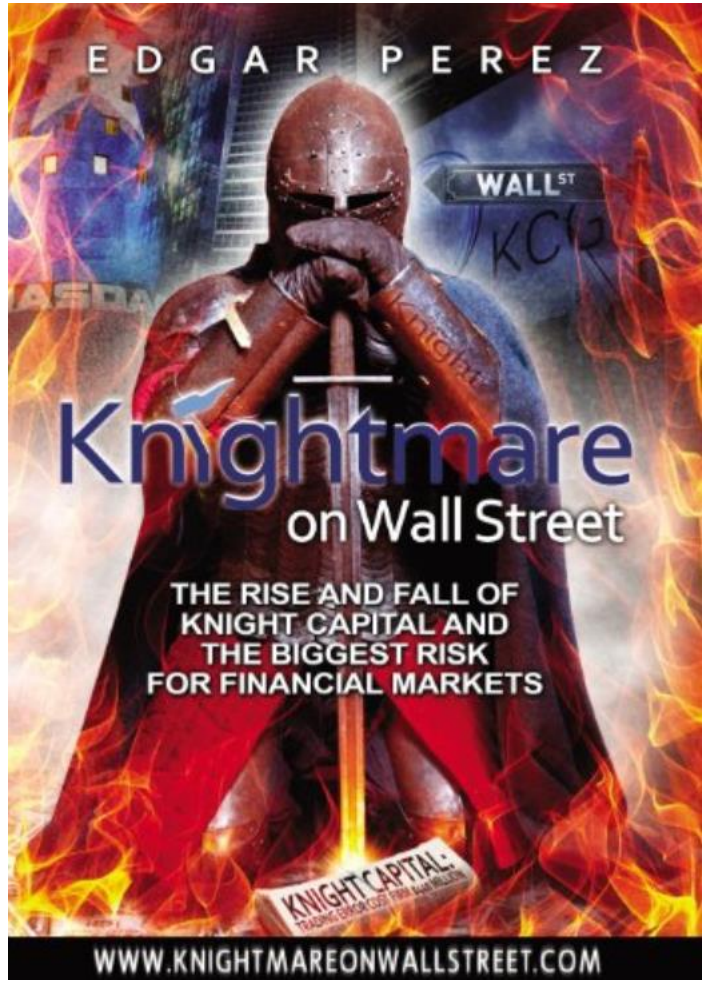
Strategic Benefits

- Competitiveness
- Innovation leader
- High Valuation
- Longevity

Cost of Poorly Engineered DevOps

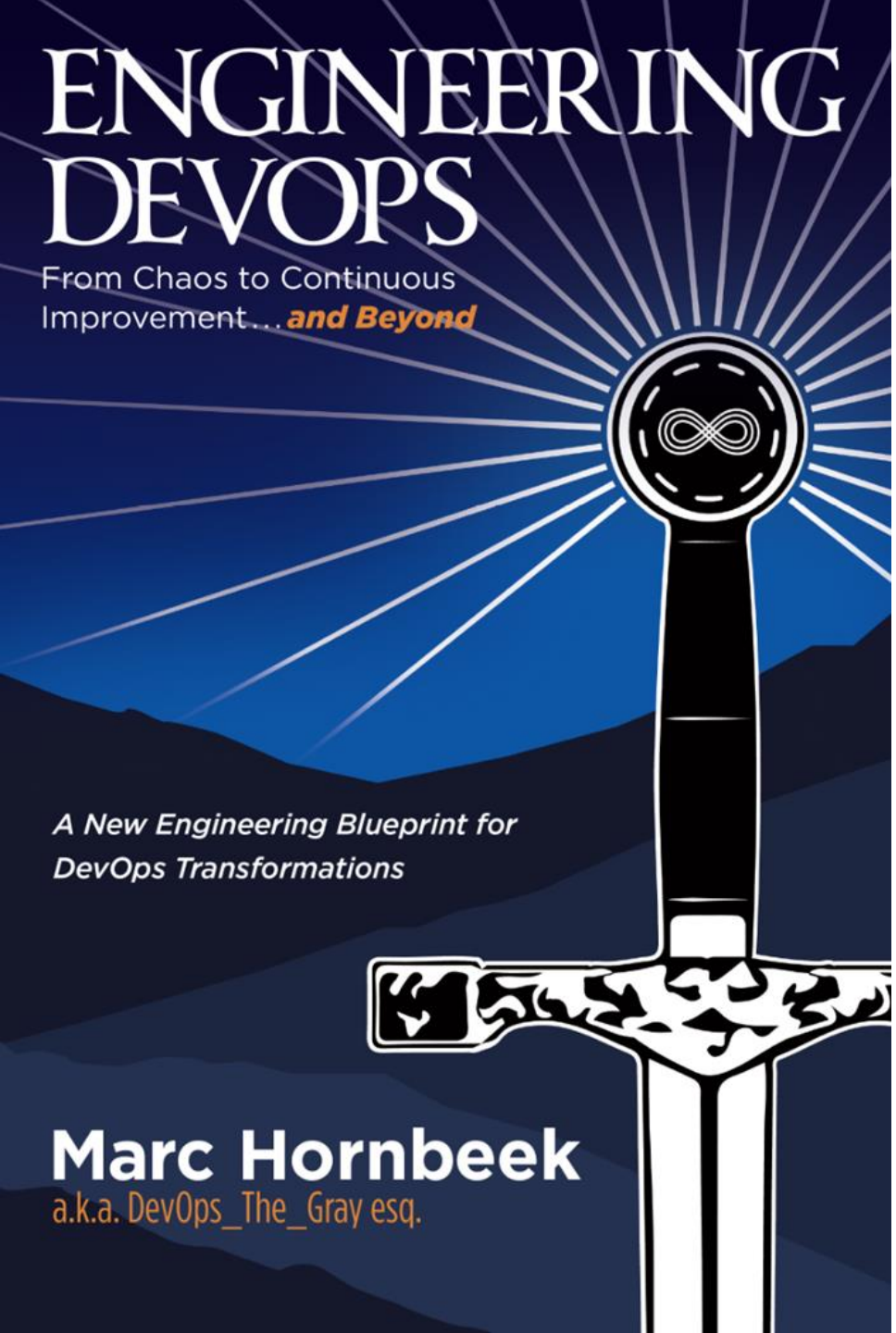
Between July 27, 2012 and July 31, 2012 Knight Capital Group manually deployed new software to a limited number of servers per day – eight (8) servers in all. “During the deployment of the new code, however, one of Knight’s technicians did not copy the new code to one of the eight computer servers.”

Knight Capital Group realized a $460 million loss in 45-minutes. Knight only had $365 million in cash and equivalents. In 45-minutes Knight went from being the largest trader in US equities and a major market maker in the NYSE and NASDAQ to bankrupt. NASDAQ and SEC fines and payments to investors ensued.



Engineering DevOps Definition

"Engineering DevOps is <u>the application of</u> lean engineering practices, (continuous flow, feedback and improvement), *to* People, Processes and Technology <u>for the benefit of</u> agility, stability, efficiency, quality, security, availability and satisfaction."



How do you Engineer People, Process and Technology for DevOps?

People + Process + Tech = DevOps

Engineering methods are key to success with DevOps

My argument is DevOps should be classified as a discipline of ***software engineering*** *rather than general computer science* and performed using engineering disciplines.

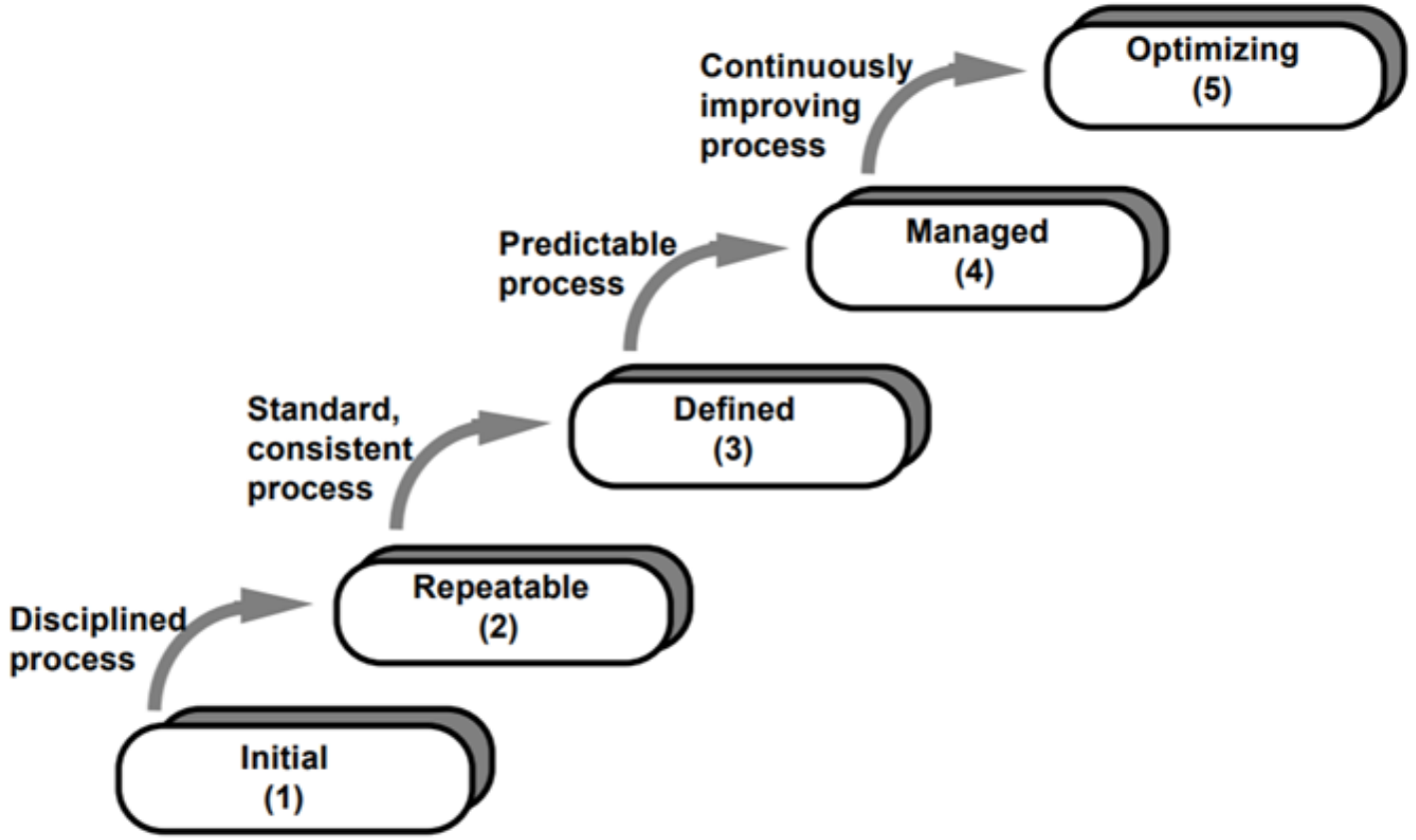
So why does it matter? DevOps Engineering emphasizes DevOps tools including ***blueprint models*** and specific ***engineering practices***, disciplined ***measured processes***, ***calibrated tools***, ***systematic progress tracking***, ***validation***, strict ***governance*** and ***reuse of artifacts***.

Embracing disciplined engineering methods and tools will lead to well-engineered, high performance DevOps.

This does NOT mean that only software engineers with an engineering degree should be doing the work of DevOps. I hope that is obvious, but I fear I better clarify that in case some readers think I am a snobbish engineering curmudgeon.

Software Engineering Institute Capability Maturity Model

Step by step improvement is the proven engineering way to achieve mature performance of any system.



Engineering DevOps Maturity Levels

1. Know your current state of DevOps maturity

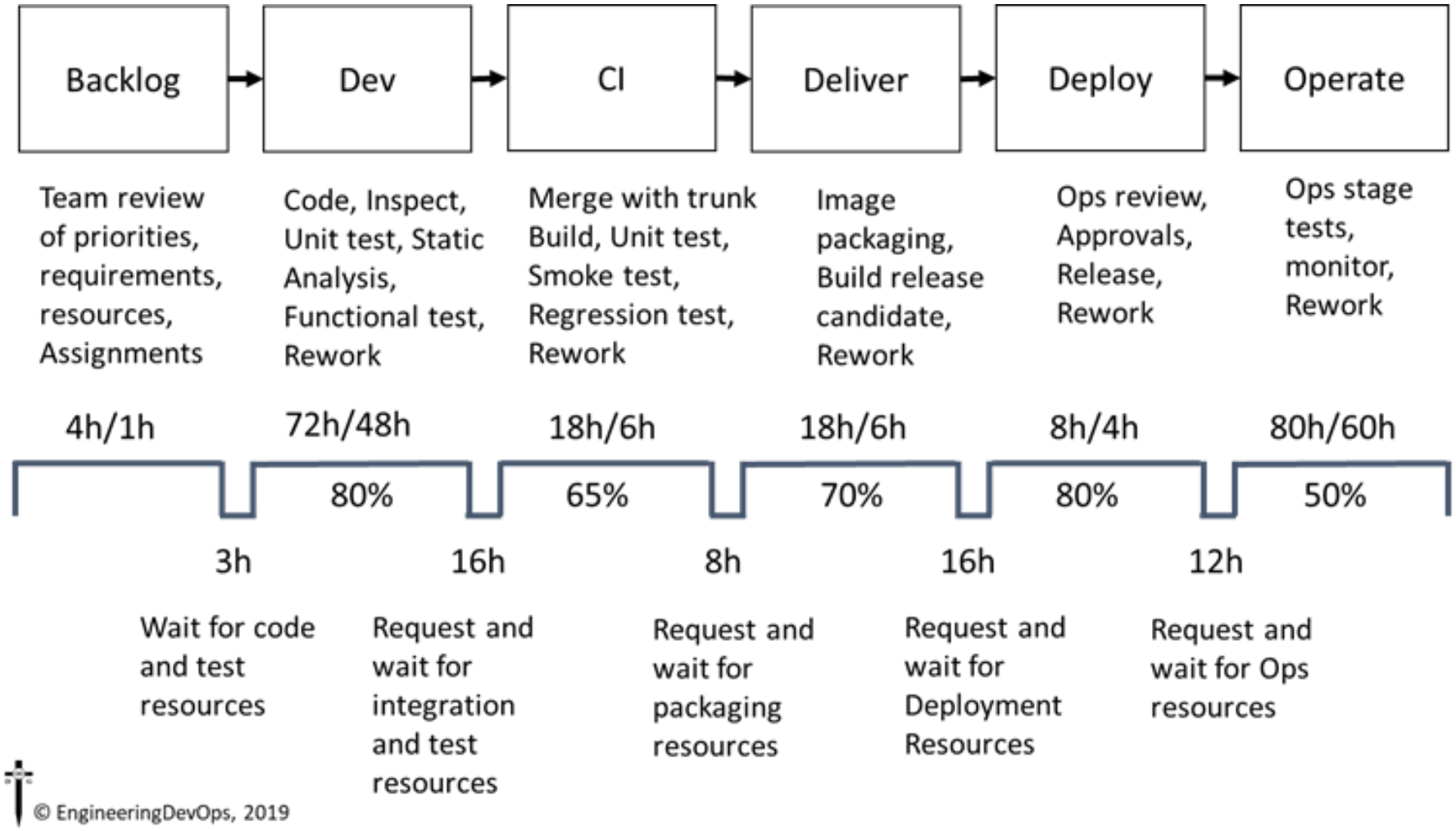
and

2. Determine desired NEXT future state of DevOps maturity

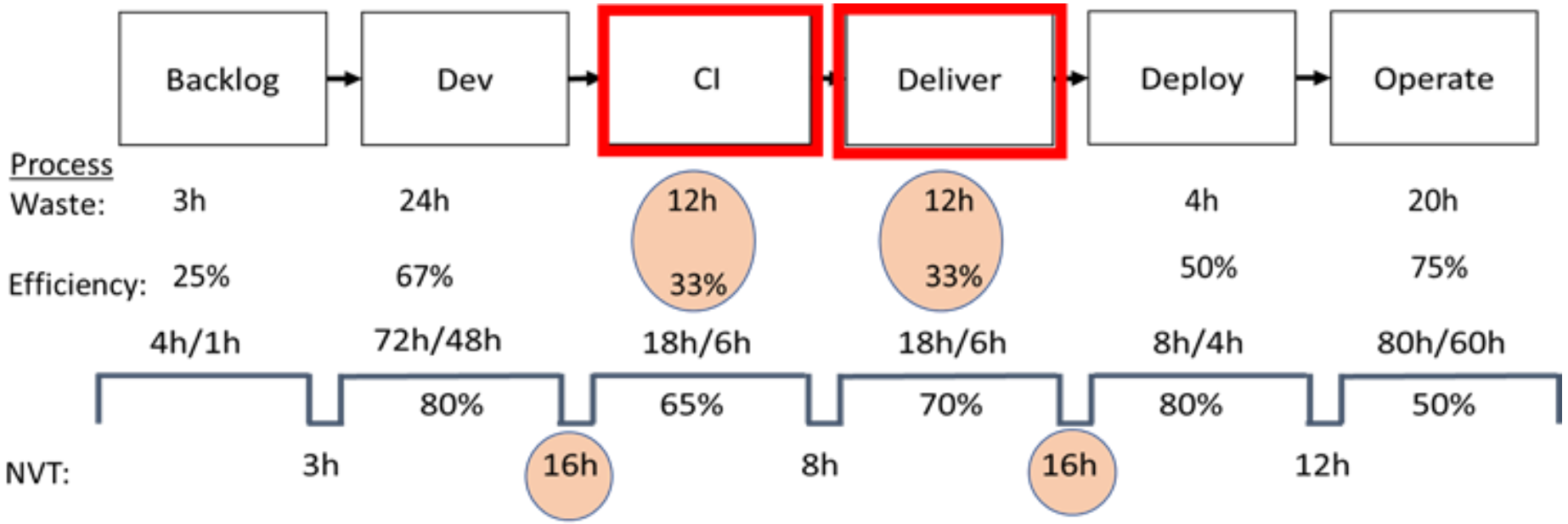
3. Pick solution components that get you to NEXT without waste.

Maturity Level	People	Process	Technology
Chaos	• Silo team and organization with little communication between silos • Blame and finger-pointing • Dependent on experts	• Requirements, planning and tracking processes poorly defined and operated manually • Unpredictable and reactive	• Manual builds and deployments • Manual quality assurance • Environment inconsistencies
Continuous Integration	• Managed communications between silos • Limited knowledge sharing • Ad hoc training	• Processes defined within silos • No standards for end-to-end processes • Can repeat what is known but can't react to unknowns	• Source code version management • Automated builds, release artifacts,& automated tests • Painful but repeatable releases
Continuous Flow ("First Way" of DevOps)	• DevOps leadership • Collaboration between cross-functional teams • DevOps training program	• End-to-end pipeline automated • Standards across the org for applications, releases processes and infrastructure	• Toolchain orchestrates and automates builds, tests and packaging deliverables. • Infrastructure is orchestrated as code. • Automated metrics and analysis for release acceptance and deployment.
Continuous Feedback ("2nd Way" of DevOps)	• Collaboration based on shared metrics with a focus on removing bottlenecks • SLIs, SLOs and SLAs • DevOps Mentors and Guilds	• Pro-active monitoring • Metrics collected and analyzed against business goals • Visibility and repeatability	• Applications, pipelines and infrastructure fully instrumented • Metrics and analytics dashboards • Orchestrated deployments with automated rollbacks
Continuous Improvement ("3rd Way" of DevOps)	• Culture of continuous experimentation and improvement	• Self service automation • Risk and cost optimization • High degree of experimentation	• Zero downtime deployments • Immutable infrastructure • Actively enforce resiliency by forcing failures

