

Course 1	2
Week 1: Introduction to DevOps	2
21.11.2022:	2
1.1. Course Introduction	2
1.2. Business Case for DevOps	2
1.3. DevOps Adoption	3
1.4. Definition of DevOps	3
1.5. Essential Characteristics of DevOps	3
1.6. Activity: Identify Categories in Application Evolution	4
1.7. Practice Quiz: Introduction To DevOps: Practice Quiz 1	4
22.11.2022	4
1.8. Leading Up to DevOps	4
1.9. XP, Agile, and Beyond	5
1.10. Brief History of DevOps	6
1.11. Practice Quiz: Introduction to DevOps: Practice Quiz 2	6
1.12. Discussion Prompt: Module 1 Discussion	6
1.13. Reading: Summary and Highlights	6
1.14. Quiz: Overview of DevOps	7
Week 2: Social Coding Principles	8
23.11.2022	8
2.1. Social Coding Principles	8
2.2. Git Repository Guidelines	8
2.3. Working in small Batches:	9
2.4. MVP (Minimum viable product)	9
2.5. Practice Quiz: Social Coding principles: Practice Quiz 1	9
2.6. Test Driven Development (TDD)	9
2.7. Behaviour Driven Development	9
2.8. Activity: Writing in Gherkin Syntax	10
2.9. Practice Quiz: Social Coding Principles: Practice Quiz 2	10
01.12.2022	10
2.10. Cloud Native Microservices	10
2.11. Designing for Failure	11
2.12. Practice Quiz: Social Coding Principles: Practice Quiz 3	11
2.13. Discussion Prompt: Module 2 Discussion	11
2.14. Reading: Summary and Highlights	12
2.15. Quiz: Thinking DevOps	12
Week 3: Taylorism and Working in Silos	13
01.12.2022	13
3.1. Taylorism and Working in Silos	13
3.2. Software Engineering vs Civil Engineering	13
3.3. Required DevOps Behaviors	13

3.4. Activity: Choosing DevOps Behaviour	14
3.5. Practice Quiz: Taylorism and Working in Silos: Practice Quiz 1	14

DevOps Courses by IBM on Coursera

Course 1

Week 1: Introduction to DevOps

21.11.2022:

1.1. Course Introduction

→ Devops is practice of Engineers working together through full SLDC, in a rapid and continuous manner. This is the heart of the DevOps Culture.

1.2. Business Case for DevOps

→ Since the Year 2000 52% of Fortune 500 Companies are gone due to disruption.
→ If it takes a bank 6 months to deploy updates to software a lot of customers might get lost.
→ Speed is very important in CI/CD
→ Uber disrupted Taxi industry with Open Source Software.



A ridesharing service is 40% cheaper than a regular cab for a 5-mile trip into Los Angeles

\$\$\$ Ridesharing

\$\$\$\$\$ Taxi

→
→ Netflix disrupted Blockbuster
→ Technology enables innovation, doesn't drive it by itself.

1.3. DevOps Adoption

→ fail fast, roll back quickly
→ A/B testing
→ Microservices are usable and recommended for this
→ January 2011 Etsy deploys every ~25min, in 1 month they:
• Code committed by 76 people
→ a total of 517 times

2016 DevOps Enterprise Summit

Ticketmaster - 98% reduction in MTTR

Nordstrom - 20% shorter lead time

Target - Full Stack Deploy three months to minutes

USAA - Release from 28 days to 7 days

ING - 500 application teams doing DevOps

CSG - From 200 incidents per release to 18

→ 2016:

1.4. Definition of DevOps

→ “Development Operation is an extension of agile development that aims to enhance the process of software delivery as a whole” - Patrick Debois, 2009

