DEVOPS FUNDAMENTALS

Dr. SELİN METİN

Orion Innovation Türkiye

DevOps Fundamentals Overview

- What is DevOps?
- Why DevOps Matters?
- DevOps Core Values: CAMS
- DevOps Principles: The Three Ways
- DevOps Methodologies
- Ten Practices for DevOps Success
- Managing DevOps
  - Five Why's
- DevOps Building Blocks:
  - Agile and Lean
  - ITIL, ITSM, SDLC
  - Infrastructure as Code
  - CI / CD
  - The Role of QA
  - Design for Operation
  - Metrics and Monitoring
  - Logging
- DevSecOps
- The Future of DevOps

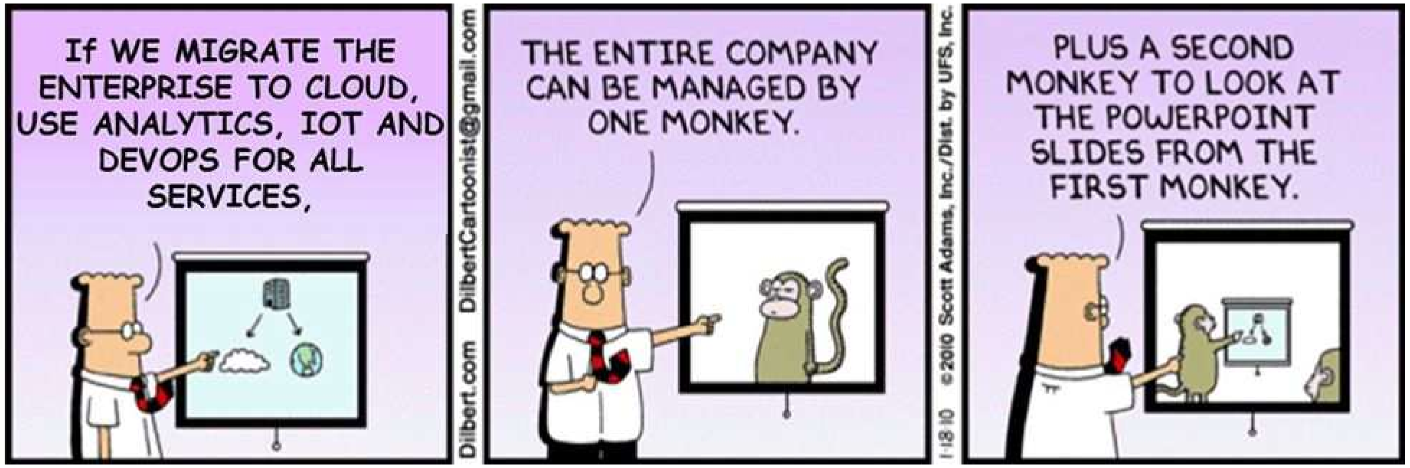
What is DevOps?

- DevOps is a culture!
- DevOps ensures collaboration and communication between software engineers (Dev) and IT operations (Ops).
  - Changes make it to production faster.
  - Resources are easier to share.
  - Large-scale systems are easier to manage and maintain.
- DevOps can help you whether you are a developer or a system admin or something in between.
- It's not a new name for an operations team, a job title or a new tool category.

A Mindshift…

- 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 Definition


If WE MIGRATE THE ENTERPRISE TO CLOUD, USE ANALYTICS, IOT AND DEVOPS FOR ALL SERVICES,
THE ENTIRE COMPANY CAN BE MANAGED BY ONE MONKEY.
PLUS A SECOND MONKEY TO LOOK AT THE POWERPOINT SLIDES FROM THE FIRST MONKEY.
Dilbert.com DilbertCartoonist@gmail.com
1-18-10 ©2010 Scott Adams, Inc./Dist. by UFS, Inc.

