DevOps

Ali Kahoot
Dice Analytics

About Me - M. Ali Kahoot

/kahootali

/kahootali

- Lead DevOps Engineer, Tarabut Gateway
- DevOps Trainer, Dice Analytics
- Ex-Team Lead, DevOps Engineer, Stakater, Aurora Solutions
- Ex-Software Engineer, Bentley Systems
- Blog with more than 81k views and 1.5k claps on Medium
- Trained more than 300 resources on DevOps & Kubernetes
- Speaker at S&P DevOps Week, Data on Kubernetes & conducted more than 10 Bootcamps on Kubernetes
- Certified in RedHat Delivery Specialist: Container Platform Deployment
- Technical Expertise: Git, Containers, Kubernetes, OpenShift, Helm, Jenkins, Github Actions, Terraform, AWS, GitOps, Sealed Secrets

About You

- Name
- Field
- Experience
- Technologies you have worked on
- What are your expectations from this course?

What is DevOps?

Problem

- Everything has/needs Software nowadays
- Software has to run on a server
- A complete cycle of how code in your repo gets deployed to the server
- A bottleneck in deploying new features frequently
- Errors in service, then have to complete the lifecycle again
- Diagnose Issues on servers
- Shifting Blames

Problem

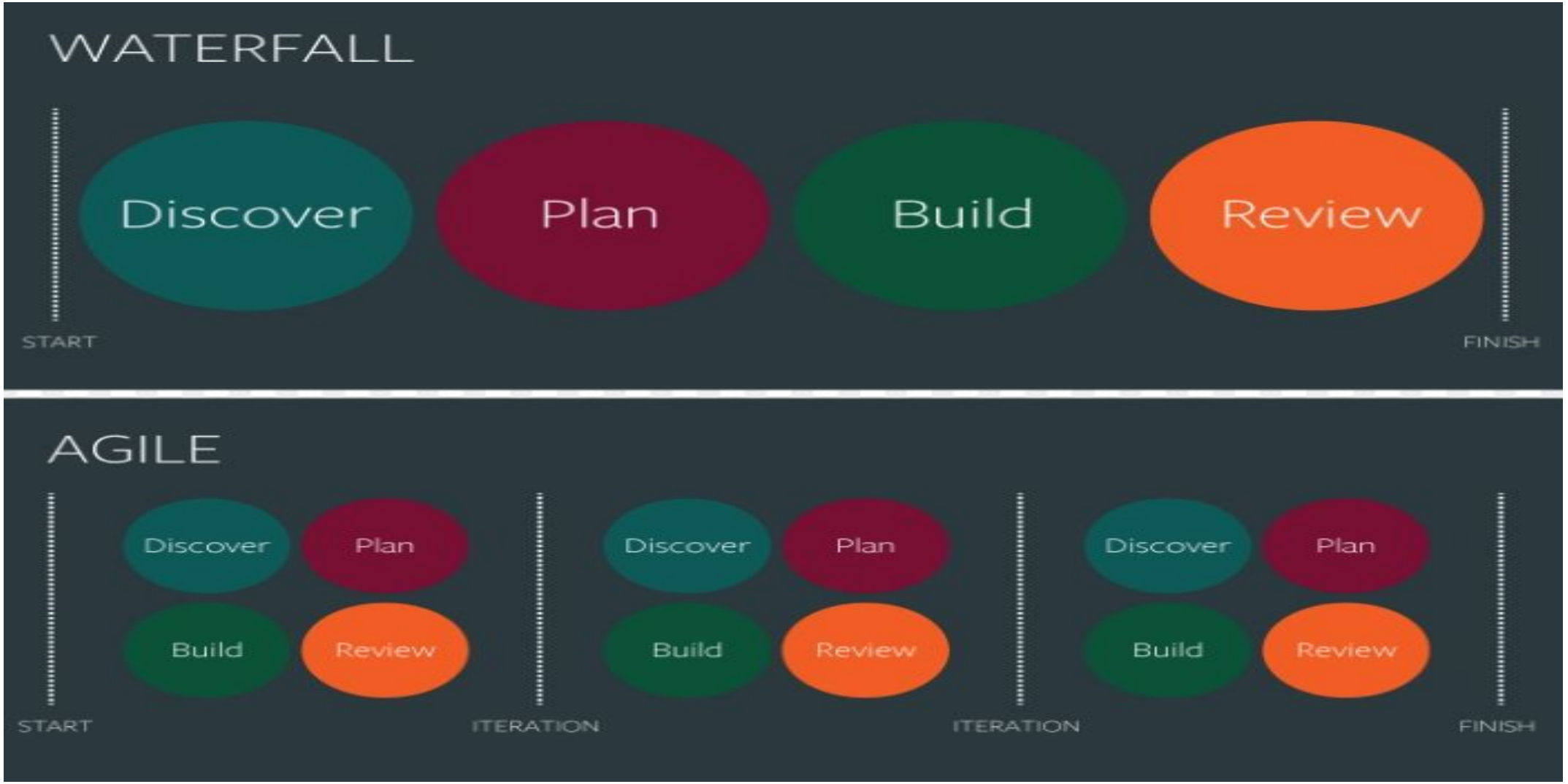
- Delays when deploying to Dev, waiting for confirmation, moving to QA, again manual confirmation then moving to Prod.
- Can take days to release a new version of software
- Then customer feedback, and again make changes and whole process is repeated

Problem

Result

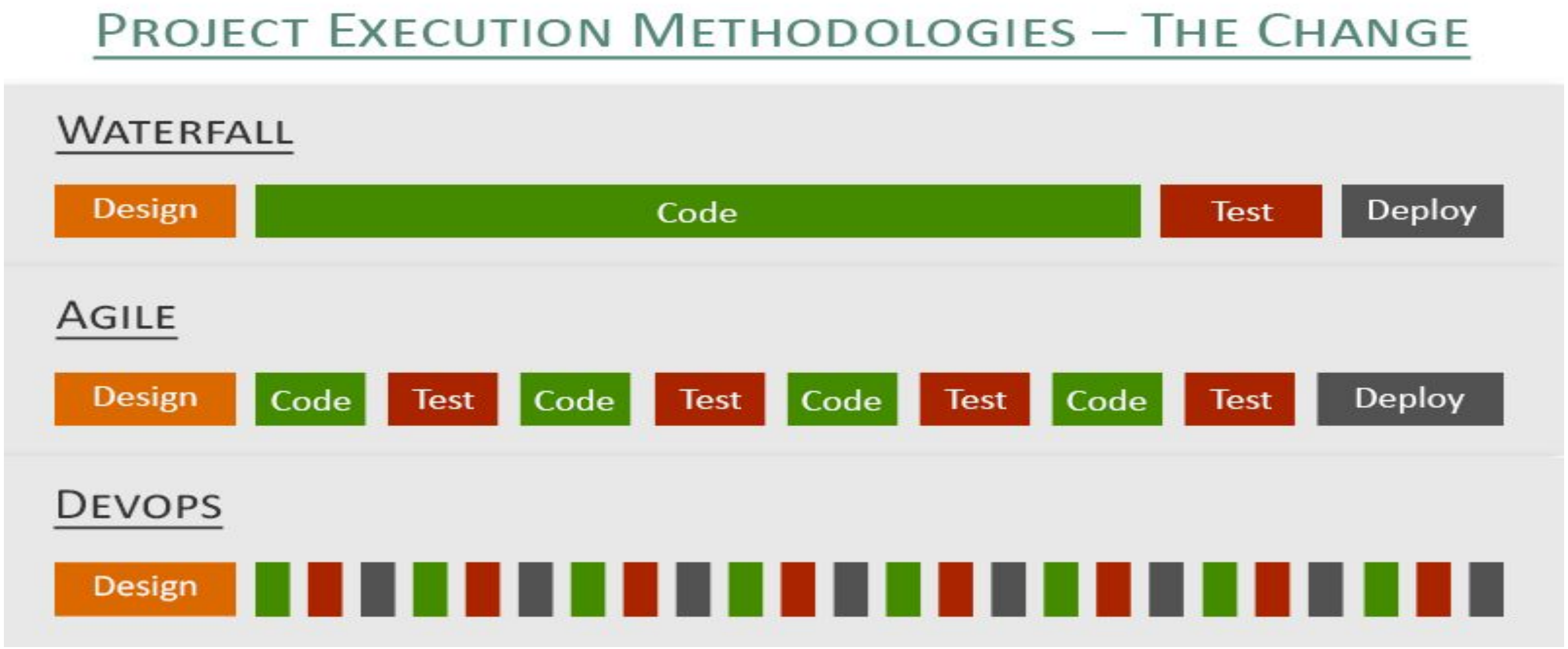
- Product delivery cycles continue to move slower and slower
- Fewer and less ambitious projects are undertaken
- The organization is no longer able to provide stable, reliable service to customers.

SDLC


WATERFALL
Discover
Plan
Build
Review
START
FINISH
AGILE
Discover
Plan
Build
Review
Discover
Plan
Build
Review
Discover
Plan
Build
Review
START
ITERATION
ITERATION
FINISH

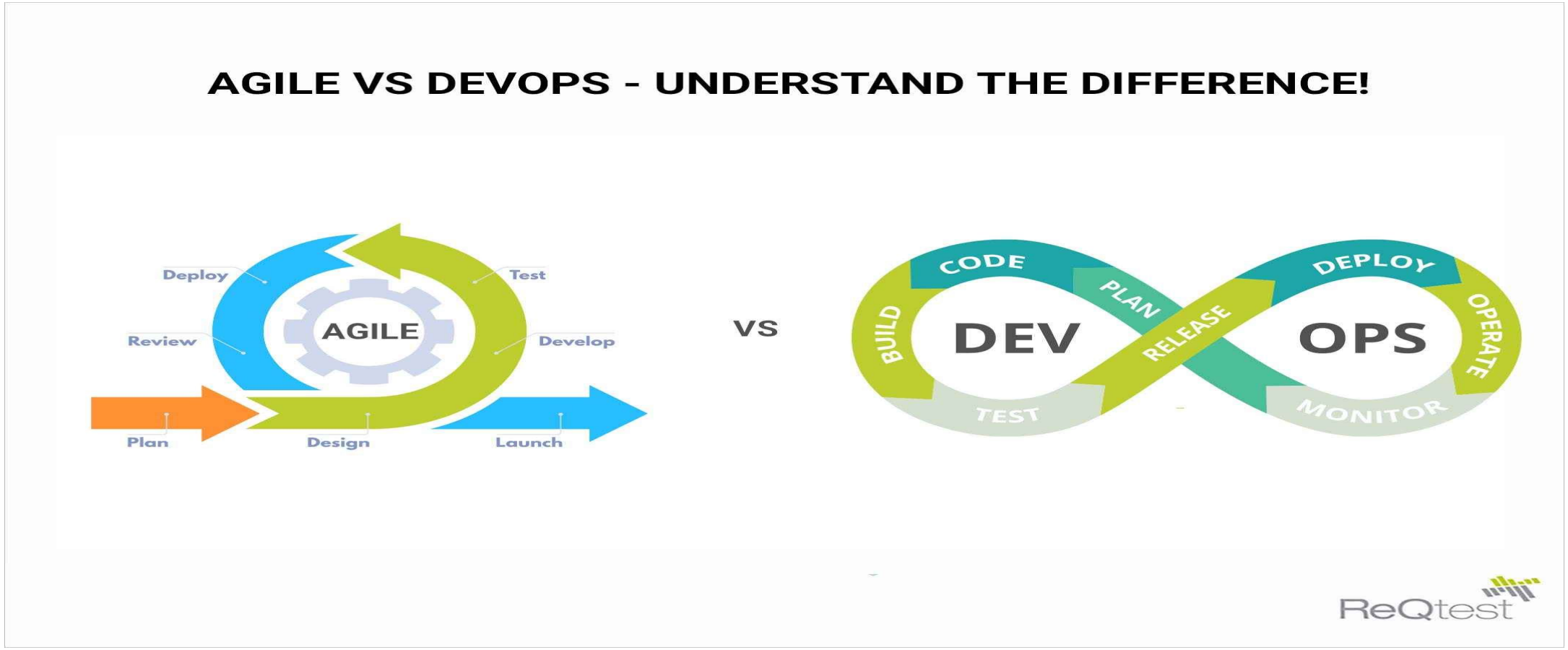
SDLC

From https://analyze.co.za/the-transition-to-devops/



Agile vs DevOps

From https://reqtest.com/agile-blog/agile-vs-devops/



DevOps

Continuous feedback and quick release/fail process

If code is correct, release and take feedback from stakeholder

If error, fail fast and give feedback to developer

Evolution of DevOps

THE AGILE MANIFESTO

Created in 2001 as lightweight set of values and principles against heavyweight software development processes

Relevant key principles:

- small batch sizes
- incremental releases instead of large, waterfall releases
- small, self-motivated teams.

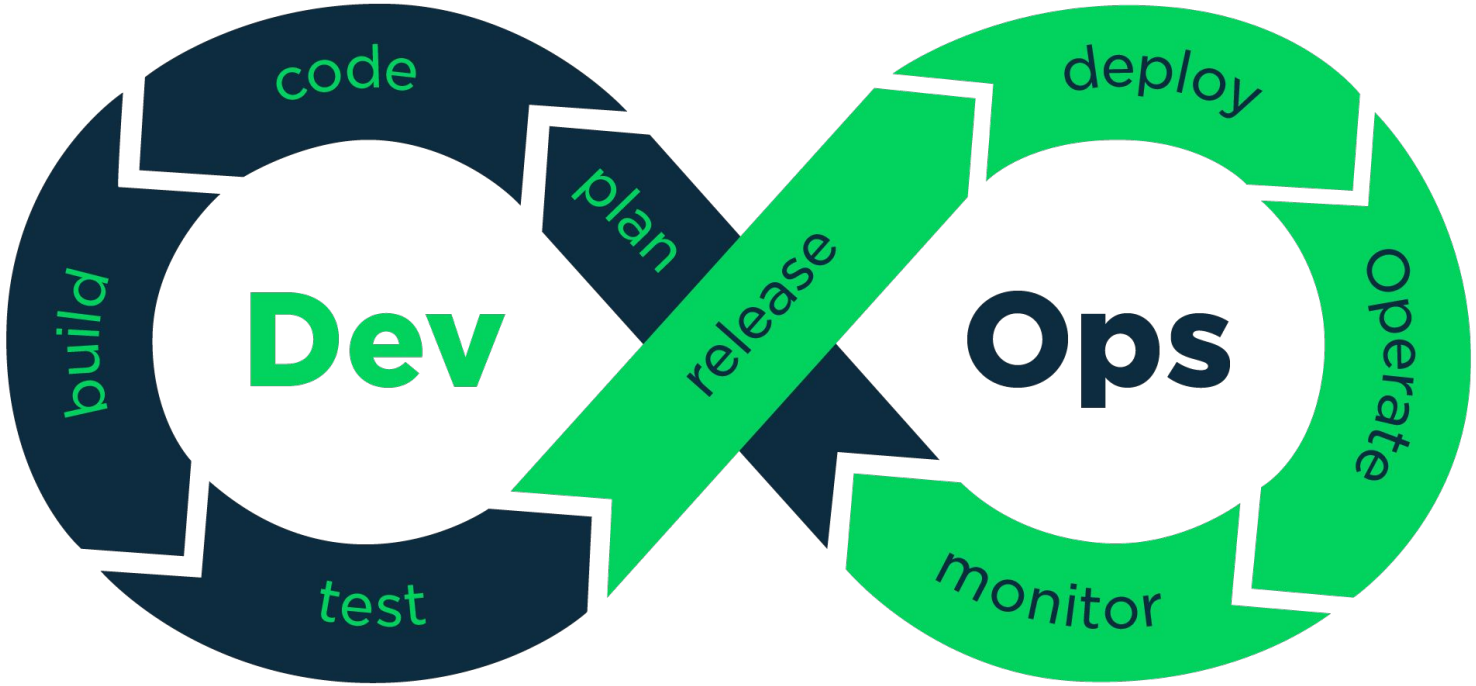
Evolution of DevOps

THE CONTINUOUS DELIVERY MOVEMENT

2006 - 2009

- Extending the concepts of continuous build, test, and integration, to continuous delivery
- “deployment pipeline”. Ensure that code and infrastructure are always in a deployable state
- code checked in to trunk can be safely deployed into production.