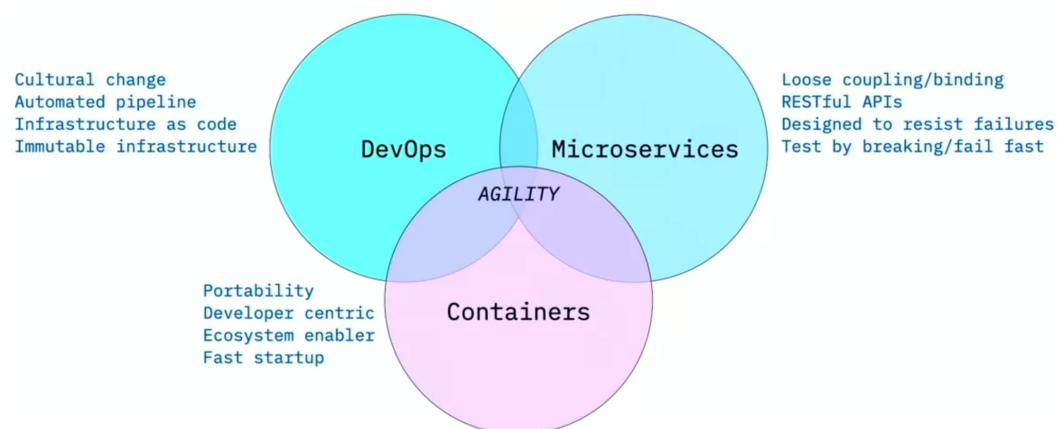
→ DevOps is Devs and Ops working together in full SLDC following agile principles

→ DevOps requires a new application design, leveraging automation for microservices and a defined programmable platform to deploy on to.

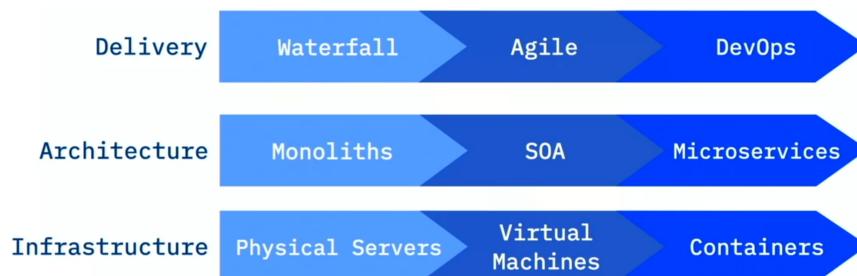
1.5. Essential Characteristics of DevOps

→ Be agile, move in market with minimum risk and maximum velocity, gain valuable insights

→ Three Pillars of Agility:



→ Application evolution:



1.6. Activity: Identify Categories in Application Evolution

Match categories and Application Evolution steps

Correct!

You have made the right matches.

Waterfall > Agile > DevOps: Waterfall, Agile, and DevOps are methods for software development and delivery.

Monoliths > SOA > Microservices: Monoliths, SOA, and Microservices are architecture: ways that software is built.

Physical Services > VMS > Containers: Physical Services, VMS, and Containers are used to create infrastructure: basic services such as communication and storage.

[Continue](#)

→

1.7. Practice Quiz: Introduction To DevOps: Practice Quiz 1

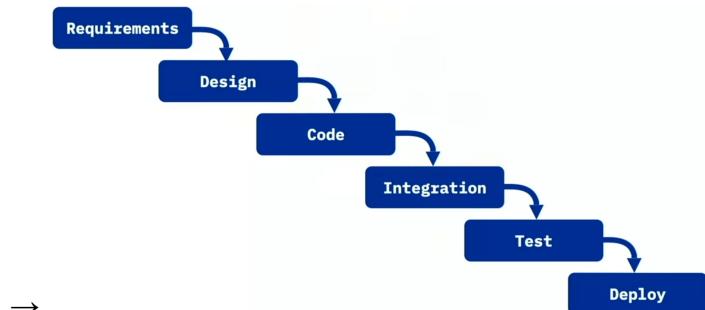
✓ Congratulations! You passed!

→ Grade received 100% To pass 70% or higher

22.11.2022

1.8. Leading Up to DevOps

→ Traditional inefficient waterfall development method:



→ ...