Value-Stream Mapping to Assess State



Example Value Stream Map – For Lead time / Agility

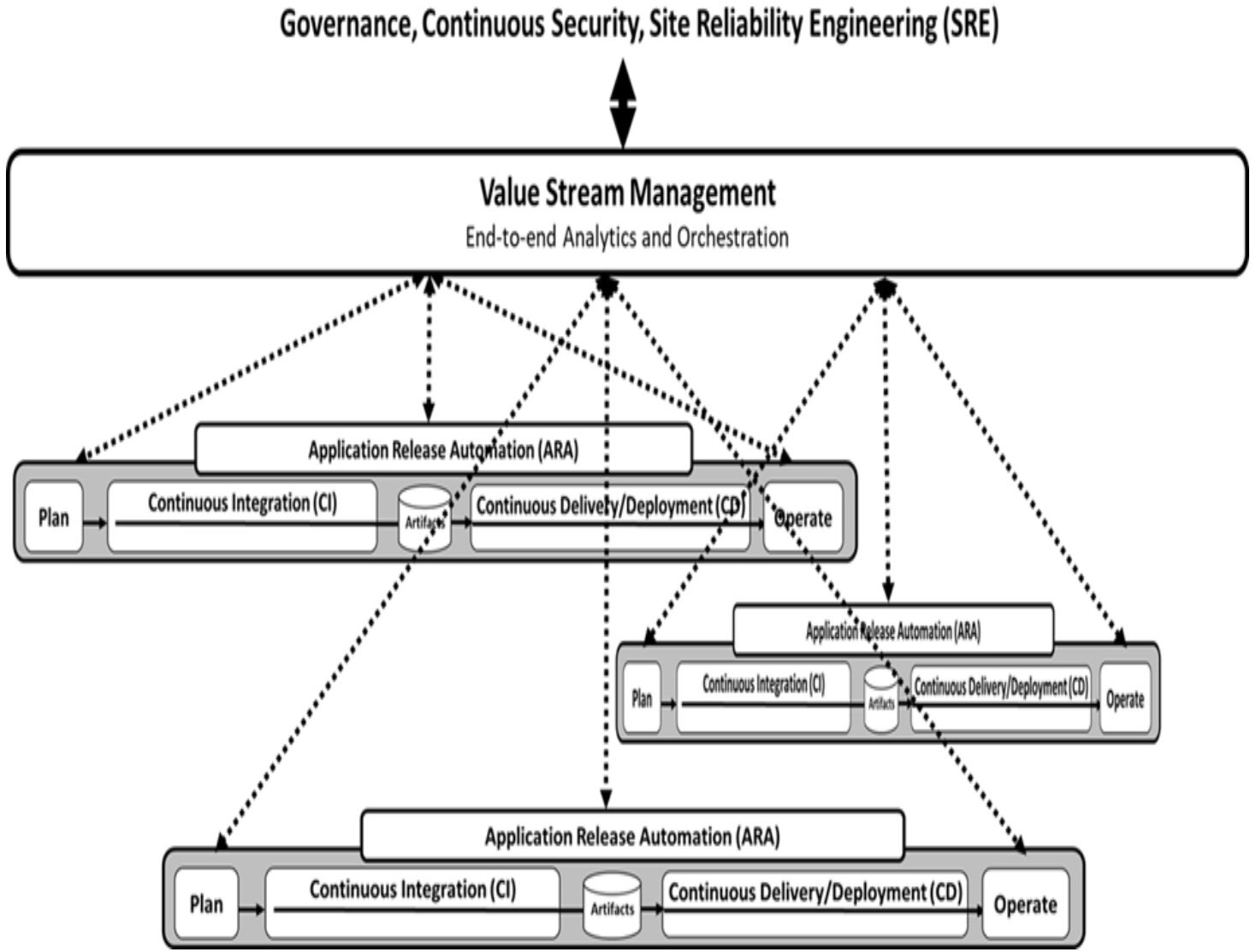


End to end lead time = Sum of LT values + NVT values = 255h
Total Waste = Sum of Process Waste + NVT values = 130h = 49%
End-to-end efficiency = (255-130)/255 = 49%
Improvement (potential) = (51%)

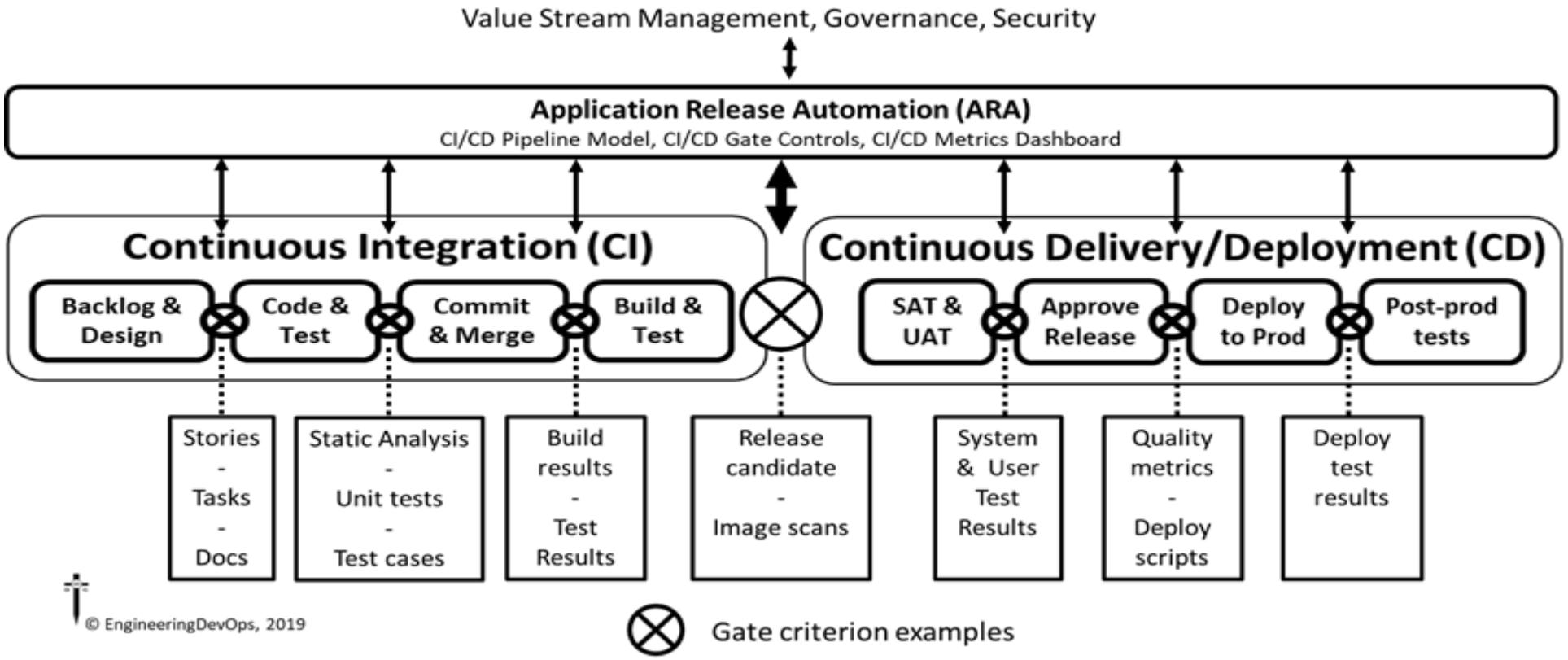
Value Stream Management (VSM) Blueprint

VSM is the top of the DevOps Engineering Blueprint

- Integration and Common Data Model
- Management, coordination & Orchestration
- Decision-making & Management of ***portfolios*** of value-streams
- Governance



Application Release Orchestration and Automation (AROA) Bluepr



Pipeline flow control, **CI/CD pipeline stages model**, **deploy application packages**, **consistent pipeline automation**, Dashboard and API make **metrics visible** and **release coordination** with humans, integrate with value stream management and governance systems

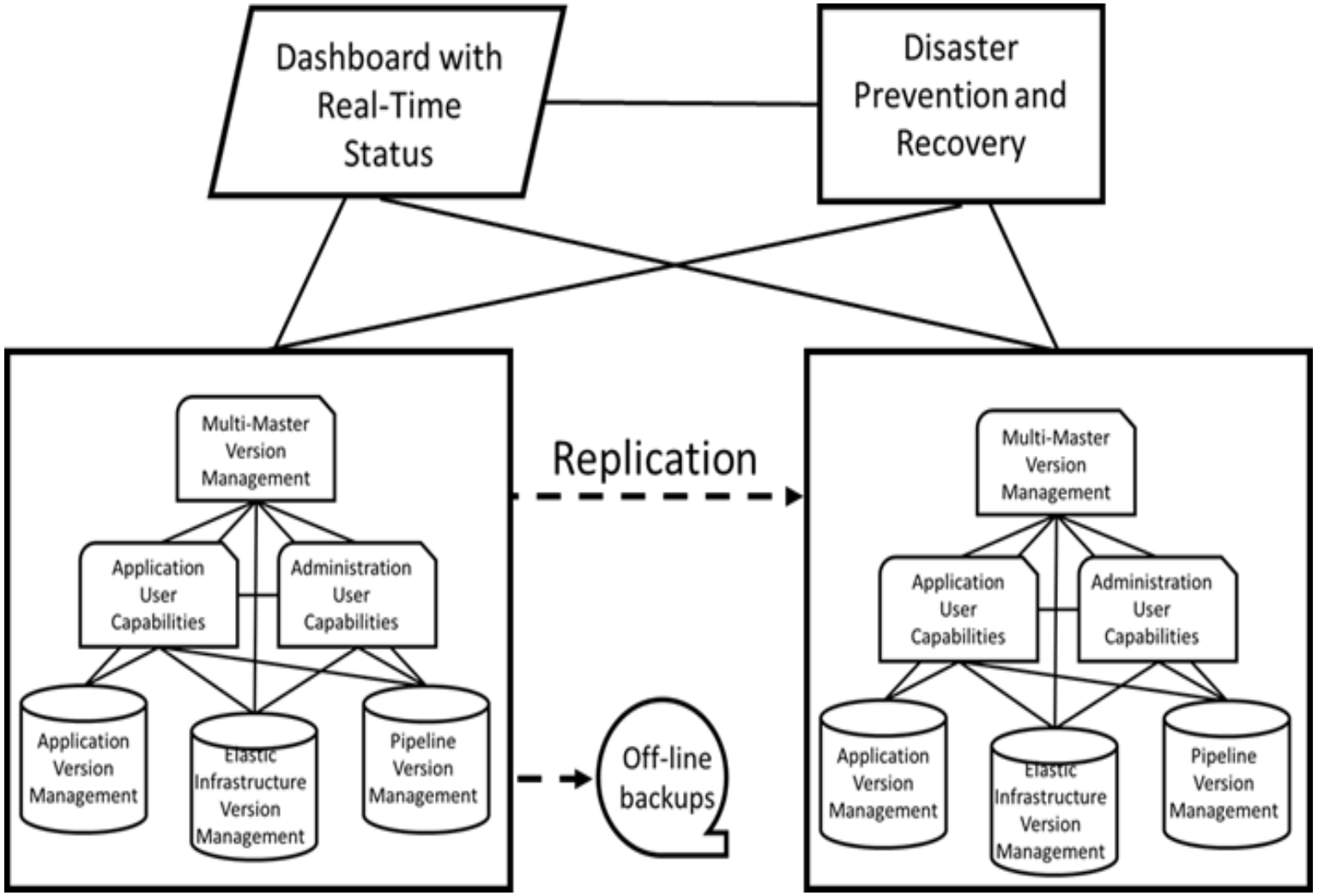
DevOps Version Management Blueprint

Versions of:

- **Application**
- **Infrastructure**
- **Pipeline**

User Capabilities:

- **Administration, B/F Rolls, DR**
- **Developers, QA, Ops**



Continuous Security (DevSecOps) Blueprint

Planning

Security requirements from business functionality, Security user stories in backlog

Tools: Cyberplanner, Service Now, Jira

VSM

End-to-end value stream security governance-policies-as-code, analytics

Tools: Plutora

ARA

Pipeline entry/exit gate policies as code, release dashboard

Tools: CloudBees & ElectricFlow

Infrastructure

Infrastructure-as-Code, network security, access security

Tools: F5, Nagios, ZenMap, CyberArk, Vault, Identity Finder

Post-prod tests

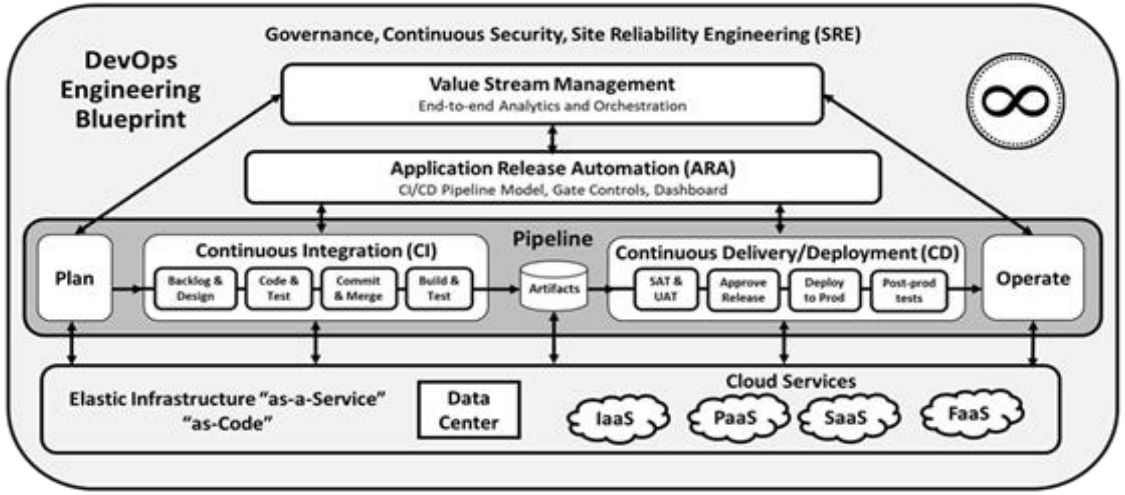
Secrets Management, Authorization and logging, Repeatable Execution

Tools: CyberArk, Vault,

Design

Threat Modeling, Secure by Default Designs (security services, frameworks), Opensource scanning

Tools: Continuum Security, ThreatModeler, Blackduck



Operate

Chaos Engineering, Social engineering attacks, in-operation vulnerability scans

Tools: Chaos Monkey, F5, Immunio, Retina, SET

Code & Test

Development Standards, Peer Review, Static Code Analysis, Unit Tests, Software supply chain, Data masking, Web code scanning

Tools: Veracode, MSSQL Data Mask, Probely, Blackduck

Deploy to Prod

Secrets Management, Dark launches, Penetration testing, SQL discovery, DB encryption

Tools: LaunchDarkly, SQLRECON, DbDefence, CyberArk, Vault, Metasploit

Commit & Merge

Version management role-based access, Validate pre-flight security results were done in pipeline prior to code commit

Tools: Git, Datical, DBMaestro, AppDetectivePRO

Build & Test

Dynamic Security Testing, Containers security , SQL Fault injection, dependencies

Tools: Contrast, Veracode, Aqua Security, BSQL Hacker, Dependency-check, Twistlock

Artifacts

Deliverable image scanning, white lists and black lists

Tools: XRay

SAT & UAT

Test automation, Web vulnerability testing

Tools: Gauntlt, Acunetix

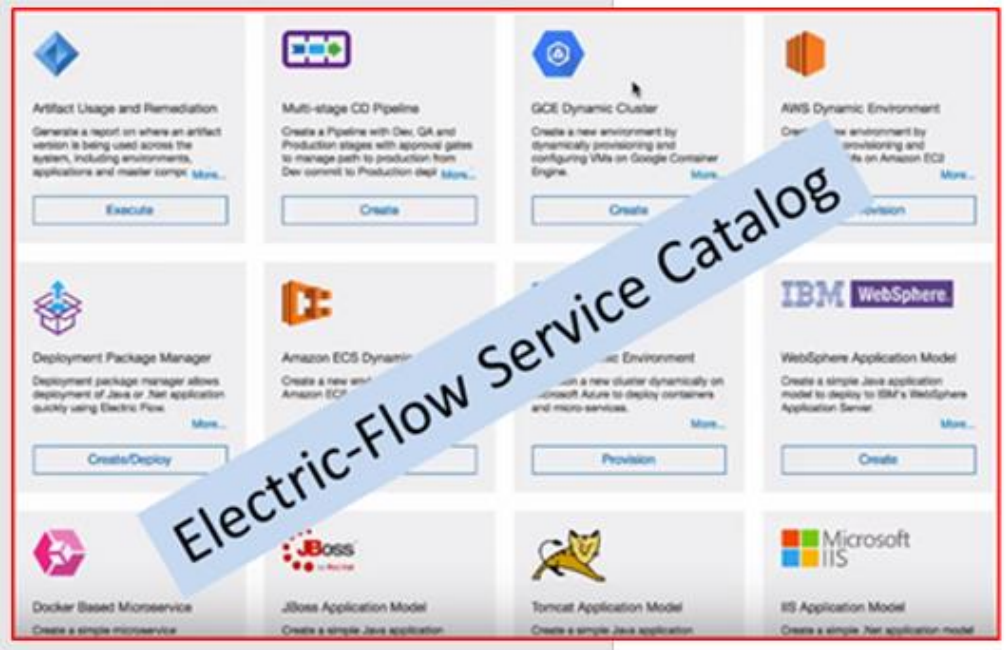
Approve Release

Validate release meets all security policies

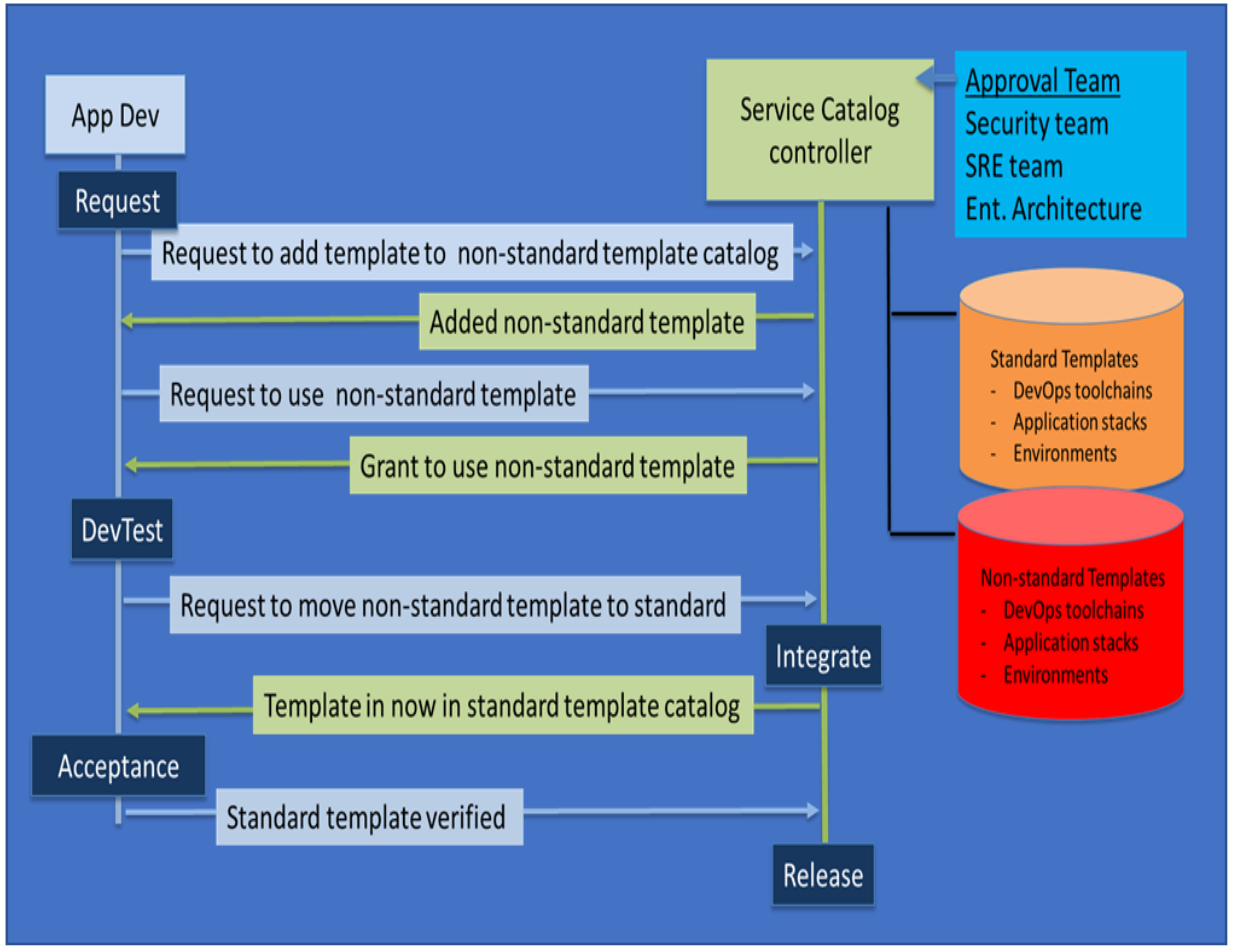
Tools: CloudBees & ElectricFlow

DevOps Service Catalog

Governance through control of choices



Controlled –flexibility workflow



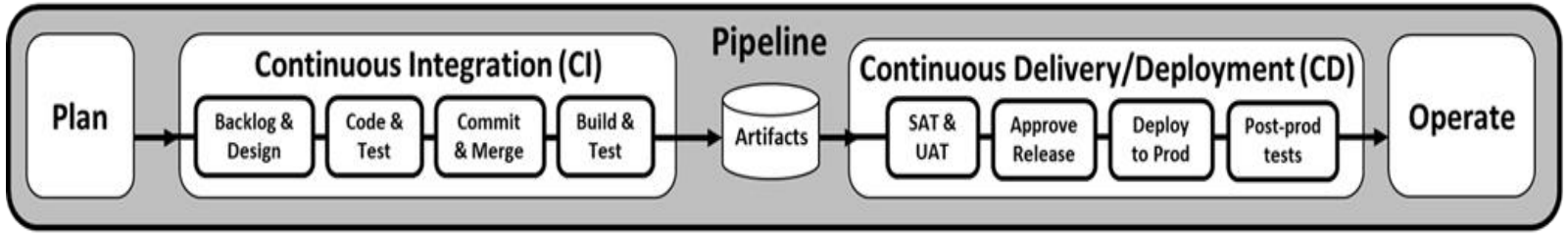
Site Reliability Engineering (SRE) Blueprint

- Assure availability of apps and services
- Assure error rate and latency meet (SLAs)
- Enable large-scale while controlling risk
- Operational Cost savings by automation
- Improved skills of developers and SREs
- Resolves conflicts between Dev and SRE
- Improved capacity planning

SRE
Response automation
Post mortems
Monitor events, alerts
Capacity Planning
Focus: Reliability