What is DevOps?



DevOps

DevOps is a combination of software development (Dev) and information technology operations (Ops). DevOps is a set of software development practices that aim to shorten the systems development life cycle while delivering features, fixes, and updates frequently in close alignment with business objectives.

Wikipedia

DevOps

DEVOPS IS THE COMBINATION OF CULTURAL PHILOSOPHIES, PRACTICES, AND TOOLS THAT INCREASES AN ORGANIZATION'S ABILITY TO DELIVER APPLICATIONS AND SERVICES AT HIGH VELOCITY: EVOLVING AND IMPROVING PRODUCTS AT A FASTER PACE THAN ORGANIZATIONS USING TRADITIONAL SOFTWARE DEVELOPMENT AND INFRASTRUCTURE MANAGEMENT PROCESSES. THIS SPEED ENABLES ORGANIZATIONS TO BETTER SERVE THEIR CUSTOMERS AND COMPETE MORE EFFECTIVELY IN THE MARKET.

AWS

DevOps Course By Ali Kahoot - Dice Analytics

DevOps

DevOps is a way to deliver software with shared pain and responsibility.

DevOps ultimately means building digital pipelines that take code from a developer's laptop all the way to revenue generating prod awesomeness!

Benefit of DevOps

Speed

- Fast feedback loops at every step of the process

Rapid Delivery

- Increase the frequency and pace of releases and bug fixes

Reliability

- Fast automated tests are run in production-like environments
- No firefighting for days or weeks

Delegation

- Teams take Responsibility and ownership

Benefit of DevOps

Scale

- Operate infrastructure and development processes at scale

Improved Collaboration

- More effective teams emphasizing values such as ownership and accountability.
- Developers and operations teams collaborate closely reducing inefficiencies and saving time

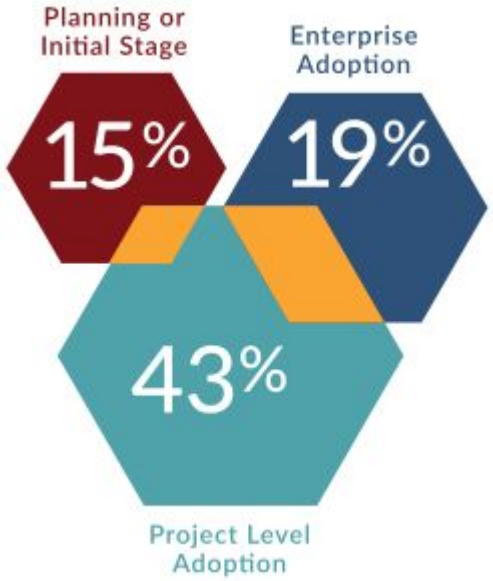
Security

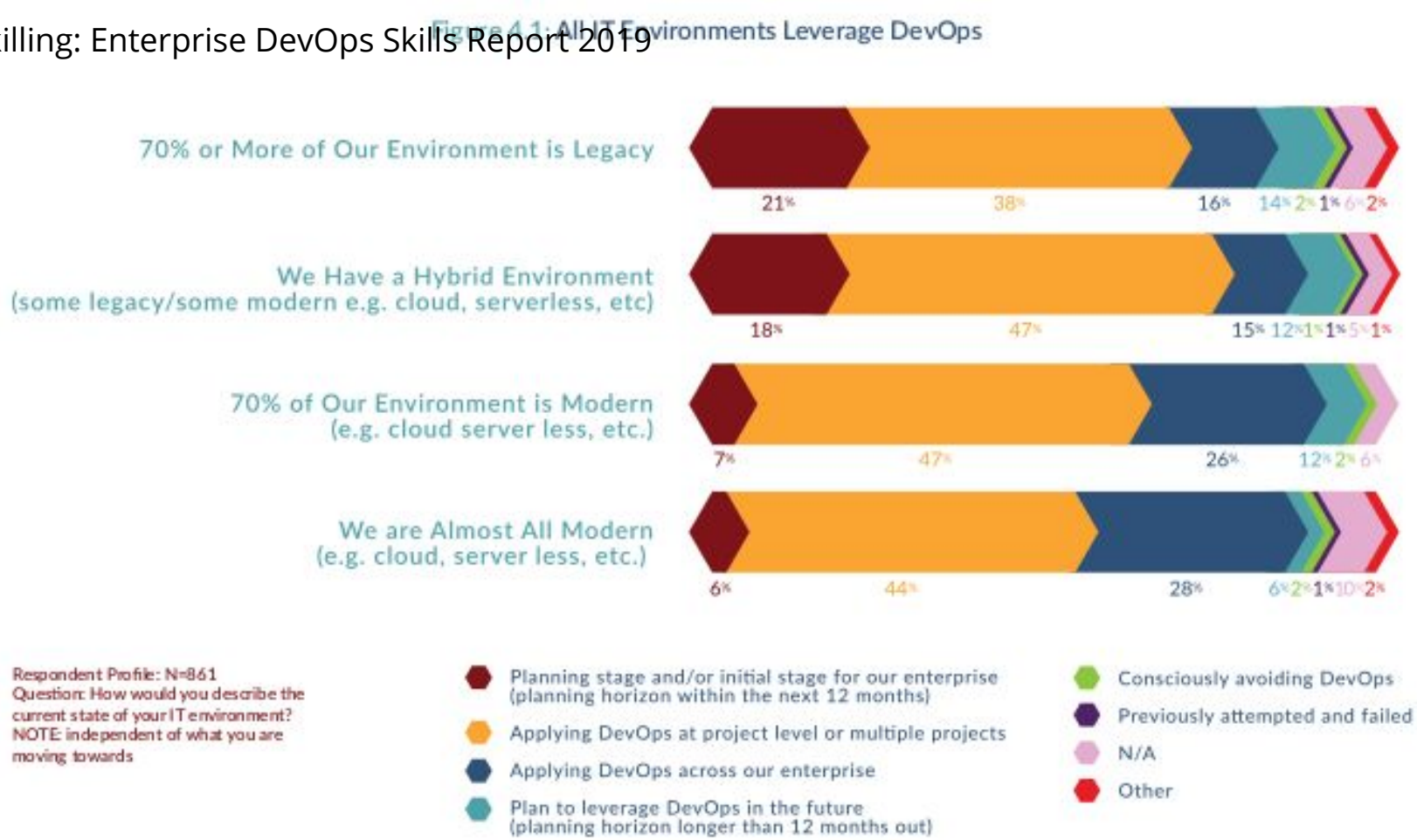
- Compliant with policies and configuration management

Why Should One Move to DevOps ?

DevOps Trends

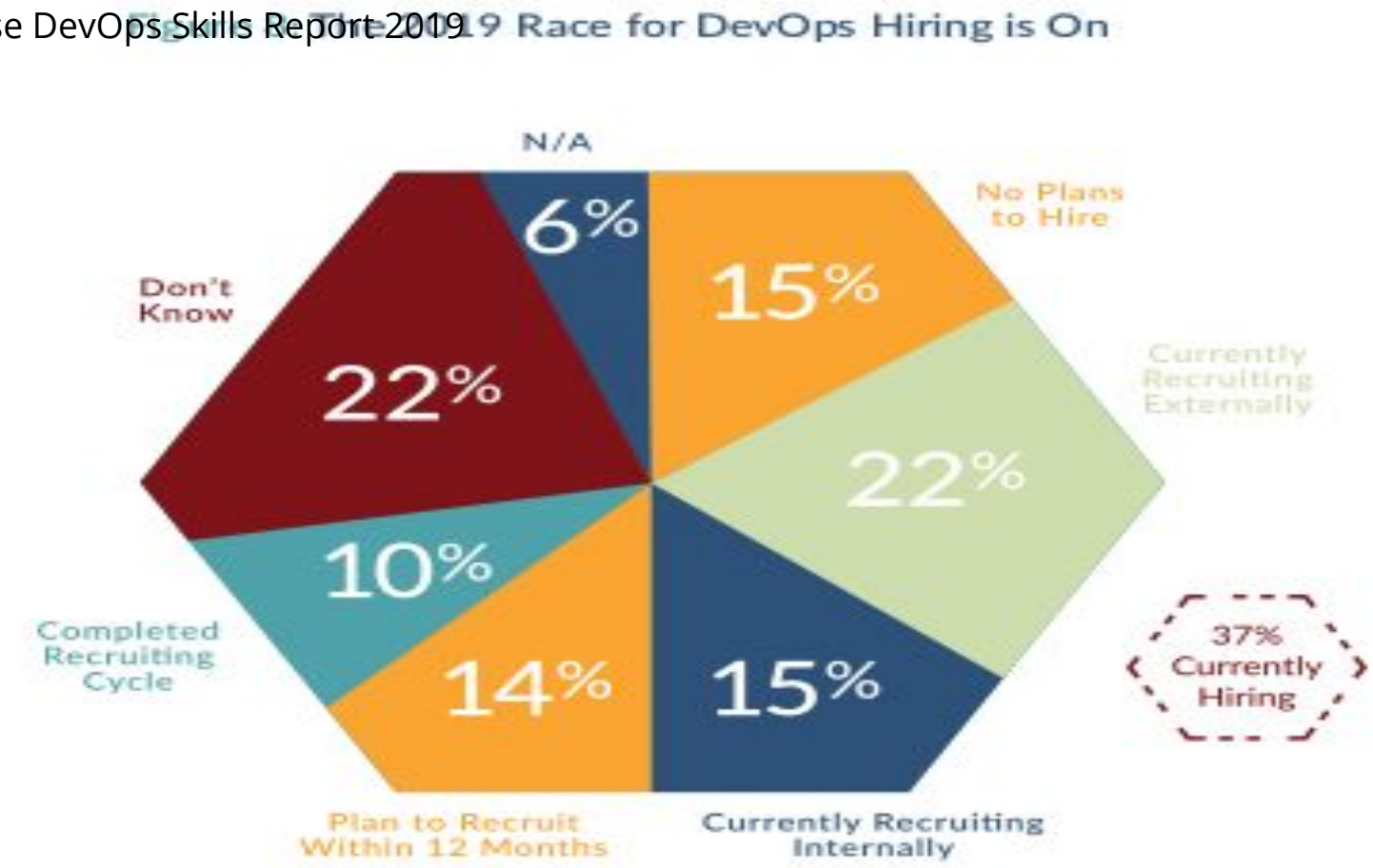
From: DevOps Institute | Upskilling: Enterprise DevOps Skills Report 2019