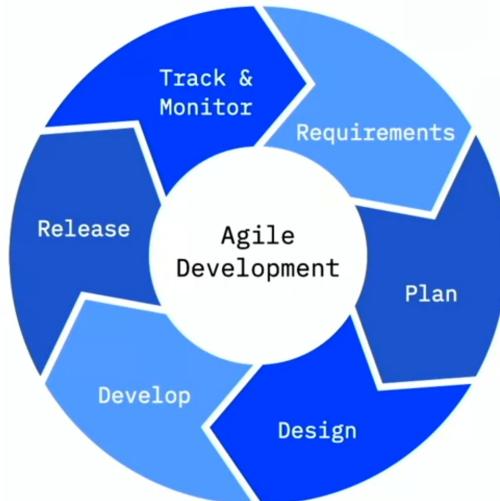
- Traditional Waterfall development creates problems such as delays, frustration, long lead times, expensive late changes, and operations managing unfamiliar code
- In the past, software developers and operations worked in silos, rather than working together

1.9. XP, Agile, and Beyond

→ The Agile Manifesto:

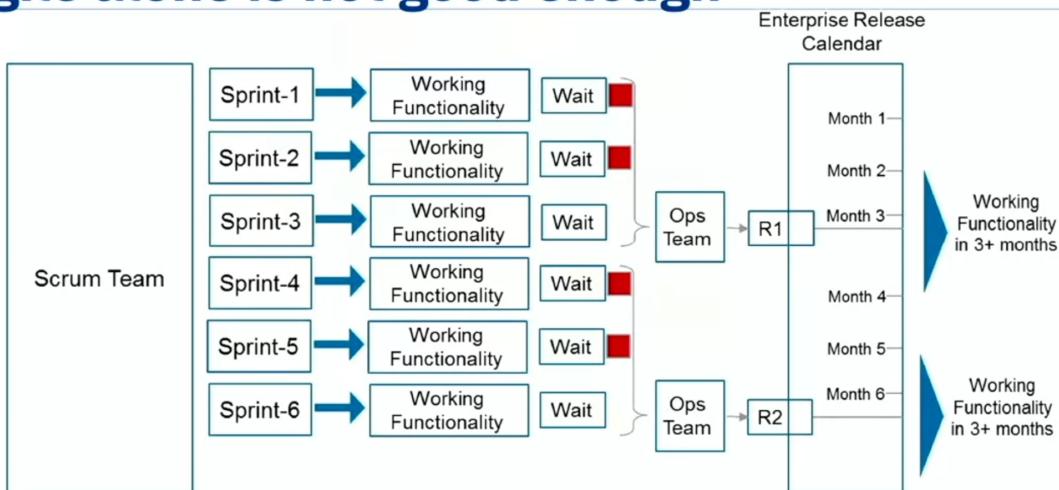
We have come to value:
Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right,
→ we value the items on the left more



→

Agile alone is not good enough



→

→ Shadow IT (Resources the business doesn't know about) comes around because of this speed Problem in Company infrastructure, people go around it.

1.10. Brief History of DevOps

- Patrick Debois started the DevOps movement in 2007 with a simple idea of getting development and operations to work better together
- DevOps grew through the efforts of influential people such as Patrick Debois, Andrew Shafer, John Allspaw, Nicole Forsgren, Bridget Kromhout, Jez Humble, Gene Kim, and John Willis

1.11. Practice Quiz: Introduction to DevOps: Practice Quiz 2

Congratulations! You passed!

→ Grade received **100%** To pass 70% or higher [Go to next item](#)

→ **Practice Quiz: Introduction to DevOps: Practice Quiz 2**
3 questions

1.12. Discussion Prompt: Module 1 Discussion

Module 1 Discussion

Please introduce yourself and share why you want to learn about DevOps. Is your organization currently practicing or considering DevOps?

Your response has been submitted. Engage and discuss with other learners below!



1.13. Reading: Summary and Highlights

→ Technology is the enabler of innovation, rather than the driver of innovation. You must have an innovative business idea to leverage technology.

→ In 2009, John Allspaw described an innovative approach to managing development and operations that enabled Flickr to complete over ten deploys per day, when many companies were completing fewer than one deploy every six months. This was a key moment in the growth of DevOps.

→ DevOps is the practice of development and operation engineers working together during the entire development lifecycle, following Lean and Agile principles that allow them to deliver software in a rapid and continuous manner.