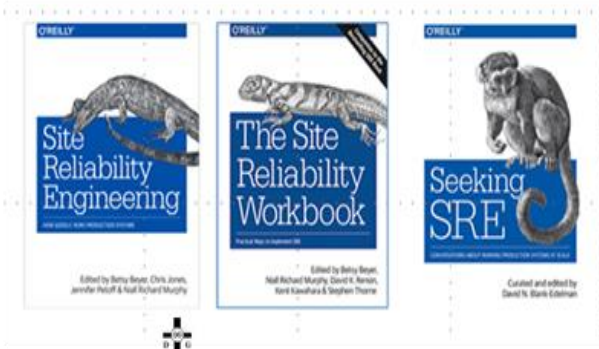
Use Cases
Report on issues
System Diagnostics
Track SLA/SLOs
Identify optimizations
Ensure reliable releases

SLAs/SLOs
End-user availability
Latency
Error rate
End-user response time
Correctness



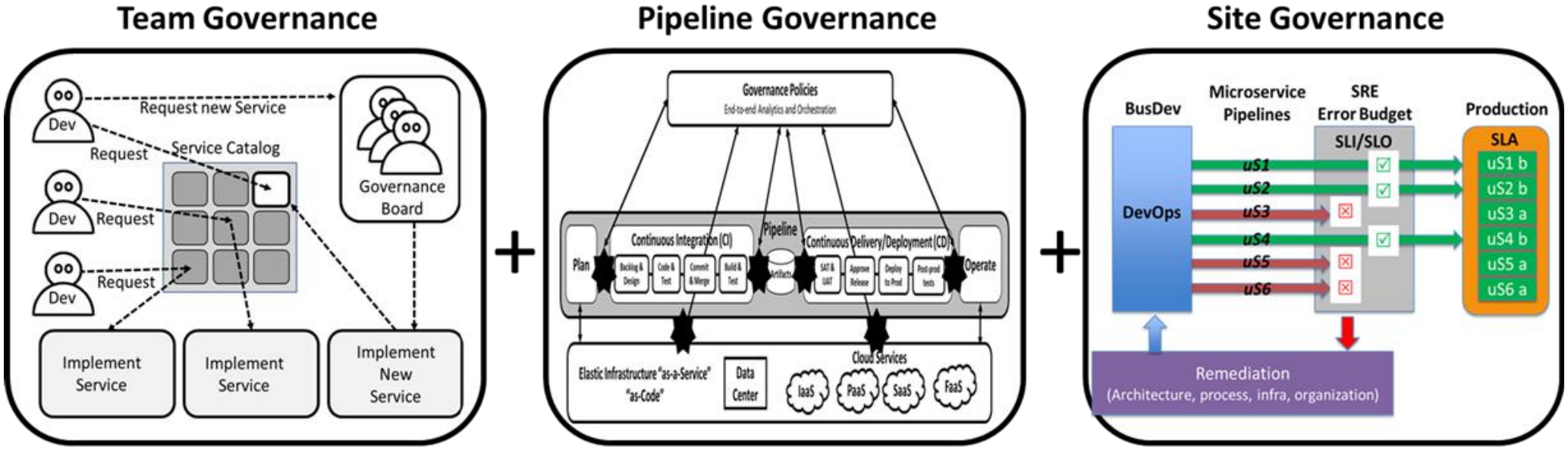
Dev
Release automation
Environment builds
Config Management
Infra—as-code
Focus: Delivery Speed

SLIs
Number of incidents
MTTR
MTBF
Deploy frequency
Revenue



© EngineeringDevOps, 2019

DevOps Governance Blueprint



Continuous Flow
(DevOps First Way)

Continuous Feedback
(DevOps 2nd Way)

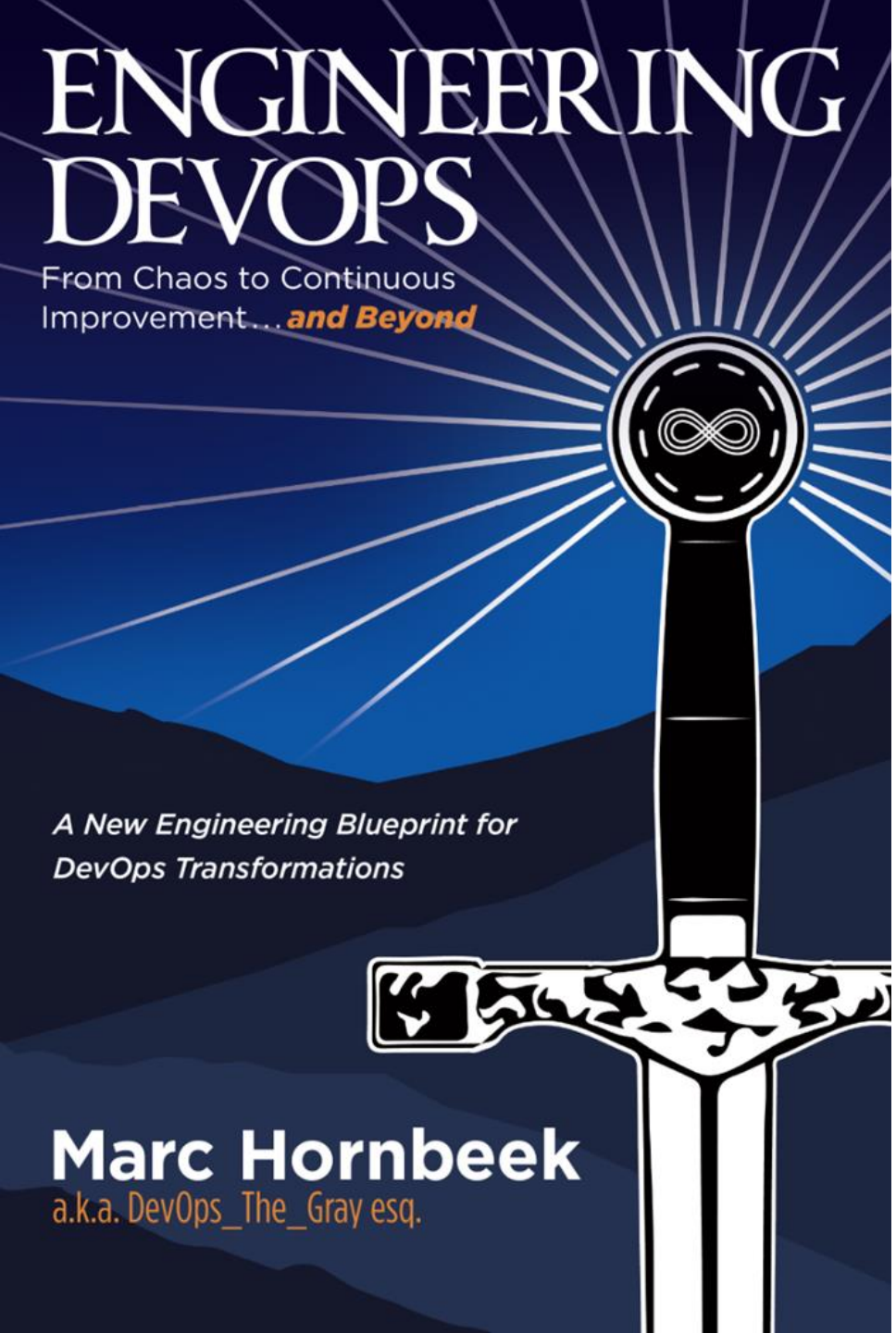
Continuous Improvement
(DevOps 3rd Way)

A sound DevOps *governance* strategy should cover organizational, financial and operational requirements. The process must be designed to be repeatable, as the DevOps journey is not just a one-off deal, but an ongoing process to facilitate continuous improvement.

DevOps Disaster Mitigation and Recovery Blueprint

Well-engineered DevOps environments are designed for disaster mitigation and recovery.

Nine Pillars of DevOps (Cause)	Application Disaster	Pipeline Disaster	Infrastructure Disaster
Leadership	Example: App crash w/bad release approval Mitigation: Version Management with VSM Recovery: Roll-back to last good App version	Example: Pipeline crash w/bad change approval Mitigation: Version Management, with VSM Recovery: Roll-back to last good Pipeline version	Example: Infra crash w/bad change approval Mitigation: Version Management, with VSM Recovery: Roll-back to last good Infra code version
Collaborative Culture	Example: App crash w/ poor change coordination Mitigation: Dependencies Version Management Recovery: Roll-back to last good App version	Example: Pipeline crash w/ poor change coordination Mitigation: Dependencies Version Management Recovery: Roll-back to last good Pipeline version	Example: Infra crash w/ poor change coordination Mitigation: Dependencies Version Management Recovery: Roll-back to last good infra code version
Design For DevOps	Example: Monolith App crash during operations Mitigation: Redundant deployments Recovery: Switch to last good version of App	Example: CI crash w/application file corruption Mitigation: Redundant image repository Recovery: Switch to last good back-up file	Example: Infra crash w/config code file corruption Mitigation: Redundant image repository Recovery: Switch to last good back-up file
Continuous Integration	Example: App crash w/coding error Mitigation: Static analysis checks Recovery: Roll-back to last good App version	Example: Bad CI automation script breaks CI process Mitigation: Version management Recovery: Roll-back to last good Pipeline version	Example: Infra crash with bad config integration Mitigation: Version management Recovery: Roll-back to last good Infra code version
Continuous Testing	Example: App crash w/test exception Mitigation: Test validation Recovery: Roll-back tests to last good version	Example: Pipeline crash w/test exception Mitigation: Test validation Recovery: Roll-back tests last good Pipeline version	Example: Infra crash w/test exception Mitigation: Test validation Recovery: Roll-back tests last good Infra version
Continuous Monitoring	Example: App crash w/performance exception Mitigation: APM design validation Recovery: Roll-back APM config to good version	Example: Pipeline crash w/performance exception Mitigation: APM design validation for Pipeline Recovery: Roll-back APM config to good version	Example: Infra crash w/performance exception Mitigation: APM design validation for Infra Recovery: Roll-back APM config to good version
Elastic Infrastructure	Example: App crash due to failed cluster Mitigation: Load balancer Recovery: Restrict load and add clusters	Example: Pipeline crash due to failed servers Mitigation: Load balance Recovery: Restrict load and add clusters	Example: Infra crash due to network failure Mitigation: Chaos monkey, alternate networks Recovery: Switch to alternate network
Continuous Security	Example: Cyber attack takes down Application Mitigation: Continuous security best practices Recovery: Recover data and deployments	Example: Hacker takes down Pipeline Mitigation: Continuous security best practices Recovery: Reset and restore pipeline	Example: Hacker takes down Infrastructure Mitigation: Continuous security best practices Recovery: Reset and restore infrastructure
Continuous Delivery	Example: New version of App fails in production Mitigation: Dark launches of App releases Recovery: Roll-back to last good App version	Example: New version of Pipeline fails Mitigation: Dark launches of Pipeline releases Recovery: Roll-back to last good Pipeline version	Example: New version of Infra fails in production Mitigation: Dark launches of Infra code Recovery: Roll-back to last good Infra version



How do you Engineer Applications, Pipelines and Infrastructures for DevOps?

Application DevOps Suitability Checklist

DevOps applies to all types of Apps: Legacy, COTs, Enterprise, embedded, IOT, platforms, services, netops, etc.

Business Factors determine whether it makes sense to engineer DevOps for an application more than technology differences between different types of applications.

- The application will benefit from faster lead times.
- Leaders over this application are open to collaboration and will be sponsors of change.
- Team players that are associated with the application team (Product owners, Dev, QA, Ops, Infra, Sec, PM) are open to collaboration and change.
- The application is currently using, or planning to use, service-oriented, modular architectures or micro-services. RB14, RB15, RB16
- There are at least ten people associated with the application including Product owners, Dev, QA, Ops, Infra, Sec, and PM. Smaller projects with smaller teams may not show sufficient impact or justify substantive DevOps investment.
- The application is expected to be undergoing changes for more than a year. Less time would not likely yield return on the investment required for DevOps.
- The application represents a good level of business impact and visibility but does not involve an extreme amount of risk for the business.
- The application experiences frequent demands for changes from the business.
- Not all tools used with the application need to be replaced to implement a DevOps toolchain. The costs of changing out all tools can negatively affect the return on investment and business case for DevOps transformations.
- Efforts to build, test or deploy releases of the application is significant and could be reduced significantly by automation.

Application Design for DevOps Recommended Engineering Practices

- Designers must understand customer use cases.
- NFRs - usability, reliability, scaling, availability, testability and supportability are more important than features! Leaders support designers with motivation, mentoring and training. No designer can be expected to know everything.

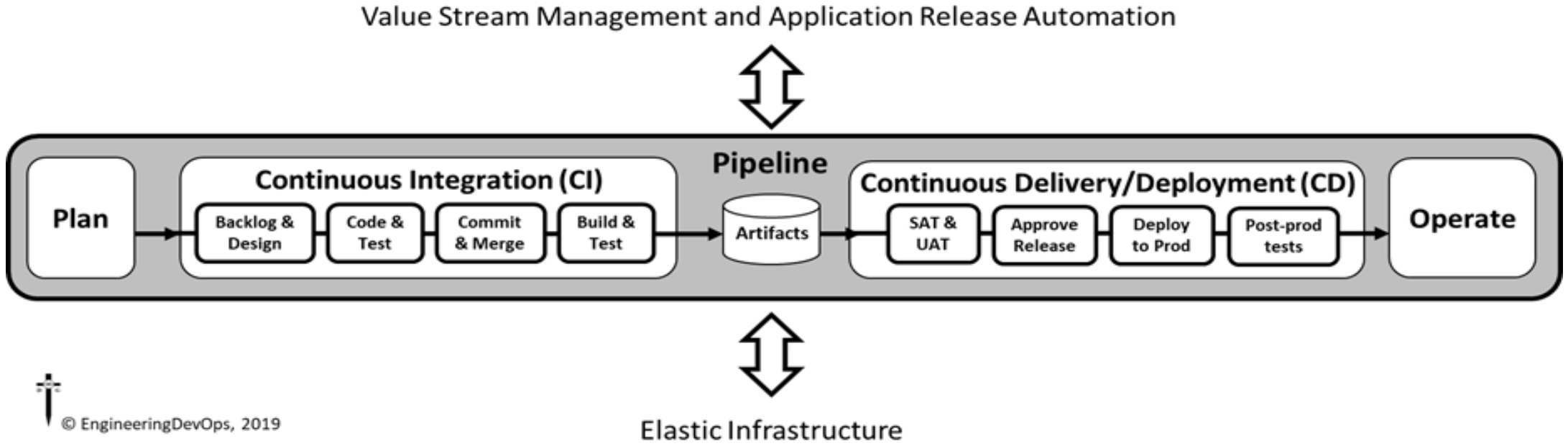
- Mistakes are ok if lessons are learned and improvement quickly follows. Continuous monitoring and quick remediation help minimize the impact.
- Design coding practices are critical: 12 factor apps, modular, etc.
- DevOps design practices support QA and Ops
- DevOps tools and infrastructure are provided as a service to designers.

Five Levels of Application Maturity

1. **Spaghetti Applications:** older monolithic legacy applications - large and complex like a big messy bowl of spaghetti.
2. **Modular Applications:** separate functionally independent, interchangeable modules.
3. **Cloud-Ready Application:** use ***The Twelve-Factor Application*** modular design methodology; declarative formats for automation; clean contracts with the operating system; and portability.
4. **Cloud-Optimized Applications:** apps take advantage of services offered by cloud providers; portable to deploy in *IaaS* or *PaaS*. Easily scale-in and scale-out for optimal utilization of resources.
5. **Cloud-Native Applications:** are *born in the cloud*. They exploit capabilities such as elasticity, event-driven, resource optimization, and faster release cycles; designed as *Federated Microservices*, packaged and deployed as containers, and managed through modern DevOps processes. The target deployment environments include *Containers -as-a-Service (CaaS)* and *Functions-as-a-Service (FaaS).* This breed of apps delivers ultimate scalability and availability.

Applications at different maturity levels will co-exist and interoperate. **Don't trick yourself into a microservices-*only* mindset.** Explore the modularity features or frameworks of your technology stack to enforce modular design, instead of relying on conventions to avoid spaghetti code. **Choose whether you want the complexity penalty of microservices.**

CI/CD Pipeline Blueprint

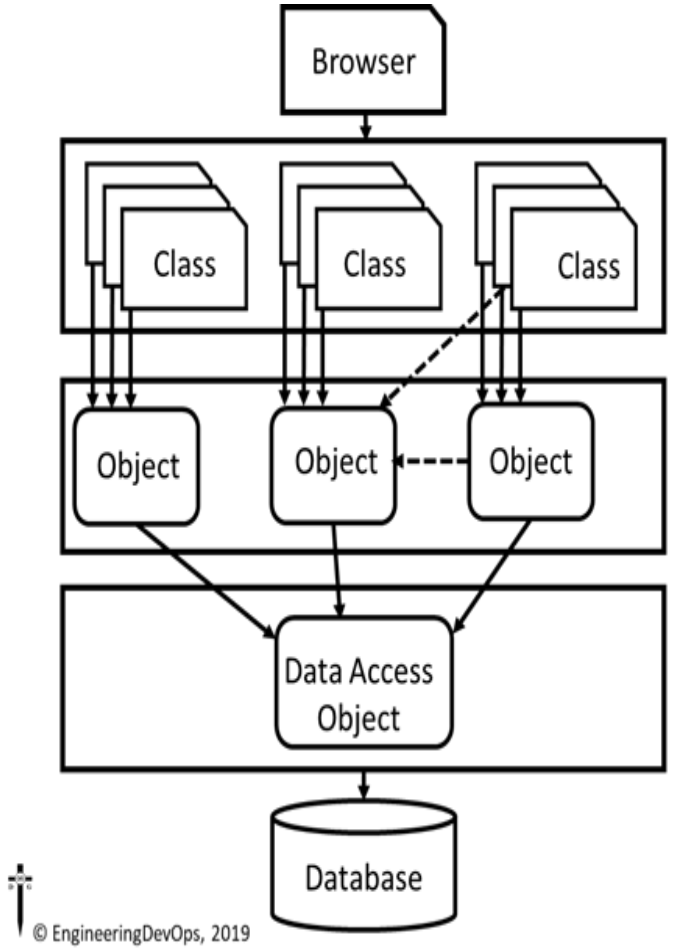


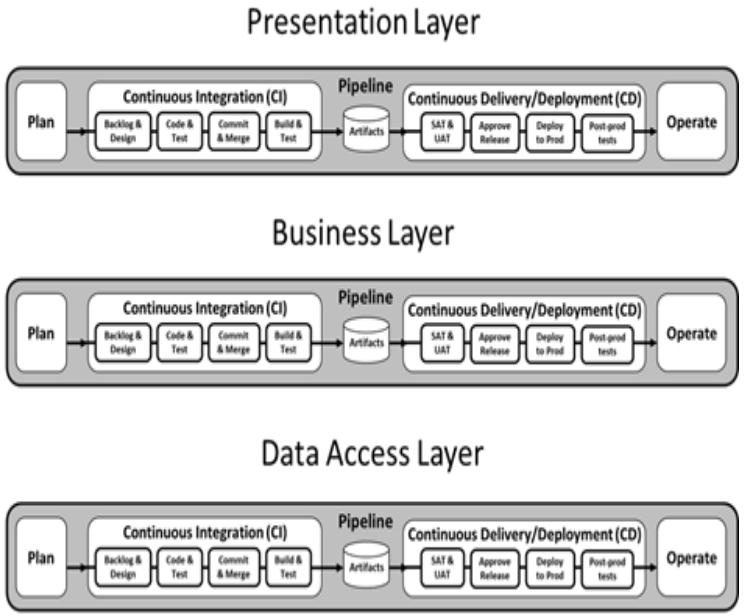
CI/CD pipelines, also known as "***continuous delivery pipelines***", constitute the core of the DevOps engineering blueprint.

CI/CD pipelines, in the world of DevOps, refers to the idea that software changes "flow" through a series of stages, from the start of the software Planning process until it is delivered to the Operations end.

CI/CD Pipeline for N-Tier App Blueprint

- **Multiple parallel *Continuous Delivery* pipelines** deliver applications that are partitioned in layers, tiers or modules that have separate teams and separate workflows.
- ***Synchronizing multiple pipelines*** to make an application *release* is considerably challenging.
- ***Choose tools and tool frameworks*** strategically to ensure the separate pipelines can be co-ordinated as *choreographed federations*.