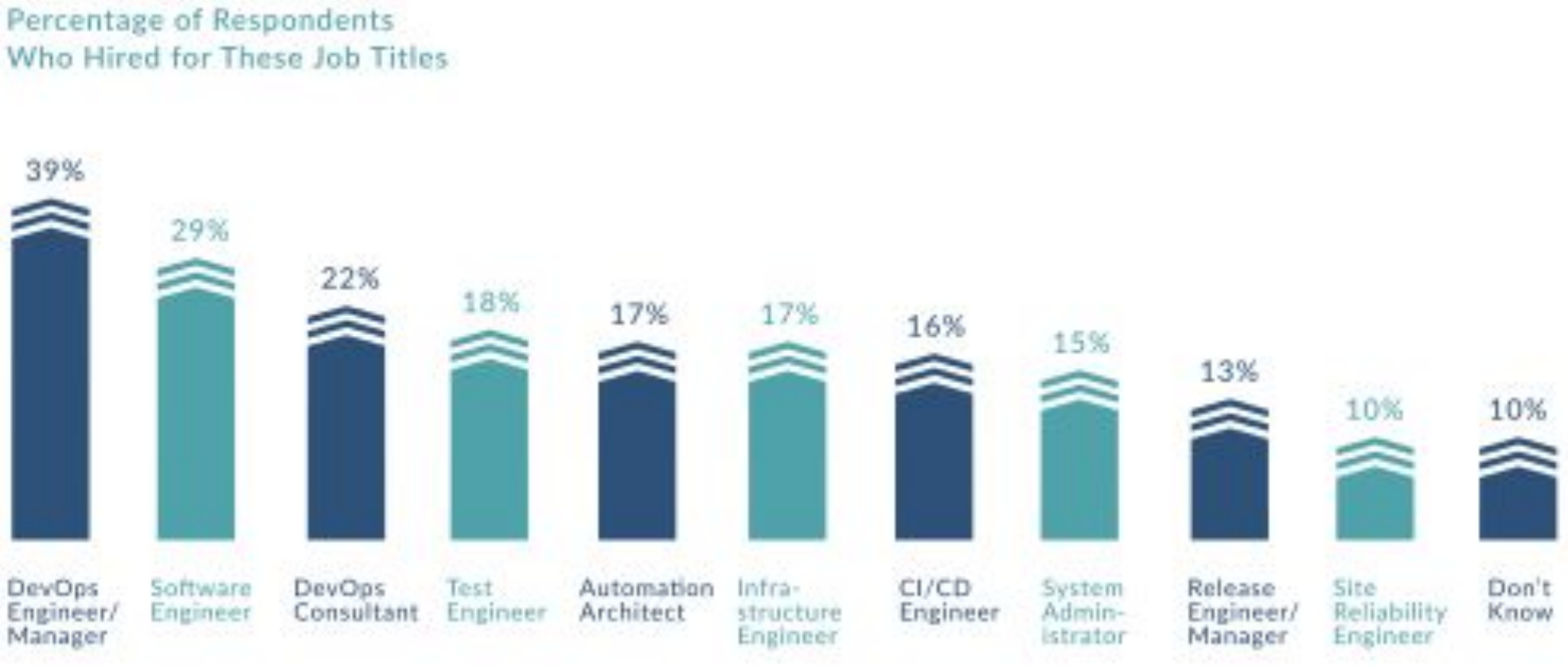
DevOps Trends

From: DevOps Institute | Upskilling: Enterprise DevOps Skills Report 2019

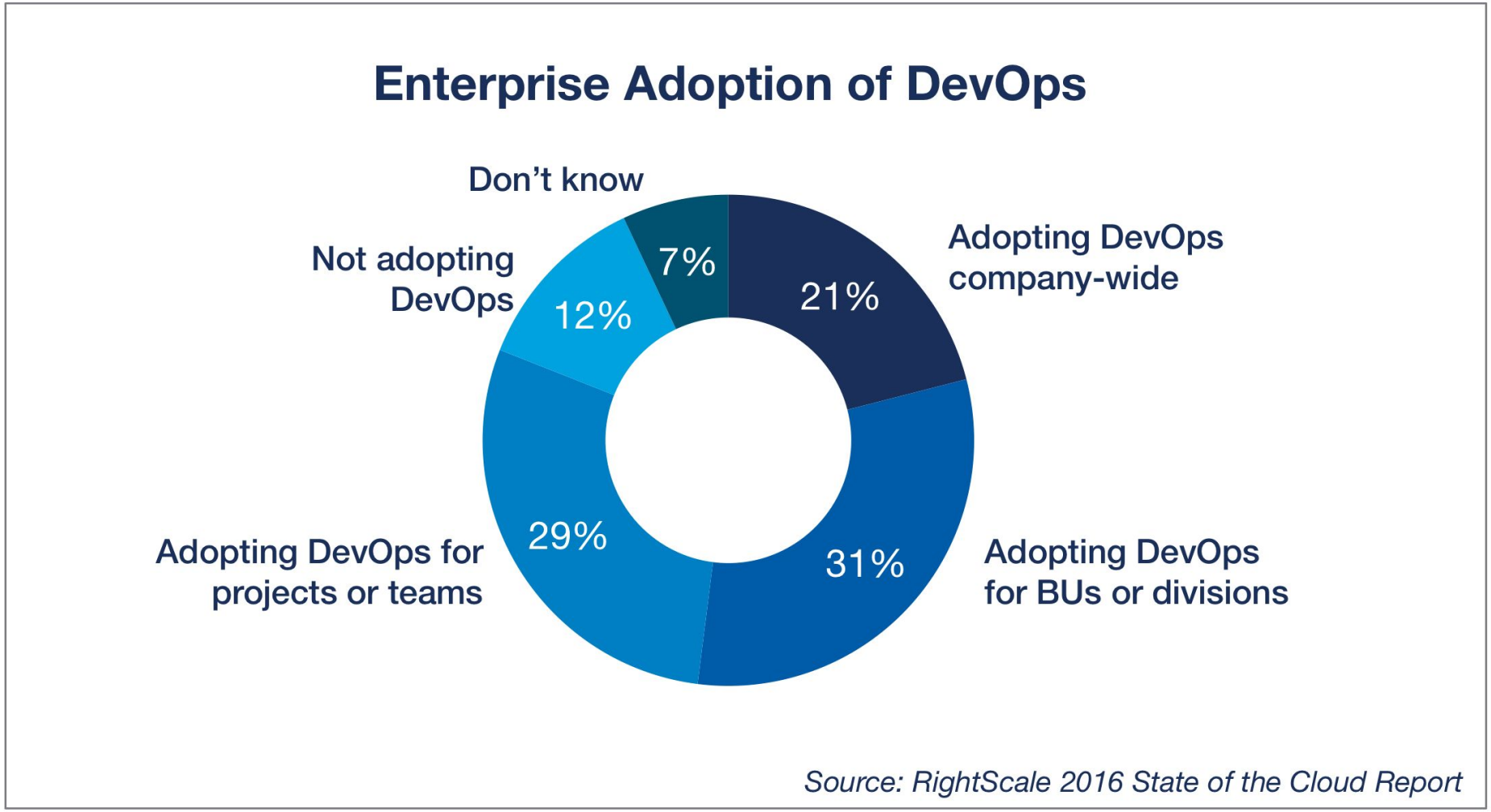


DevOps Trends

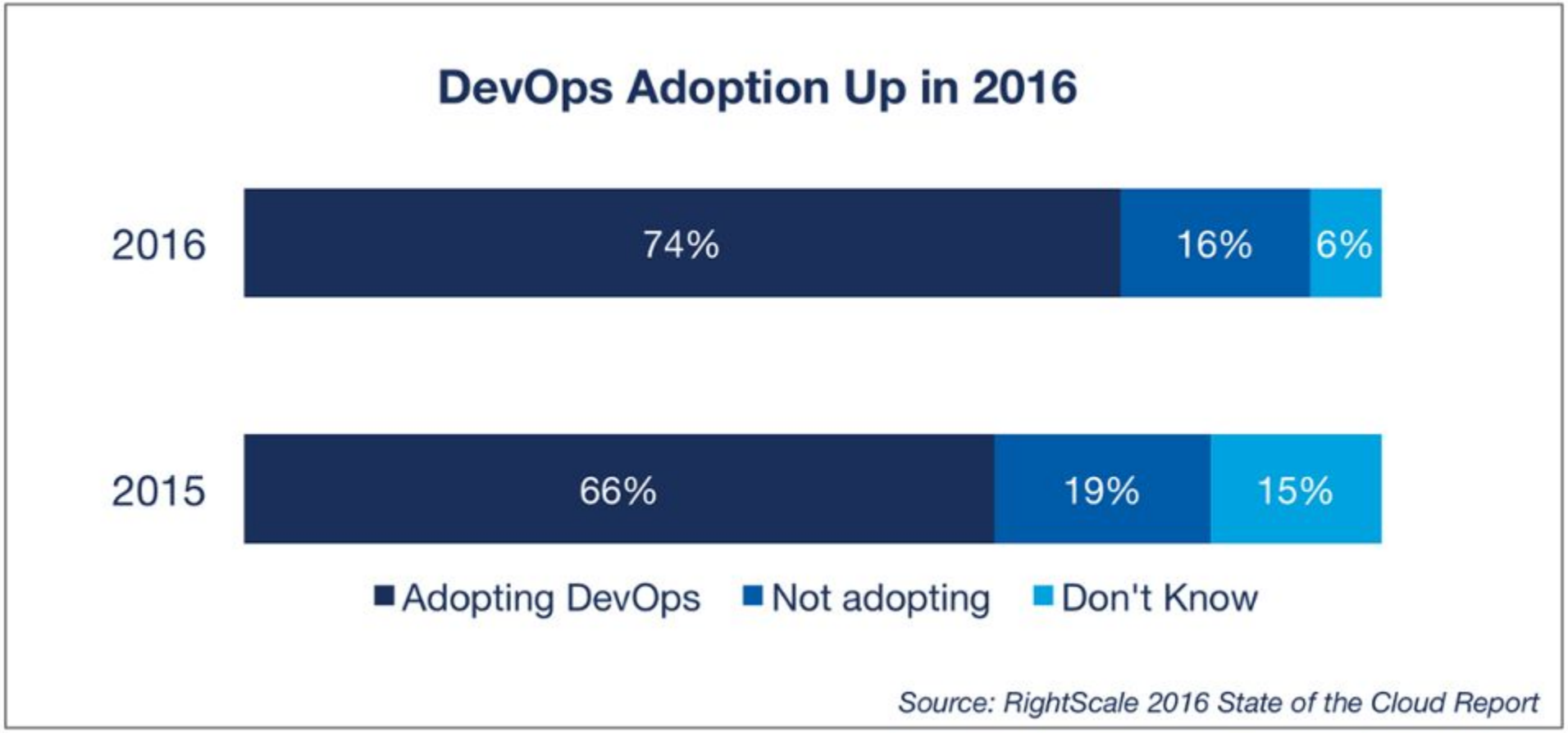
From: DevOps Institute | Upskilling: Enterprise DevOps Skills Report 2019



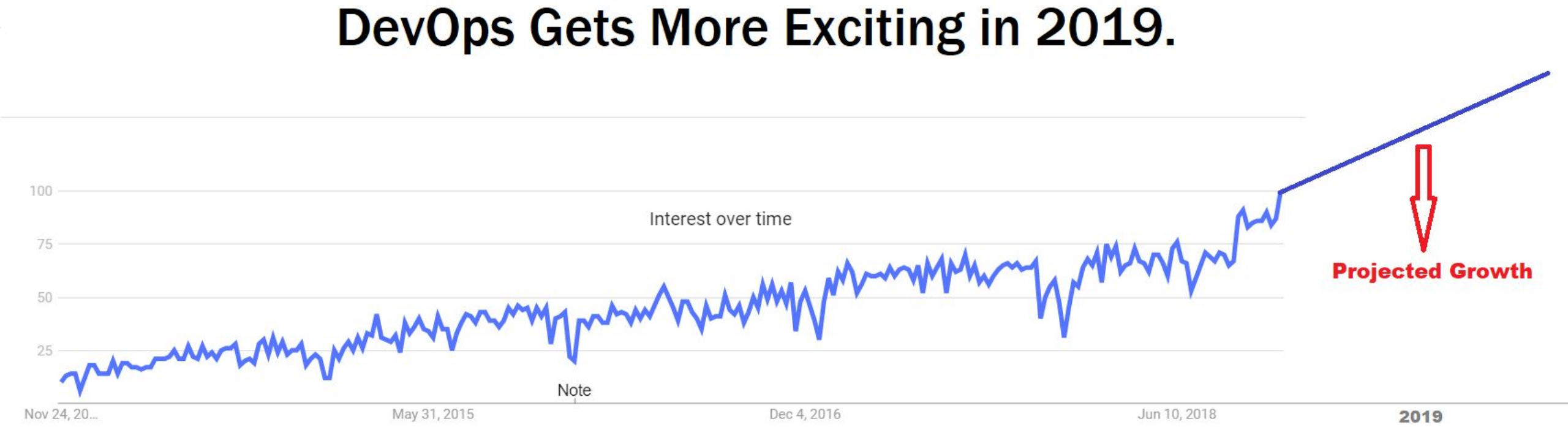
DevOps Trends



DevOps Trends



DevOps Trends



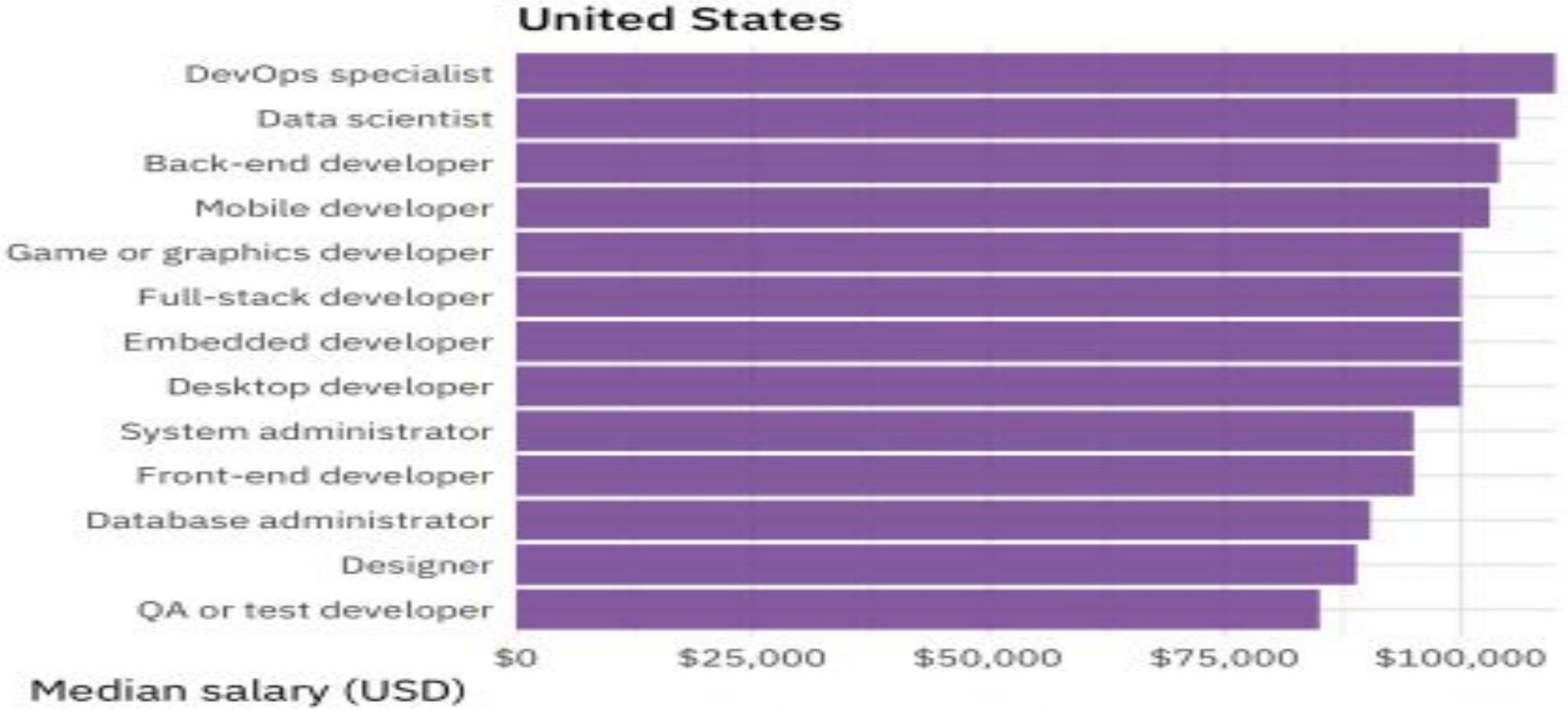
DevOps Trends

From: https://www.datacareer.de/blog/devops-engineer-salaries-in-europe-in-2018/

Country	Annual Salary in 1000 EUR	OECD Price level	Adjusted Salary
Italy	26.3	91	28.9
Belgium	34.2	101	33.8
France	42.7	101	42.3
Spain	36.8	83	44.4
Ireland	47.5	102	46.6
United Kingdom	52.4	108	48.5
Netherlands	52.5	103	50.9
Germany	55.3	98	56.5
Switzerland	83.5	142	58.8
Austria	60	101	59.4