→ DevOps is not just Dev and Ops working together. It is a cultural change and a different way to work. DevOps has three dimensions: culture, methods, and tools. Of these, culture is the most important.

→ The essential characteristics of DevOps include cultural change, automated pipelines, infrastructure as code, immutable infrastructure, cloud native application design, the ecosystem of containers, and how to deploy with immutable infrastructure.

→ DevOps started in 2007 when Patrick Debois and Andrew Clay Shafer began to gather like-minded people together at conferences to talk about common experiences.

→ In 2009, Allspaw delivered his now famous “10+ Deploys Per Day – Dev and Ops Cooperation at Flickr” presentation and the idea gained ground. Also in 2009, Patrick Debois started a conference called DevOpsDays that helped spread the DevOps message.

- Books such as *Continuous Delivery* in 2011, *The Phoenix Project* in 2015, and *The DevOps Handbook* in 2016, helped practitioners understand how DevOps worked.
- The major influential people of the early DevOps movement: Patrick Debois, Andrew Clay Shafer, John Allspaw, Jez Humble, Gene Kim, John Willis, Bridget Kromhout, and Nicole Forsgren, went out and made a difference, showing the results that could be achieved with DevOps.
- The message spread from practitioner to practitioner until they began to realize what was possible with DevOps and that it was a better way to work.

1.14. Quiz: Overview of DevOps

→  **Congratulations! You passed!**

Grade received 100%	Latest Submission Grade 100%	To pass 70% or higher	Go to next item
-------------------------------	--	--------------------------	---------------------------------

Week 2: Social Coding Principles

23.11.2022

2.1. Social Coding Principles

- Social coding is done communally with public repositories and all team members are encouraged to contribute
- Pair programming results in higher quality code because
 - Defects are found earlier
 - Costs are lowered
 - There is a broader understanding of the code base

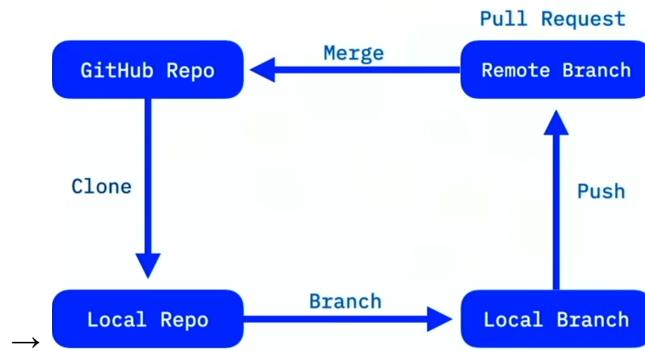
→

2.2. Git Repository Guidelines

→ Every Microservice in a separate repository, new Branch for every issue

→ Pull Request to merge to master, every Pull Request should be reviewed

→ The Git Feature Branch Workflow:



2.3. Working in small Batches:

- Feature size supports frequent releases
- Features should be completed in a sprint
- Features are a step toward a goal, so keep them small

2.4. MVP (Minimum viable product)

- An MVP is the minimal thing that you can do to test your hypothesis
- An MVP is not about delivery as much as learning
- It's okay if the MVP fails—make sure that you learn from it

→

2.5. Practice Quiz: Social Coding principles: Practice Quiz 1

✓ Congratulations! You passed!

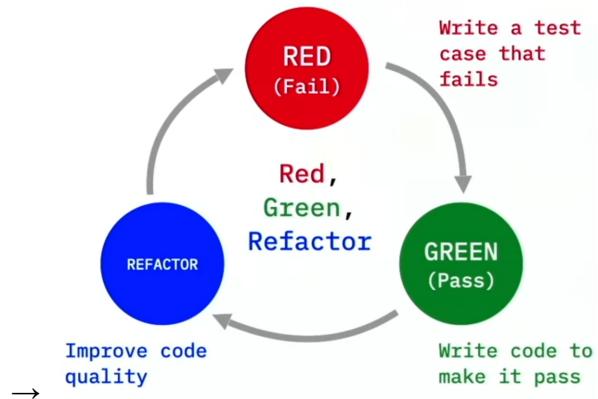
→ Grade received 100% To pass 70% or higher