Definition

- Practice of operations and development engineers, participating together through the entire service lifecycle, from the design and development process all the way to production support.
  - Replaces the model where you have separate teams to write, test, to deploy, and operate the code.
  - Traditionally

    Dev = everyone usually on the code side (developers, front end designers, QA) and

    Ops= everyone traditionally on the system side (Linux admins, network admins, …)
- DevOps is also characterized by operations staff, making use of many of the same techniques as developers for their systems work.
- Effective in improving both IT and business outcomes.
  - The Puppet Labs «State of DevOps» survey indicated the teams using DevOps practices deployed changes 30x more frequently, with 200x shorter lead times. For quality issues, they had 60x fewer failures, and recovered from issues 168x faster than other organizations.
- It makes your daily life easier.
  - DevOps approach reduces unplanned work.
  - It increases friendly relationships between coworkers, and it reduces stress on the job.

Why DevOps Matters?

- Software no longer merely supports a business; rather it becomes an integral component of every part of a business.
- Companies interact with their customers through software delivered as online services or applications and on all sorts of devices.
- Software is used to increase operational efficiencies by transforming every part of the value chain, such as logistics, communications, and operations.
- In the 20th century industrial automation transformed the way to design, build, and deliver products.

  →In the 21st century companies must transform how they build and deliver software.

DevOps Core Values: CAMS

- The CAMS model, created by DevOps pioneers, John Willis and Damon Edwards.
  - Culture
  - Automation
  - Measurement
  - Sharing.

- DevOps is a human, cultural and business problem.

Culture

- Culture is driven by behavior.
  - Culture exists among people with a mutual understanding of each other and where they're coming from.
- Early on in IT organizations two majors groups existed: Development (creating features) and operations (maintaining stability).

  →Distance, conflicting interests, …
- DevOps is not just about automated tooling, people and process.
  - **People over process over tools**

Automation

- Once you begin to understand your culture, you can create a fabric of automation that allows you to control your systems and your applications.
- Automation is the accelerator that's going to get you all the other benefits of DevOps.
- Make sure to prioritize automation as your primary approach to the problem.

Measurement

- One of the keys to rational approach to our systems is the ability for us to measure them.

- Two major pitfalls in metrics:
  - Sometimes we choose the wrong metrics to watch.
  - Sometimes we fail to incentivize them properly.

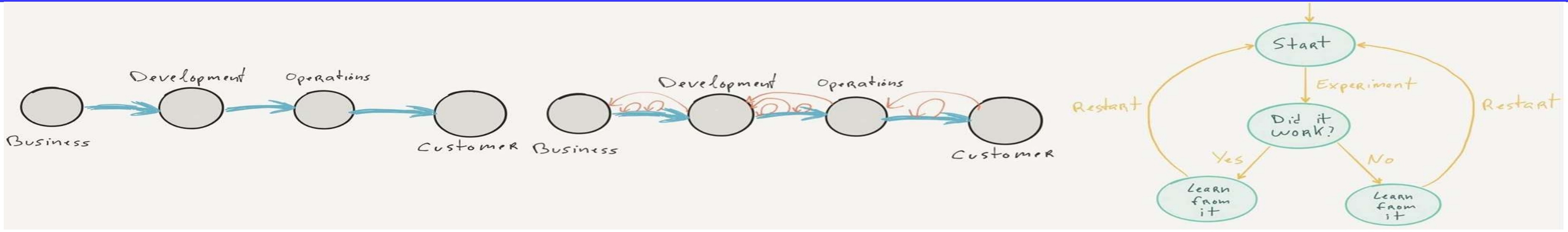
- DevOps strongly advises you to measure key metrics across the organization.
  - Look for things like MTTR, the Mean Time to Recovery or cycle time, look for costs, revenue, even something like employee satisfaction.
  - All of these are part of generating a holistic insight across your system.

- These metrics help engage the team and the overall goal.

Sharing

- Sharing ideas and problems is the heart of collaboration.
- DevOps expect to see a high premium placed on openness and transparency.
- This drives Kaizen (discreet continuous improvement).