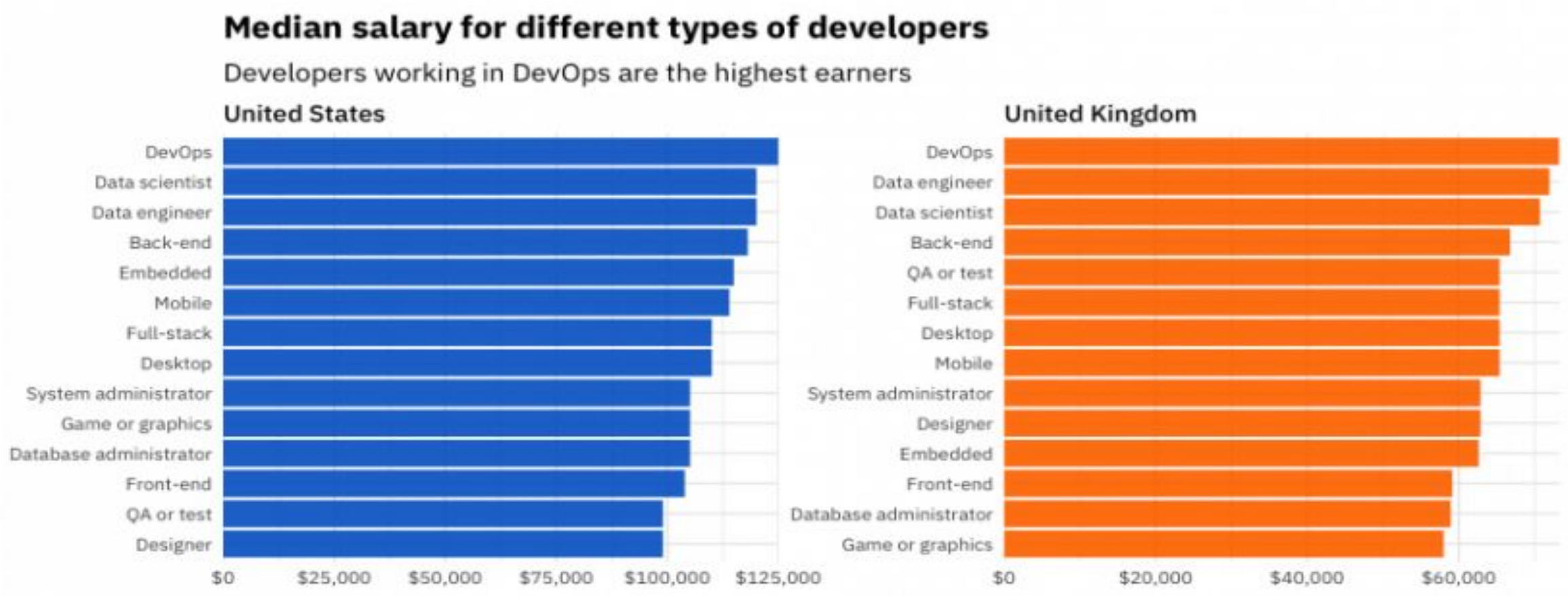
DevOps Trends

From: https://jaxenter.com/stack-overflow-2018-salary-149230.html



DevOps Trends

From: https://stackoverflow.blog/2019/10/16/coding-salaries-in-2019-updating-the-stack-overflow-salary-calculator/

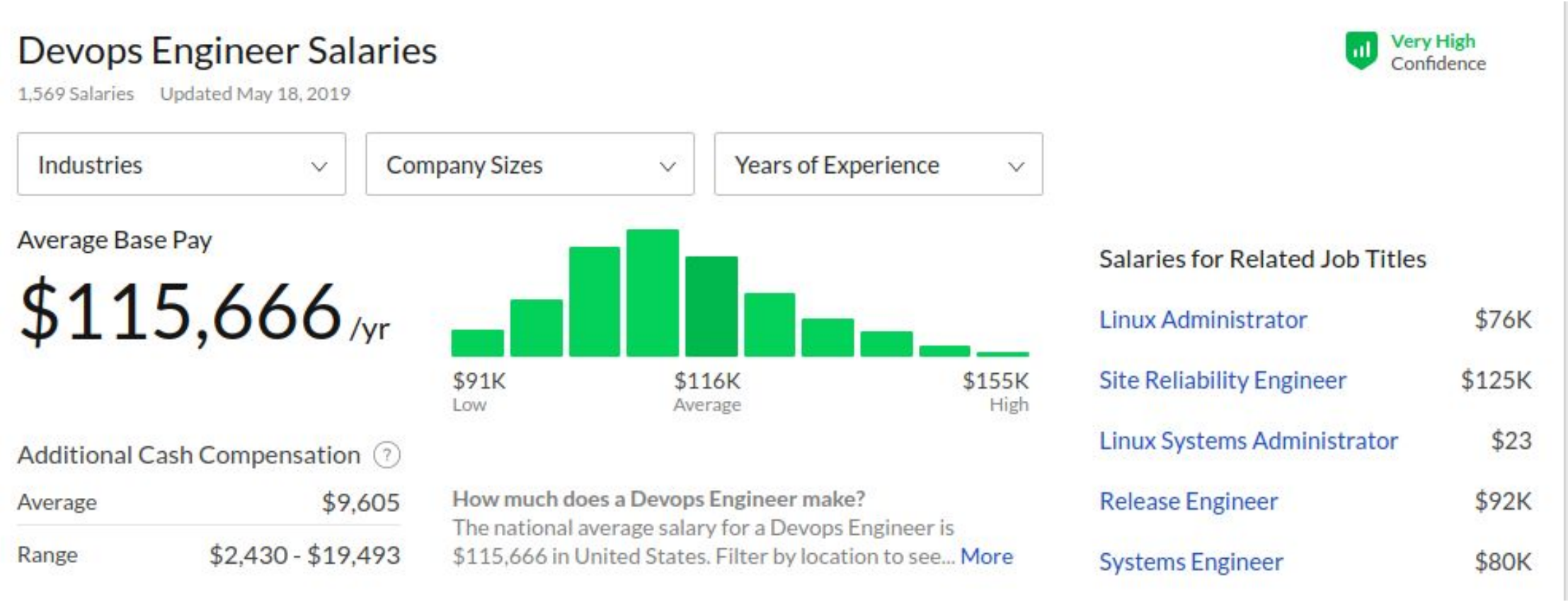


DevOps Trends

From: https://stackoverflow.blog/2019/10/16/coding-salaries-in-2019-updating-the-stack-overflow-salary-calculator/

DevOps Trends

From: https://www.glassdoor.com/Salaries/devops-engineer-salary-SRCH_KO0,15.htm



DevOps Trends

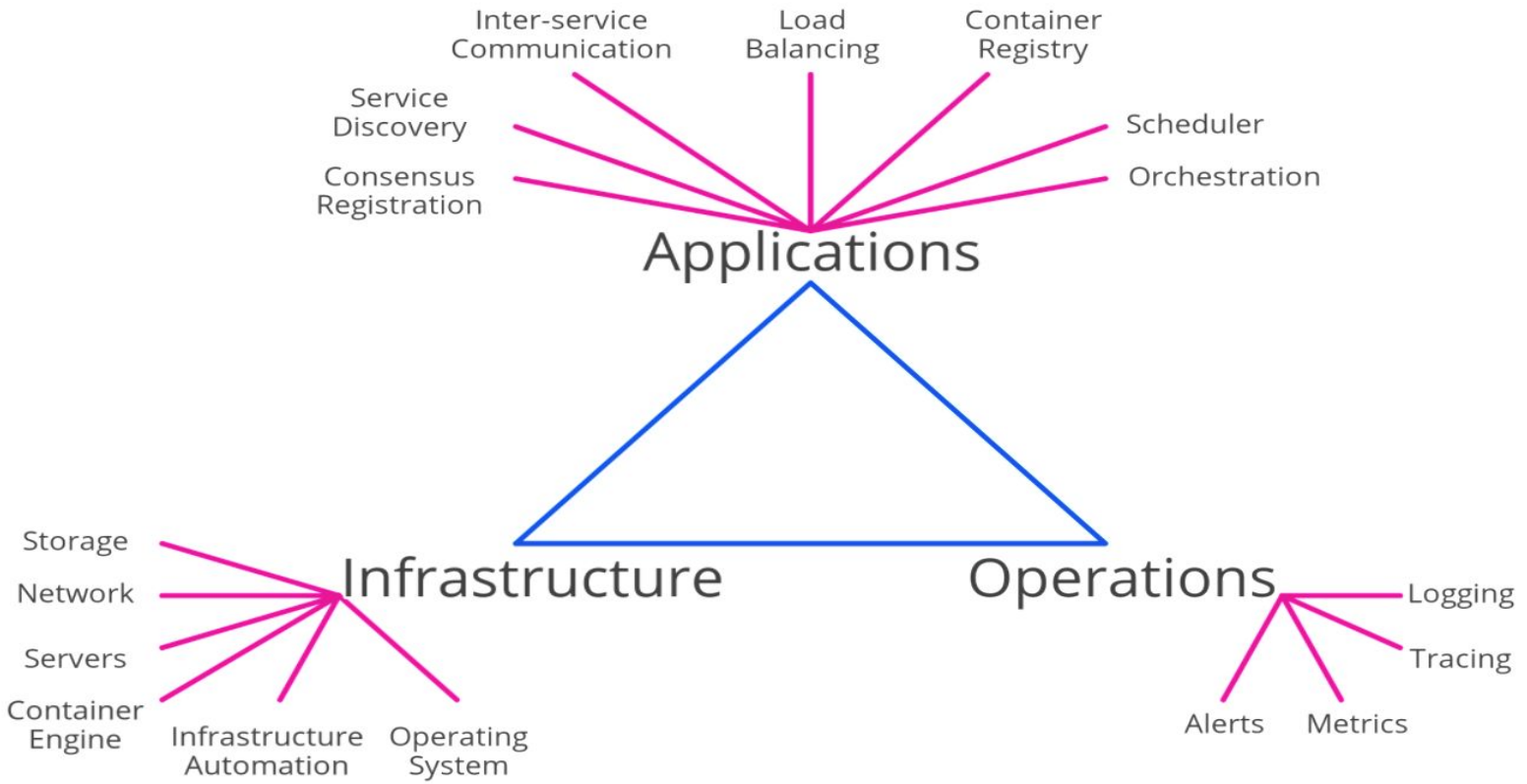
DevOps Engineer: #1 Hardest Job to Fill

Article: https://www.logicworks.com/blog/2016/06/devops-engineer-hardest-job-find-skills-shortage/

Recent Acquisitions

- Microsoft acquired Github
- IBM acquired Redhat
- Mirantis acquired Docker

DevOps Artifacts



DevOps Course By Ali Kahoot - Dice Analytics

DevOps Best Practices

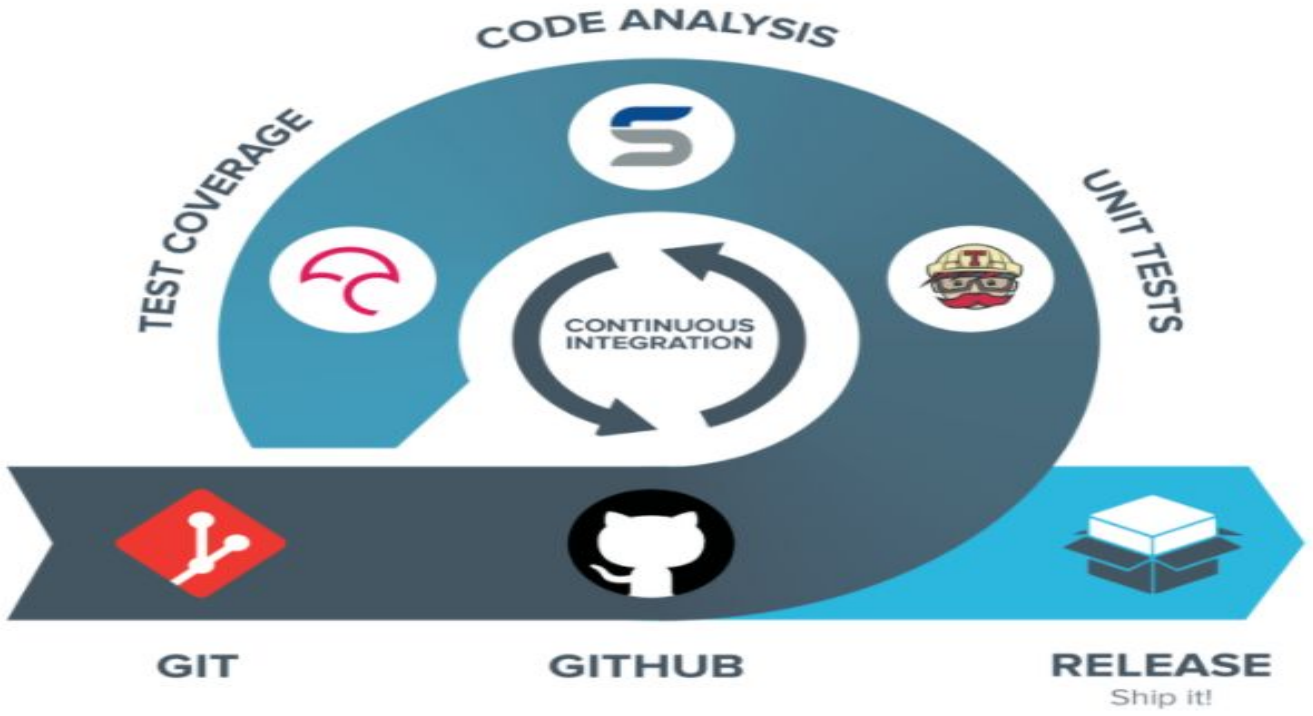
1. Continuous Integration
2. Continuous Delivery/Continuous Deployment
3. Microservices
4. Cloud Computing
5. Infrastructure as Code
6. Monitoring and Logging
7. Communication and Feedback

1. Continuous Integration

- Merge code changes from different developers into a central repository
- Automated builds and tests are run
- Key goals are to fail fast and find and address bugs quicker
- Benefits Developers most, Less merge conflicts

1. Continuous Integration

From https://www.silverstripe.org/blog/developers-how-we-use-continuous-integration-at-silverstripe/



2. Continuous Delivery

- CI stage is approved
- A small build cycle for short sprints for releasing small features
- Code changes are automatically built & tested
- Can be deployed to a test environment
- Can use branching strategy (other than master)
- Mindset to always have a deployment-ready build artifact.