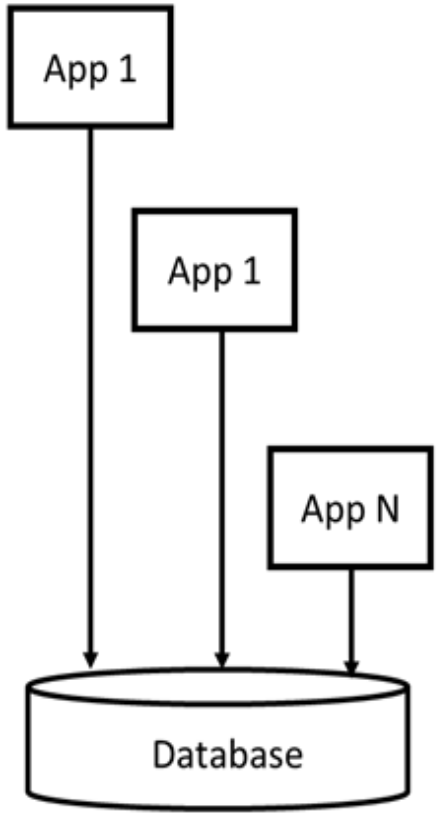


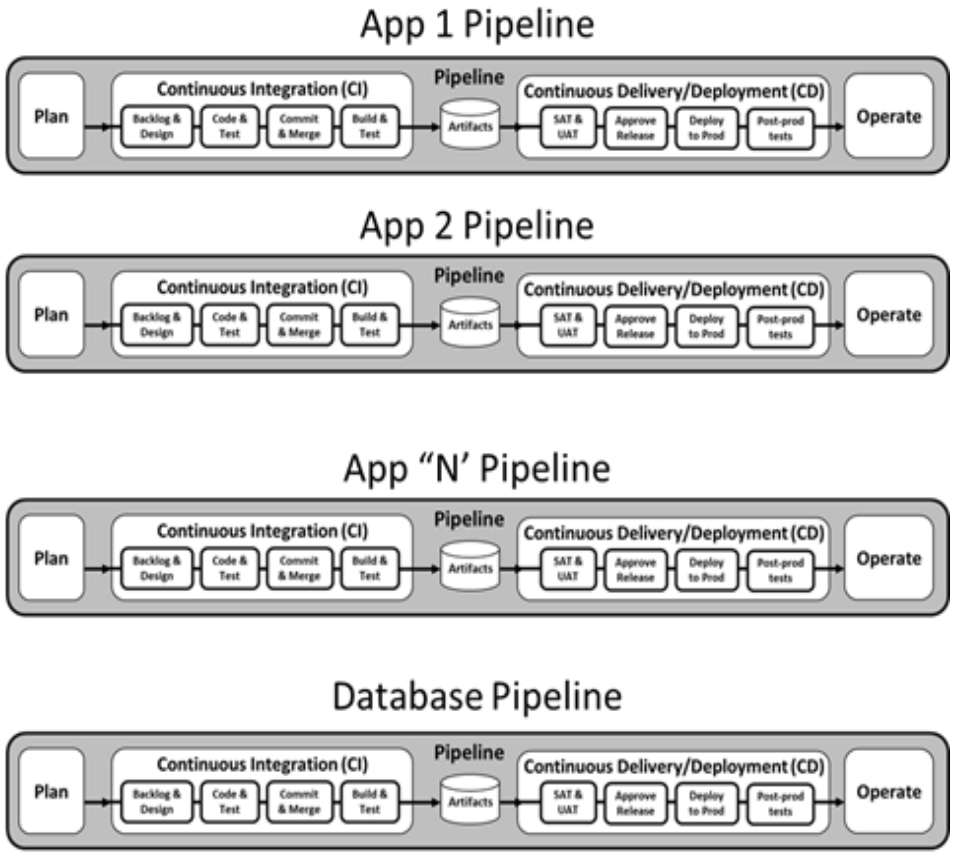


CI/CD Pipeline for Databases Blueprint

Database changes used by several applications must be co-ordinated with applications that depend on the *database* structure.

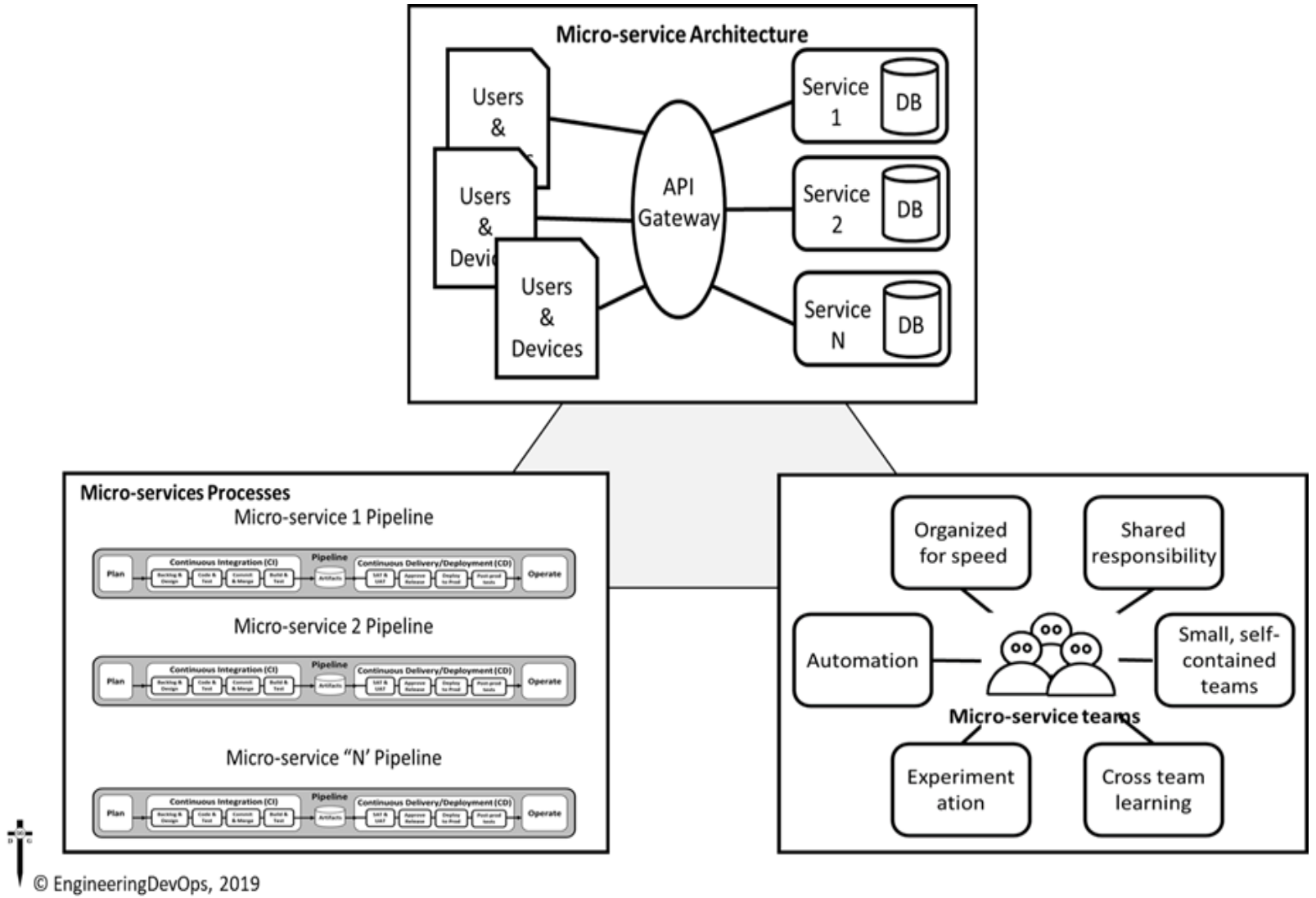
Handling failures in *database* code need a ***database pipeline*** that can orchestrate ***database roll-backs and forward deployments*** in concert with application pipeline versions.





CI/CD Pipeline for Microservices Blueprint

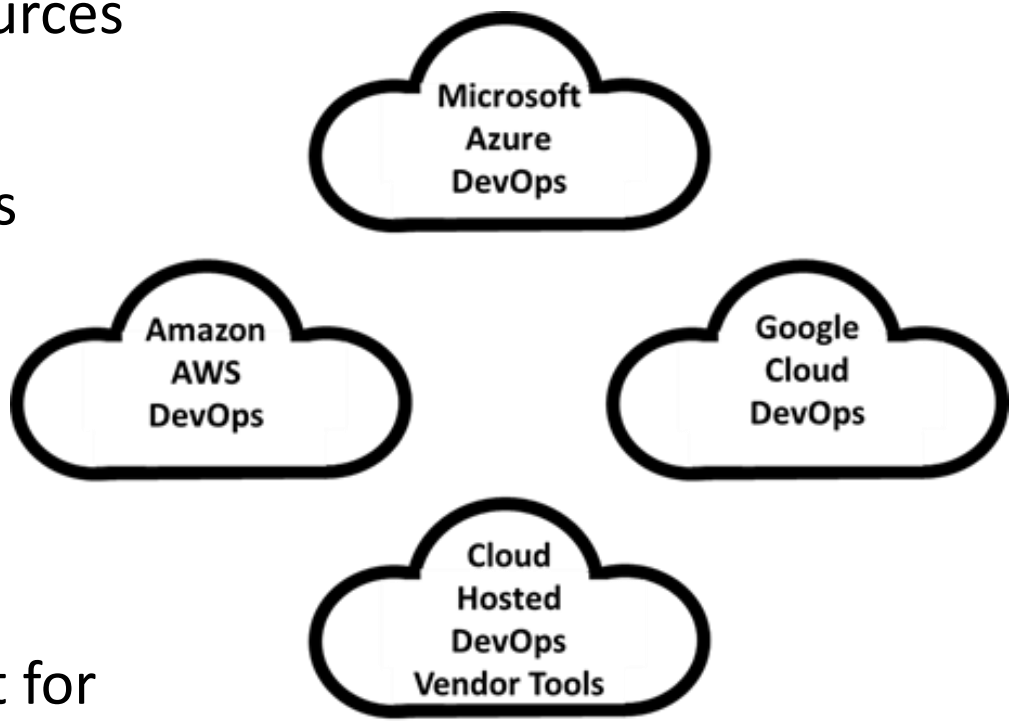
Microservices require DevOps *CI/CD pipelines* to work as a federation together in a choreographed concert with the microservice architecture of the application and the organizations responsible for them.



CI/CD Pipelines in the Cloud

Benefits:

- Free to choose tools, tool frameworks and craft toolchains from any number of vendors or open sources
- Rapid feature evolution for specific vendor tools
- Support for Hybrid-Cloud and Multi-Cloud scenarios



Disadvantages:

- Lock-in to tool vendor solutions
- Need to create and maintain toolchain yourself
- Need to manage deployments and regional support for the toolchains
- Scaling of some tools may not be enough for some tools

Pipeline Maturity Levels

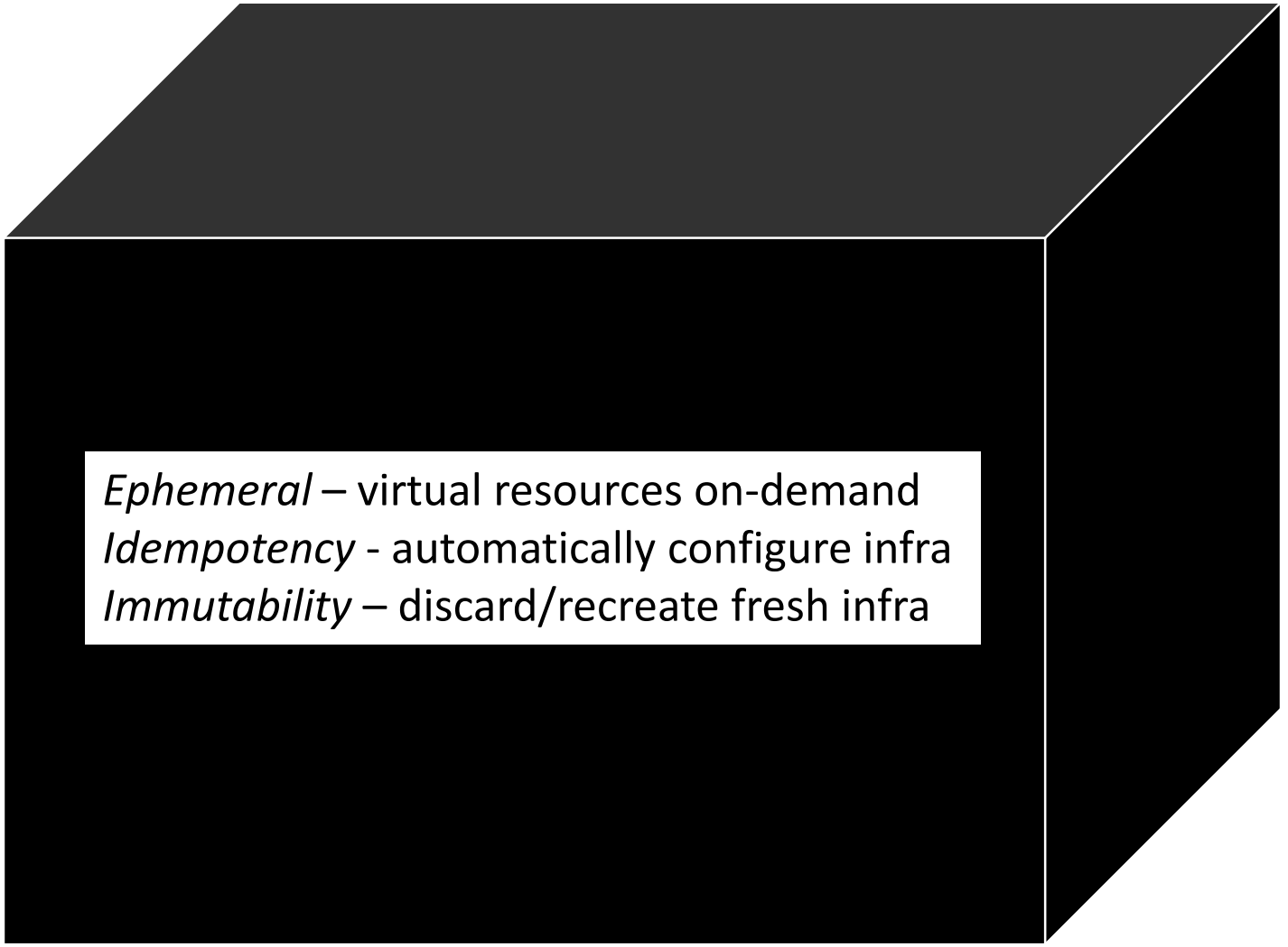
1. **Chaos Reigns:** the *value stream* does not have a fully connected pipeline or *toolchain* for CI or CD. Manual interactions cause bottlenecks and errors; low repeatability and reliability.
2. **Continuous Integration:** *CI* stages are supported by an operational *toolchain*. Developers commit changes; the system automates integration, builds, tests, artefacts and flags failures.
3. **Continuous Flow (*The First Way* of DevOps):** CI and CD pipeline stages have *toolchains*. *Release* candidates are prepared and qualified for deployment with minimal manual effort.
4. **Continuous Feedback (*The Second Way* of DevOps):** *CI/CD pipeline* is instrumented with metrics that make visible performance of the pipeline, the applications and the infrastructure.
5. **Continuous Improvement (*The Third Way* of DevOps):** *CI/CD pipeline* is reliable, fast and uses advanced DevOps practices for advanced *continuous testing*, *continuous security*, and advanced *continuous deployment*. The robust, sophisticated pipelines provide confidence for proactive experimentation with new pipeline tools, integrations and workflows while managing risk.

DevOps Infrastructure Blueprint

DevOps needs infrastructure to run application and pipelines - *obviously*!

Developers prefer not to have to think about infrastructures or CI/CD pipelines.

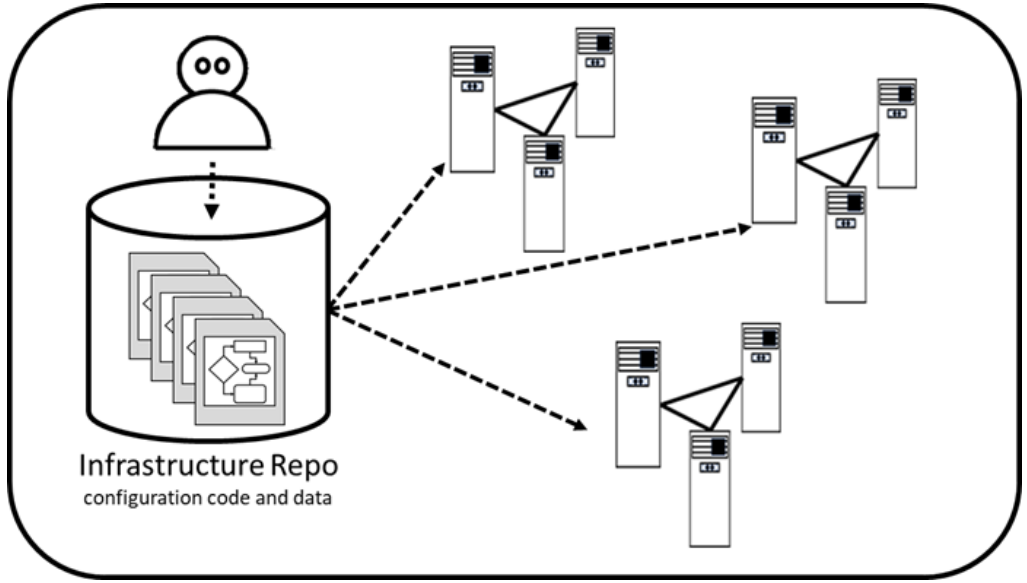
They have enough to do developing and operating their applications.



Infrastructure as Code (IAC)

DevOps *IAC* automation use cases:

- *Pre-Flight* test environment
- Staging automation
- Deployment automation
- Safety: *Green/Blue* deployments
- Quality: *A/B testing* methodologies
- Restore and Recovery
- Cost control
- Utilization
- *Governance*

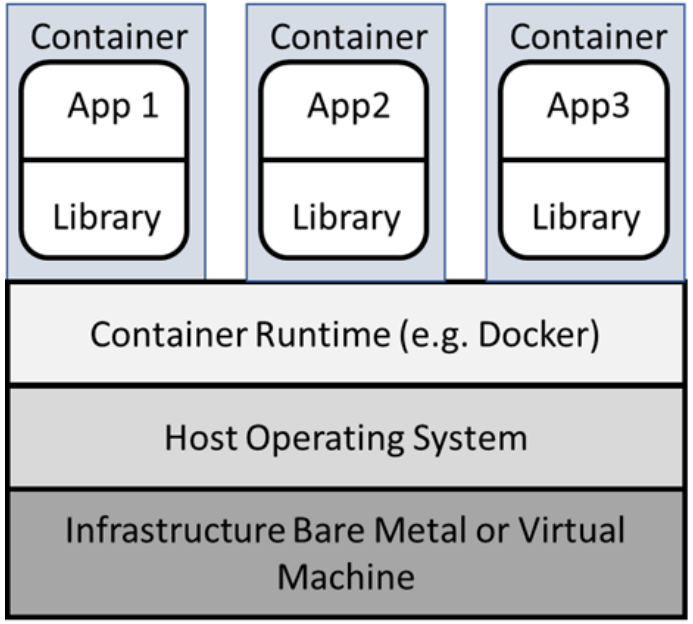


- *Orchestration* – create and release resource
- *Automation* – execute tasks
- Ease of use and Admin - unified practices and tools
- Speed - deliver stable environments rapidly
- Scale - vertically and horizontally on-demand
- Repeatability - resolves environment drift
- Maintenance: version manage infra code and data

Containers and Serverless Infrastructures are Preferred for DevOps

Containers:

- Small size allows quick deployment
- Portability across machines
- Easy to track and compare versions
- More containers than virtual machines can run simultaneously on a host machine.

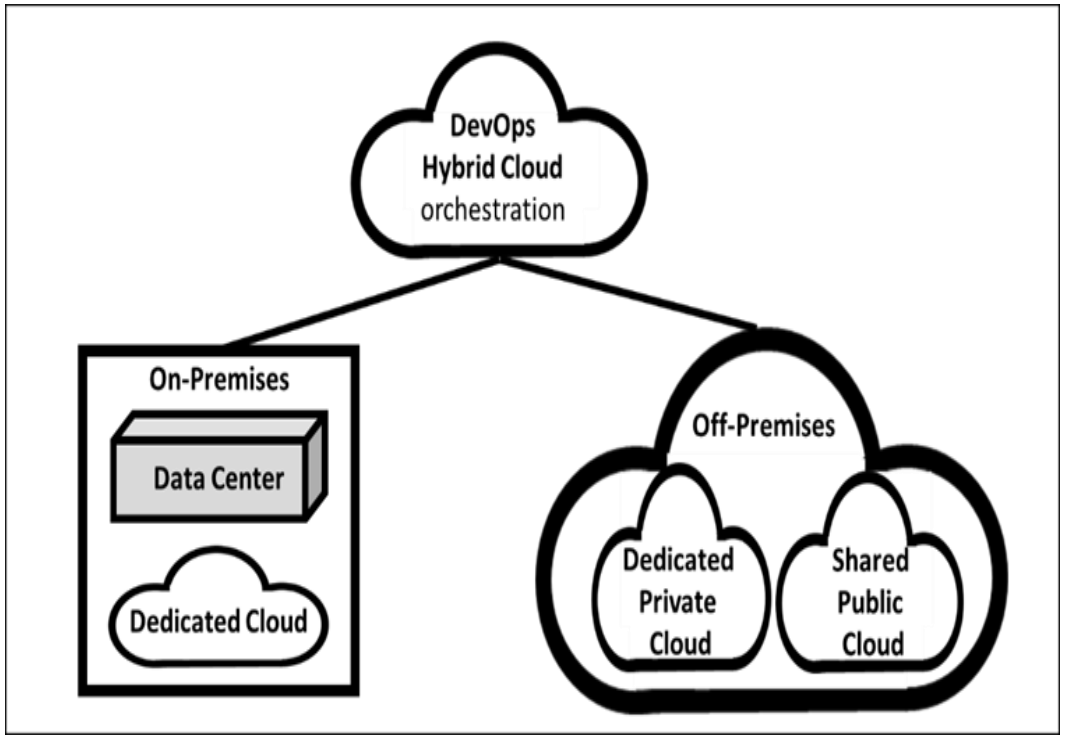


Serverless computing:

- Cloud provider runs the server, and dynamically manages resources
- Pricing is based on consumption
- Simpler deploy into production
- Scaling, capacity planning and maintenance operations may be hidden
- Applications can be written to be purely *serverless* and use no provisioned servers at all.