→ Practice Quiz: Social Coding Principles: Practice Quiz 1
2 questions

[Go to next item](#)

2.6. Test Driven Development (TDD)

→ Workflow is as follows, it goes as red → green → refactor



→ Why is TDD important:

- TDD means test cases drive the design and development of code
- TDD allows you to develop faster and with more confidence
- The Red, Green, Refactor workflow has to do with red as fail, green as pass, and it increases the quality of code
- In order to create a DevOps CI/CD pipeline, you must first automate your testing

2.7. Behavior Driven Development

→ Difference to TDD:

- BDD ensures that you are building the "right thing"
- TDD ensures that you are building the "thing right"

- BDD focuses on the behavior of the system from the outside in. It looks at the system as a consumer of it.
- BDD uses an approachable syntax that everyone can understand.
- Key benefits of BDD include improvement in communication, a common syntax, and acceptance criteria for user stories.

→ BDD Workflow:

- Explore the problem domain and describe the behavior
- Document the behavior using Gherkin syntax
- Use BDD tools to run those scenarios
- One document that's both the specification and the tests

→ Gherkin:

- An easy-to-read natural language syntax
- Given... When... Then...
- Understandable by everyone

- Given (some context)
- When (some event happens)
- Then (some testable outcome)
- And (more context, events, or outcomes)

2.8. Activity: Writing in Gherkin Syntax

Correct!

You have made the right matches.

Given: Some context. Given steps describe the initial context of the system.

When: Some event happens. When steps describe an event or an action.

Then: Some testable outcome. Then steps describe an expected outcome or result.

And: More context, events, or outcomes. And is used for continuations.

→ [Continue](#)

2.9. Practice Quiz: Social Coding Principles: Practice Quiz 2

Congratulations! You passed!

Grade received 100% To pass 70% or higher

→ [Go to next item](#)

→ [Practice Quiz: Social Coding Principles: Practice Quiz 2](#)
2 questions

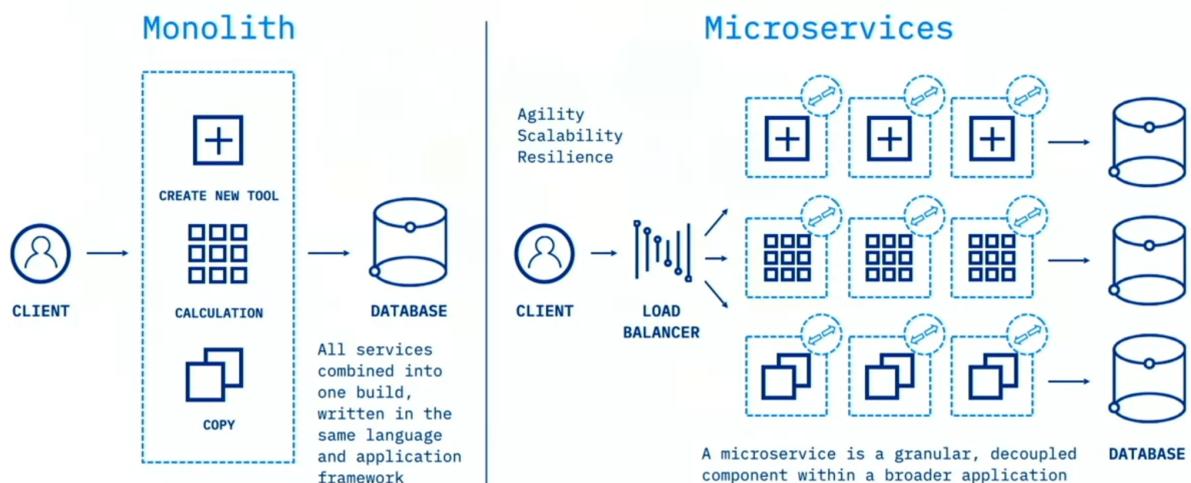
01.12.2022

2.10. Cloud Native Microservices

"...the microservice architectural style is an approach to developing a single application as a **suite of small services**, each **running in its own process** and communicating with lightweight mechanisms, often an HTTP resource API. These services are **built around business capabilities** and **independently deployable** by fully automated deployment machinery."