2. Continuous Deployment

- CI/CD stages are approved
- The change approved from CI/CD are deployed to production.
- Can use branching strategy(master)
- Release features to get feedback from user

Continuous Delivery != Continuous Deployment

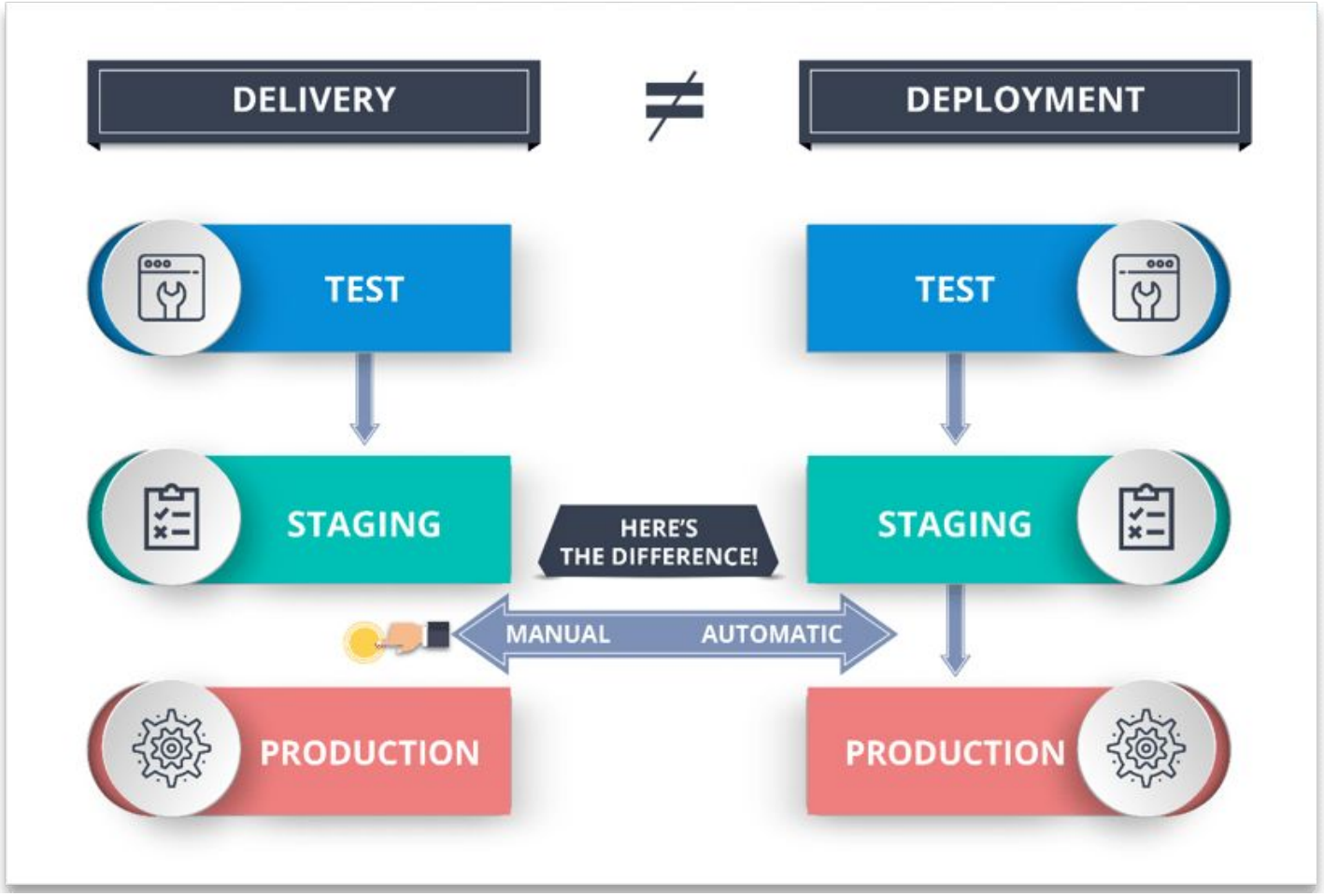
Continuous Integration

+

Automated software release

+

Manual Deployment To Production



Continuous Integration

+

Continuous Delivery

+

Automated Deployment To Production

From Edureka: https://www.edureka.co/blog/continuous-delivery-vs-continuous-deployment/

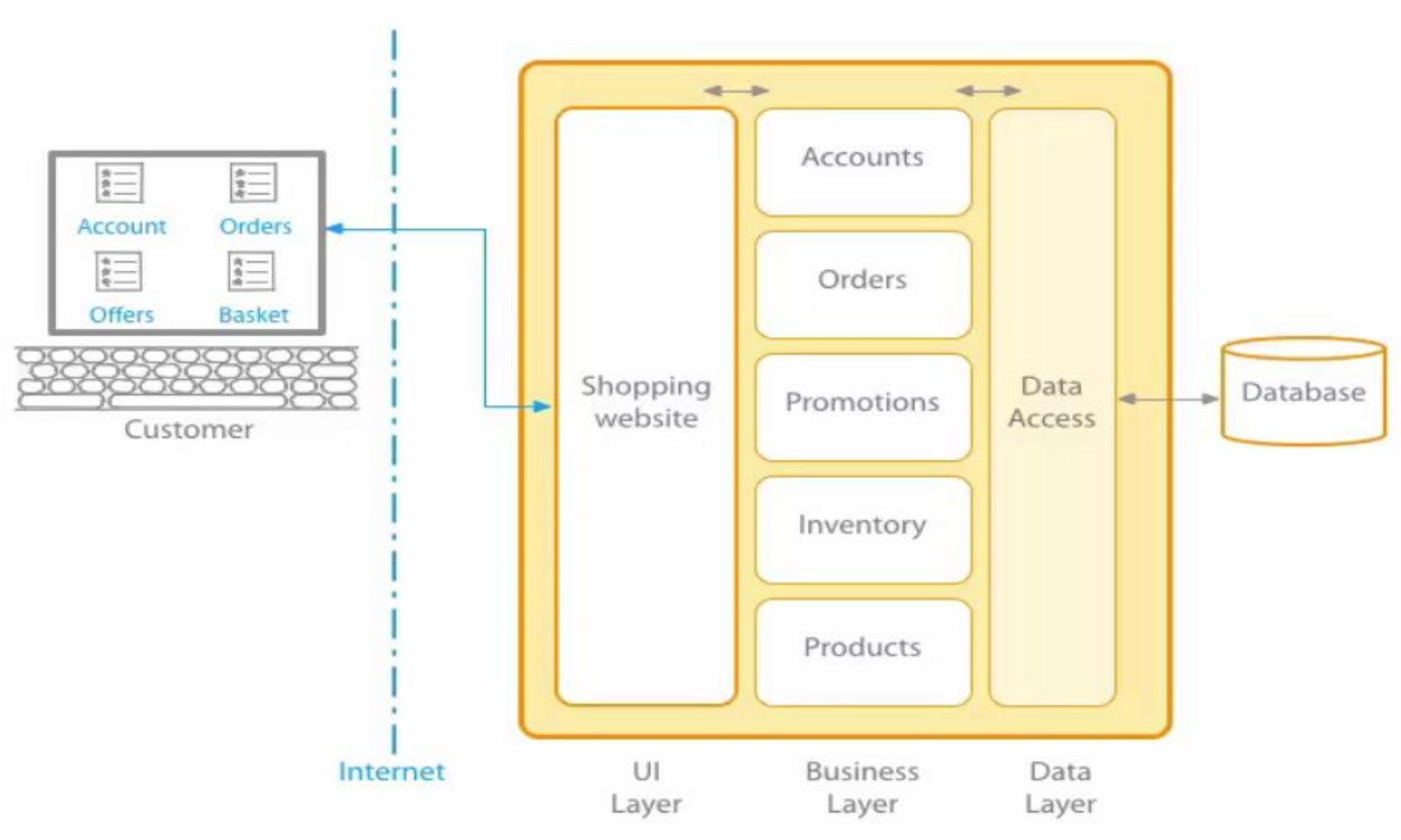
CI/CD Tools

- Git / SVN / TFS
- Github / Gitlab / Bitbucket
- Jenkins
- Github Actions
- Gitlab CI
- Tekton CI
- Circle CI
- IBM DevOps
- AWS CodePipeline
- Azure DevOps
- Flux
- ArgoCD

3. Microservices

- Design approach to build a single application as a set of small services.
- Each service runs in its own process and communicates with other services through a well-defined interface.
- Each service can be updated and deployed independently, decreasing risk of update and impact of errors.
- Separate services meaning
  - Independent CI/CD pipelines
  - Independent Ownership
  - Independent Responsibility

Monolithic Service



Concerns of Monolithic

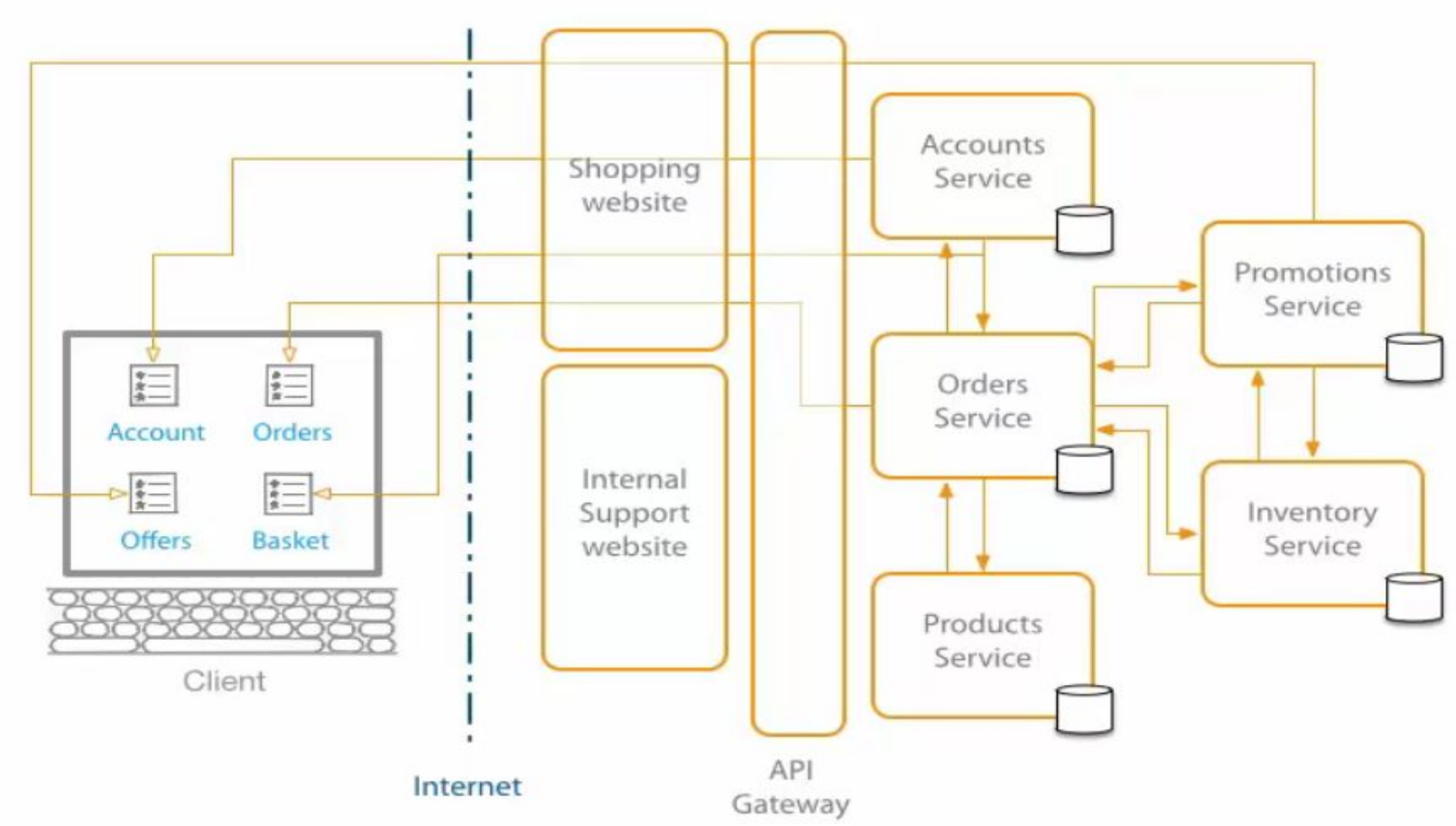
- Difficult to Scale
- Long Time to Ship
- Complexities of Growing Applications
- No Clear Ownership
- Failure Cascade
- Wall Between Dev and Ops
- Stuck in a Technology/Language

Deciding a Microservice?

Factors deciding if it should be a Microservice

- Multiple Rates of Change
- Independent Life Cycles
- Independent Scalability
- Isolated Failure
- Technology/Language stack
- Separate Team/Ownership