DevOps with Hybrid-Cloud Deployments

- ***DevOps-as-a-Service*** uses a mix of on-premises, private cloud and third-party, public cloud computing services.
- **Hybrid Cloud orchestration tools** abstract resources (E.g. VMs, containers, microservices and databases).
- **Hybrid cloud orchestration communicate** with the supplicant functions through an API framework.
- **Hybrid Cloud Orchestration stacks** that are supported are defined.
- **A dedicated reliable, high-speed / low-latency, redundant *WAN*** connection to the public cloud is installed.
- **Stakeholders have visibility** into the deployment infrastructure in the desired private and public clouds.
- **Tools help transition workloads** to the cloud by orchestrating virtual machine movement from a private environment to a public cloud.



DevOps Hybrid Cloud

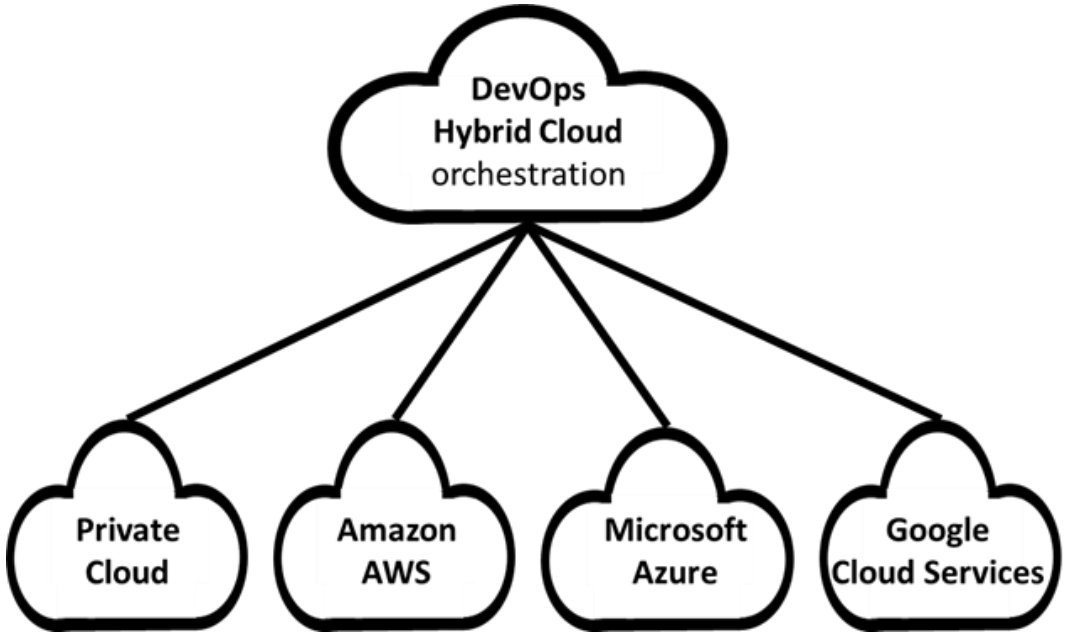
DevOps with Multi-Cloud Deployments

Multi-Cloud provides the ultimate DevOps infrastructure:

- Flexible, widest range of application deployments requirements.
- Enterprises that assemble harmonized *Multi-Cloud* platforms are positioned for competitive advantage and lower costs.

Cloud-Native is critical in a Multi-Cloud environment. Moving DevOps from a single or Hybrid Cloud to a Multi-Cloud world requires platforms and tools for capacity management and cost management.

A globally aware traffic management strategy monitors infrastructure health across data centers and end-user experiences globally, while responding to control changes and system specifications at the speed of DevOps Continuous Improvements.



DevOps Multi-Cloud

DevOps with Multi-Cloud *Services*

Multi-Cloud environments can be more complicated than managing a single *Cloud* environment. It is not a “write once, run everywhere” situation.

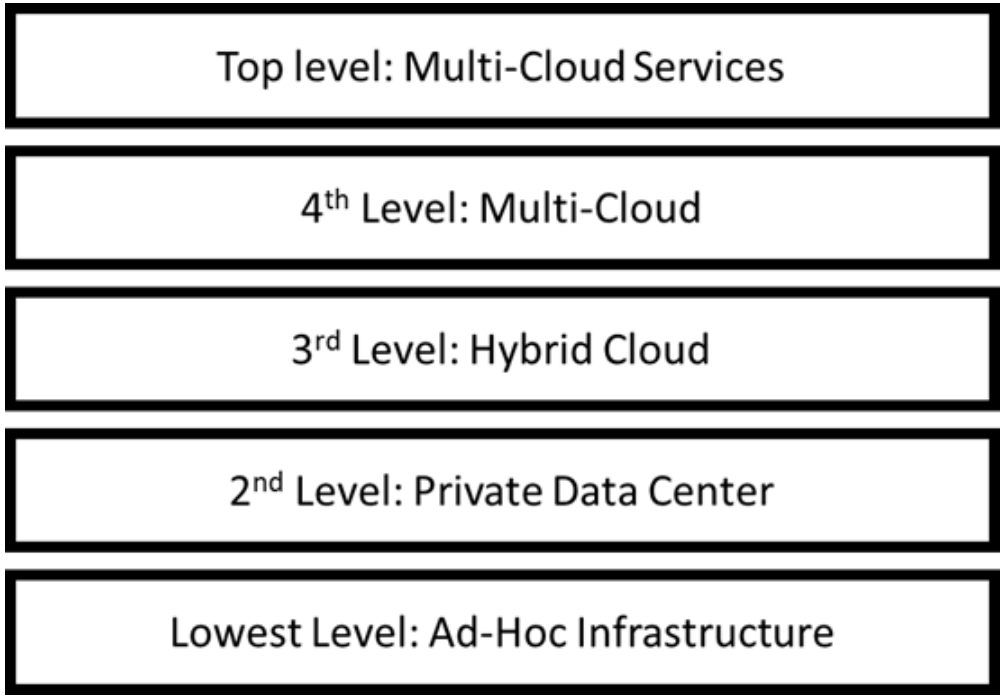
- CSPs have their own feature sets, GUIs and access protocols.
- A *Multi-Cloud* management system must do the work to provide deep integration with each CSP.
- A success strategy considers challenges, benefits and a clear path to successful *Multi-Cloud* management.

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Virtual Servers	Instances	VMs	VM Instances
Platform-as-a-Service	Elastic Beanstalk	Cloud Services	App Engine
Serverless Computing	Lambda	Azure Functions	Cloud Functions
Docker Management	ECS	Container Service	Container Engine
Kubernetes Management	EKS	Kubernetes Service	Kubernetes Engine
Object Storage	S3	Block Blob	Cloud Storage
Archive Storage	Glacier	Archive Storage	Coldline
File Storage	EFS	Azure Files	ZFS / Avere
Global Content Delivery	CloudFront	Delivery Network	Cloud CDN
Managed Data Warehouse	Redshift	SQL Warehouse	Big Query

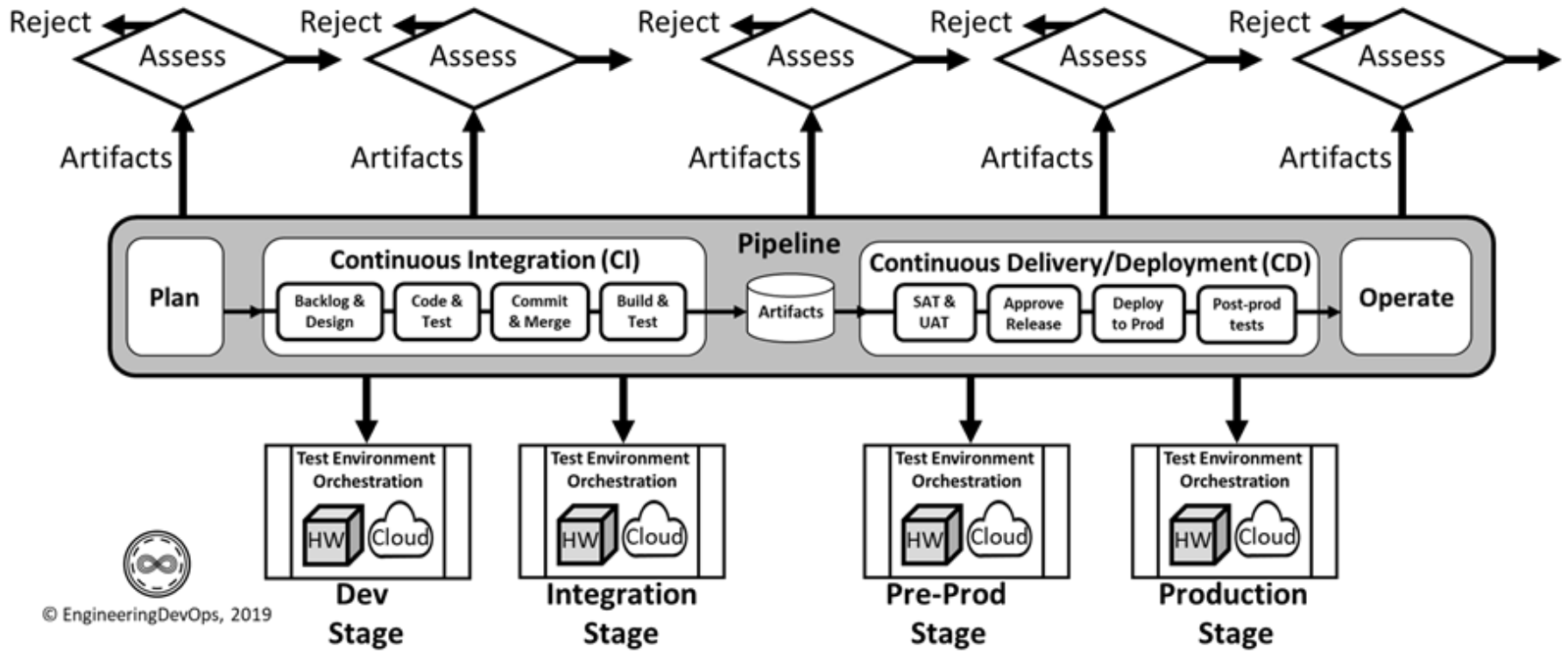
DevOps Multi-Cloud Services

DevOps Infrastructure Maturity Levels

The more elastic infrastucture is, the better, or more mature it is, from the point of view of DevOps capabilities.

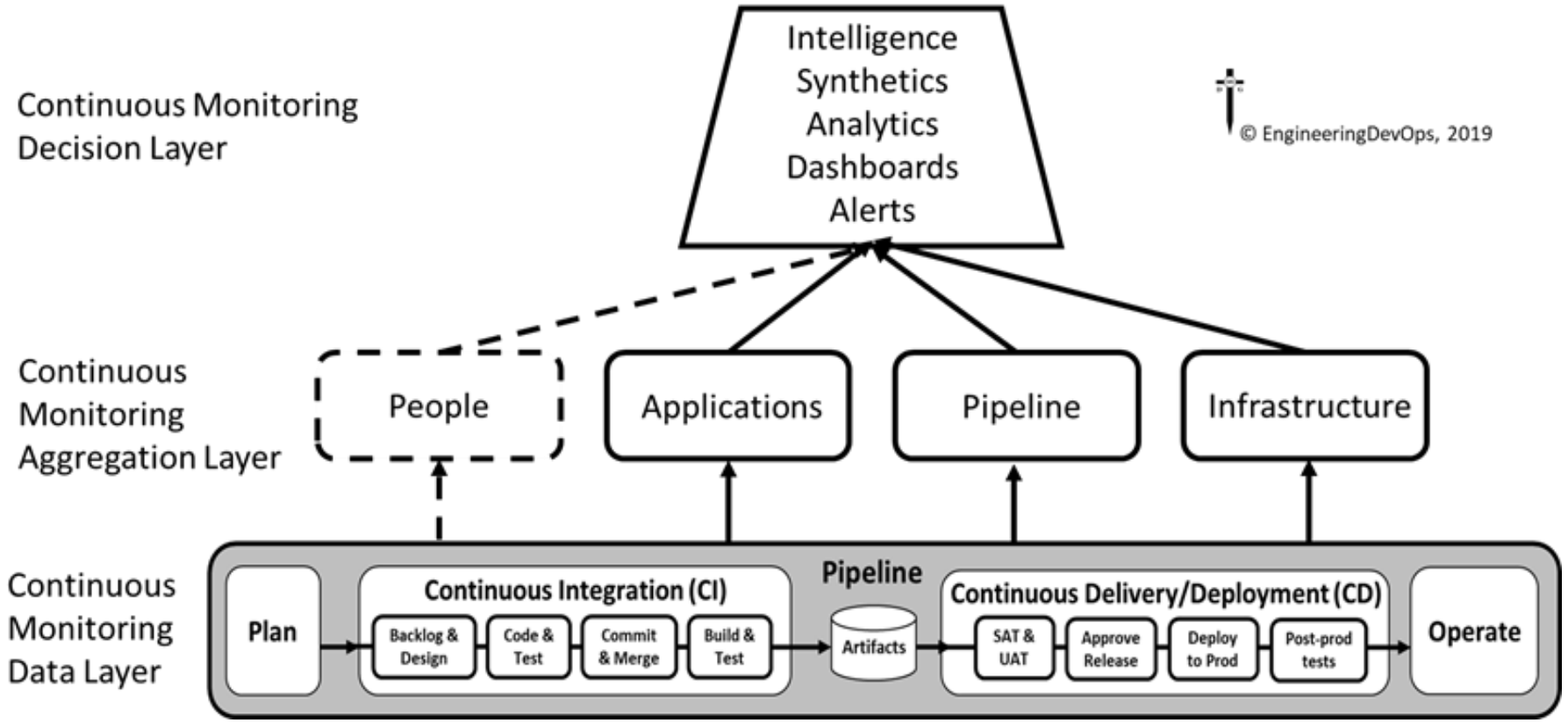


Continuous Test Engineering Blueprint



Continuous Test Engineering is a ***quality assessment strategy*** in which most tests are automated and integrated as a core and essential part of DevOps. Continuous testing is much more than simply “automating tests”.

Continuous Monitoring Engineering Blueprint



Continuous Monitoring provides visibility of the health of all things that are important to the operation and performance of the (1) application, (2) pipeline and (3) the infrastructure.

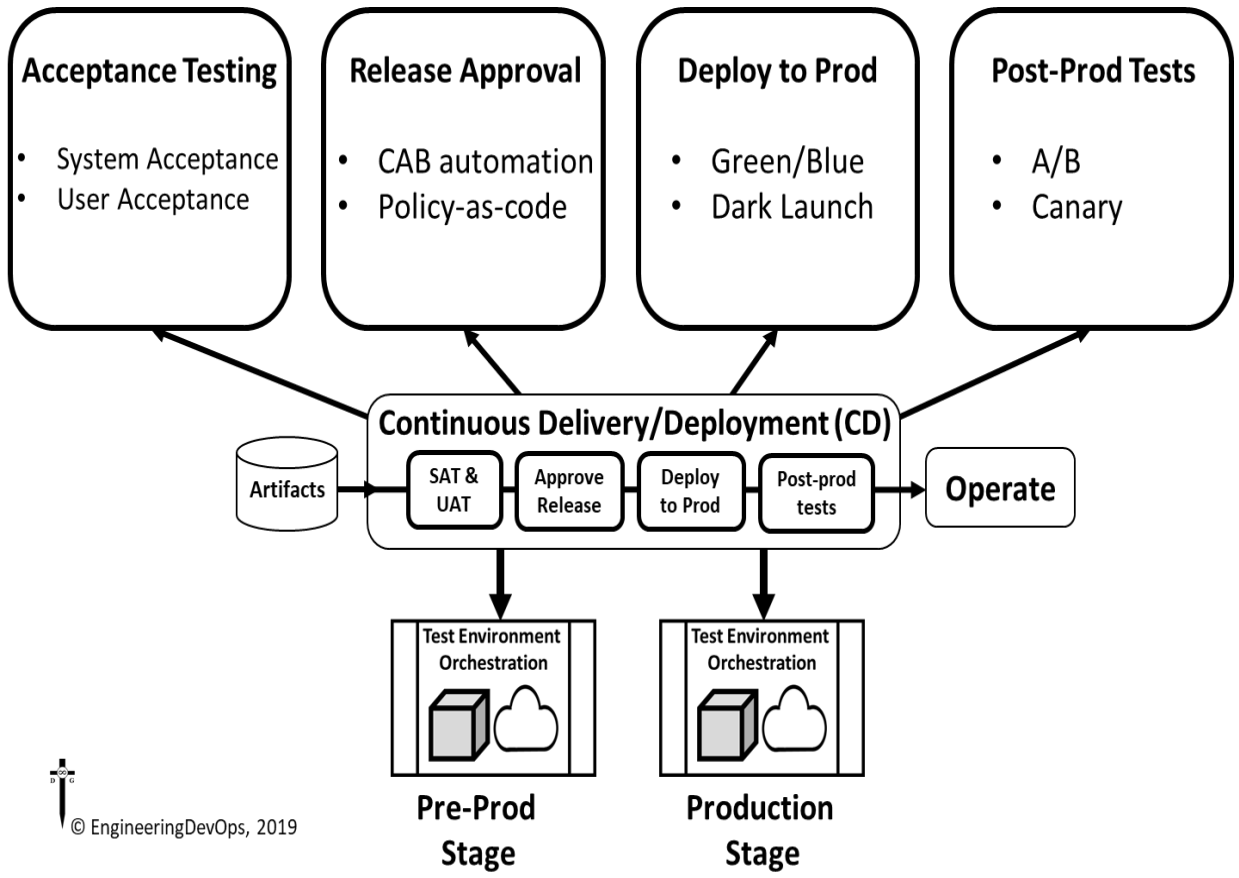
Continuous Delivery & Deployment Engineering Blueprint

***Continuous Delivery*:**

- Software is deployable throughout its lifecycle
- Your team prioritizes keeping the software deployable over working on new features
- Anybody can get fast, automated feedback on the production readiness of their systems any time somebody makes a change to them
- You can perform push-button deployments of any version of the software to any environment on-demand