→ Monolith vs microservices:

→



→ Communication between services is done through (REST) APIs

2.11. Designing for Failure

- Failure is inevitable, so we design for failure rather than trying to avoid failure
- Developers need to build in resistance to be able to recover quickly
- Retry patterns work by retrying failed operations
- Circuit breaker patterns are designed to avoid cascading failures
- Bulkhead patterns can be used to isolate failing services
- Chaos engineering is deliberately causing services to fail to see how other services are affected

2.12. Practice Quiz: Social Coding Principles: Practice Quiz 3

→ Congratulations! You passed!
Grade received 100% To pass 70% or higher [Go to next item](#)

→ Practice Quiz: Social Coding Principles: Practice Quiz 3
2 questions

2.13. Discussion Prompt: Module 2 Discussion

Module 2 Discussion

How could your organization or team benefit from DevOps? If you are practicing DevOps, please share an experience of what has and has not worked.

→ Your response has been submitted. Engage and discuss with other learners below!

[View My Response](#)

2.14. Reading: Summary and Highlights

- Social coding is coding as a community and public repositories and pair programming result in higher code quality.
- Working in small batches reduces waste and means quickly delivering something useful to the customer.
- Minimum viable product is as much about delivery as it is about building what the customer really desires.
- Test driven development is writing the test for the code you wish you had, then writing the code to make the test pass. It allows you to develop faster and with more confidence.
- Behavior driven development focuses on the behavior of the system from the outside in. It looks at the system as a consumer of it.
- Behavior driven development improves communication by using an approachable syntax that developers and stakeholders can understand.

- Microservices are built around business capabilities and are independently deployable by fully automated deployment machinery.
- Cloud native architecture enables independently deployable microservices that take advantage of horizontal scaling and result in more resilient services.
- Failure is inevitable, so we design for failure rather than trying to avoid failure.
- It is important to embrace failure and quickly recover when failures happen.

2.15. Quiz: Thinking DevOps

→ **Congratulations! You passed!**

Grade received 80% Latest Submission Grade 80%

To pass 70% or higher

[Go to next item](#)

→ **Quiz: Thinking DevOps**
10 questions

Week 3: Taylorism and Working in Silos

01.12.2022

3.1. Taylorism and Working in Silos

- Working DevOps means pushing small releases faster in order to get feedback, minimize risk, and maximize learning
- Taylorism was designed for factory work, while software development is like craft work
- Handoffs created by working in silos results in → mistakes, bottlenecks, and delays

3.2. Software Engineering vs Civil Engineering

- Software development is often viewed as a project to be completed and then passed to operations to be maintained
- Software engineering and the behavior of the system are constantly changing
- Team ownership and stable teams make software development more like product development and → less like project management

3.3. Required DevOps Behaviors

→ Traditional vs DevOps:

Manual configuration changes to critical infrastructure	Automated deployment to all environments
Application architectures defined by network design	Network design defined by application architectures
Bespoke infrastructure built once, then maintained	Ephemeral infrastructure created for each new deployment
Risk managed through change windows	Risk managed through progressive activation
Process biased toward "build once"	Builds are repeatable leveraging infrastructure as code

→ Wall of Confusion:

- Development wants innovation
- • Operations wants stability

→ Required Behaviour:

Organizational silos and hand-offs	Shared ownership and high collaboration
Fear of change	Risk management by embracing change
Build once, hand-crafted "snowflakes"	Ephemeral infrastructure as code
Manual fulfillment	Automated self-service
Alarms, callbacks, and escalations	Feedback loops and data-driven responses

3.4. Activity: Choosing DevOps Behaviour

Recap

→ Congratulations! You have discovered how some behaviors help organizations fully become DevOps.

→  Activity: Choosing DevOps Behavior
10 min

3.5. Practice Quiz: Taylorism and Working in Silos: Practice Quiz 1

→  Congratulations! You passed!

→ Grade received 100% To pass 70% or higher

→  Practice Quiz: Taylorism and Working in Silos: Practice Quiz 1
3 questions