Microservices

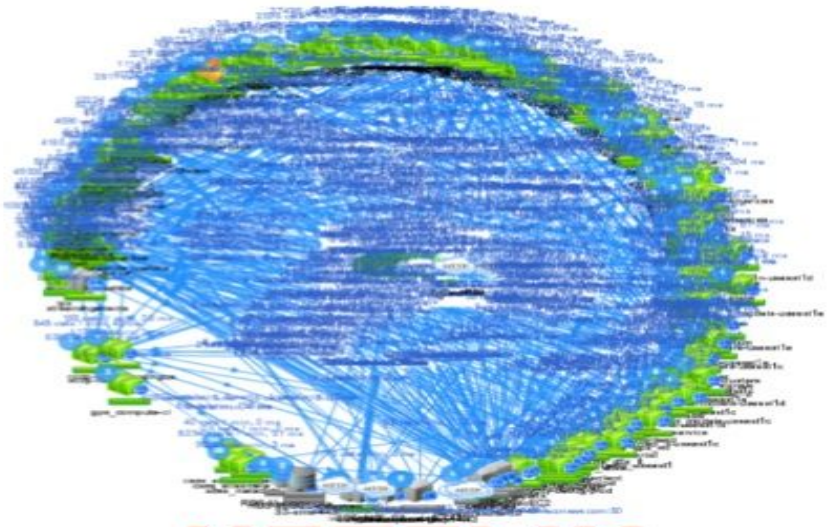


Examples

450 microservices

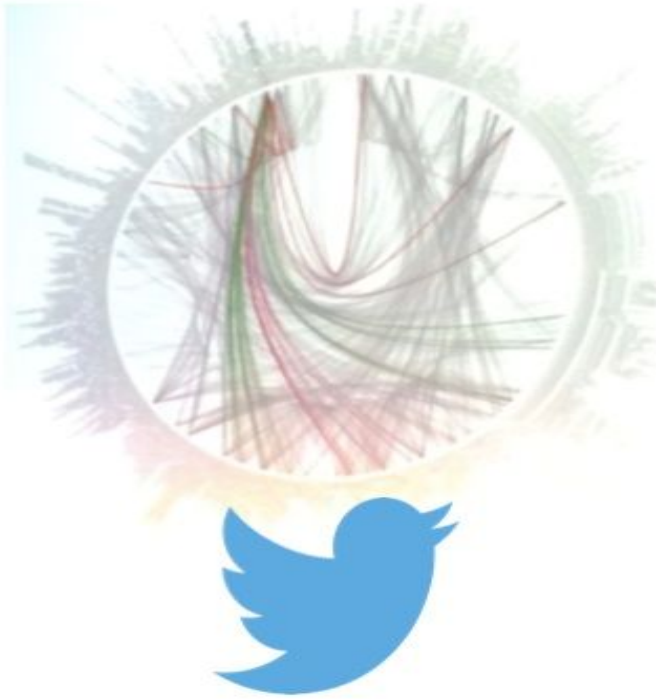
HAIL O

500+ microservices



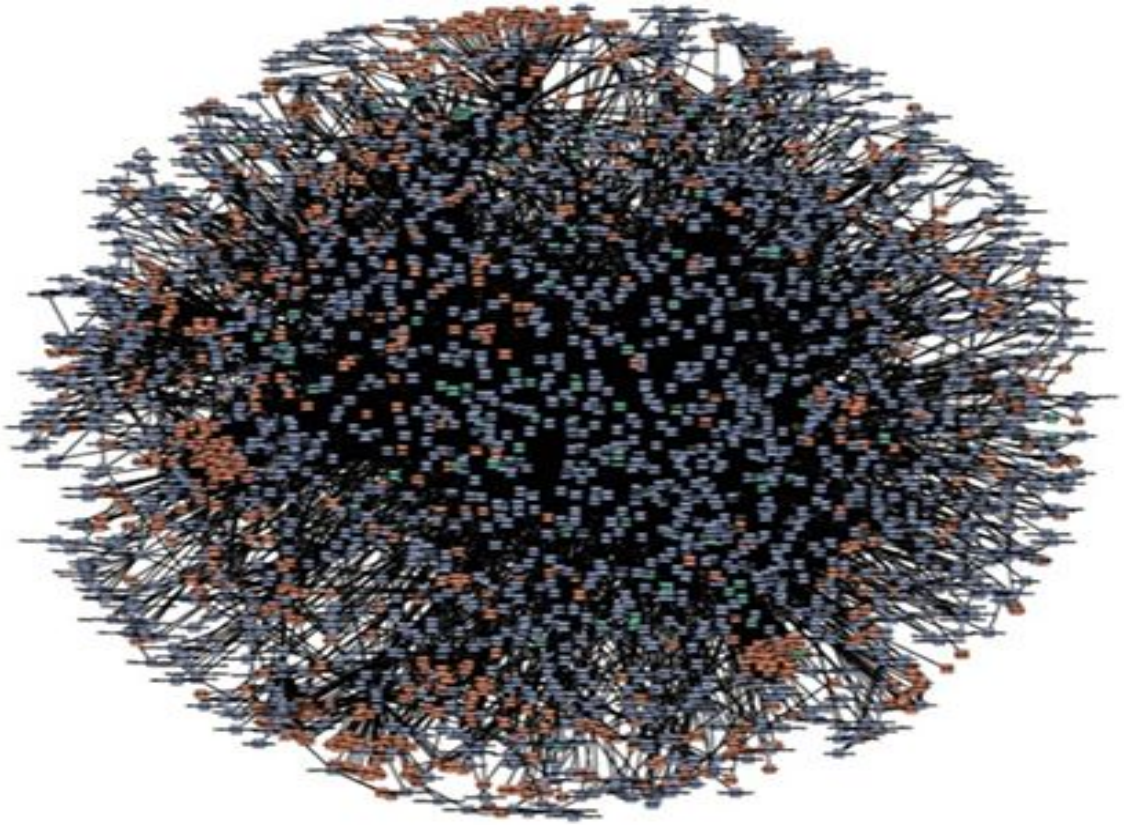
NETFLIX

500+ microservices

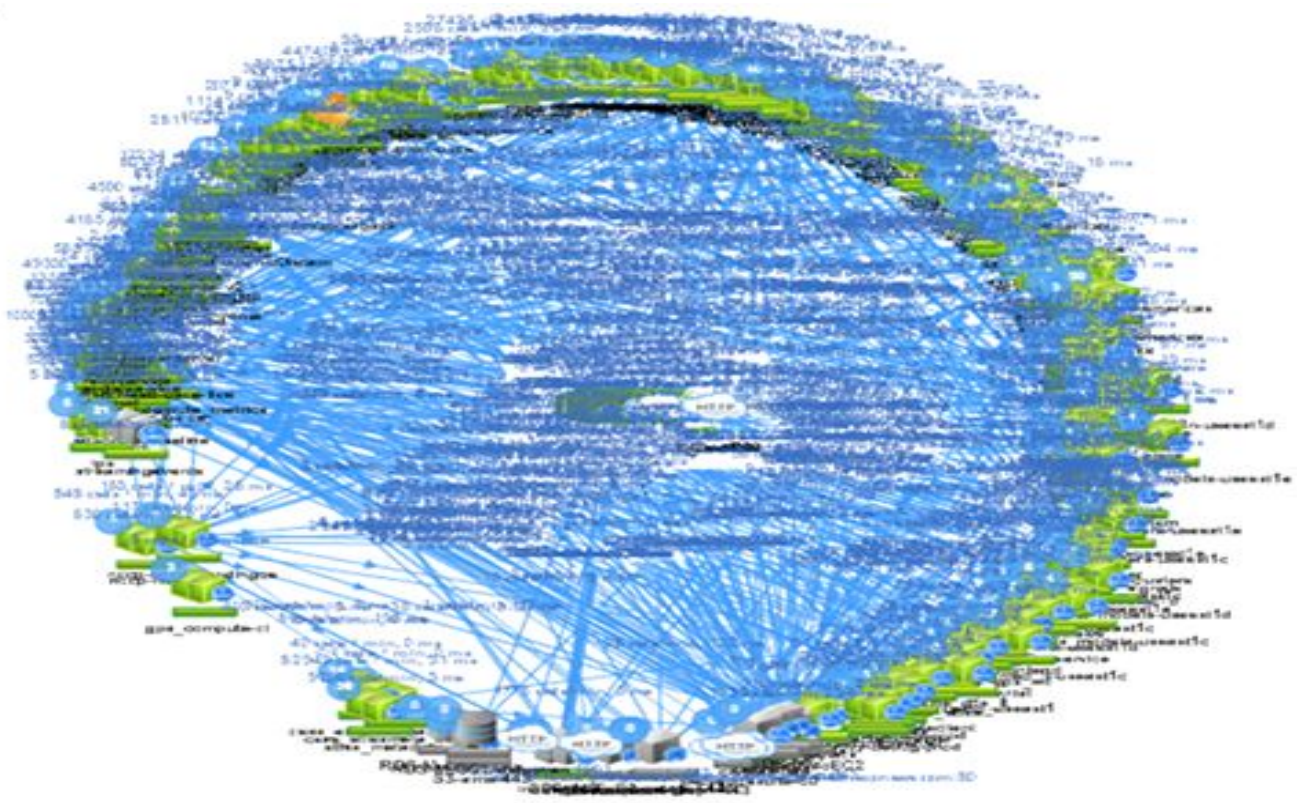


Source:
Netflix: http://www.slideshare.net/BruceWong3/the-case-for-chaos
Twitter: https://twitter.com/adrianco/status/441883572618948608
Hail-o: https://sudo.hailoapp.com/services/2015/03/09/journey-into-a-microservice-world-part-3/

Examples



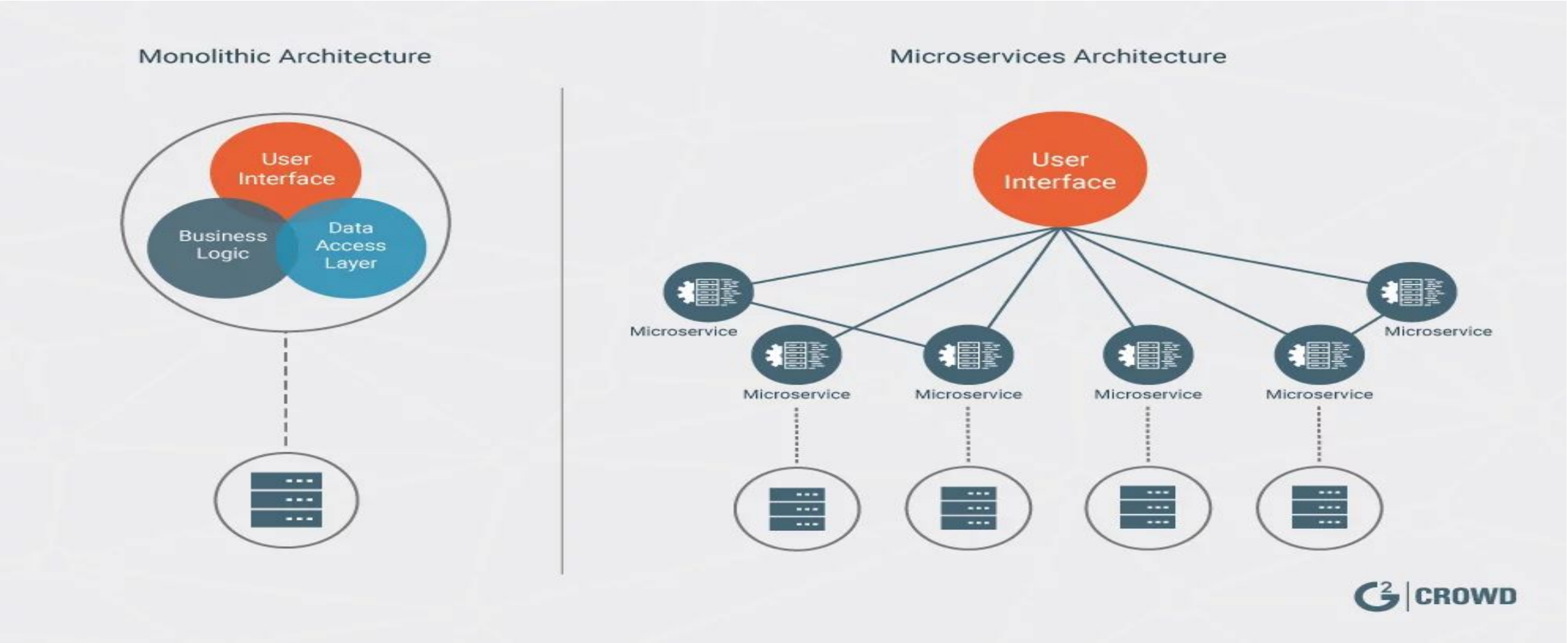
amazon.com



NETFLIX

Microservices vs Monolithic

From: https://blog.g2crowd.com/blog/trends/digital-platforms/2018-dp/microservices/



Pros & Cons of Microservices

Pros

- Freedom to use technology
- Responsible of single business capability
- Separate Ownership & Tracking
- Frequent Software Releases
- Parallel releases & feature requests
- No Single Point of Failure
- Code Understanding
- Each service scaled independently
- No Delay for Developers
- Can be reused

Cons

- Complex architecture
- Single functionality becomes distributed so latency
- Difficult to trace a call and which microservice is taking time
- A good amount of integration/e2e tests are required
- Data division for Microservices
- Difficult to maintain transaction safety

Containers & Kubernetes

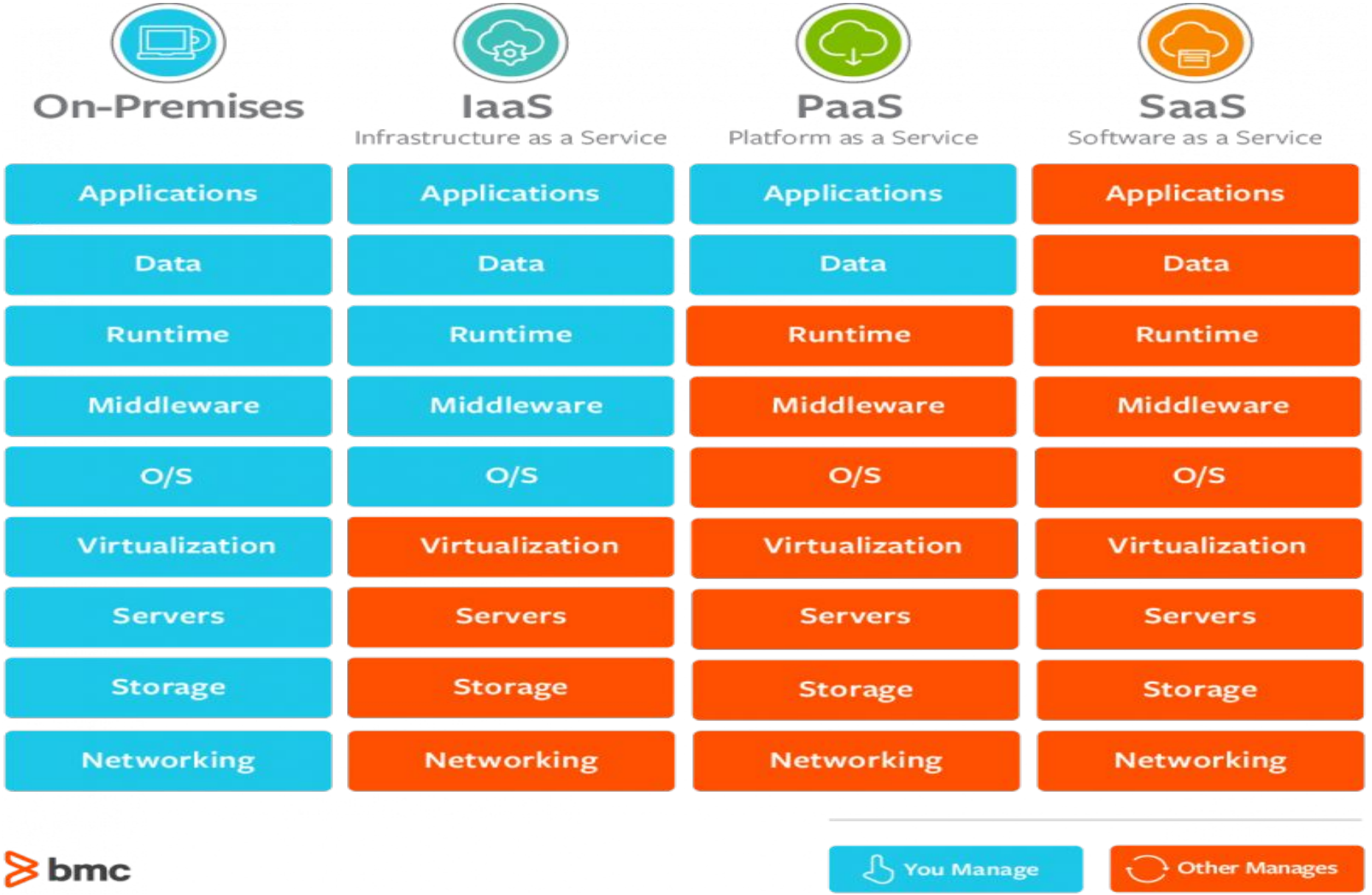
Microservices Related Tools

- API Gateways
  - Own, Mulesoft, Kong, 3scale, Cloud Meshery, Tyk, Azure/AWS API Gateway
- Messaging:
  - Kafka, Rabbitmq, Amazon Simple Queue Service
- Kubernetes:
  - Helm, EKS, AKS, IKS, GKE, Openshift
- Docker Containers
  - Azure CI, AWS ECS, Google CE, Docker Compose, Docker Swarm
- Service Mesh
  - Istio, Linkerd
- Serverless
  - AWS Lambda, Azure Functions, Kubeless,