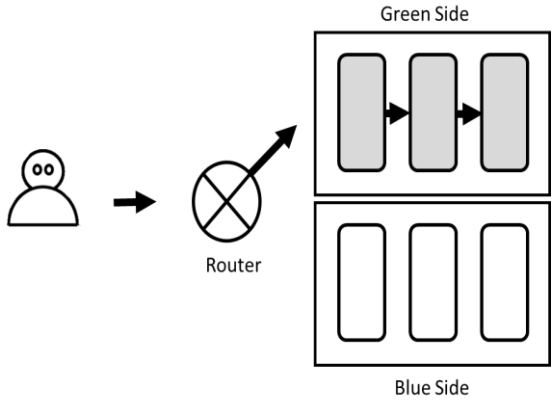
***Continuous Deployment* properly:**

- Your software is automatically deployed after continuous delivery

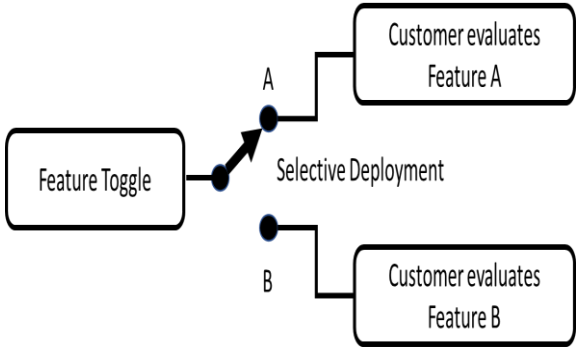


Deployment Strategies

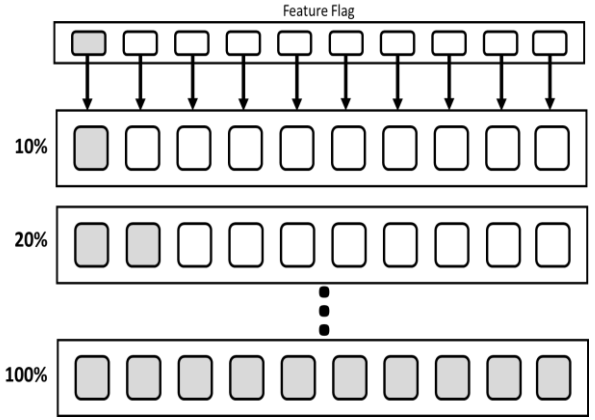
Blue-Green

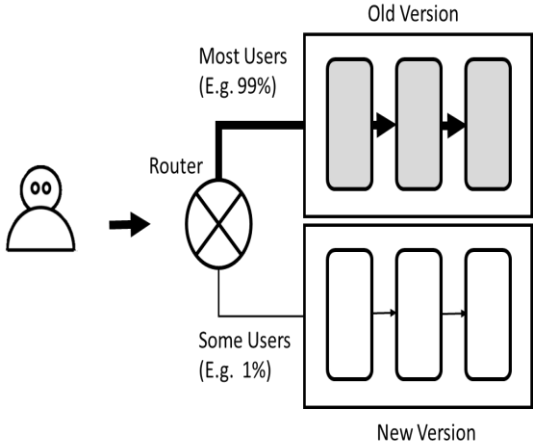


A/B Testing

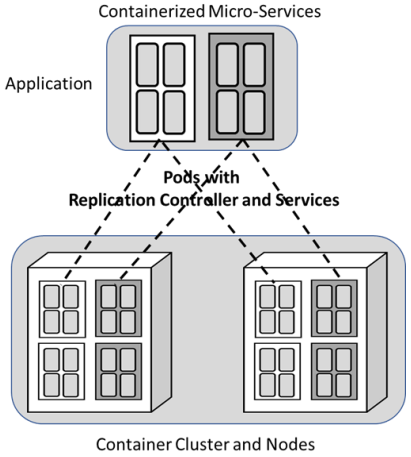


Feature Flag Rollout

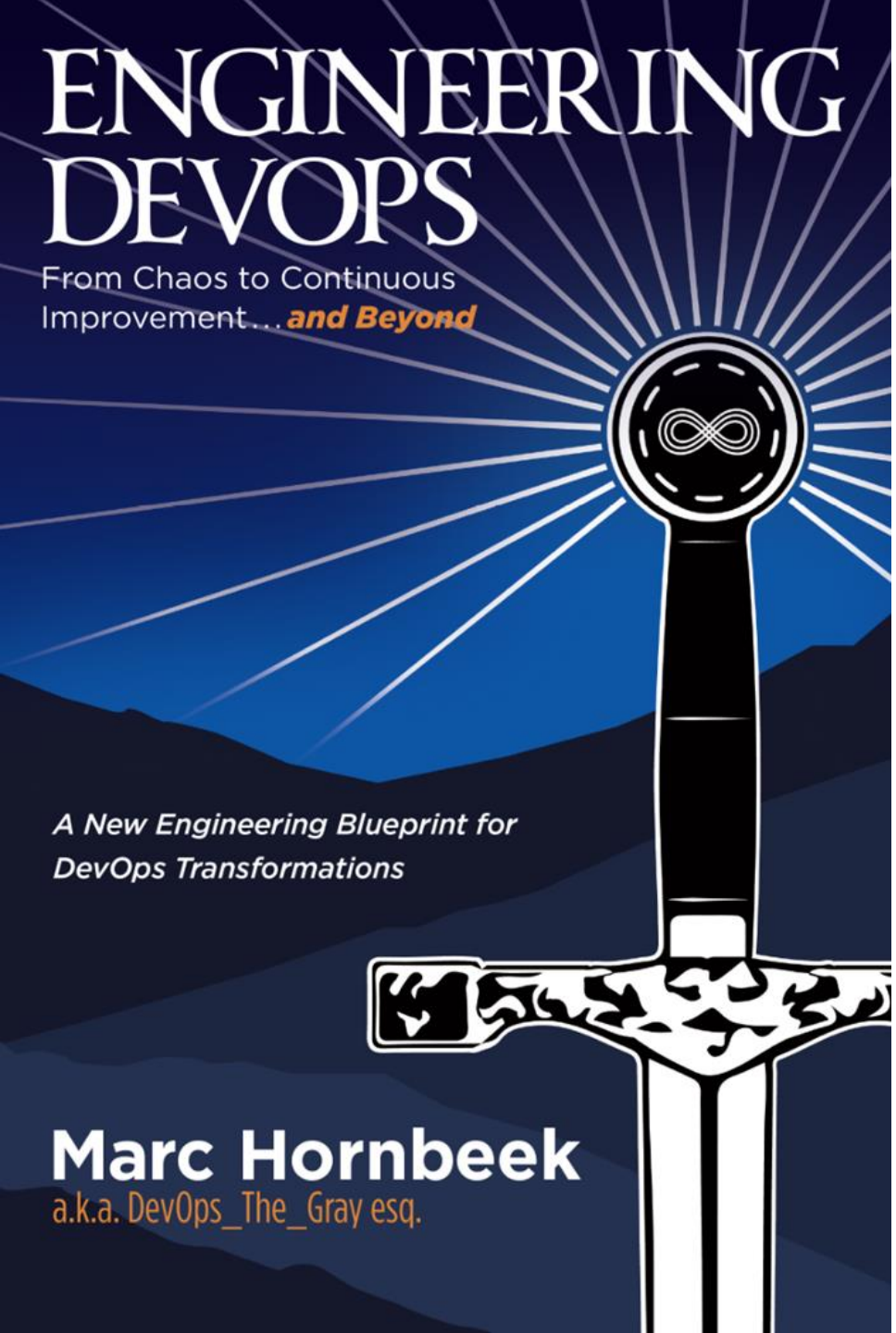




Canary Roll-out



Cluster Deployment

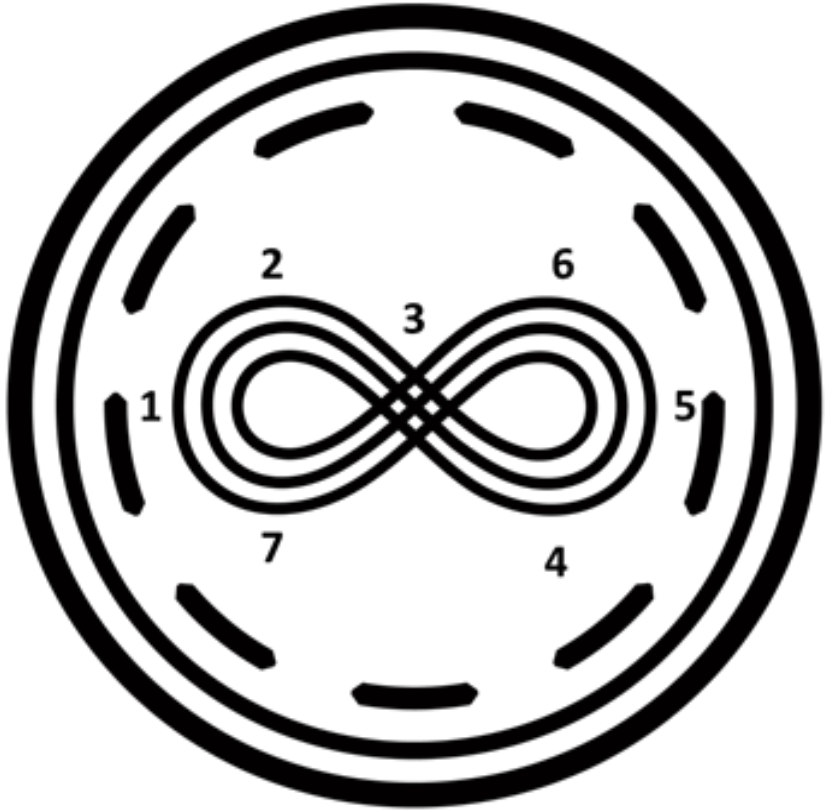


Seven-Step DevOps Transformation Engineering Blueprint

DevOps Seven-Step Transformation Engineering Blueprint

Infinite cycle of seven steps achieve your DevOps goals methodically, no matter what your goals or level of DevOps maturity are currently.

1. Visioning
Strategic, Sponsors, Partners
2 Alignment
Leadership, Team, Applications
3. Assessment
Discovery, 9 Pillars Maturity, Deep-Dive, Value-Stream Map
4. Solution
Future State, Road-map, Epics, Re-alignment



5. Realize
Projects, Stories, Tasks, POC, Validation, Training, Deployment, Governance
6. Operationalize
Monitor SLI/SLO/SLA, SRE Controls, Retrospectives
7. Expansion
Continuous Flow, Enterprise Adoption, Continuous Feedback, Continuous Improvement

Step One: Visioning

Sponsors and strategic leaders that will own the DevOps transformation, from one level of DevOps maturity to the next, are identified, key partner organizations that need to be strategically aligned to the DevOps transformation are identified, strategic goals for DevOps for the organization are defined, and actions towards the next step are committed.

1. Use the ***DevOps Transformation Application Scorecard*** to select the application or set of applications that will be targeted for the DevOps Transformation, and identify a model application to start the Transformation.
2. Define and document **strategic goals** for the DevOps transformation.
3. Agree to follow the ***DevOps Transformation Engineering Blueprint*.**
4. ***Identify strategic level leaders and partners*** for the DevOps Transformation Team that cover the cross-section of organization functions that are most relevant to the DevOps transformation.
5. ***Commit actions*** towards the *Alignment* step.
   - Engage DevOps *Expert*
   - Communicate membership, strategic goals and kick-off meeting schedule to the DevOps *Transformation Team*
   - Prepare DevOps Transformation *Alignment* Kick-off Meeting

DevOps Transformation Application Scorecard

Name of Application :

Some ***applications*** can benefit from DevOps more readily than others. An average rating score of 3 or more is preferred.

Rating scores: 0 = "Don't know/unsure", 1 = "Doesn't fit this characteristic", 2 = "Somewhat fits", 3 = "Mostly fits", 4 = "Good fit", 5 = "Perfect fit"	**Rating (0, 1 to 5)**
1. **Lead time**: The ***application*** will benefit from faster lead times, where lead time = time from backlog to deployment.	
2. **Leadership**: Leaders over this ***application*** are open to collaboration and will be sponsors of change.	
3. **Culture**: Team players that are associate with the ***application*** team (Product owners, Dev, QA, Ops, Infra, Sec, PM) are open to collaboration and change.	
4. **Application architecture**: The ***application*** is currently using, or planning to use, service-oriented, modular architectures.	
5. **Product Team size**: At least 15 people associated with the ***application*** team (includes Product owners, Dev, QA	

Step Two: Alignment

Sponsor, conducts a DevOps Transformation Alignment Meeting with the DevOps Transformation Team.

1. DevOps **Strategic Goal is communicated** by the Sponsor
2. DevOps **Transformation Team and roles are introduced**.
3. A **Definition of DevOps** is discussed and agreed.
4. The **DevOps Engineering Blueprint** is presented.
5. **Present chosen applications** for the Transformation
6. **Align around DevOps** Transformation Engineering **Blueprint**
7. DevOps Expert conducts DevOps **Transformation Goal Workshop**.
8. Decide key recommended focus for assessment step using a **DevOps Transformation Practices Topics Scorecard**.
9. **Assessment input requirements** for the next step are discussed.
10. **Next steps actions** towards the Alignment step are committed including:
    Schedule date to complete Alignment Data
    Schedule date to complete Assessment inputs
    Set Date for DevOps Transformation Assessment Workshops
    Logistics for DevOps Transformation Assessment Workshops

DevOps Transformation Goals Scorecard	Metric Unit	Importance (I) (1-5) (1=low, 5=critical)	Current State	Desired State	Percent Improvement (P%)	SCORE I x P%	RANK
Agility		4.4			205%	9	1
Lead time: Duration from code commit until code is ready to be deployed to production.	# days	5	5.0	2.0	250%	13	2
Release Cadence: Frequency of having releases ready for deployment to live production.	# releases / month	5	0.2	0.6	300%	15	1
Fraction of Non-Value Added-Time Fraction of their time employees are not spending on new value enhancing work such as new features or code.	Fraction %	3	30%	20%	150%	5	15
Batch size: Product teams break work into small batch increments.	1-10 (1 rarely, 10 usually)	5	3.0	5.0	167%	8	3
Visible Work: Workflow is visible throughout the pipeline.	1-10 (1 rarely, 10 usually)	4	5.0	8.0	160%	6	6
Security		2.7			233%	6	2
Security Events: # times that a serious business impacting security event occur over a set period	# per year	2	2	1	400%	8	4
Unauthorized Access: # times per period that unauthorized users accessed unathorized information.	# per year	3	1	1	100%	3	21
Fraction of time remediating security problems: Average % of time that employees spend remediating security issues.	%	3	5%	3%	200%	6	9
Satisfaction		3.8			137%	5	5
Employee satisfaction with team: Employees are likely to recommend their team as a great to work with.	1-10 (1 not likely, 10 most likely)	3	7.0	8.0	114%	3	20
Employee satisfaction with organization: Employees are likey to recommend their organization as a organization to work in.	1-10 (1 rarely, 10 most likely)	4	6.0	8.0	133%	5	13
Organization type: The culture is of the organization is a ***generative*** type, with good communication flow, cooperation and trustful.	1-10 (1 rarely, 10 very much)	4	5.0	7.0	140%	6	11
Leader Style for Recognition: Leaders promote personal recognition by commending team for better-than-average work, acknowledging improvement in quality of work and personally compliments individuals' outstanding work.	1-10 (1 rarely, 10 very much)	4	5.0	8.0	160%	6	6
Stability		4.0			152%	6	3
MTTR: Mean-Time-To-Recover (MTTR) from failure/service outage in production.	hours	4	1.0	0.7	154%	6	8
Code merges problems: % of code merges from development branches to the trunk branch break the trunk branch.	%	4	15%	10%	150%	6	10
Quality		3.7			148%	5	4
Failures In Production: Frequently of failures requiring immediate remediation occur in live production.	# per week	4	0.1	0.05	200%	8	4
Test and Data Available: Tests and test data are sufficient and readily available when needed.	1-10 (1 rarely, 10 usually)	3	7.0	9.0	129%	4	19
Customer Feedback: The organization regularily seeks customer feedback and incorporates the feedback into design.	1-10 (1 rarely, 10 usually)	4	7.0	8.0	114%	5	14
Efficiency		3.3			143%	5	6
Unplanned work: % of time do employees spend on all types of unplanned work, including rework.	%	3	15%	10%	150%	5	15
Operating Costs are Visible: Comprehensive metrics are kept for **operating** costs of development and operations.	1-10 (1 rarely, 10 usually)	3	5.0	7.0	140%	4	17
Capital costs are visable: Comprehensive metrics are kept for **capital** costs of development and operations.	1-10 (1 rarely, 10 usually)	3	5.0	7.0	140%	4	17
Backlog Visibility: Lean product management is practiced using highly visible, easy-to-understand presentation formats that show work to be done.	1-10 (1 rarely, 10 usually)	4	5.0	7.0	140%	6	11

The DevOps Transformation Goal Scorecard spreadsheet is posted on www.EngineeirngDevOps.com

DevOps Transformation Practices Topics

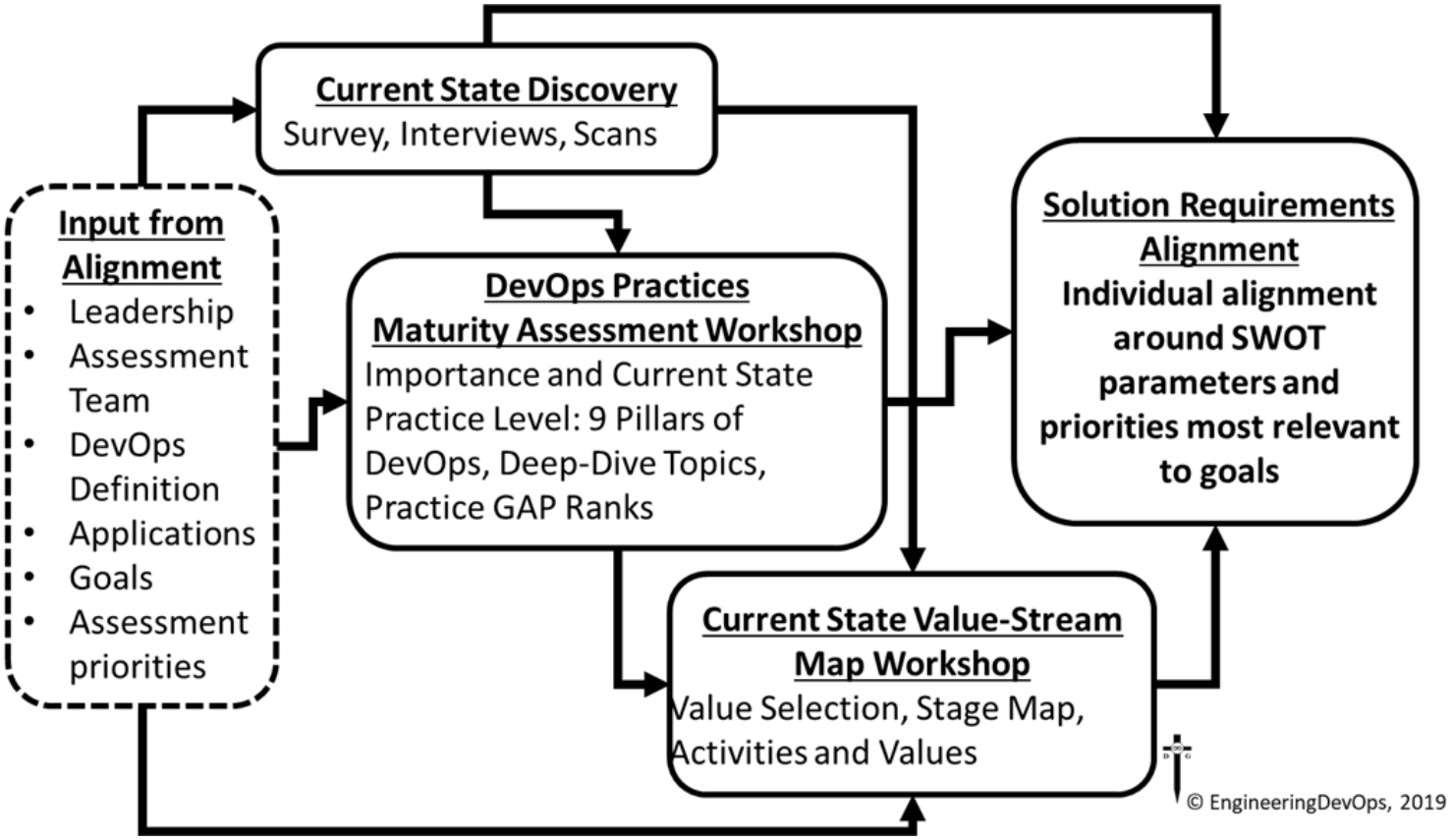
Nine Pillars of DevOps Practice Topics	Importance (I) (1-5) (1=low, 5=critical)	Current Level of Practice (P) (1-5) (1=not yet, 5=always)	SCORE (1-5) 6-(((Ix(5-P))/5)+1)	Rank
Collaborative Leadership Practices	2	3	4.20	7
Collaborative Culture Practices	4	3	3.40	4
Design for DevOps Practices	5	2	2.00	2
Continuous Integration Practices	3	3	3.80	4
Continuous Testing Practices	3	4	4.40	7
Elastic Infrastructures Practices	3	4	4.40	7
Continuous Monitoring Practices	5	2	2.00	2
Continuous Security Practices	4	1	1.80	1
Continuous Delivery/Deployment Practices	3	3	3.80	7

Special DevOps Deep Dive Practice Topics	Importance (I) (1-5) (1=low, 5=critical)	Current Level of Practice (P) (1-5) (1=not yet, 5=always)	SCORE (1-5) 6-(((Ix(5-P))/5)+1)	Rank
DevOps Version Management Practices	5	4	4.00	11
Value Stream Management DevOps Practices	4	2	2.60	2
Application Release Automation Practices	4	3	3.40	8
DevOps Infrastructure-As-Code	3	2	3.20	4
Hybrid Cloud DevOps Practices	3	2	3.20	4
Multi-Cloud DevOps Practices	3	2	3.20	4
Application Performance Monitoring Practices	4	3	3.40	8
DevOps Training Practices	3	2	3.20	4
Site Reliability Engineering Practices	3	1	2.60	2
DevOps Service Catalog	3	3	3.80	10
DevOps Governance Practices	4	1	1.80	1
Other >	0	0	5.00	12

The DevOps Transformation Practices Topics Scorecard is posted in spreadsheet form on www.EngineeringDevOps.com

Step Three: Assessment

- Discover current state
- DevOps Practices Maturity Assessment
- Current State Value-stream Map
- Align priorities for solution requirements