DevOps Principles: The Three Ways



- The Three Ways model, developed by Gene Kim and Mike Orzen, provide a practical framework.

- **Systems thinking** tells us that we should make sure to focus on the overall outcome of the entire pipeline or value chain.
  - You have to understand the whole system to optimize it well.
  - From Concept to Cash: If you write all the software in the world, but you can't deliver it to a customer in a way that they can use it, you lose.
  - Use systems thinking as guidance when defining success metrics and evaluating the outcome of changes.

- **Amplifying feedback loops**: Creating, shortening, and amplifying feedback loops between the parts of the organization that are in the flow of that value chain.
  - Use amplifying feedback loops to help you when you're creating multi-team processes, visualizing metrics, and designing delivery flows.

- Create a **work culture** that allows for both **continuous experimentation and learning**.
  - Be open to learning new things by actively trying them out to see what works and what doesn't
  - Engage in the continuous practice required to master the skills and tools that are already part of your portfolio.
  - Use when creating team processes and standards, and as part of your leadership style.

- No technology is a silver bullet that solves all your problems. It's how you use them that matters most.

DevOps Methodologies

- **People over process over tools.**
  - Identify who's responsible for a job function first, then define the process that needs to happen around them, and then select and implement the tool to perform that process.
- **Continuous Delivery**
  - Practice of coding, testing and releasing software frequently and really small batches so that you can improve the overall quality and velocity.
  - It's been shown in studies, that in CD environments, the team spends 22% less time on unplanned work and rework. Changes have a three times lower failure rate and the team recovers 24 times faster from failures.
- **Lean Management**
  - Use small batches of work, work in progress limits, feedback loops and visualization.
  - Lead to both better organizational outputs, including system throughput and stability. And less burnout and greater employee satisfaction at the personal level.
- **Changed Control**
  - There is a direct correlation between operational success and control over changes in your environment.
  - There's a lot of old school heavy change control processes out there that do more harm than good.
  - Focus on an emphasis of eliminating fragile artifacts, creating a repeatable build process, managing dependencies and create an environment of continuous improvement.
- **Infrastructure as code**
  - Systems can and should be treated like code. System specifications should be checked into source control, go through a code review, whether a build an automated test, and then we can automatically create real systems from the spec and manage them programmatically. With this kind of programmatic system, we can compile and run and kill and run systems again, instead of creating handcrafted permanent fixtures that we maintain manually over time.

Ten Practices for DevOps Success

1. Incident Command System: Bad things happen to our services → incidents. Use a tracking and prioritization system.

2. Developers on Call: Teams have begin putting developers on call for the service they created. This creates a very fast feedback loop. Logging and deployment are rapidly improved and core application problems get resolved quickly, instead of lingering for years

3. Status Pages: Services go down. Have a status page so that users can be notified of problems, understand what's being done, and hear what you've learned from the problem afterwards.

4. Blameless Postmortems: examine failures and learn from them, while avoiding logical fallacies or relying on scapegoating

5. Embedded Teams: Dev team wants to ship new code and the Ops team wants to keep the service up → conflict of interest. Some teams reorganized to embed an Operations Engineer on each development team to make the team responsible for all its own work for close coordination.

Ten Practices for DevOps Success

6. Cloud: Cloud technologies give you an entirely API-driven way to create and control infrastructure. This allows you to treat your systems infrastructure exactly as if it were any other program component.

7. Andon cords: This is an innovation originally used by Toyota on its production line. A physical cord like the stop request cord on a bus that empowers anyone on the line to pull to stop ship on the production line because they saw some problem → a fundamental part of their quality control system to this day.

8. Dependency injection: Connections to external services like databases, rest services, etc. cause most of the runtime issues. Focus on loosely coupled dependencies that are passed into the app at runtime.