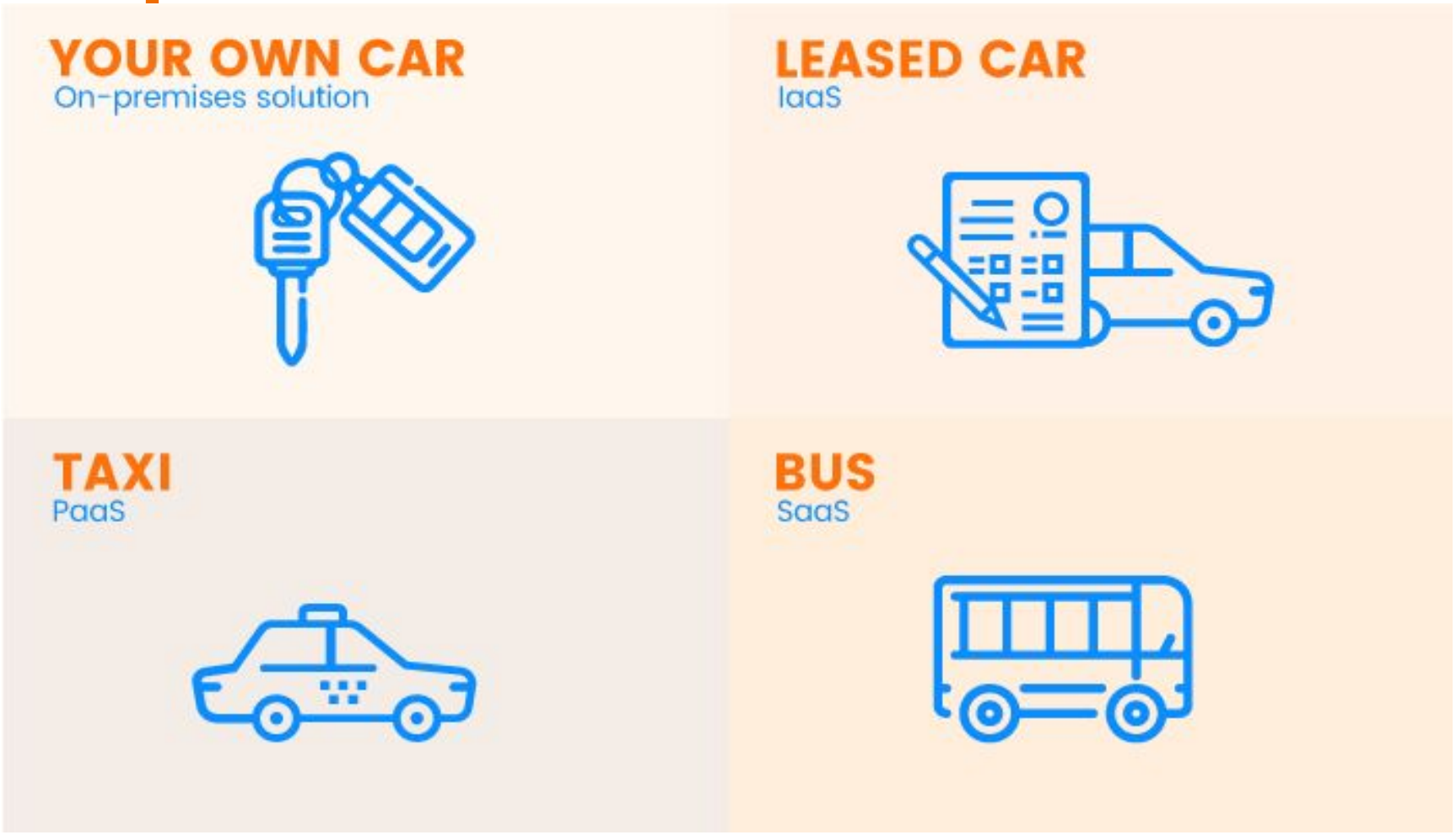
4. Cloud Computing

- On-demand computing resources
- Don’t have to buy resources
- Elastic resources - Scale up or down quickly and easily to meet demand
- Pay for what you use only
- Self service - All the IT resources you need with self-service access

From https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/



Examples


YOUR OWN CAR
On-premises solution
LEASED CAR
IaaS
TAXI
PaaS
BUS
SaaS

Examples

- IAAS: Amazon EC2, Rackspace, Google Compute Engine etc

- PaaS: Google App Engine, Cloud Foundry, Engine Yard, EKS, AKS etc.

- SaaS: Salesforce, Google Docs, Office 365, Basecamp, Facebook etc

FaaS/Serverless

FaaS: Function as a Service

Can run a function in cloud

Don’t need to worry of platform

Platform scaling and other issues managed by Provider

Azure Functions, AWS lambda, AWS Fargate, Azure Container Instance etc

Clouds

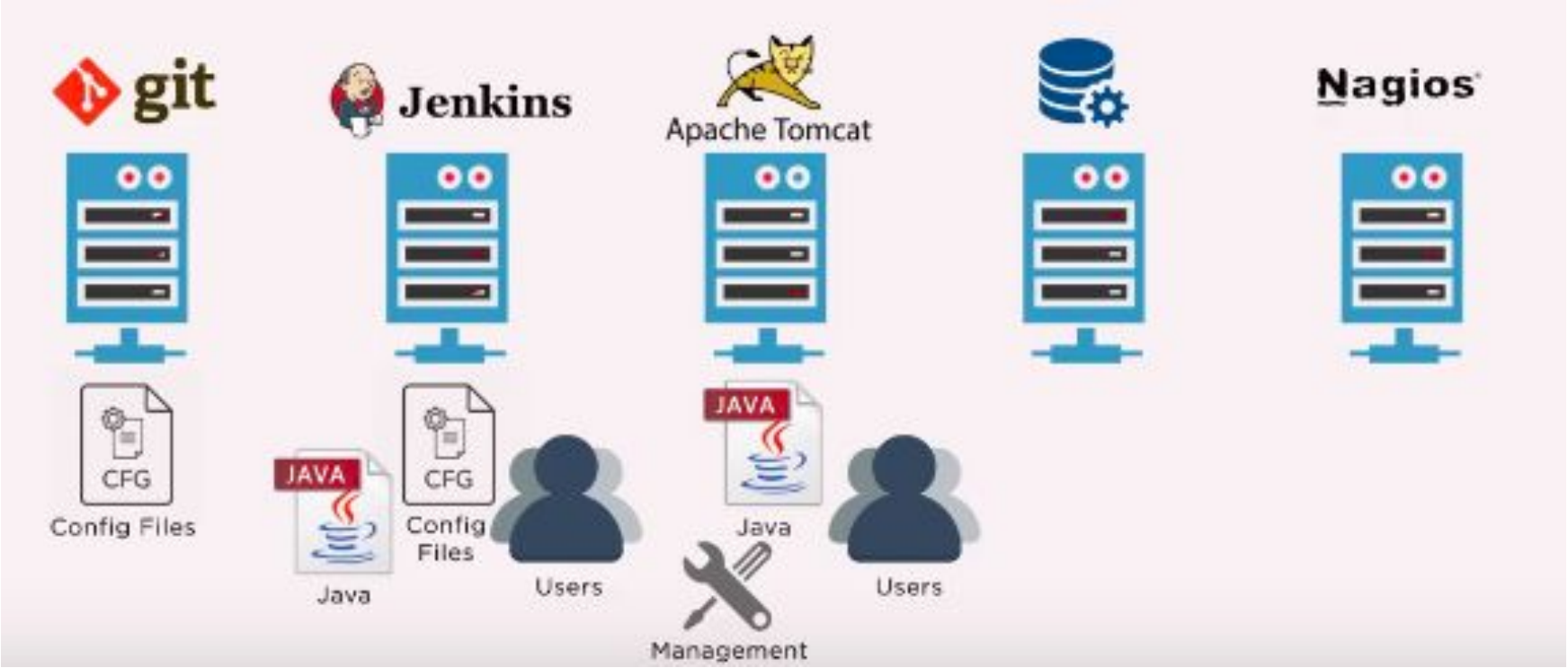
- AWS
- Azure
- Google Cloud
- Oracle
- IBM Cloud
- Digital Ocean
- Openstack

5. Infrastructure as Code

- You need infrastructure to deploy applications
- Can be on-prem or cloud
- Infrastructure can be managed using version control and continuous integration
- Infrastructure and servers can quickly be created or recreated using standardized patterns.
- No need for manual configuration of OS, system software, or applications.
- Easier to govern changes in infrastructure resources

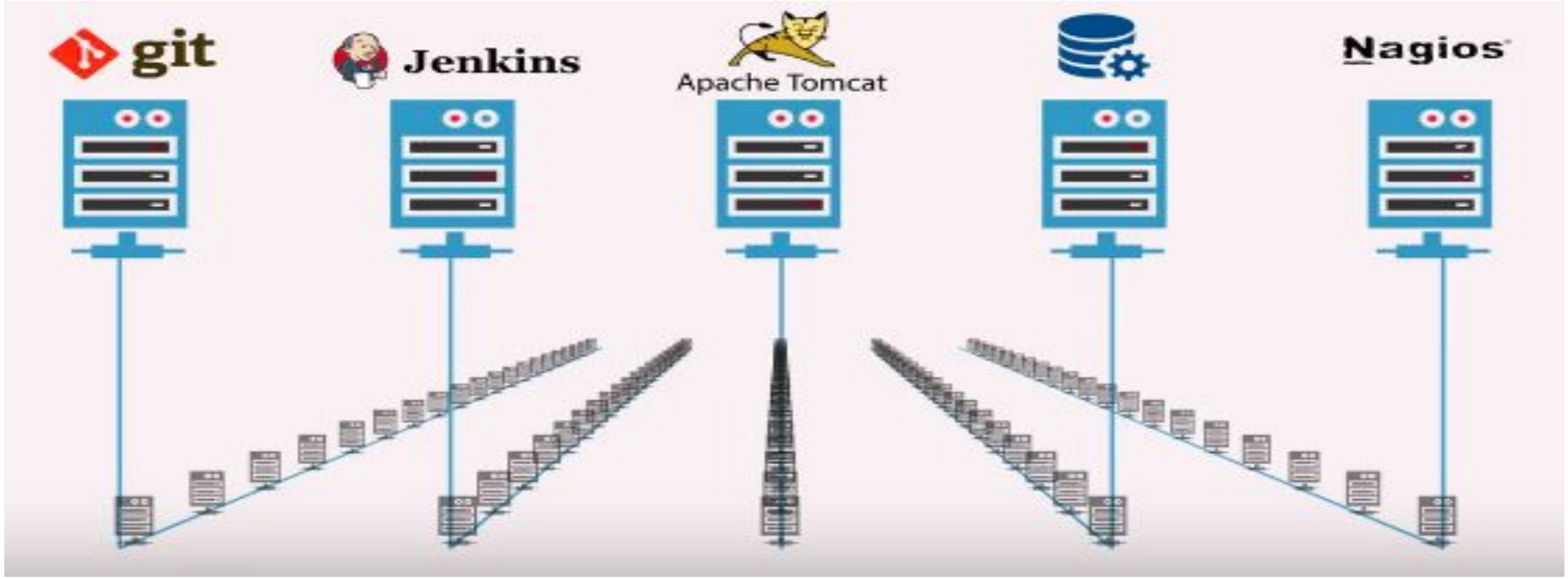
5. Infrastructure as Code

- You need infrastructure to deploy applications
- Can be on-prem or cloud



5. Infrastructure as Code

- Automation across multiple envs



5. Infrastructure as Code

- Recreation of environment
- Deploy everything in an automated way or through scripts
- Main purpose is to have state & deployment manifests in any central place
- Terraform, Ansible, Puppet, Chef
- Lead to GitOps

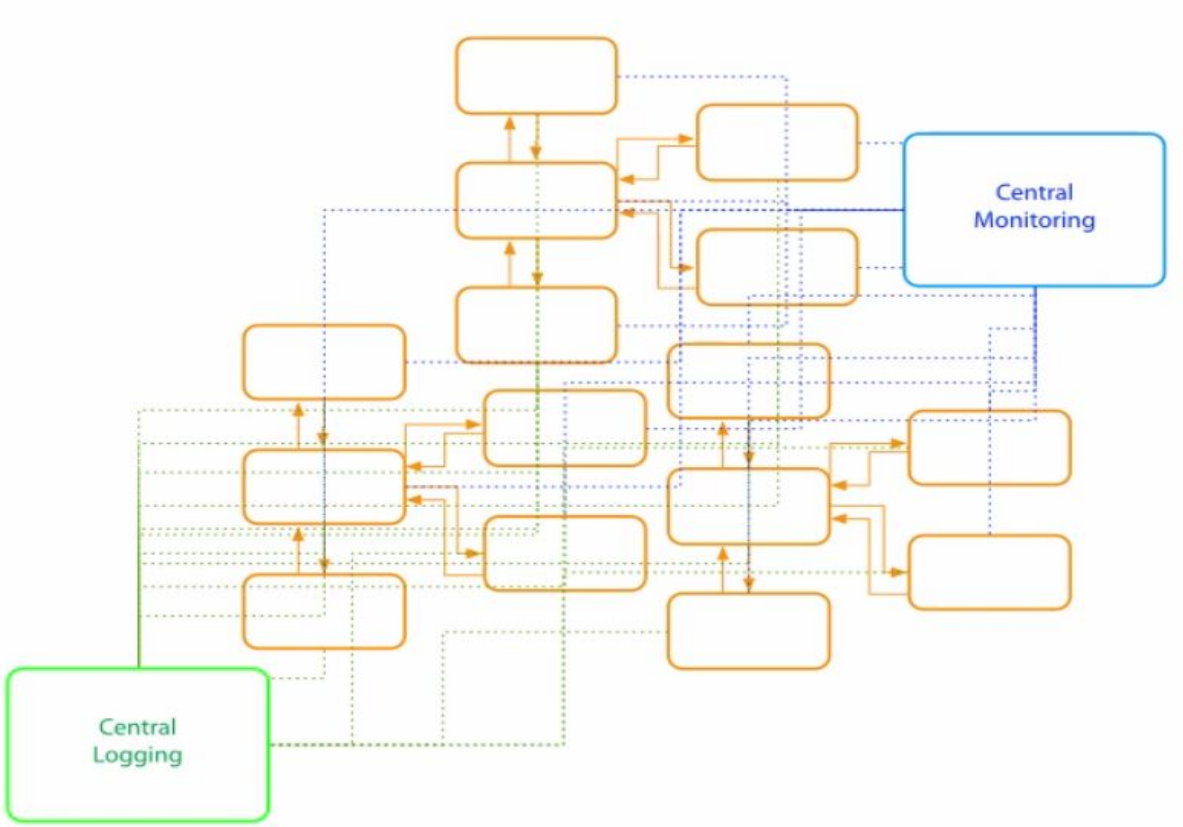
6. Monitoring, Logging, Tracing

- Deploying Infrastructure is not an issue, maintaining it is
- You will be responsible when
  - Infrastructure is throttled
  - Network went down
  - Machines are not working
- Deploy tools to monitor & see the logs sent by application
- Shed insights into the root causes of problems or unexpected changes.