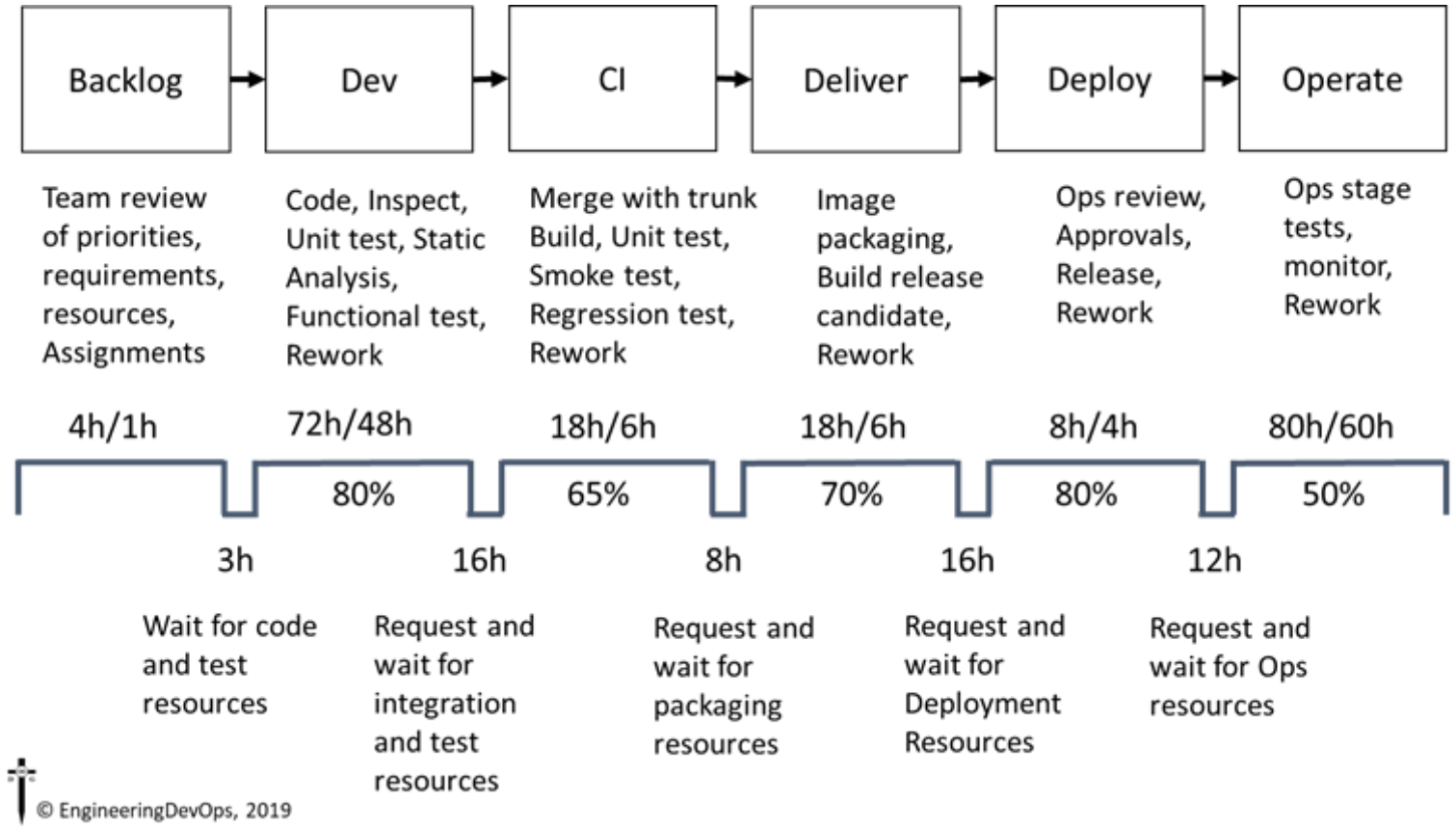
DevOps Engineering 9 Pillars Practices Maturity Assessment

Enter Calculate

Nine Pillars of DevOps Practice Topics	Importance (I) (1-5) (1=low, 5=critical)	Current Level of Practice (P) (1-5) (1=not yet, 5=always)	SCORE (0-5) 6-(((Ix(5-P))/5)+1)	GAP Rank (1-N)
Collaborative Leadership Practices	3.80	4.40	4.54	9
Leadership demonstrates a vision for organizational direction, team direction, and 3-year horizon for team.	1	5	5.00	4
Leaders intellectually stimulate the team status quo by encouraging asking new questions and question the basic assumptions about the work.	3	4	4.40	3
Leaders provide inspirational communication that inspires pride in being part of the team, says positive things about the team, inspires passion and motivation and encourages people to see that change brings opportunities.	5	5	5.00	4
Leaders demonstrate supportive style by considering others' personal feelings before acting, being thoughtful of others' personal needs and caring about individuals' interests.	5	4	4.00	1
Leaders promote personal recognition by commending teams for better-than-average work, acknowledging improvements in the quality of work and personally compliment individuals' outstanding work.	5	4	4.00	1
Collaborative Culture Practices				
The culture encourages cross-functional collab... silos between Dev & ...				

Practices for each of the 9 Pillars: Leadership, Collaborative Culture, Design for DevOps, Continuous Integration, Continuous Testing, Elastic Infrastructure, Continuous Monitoring, Continuous Security and Continuous Delivery/Deployment

Current State Value Stream Map Workshop



Engineering Solutions Alignment

Priorities from Workshops

Nine Pillars of DevOps Practice Topics	Importance (I) (1-5)	Current Level of Practice (P) (1-5)	SCORE (0-5)	GAP Rank (1-N)
Collaborative Culture Practices	4.29			
Continuous Security Practices	3.88			
Continuous Monitoring Practices	4.00			3

DevOps Topics Maturity Assessment Scorecard	Importance (I) (1-5)	Current Level of Practice (P) (1-5)	SCORE (0-5)	GAP Rank (1-N)
DevOps Training Practices	3.86			
DevOps Application Performance Monitoring (APM) Practices	4.20			
DevOps Governance Practices	4.11			1

Additional items from Value Stream Workshop

DevOps Practices	DevOps Sponsor	DevOps Expert	Product Development	Product Quality	Operations Support	Infrastructure	DevOps Tools	Partner Management	Product Owner	Security	Project Management	Training	Finance	Human Resources	Governance	Votes	Group Rank
Changes to End-to-End DevOps workflows are led by an expert team, and reviewed by a coalition of stakeholders including Dev, Ops and QA.	1		1	1	1				1			1	1		1	8	1
Key Performance Indicators (KPIs) for the DevOps infrastructure components are automatically gathered, calculated and made visible to anyone on the team that subscribes to them. Example metrics are availability (up time) of computing resources for CI, CT, CD processes, time to complete builds, time to complete tests, # of commits that fail, # changes that need to be reverted due to serious failures.			1		1		1			1			1		1	6	2
DevOps System changes follow a phased process to ensure the changes do not disturb the current DevOps operation. Examples of implementation phases in include: 1) proof of concept (POC) phase in a test environment, 2) Limited production 3) Deployment to all live environments.	1	1					1				1			1		5	3
Predictive analytics are used to dynamically adjust DevOps pipeline configurations. For analysis of test results, data may indicate a need to concentrate more testing in areas that have a higher failure trend.		1						1			1	1		1		5	3
Security assurance automation and security monitoring practices are embraced by the organization.	1							1						1		3	5
All information security platforms that are in use expose full functionality via APIs for automatability.					1												
Communication flows fluidly across the end-to-end cross-functional team using collaboration tools where appropriate (E.g. SLACK,																	

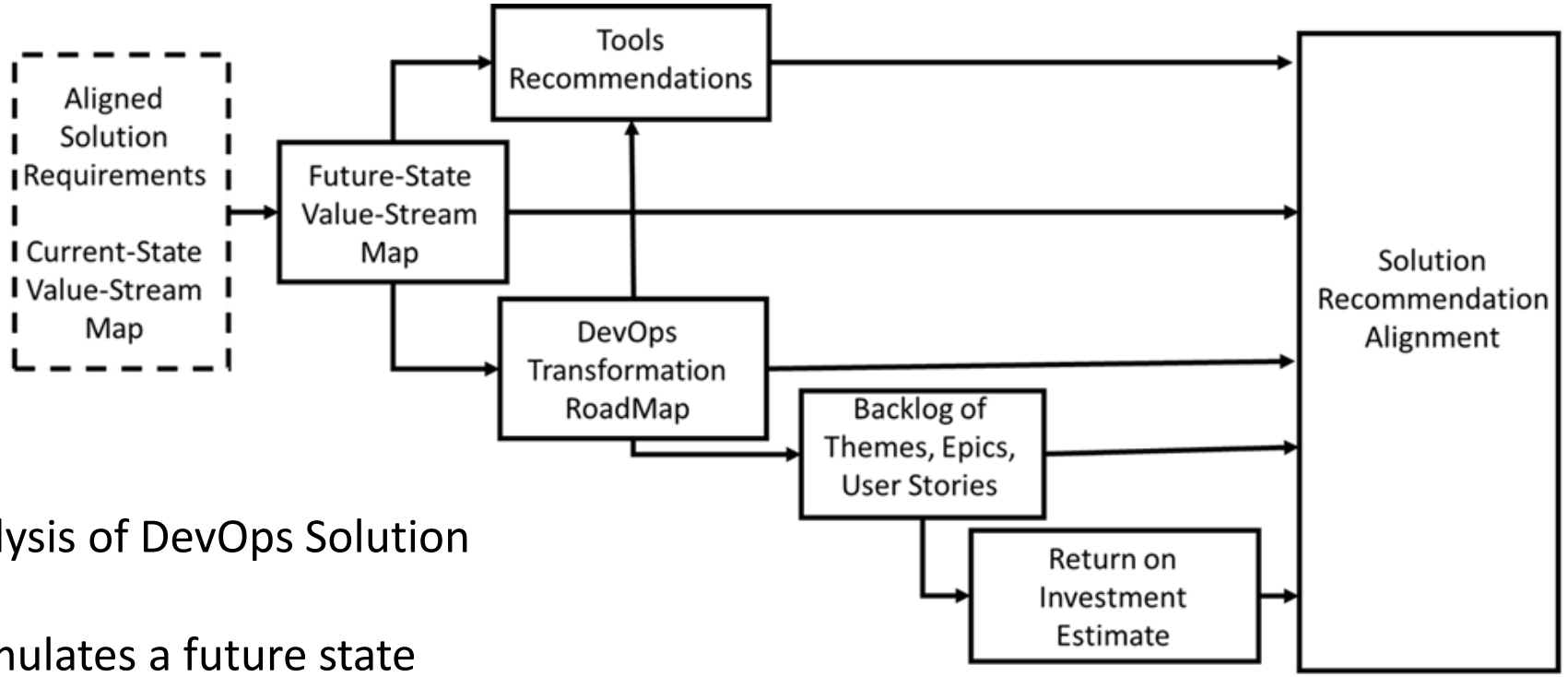
Stakeholder Surveys Determine Solution Requirements Priorities

Solution Alignment Meeting

Stakeholders need to agree to the DevOps Transformation.
- DevOps Sponsor, Initiator, Expert, DevOps Transformation Team Leaders, Business Leaders.

- DevOps Transformation Goals
- Summary of Current State of DevOps
- Summary of DevOps Solution Requirements
- Future-State Value-Stream Map
- DevOps Transformation RoadMap
- Return-On-Investment
- Solution Recommendation Summary
- Next Steps

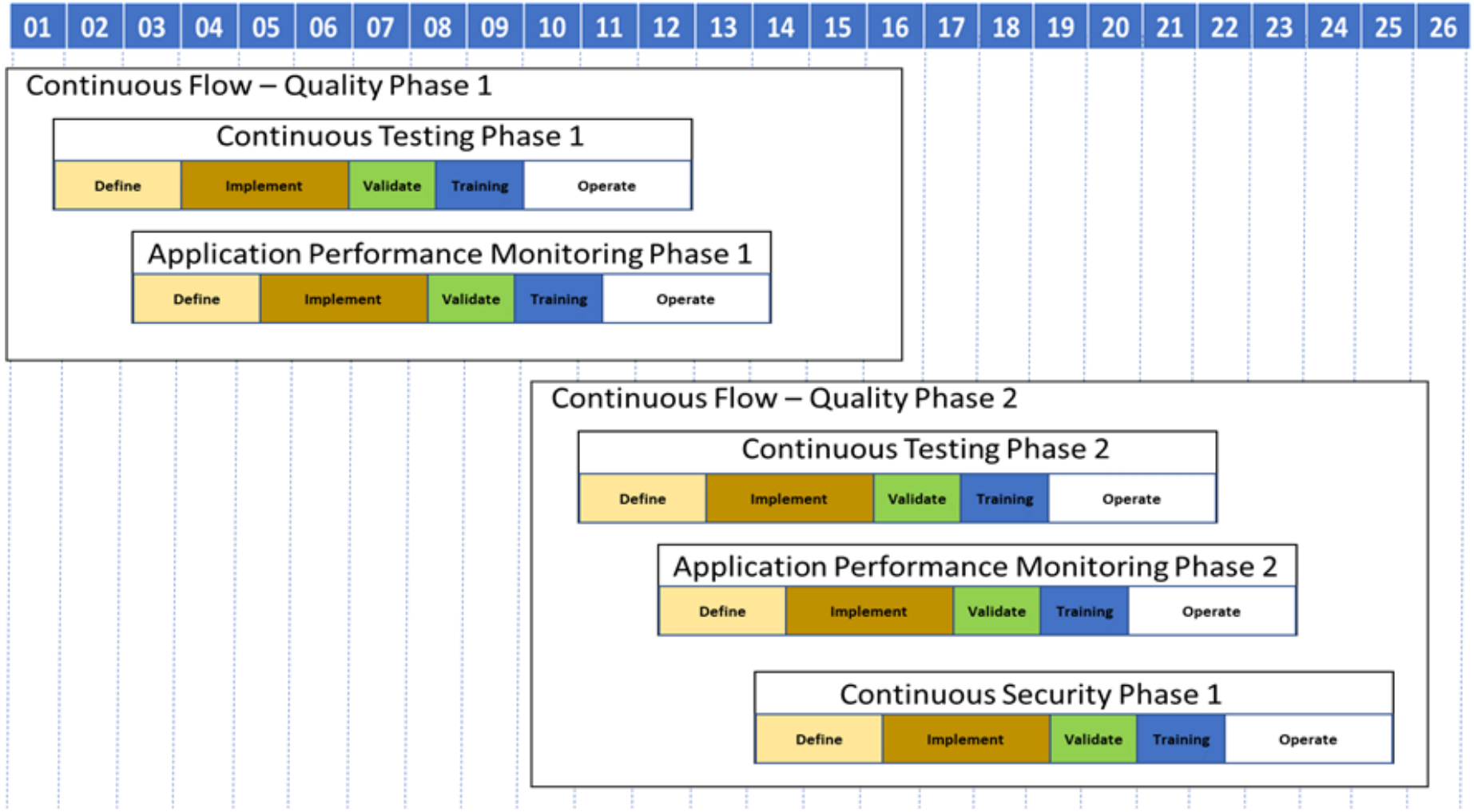
Step Four: Solution



DevOps Expert analysis of DevOps Solution Requirements.
DevOps Expert formulates a future state *Value-Stream Map* and *Solution Roadmap* including Themes, Epics, and User-Stories to obtain solution alignment with the *DevOps Transformation Team*.

Engineering DevOps Transformation RoadMap

Epics
Themes
User Stories



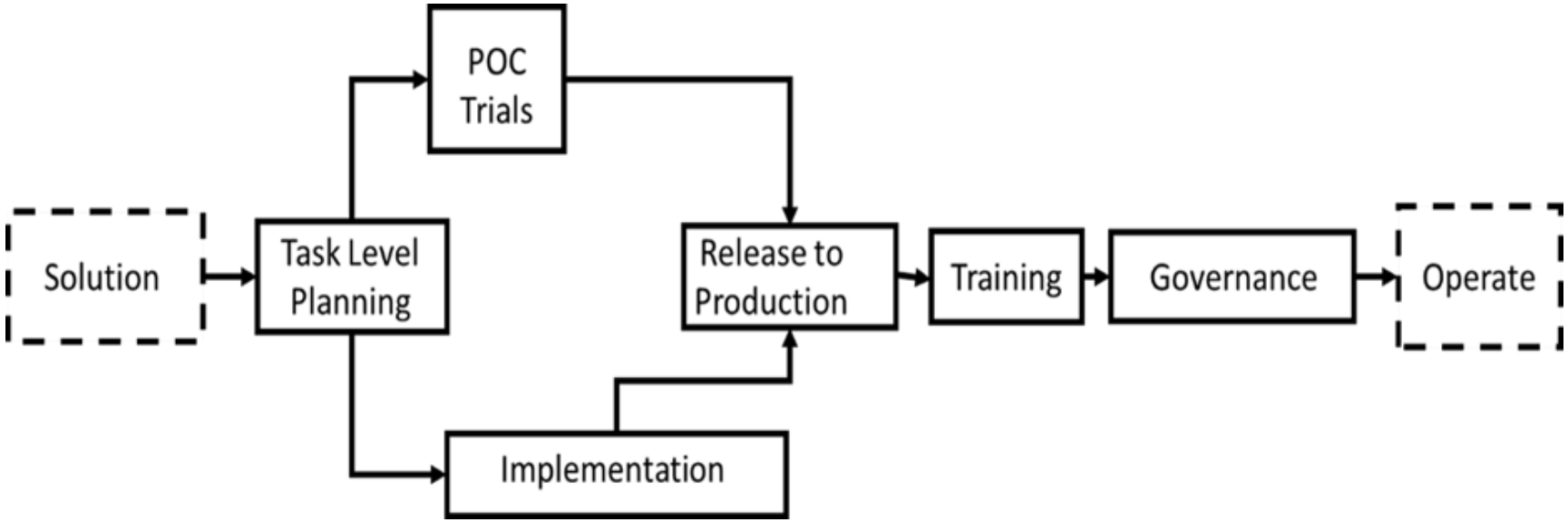
DevOps Transformation Engineering Backlog

EPIC	Phase	User Story	Responsible
Continuous Testing Phase 1	Define	As a developer I want to use Test Driven Development to define my unit tests before I write my code.	Developer
Continuous Testing Phase 1	Implement	As a developer I want to use Test Driven Development to define my unit tests before I write my code.	Project Manager
Continuous Testing Phase 1	Validation	As a developer I want to use Test Driven Development to define my unit tests before I write my code.	QA Engineer
Continuous Testing Phase 1	Training	As a developer I want to use Test Driven Development to define my unit tests before I write my code.	QA Engineer
Continuous Testing Phase 1	Operate	As a developer I want to use Test Driven Development to define my unit tests before I write my code.	Developer

Estimate Return On Investment (ROI)

Per Application (3 years Period)	Formula	NO DevOps Enhancements (n)	DevOps Enhancements (d)
R) Average # releases/year	**Estimate Number**	**6**	**10**
A) Total Costs (3 years) $K	**B+C**	**$7,550**	**$6,480**
B) Cost for 3 years of releases $K	**B1 + B4**	**$7,550**	**$5,220**
B1) Labor Costs $K	3 x B2 x B3	$6,600	$4,620
B2) Average Labor rate $K/year	Estimate Number	$110	$110
B3) Average # workers	Estimate Number	20	14
B4) Capital Depreciation (3 years amortization) $K	Estimate Number	$950	$600
C) DevOps Enhancements costs $K	**C1 + C4**	**$0**	**$1,260**
C1) Labor Costs$K	3 x C2 x C3	$0	$660
C2) Average Labor rate $/year	Estimate Number	$110	$110
C3) # workers for DevOps enhancement	Estimate Number	0	2
C4) Capital Depreciation (3 years amortization) $K	Estimate Number	$0	$600
D) Direct Savings Attributed to DevOps Enhancements	**An-Ad**		**$1,070**
F) Cost per release	**A/R**	**$419**	**$216**
H) # Releases over 3 years	3 x R	18	30
I) Additional releases due to DevOps Enhancements	Hd - Hn		12
J) Costs of Equivalent # releases with DevOps enhancements (3 years) $K	F x Hd	$12,583	$6,480
K) Equivalent Savings due to DevOps Enhancements (3 years) $K	**Jd - Jn**		**$6,103**
L) Return on Investment (ROI)	**K / C**		**5**
M) Payback period (Months)	**C / (K/36)**		**7**
E) Number of Applications	**Estimate Number**	**5**	**5**
O) Direct Savings for all applications (3 years) $K	**E x Dd**		**5,350**
P) Equivalent Savings for all applications (3 years) $K	**E x Kd**		**30,517**

Step Five: Realize

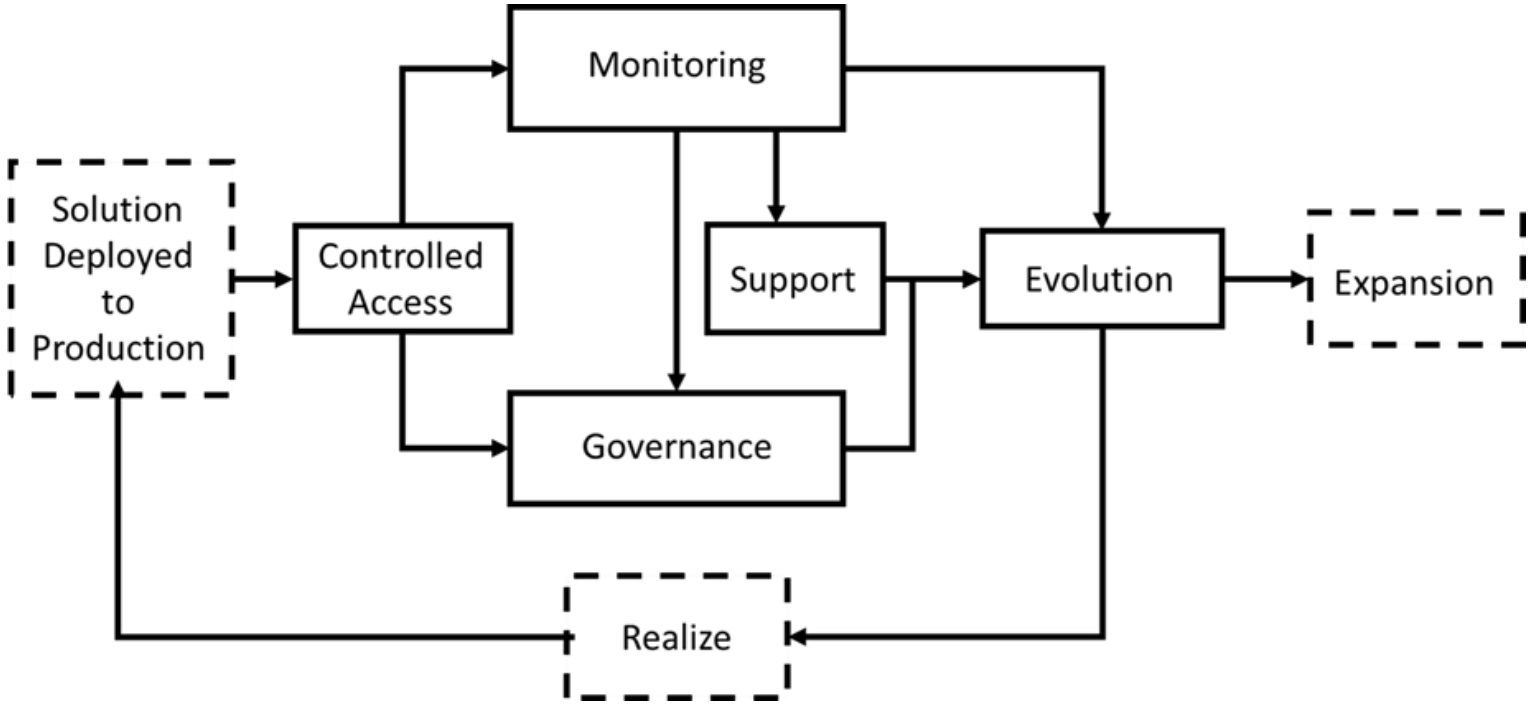


Both DevOps projects and Application development projects involve designing, coding, integration, testing, validation, deployment and operations of code changes.
Many of the same processes and tools can be shared between developers of DevOps changes and developers of applications.
Differences however are critical to fully understand and consider during the Realize step when engineering a DevOps Transformation.