9. Blue-green deployment: Instead of testing a release in a staging environment and then deploying it to a production environment and hoping it works, have two identical systems, blue and green, one is live and the other isn't. You upgrade the offline system, test it, and then shift production traffic over to it. If there's a problem, you shift back. This minimizes both downtime and risk.

10. Chaos monkey: Focus on making the overall system highly reliable even with unreliable components.
    - Netflix invented a piece of software called the chaos monkey that watches the Netflix system that runs in the Amazon cloud and occasionally reaches out and trashes a server, just kills it. This forces the developers and operators creating the systems to engineer resiliency into their services instead of making the mistake of thinking that their infrastructure is always on.

A Culture Problem

Managing DevOps

- Organizational issues
  - DevOps strongly advocates eliminating silos and thrives in an environment of autonomous teams, by putting the right people on a team, focusing them on the business problem at hand, and setting up self service tooling to allow them to develop, test, deploy and operate their own service.
  - → You maintain a very high pace

- Conway's Law: your systems will basically align themselves to your communication boundaries.
  - Organizational boundaries
  - Process boundaries: Agile processes compliment a shift to DevOps.
  - You don't add process unless everyone thinks some more process is really the solution. And you keep an eye out for processes that you can remove.

- Use lean agile processes to keep your effort focused on value creation.
  - The 2016 State of DevOps Report found that a Lean management approach correlated well with organizational performance.

Japanese Terms

- **Kaizen** literally means change for the better
- **Gemba**: the real place
- Kaizen emphasizes going to look at the actual place where the value is created, or where the problem is.
- Teach people critical thinking skills
  - Toyota refers to this as building people before building cars.

Five Why’s

- Another Kaizen tool to identify root cause
- When there's a problem, you ask the question, why did it happen?
  - When you get an answer, ask again.
  - Five times is generally enough to exhaust the chain down to a root cause.
- Keep in mind:
  - Focus on underlying causes not symptoms.
  - Don’t accept answers like not enough time, we need to know what caused us to exceed our constraints.
  - Usually there will be multiple causes contribute to one element. A diagram can be used to track all of these.
  - Do not accept human error as a root cause. That always points to a process failure or a lack of a process with sufficient safeguards.

    «People don't fail, processes do.»

DevOps Building Blocks: Agile and Lean

- Refer to week 9!

DevOps Building Blocks: ITIL, ITSM, SDLC

- ITIL (Information Technology Infrastructure Library) is a set of detailed practices for IT activities such as IT service management (ITSM) and IT asset management (ITAM) that focus on aligning IT services with the needs of business.
  - Processes, procedures, tasks, and checklists which are not organization nor technology specific, but can be applied by an organization toward strategy, delivering value, and maintaining a minimum level of competency.

Infrastructure as Code

- A large number of operations teams don't use source control at all. Even for their scripts or other code.
- Tools exist to configure servers from the bare metal up completely automatically. And with virtual servers, cloud servers or containers, everything can be created, changed and destroyed programmatically.
  - AWS: You can completely describe your systems using a JSON format called CloudFormation.
  - It brings confidence that your Dev, Staging and Production environments are identical because they were created from the same specification.

Small + Fast = Better

- You have an application that is built automatically on every code commit.
  - Unit tests are run and the application is deployed into a production like environment.
  - Automated acceptance tests are run and the change either passes or fails testing minutes after it's checked in.

  → Code is always in a working state with every Continuous Delivery.

- **Continuous Integration** is the practice of automatically building and unit testing the entire application frequently, ideally on every source code check-in.

- **Continuous Delivery** is the additional practice of deploying every change to a production like-environment and performing automated integration and acceptance testing.

- **Continuous Deployment**: every change goes through full enough automated testing that it's deployed automatically to production.

Continuous Delivery Benefits

- The 2016 State of DevOps Report found that high performing IT organizations could deploy on demand as compared to their peers that took days, weeks or months.