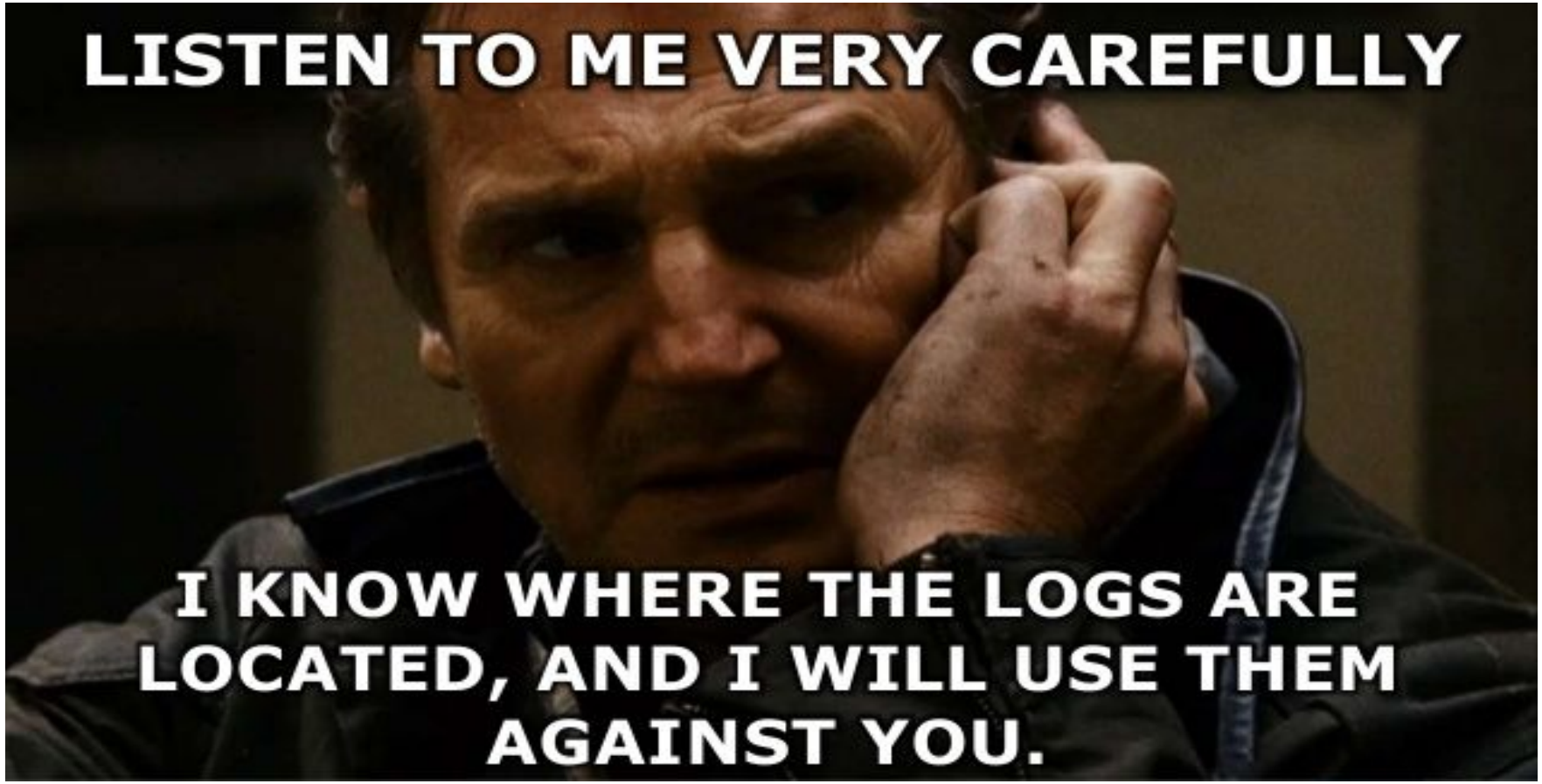
6. Monitoring, Logging, Tracing, Alerting

- Active monitoring to ensure 24/7 availability
- Alerts or perform real-time analysis of data for making rectifications
- Central place so that team can see logs of their application
- Teams can monitor their application stats
- Tracing means how much time is spent and where is it spent?

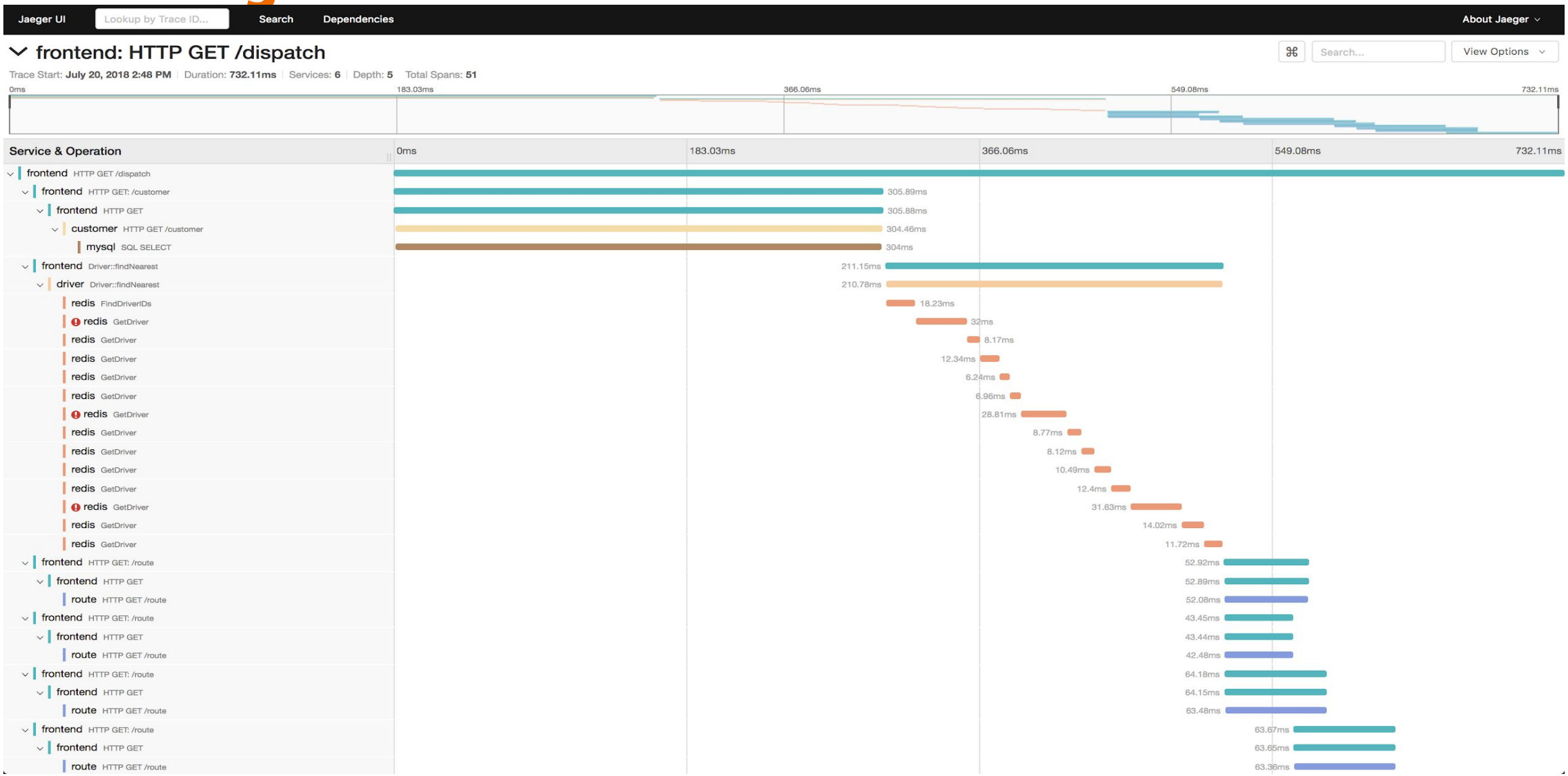
Centralized monitoring & logging



Centralized monitoring & logging



Tracing


Jaeger UI
Lookup by Trace ID...
Search
Dependencies
About Jaeger
frontend: HTTP GET /dispatch
Search...
View Options
Trace Start: July 20, 2018 2:48 PM | Duration: 732.11ms | Services: 6 | Depth: 5 | Total Spans: 51
0ms
183.03ms
366.06ms
549.08ms
732.11ms
Service & Operation
frontend HTTP GET /dispatch
frontend HTTP GET: /customer
305.89ms
frontend HTTP GET
305.88ms
customer HTTP GET /customer
304.46ms
mysql SQL SELECT
304ms
frontend Driver::findNearest
211.15ms
driver Driver::findNearest
210.78ms
redis FindDriverIDs
18.23ms
redis GetDriver
32ms
redis GetDriver
8.17ms
redis GetDriver
12.34ms
redis GetDriver
6.24ms
redis GetDriver
6.96ms
redis GetDriver
28.81ms
redis GetDriver
8.77ms
redis GetDriver
8.12ms
redis GetDriver
10.49ms
redis GetDriver
12.4ms
redis GetDriver
31.63ms
redis GetDriver
14.02ms
redis GetDriver
11.72ms
frontend HTTP GET: /route
52.92ms
frontend HTTP GET
52.89ms
route HTTP GET /route
52.08ms
frontend HTTP GET: /route
43.45ms
frontend HTTP GET
43.44ms
route HTTP GET /route
42.48ms
frontend HTTP GET: /route
64.18ms
frontend HTTP GET
64.15ms
route HTTP GET /route
63.48ms
frontend HTTP GET: /route
63.67ms
frontend HTTP GET
63.65ms
route HTTP GET /route
63.36ms

Tools

- Prometheus
- Grafana
- Nagios
- Datadog
- Jaeger
- Zipkin
- ELK/EFK stack
- AlertManager
- Splunk
- PagerDuty

7. Communication & Feedback

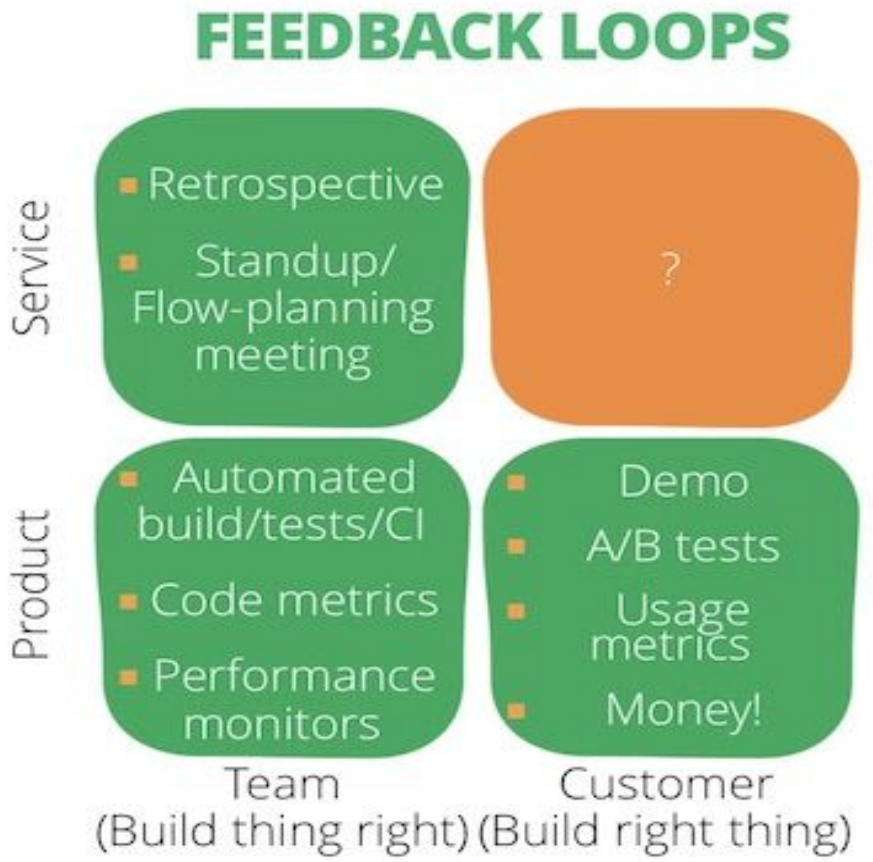
- The most important part of DevOps
- DevOps was introduced due to conflicts between Developers & Operations team blaming each other
- Physically bringing together the workflows and responsibilities of development and operations.
- Maintain visibility across organizations for various events such as deployments, bugs, server downtimes, etc.

7. Communication & Feedback

- Give continuous feedback to respective stakeholder
- If feature is correct and deployed, take feedback from consumer
- If issue in code, give feedback to developer
- If issue in infrastructure, communicate to teams as soon as possible

7. Communication & Feedback

- From: https://www.infoq.com/articles/service-delivery-review-missing-devops-feedback-loop



Tools

- Slack
- Teams
- Jira
- AlertManager

DevOps Requirements

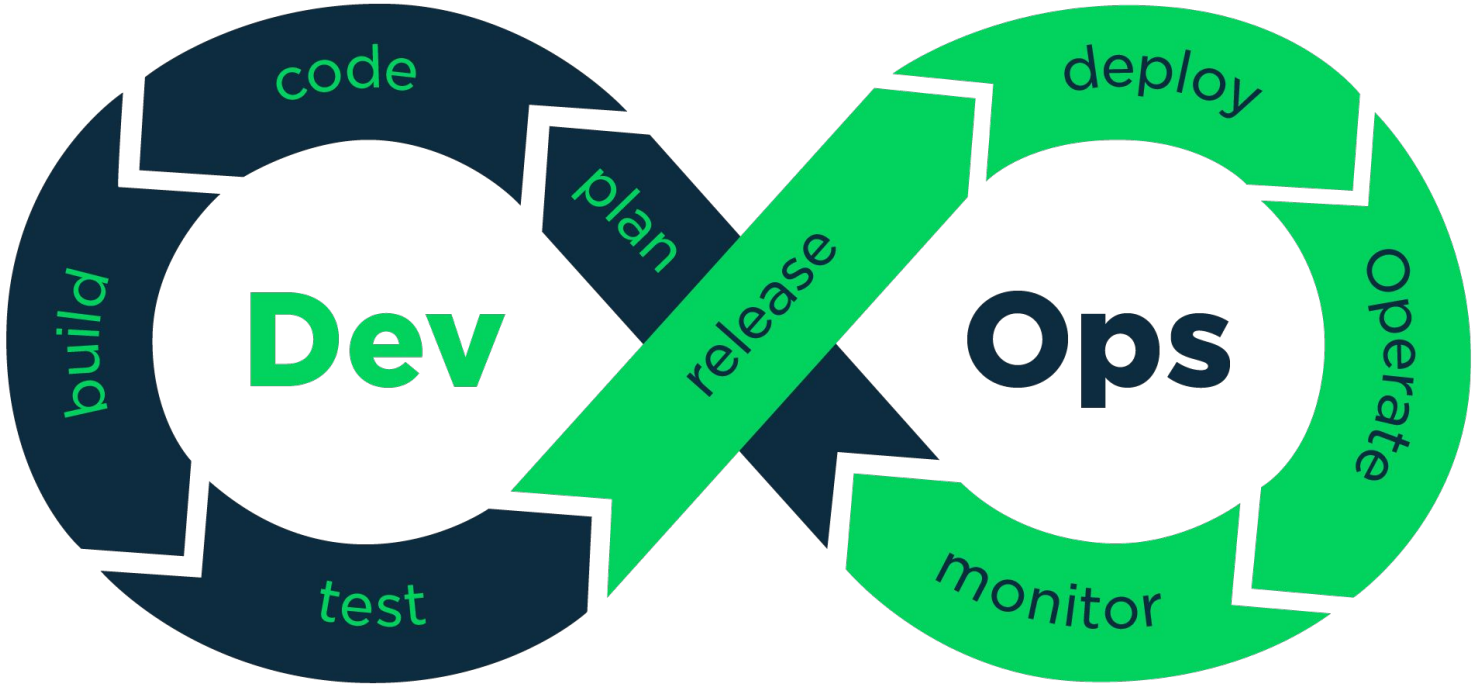
Summary

- IT operations and software development communities realized a dysfunction in the industry.
- Developers and Ops had separate (and often competing) objectives
- The result was siloed teams, long hours, botched releases, and unhappy customers.

Summary

- Combination of cultural philosophies, practices, and tools
- Increase organization's ability to deliver applications/services at high velocity
- Dev and Ops teams are no longer "siloed." Sometimes even merged into a single team
- Quality assurance and security teams may also become more tightly integrated
- Teams use practices and tools to automate processes

Summary


code
plan
build
Dev
test
release
deploy
Ops
operate
monitor

Summary

- **CI/CD:** Release once a week, higher quality of code
- **Microservices**: Loosely coupled components deployed in automated way without waiting on individual component
- **Responsibility**: Shared Responsibility, common goals, process and culture
- **Infrastructure**: Cloud, Containers, Kubernetes, Build once run anywhere.

DevOps Engineer Summary

- **Be tool Independant**
- **Adaptable**
- **Development Experience**
- **Able to Learn & Unlearn**