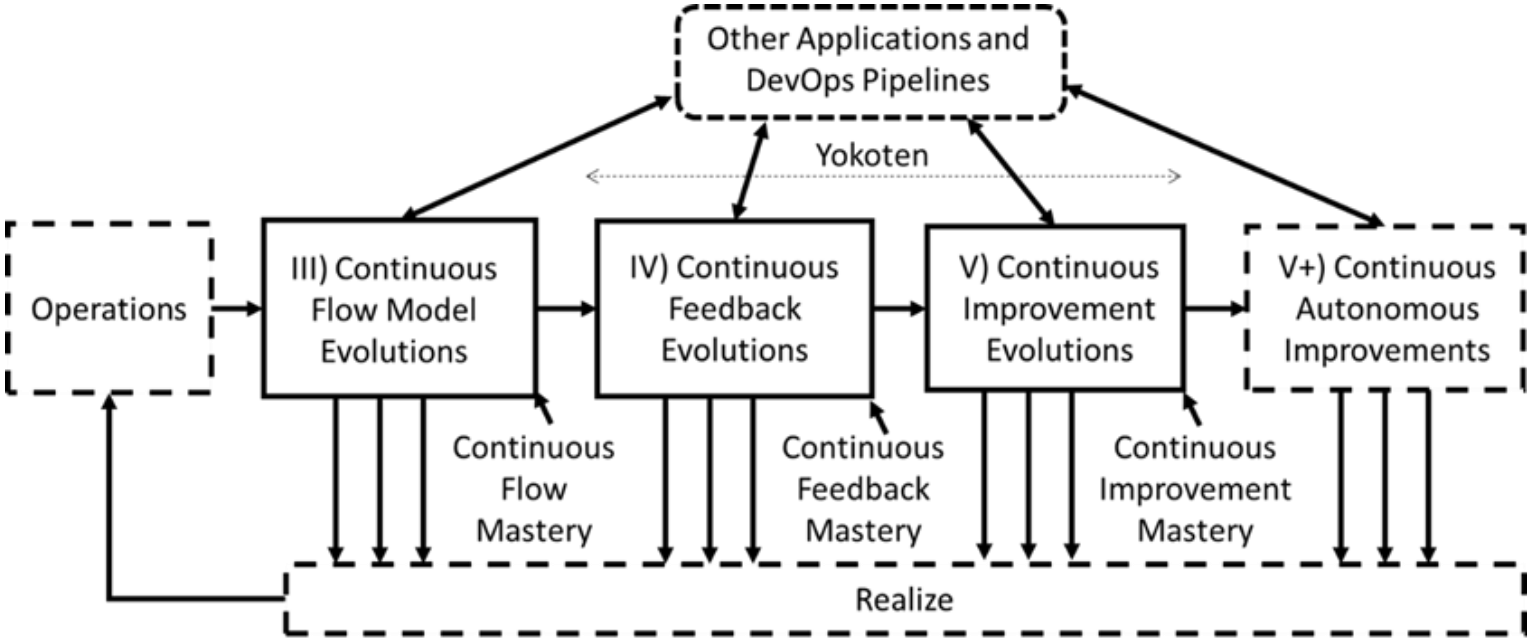
Step 6: Operationalize

Operating a DevOps Solution that has been deployed for use by Applications developers has similar requirements to operating infrastructures for Applications.

The key difference is the focus of what is being operated. For DevOps operations it is the DevOps solution rather than the Application that is being developed that is the focus of operations.



Step Seven: Expansion



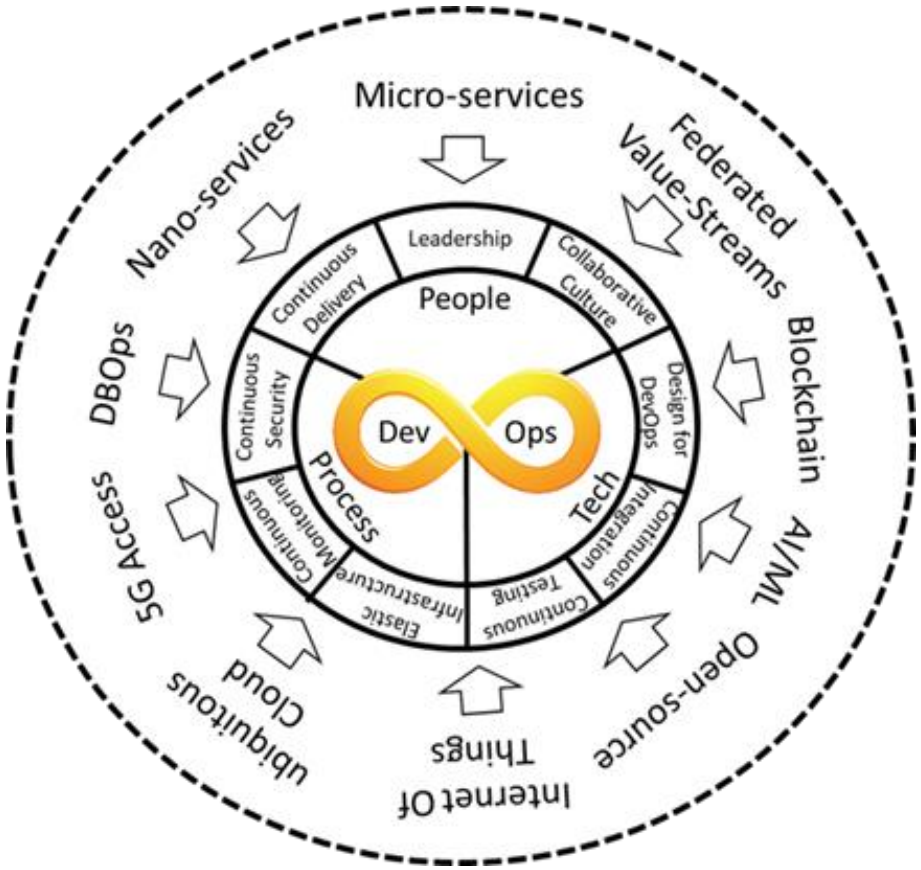
The blueprint shows a progression of DevOps to higher levels of maturity from left to right with a continuous loop of realized improvements with operations experience of each improvement driving the next evolution.

Future of Engineering DevOps - Beyond Continuous Improvement Influencing Factors

Applications: more distributed and dependent on highly secure, high-bandwidth, low-latency networks that can host their workloads and serve as their DevOps delivery platforms.

Pipelines: high capacity, highly parallel and distributed application deployment architectures. Federated value-stream management systems are become more important. The volume of data is growing. Pipeline bottlenecks that are exacerbated by a distributed architecture, such as communications, analysis and testing must leverage AI/ML to keep up. Open-source DevOps tool solutions are on the rise. The rapid rise of open-source containers and Kubernetes are a harbinger of the future of DevOps tools.

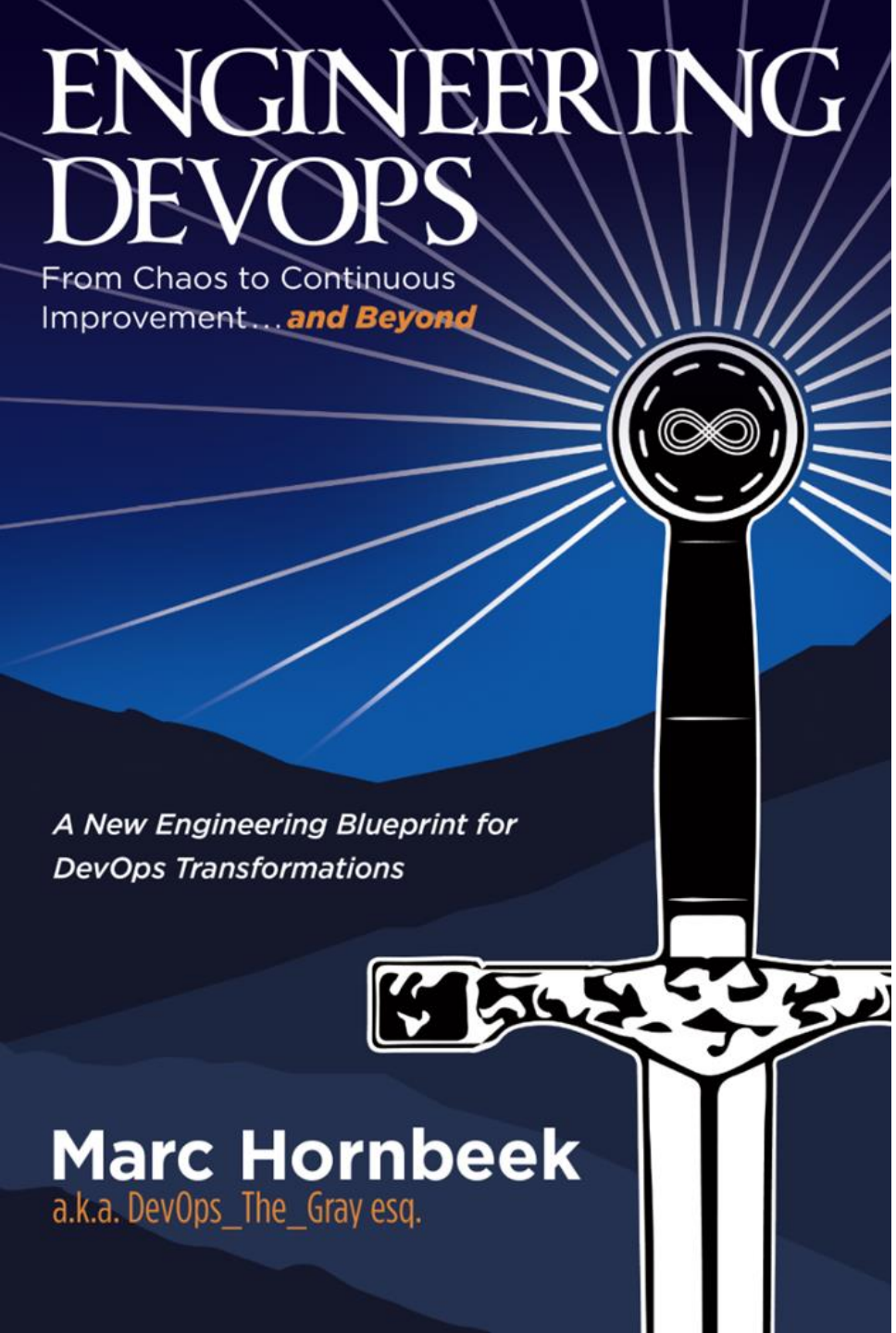
Infrastructures are becoming even more elastic. The rapid rise of ubiquitous cloud computing, 5G access, and IOT indicate infrastructures are moving from software configurable systems at a macro device level to a more distributed micro-level. Alone with this highly-distributed ephemeral infrastructure paradigm, new security attack surfaces will need to leverage highly secure and distributed blockchain technology.



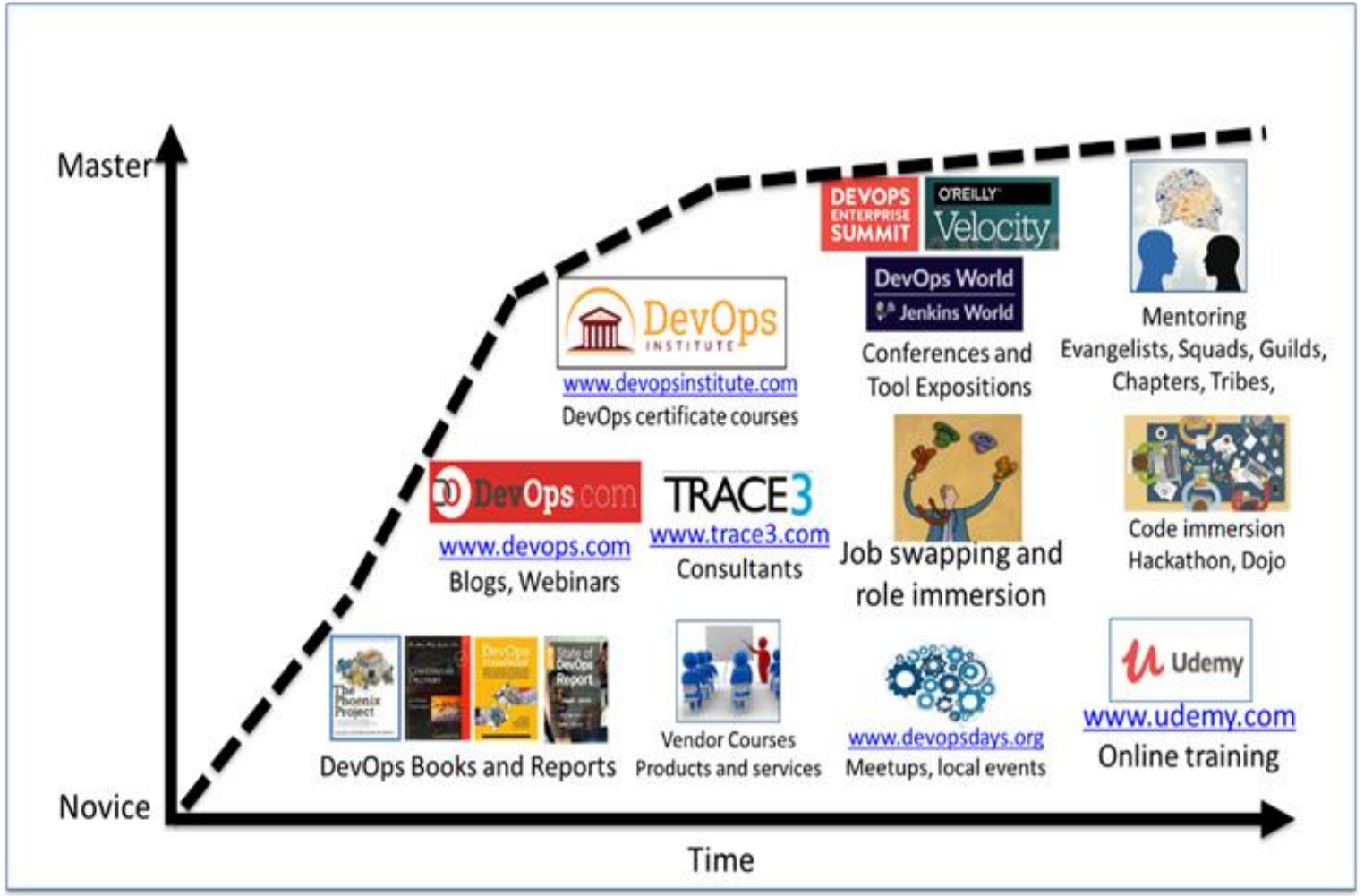
Future of Engineering DevOps - Beyond Continuous Improvement Prediction

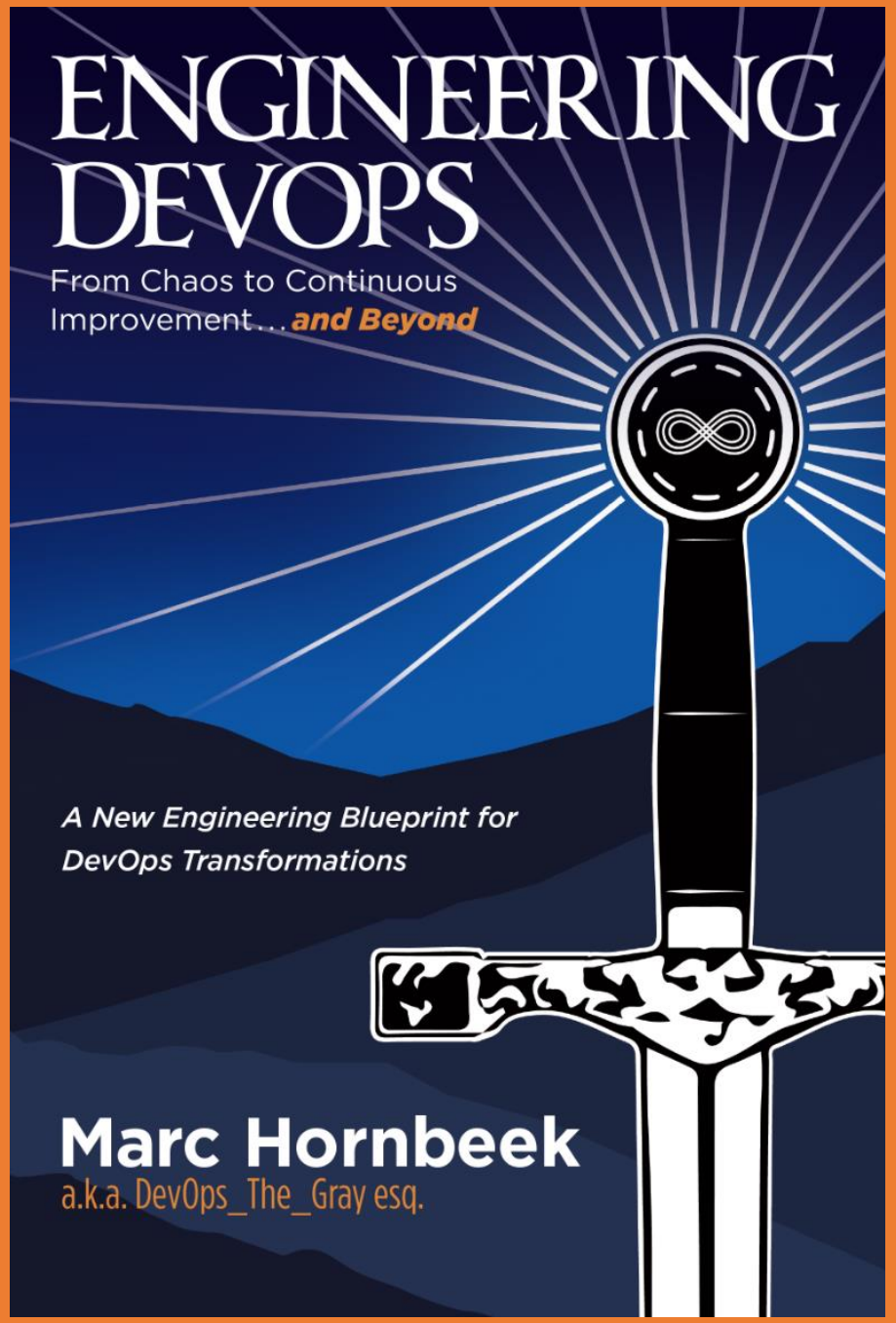
- New bottlenecks; new opportunities to reduce bottlenecks
- Pipelines and orchestrated infrastructures evolve for more distributed Applications
- DevOps tools cloud-optimized and configurable into highly federated, networked toolchain topologies
- AI/ML algorithms provide insights how to reduce existing bottlenecks and prevent new networked structures from adding new bottlenecks

Prospect of **eliminating the biggest bottleneck – human interactions** – from DevOps pipelines could result in a ***new DevOps maturity level 6, that I refer to as "Continuous Autonomous Improvement"***.



How can you learn more about Engineering DevOps?





"***Engineering DevOps***", provides you the processes and steps to engineer your own DevOps practice." **- Michael Morris, Senior Director over IT Cloud and DevOps Transformation at NetApp**

"***Engineering DevOps*** is truly an Engineering Reference Guide." **– Sanjeev Sharma, Cloud and DevOps Transformation and Strategy Executive, Thought Leader, Startup Advisor, Keynote Speaker and Author**

"***Engineering DevOps*** is a must-read for everyone who is planning, implementing, operating and scaling DevOps." – **Niladri Choudhuri, Founder and CEO of Xellentro Consulting Services LLP., and DevOps India Summit.**

Reserve your copy today on
www.EngineeringDevOps.com

Book price: 6"x9" Softcover print $19.95, Downloadable eBook $9.95

Engineering DevOps is scheduled to release October 2019.

ENGINEERING DEVOPS

From Chaos to Continuous Improvement…*and Beyond*

Marc Hornbeek
Principal Consultant

EngineeringDevOps.com

DevOps and Cloud
Collaboration for Business Success

QUESTIONS ?
SUGGESTIONS ?
COMMENTS ?

mhexcalibur@yahoo.com

www.EngineeringDevOps.com