- Increase in quality
  - Instead of doing inspection at the end of the development life cycle, we're integrating testing earlier in the delivery pipeline.
  - Instead of one huge go live consisting of hundreds of changes, we evaluate and deploy changes one by one testing every commit and making sure the software is in a running state.

- Two factors making your meantime to recover shorter:
  - Once you've come up with fix, it can be treated just like any other change and rolled out quickly.
  - Find the cause of failures: which commit broke the code?

The Role of QA

- **Unit testing:** the tests done at the lowest level of the language or framework that it supports. Write a test inside the code base to validate that function - Usually the fastest testing available.
- **Code hygiene:** Code hygiene is the sum of the best practices from your development community for the particular language or framework that you're using.
- **Integration testing:** technical testing similar to unit testing, but performed with all the apps components and dependencies operating in a test environment.
- **Testing from an outside in perspective:**
  - Test driven development (TDD): writing tests before you write any code. You start with the desired outcome written as a test. You then write code to pass the test. Bonus: while the app is being written, a comprehensive test suite is also being developed.
  - Behavior driven development (BDD): encourages the developer to work with a business stakeholder to describe the desired business functionality of the app and expresses the test in a script.
  - Acceptance test driven development (ATDD): builds on TDD and BDD and it's involved defining scenarios from the end user's perspective. Automated tests with examples of these use cases are written and run.
- **Infrastructure testing:** involves starting up a host and running the configuration management code, running all the tasks and then turning it all off.
- **Performance testing:** load tests, stress tests, soak tests, and spike tests - great for out-of-band nightly runs, they're usually long running and they consume a lot of resources.
- **Security testing:** might be useful to think about this as a simulated attack test.

Design for Operation

- Software architecture patterns
- In modern architectures, especially in microservice architectures apps end up having so many integration points that the likelihood of one of them going bad is very high.
- Identify risk items and develop platform level fixes.
- A good manifesto for how to produce service ready software is called the 12 factor app, described at https://12factor.net/ .

The Twelve Factors

I. Codebase
II. Dependencies
III. Config
IV. Backing services
V. Build, release, run
VI. Processes
VII. Port binding
VIII. Concurrency
IX. Disposability
X. Dev/prod parity
XI. Logs
XII. Admin processes

Operate for Design: Metrics and Monitoring

- Complexity creates risk for failure
- Create a minimum viable monitoring stack just enough to accomplish our goals.
- Measure: get a metric from each area you monitor
  - Service performance and uptime monitoring: implemented at the very highest level
  - Software component metrics: done on ports or processes, usually located on the host.
  - System metrics: CPU, memory, etc. time series metrics which can be stored in graph that help to answer the question, is this service or host or process, is it functioning normally?
  - Application metrics: telemetry about what your application is actually doing (how long a certain function call is taking, or maybe the number of logins in the last hour or account of all the error events that have happened). These are custom to your app.
  - Performance metrics: usually uses front-end instrumentation and captures the performance observed by the users of the actual system.
  - Security monitoring: includes four key areas. System, application security, custom events in the application (things like password resets, invalid logins, or new account creations), and anomalies

Operate for Design: Logging

- Logs are a great DevOps monitoring tools
  - They can be consumed by an operations team
  - They can provide meaningful feedback to developers

- Five Ws of logging:
  - What happened
  - When it happened
  - Where did it happen
  - Who was involved
  - Where the entity came from

- Logging principles:
  - Do not collect log data that you're never planning to use.
  - Retain log data for as long as it's conceivable that it could be used or longer if it's prescribed by regulations.
  - Log all you can, but alert only on what you must respond to. Clearly define your log levels: error, warn, info, etc.
  - Logging should meet business needs, not exceed them.
  - Logs change, as in their format or their messages with new versions of software.

DevSecOps

- Development, security, and operations
- In the collaborative framework of DevOps, DevSecOps is an approach to culture, automation, and platform design that integrates security as a shared responsibility throughout the entire IT lifecycle.
  - Think about application and infrastructure security from the start.
  - Automate some security gates to keep the DevOps workflow from slowing down.
  - Select the right tools to continuously integrate security.

DevOps security is built-in

- Build in information security and set a plan for security automation.
- Help developers code with security in mind.
- Identify risks to the software supply chain by emphasizing the security of open source software components and dependencies early in the software development lifecycle.
- A good DevSecOps strategy:
  - Determine risk tolerance and conduct a risk/benefit analysis.
  - What amount of security controls are necessary within a given app?
  - How important is speed to market for different apps?
  - Automate repeated tasks.

DevOps security is automated

- Maintain short and frequent development cycles
- Integrate security measures with minimal disruption to operations
- Keep up with innovative technologies like containers and microservices
- Foster closer collaboration between commonly isolated teams
- What to automate: source control repositories, container registries, the continuous integration and continuous deployment (CI/CD) pipeline, application programming interface (API) management, orchestration and release automation, and operational management and monitoring

DevOps security for containers and microservices

- Cloud-native technologies don’t lend themselves to static security policies and

Environment and data security	Introduce secure API gateways
Standardize and automate the environment	Integrate security scanners for containers
Centralize user identity and access control capabilities	Automate security testing in the CI process
Isolate containers running microservices from each other and the network	Add automated tests for security capabilities into the acceptance test process
Encrypt data between apps and services	Automate security updates, such as patches for known vulnerabilities
Introduce secure API gateways	Automate system and service configuration management capabilities

DevSecOps Tools

- Veracode: cloud-based, offers remediation tips for detected vulnerabilities
- Checkmarx: AI-powered software security solutions
- OWASP ZAP: free and open-source web application security scanner, highly customizable, works by intercepting and modifying HTTP and HTTPS traffic between the web application and client
- Burp Suite: for web application security testing
- SonarQube: popular code quality tool that offers security-focused plugins
- Fortify: static, dynamic, and interactive application security testing
- Snyk: developer-first application security tool
- Coverity: static analysis tool
- AppScan: AI-powered solution, supports both static and dynamic applications

The Future of DevOps

- Cloud to containers to serverless
- Security
- AI/ML in the DevOps framework
- Low-code application
- GitOps growing adoption

Summary

- DevOps is the combination of cultural philosophies, practices, and tools that increases an organization’s ability to deliver applications and services at high velocity.

- DevOps is a human, cultural and business problem.
  - Culture
  - Automation to increase pace and eliminate errors
  - Measurement to understand and locate
  - Sharing to drive Kaizen

- The Three Ways model: Systems thinking , amplifying feedback loops, a work culture for continuous experimentation and learning.

- Managing DevOps: Closely aligns with Lean and Agile principles
  - Conway's Law: your systems will basically align themselves to your communication boundaries.

- Ask «Why?» recursively five times to identify root cause

- Infrastructure as Code: Use configuration management and source control for your Ops tools.

- Continuous Integration and Continuous Delivery

- The Role of QA – different types of tests for control and monitoring

- Use architectural patterns to minimize fragility of your software. Measure metrics that you will use, don’t forget to update logging as your software evolves.

- Build in information security and set a plan for security automation.

References

- DevOps Foundations, by Ernest Mueller and James Wickett
  https://www.linkedin.com/learning/devops-foundations/
- DevOps Foundations: Continuous Delivery/Continuous Integration
  https://www.linkedin.com/learning/devops-foundations-continuous-delivery-continuous-integration-14449917?u=89254810
- 12 factor app: https://12factor.net/
- Bite-sized introduction to architectural terms: https://martinfowler.com/
- How Complex Systems Fail https://how.complexsystems.fail/
- What is DevSecOps? https://www.redhat.com/en/topics/devops/what-is-devsecops
- An Unlikely Union: DevOps and Audit https://itrevolution.com/product/an-unlikely-union-devops-and-audit/
- The Future of DevOps: 2023 and Beyond https://www.knowledgehut.com/blog/devops/future-of-devops