title: DevOps

DevOps

Development and Operations

Overview

- Introduction to DevOps
- Continuous Integration / Continuous Delivery (CI/CD)
- Infrastructure as Code (IaC)

Learning Objectives

- Explain the role DevOps plays in modern software
- Identify the role CI/CD plays in modern software
- Create a Continuous Integration workflow with GitHub Actions

DevOps - A History

Developers and IT Ops professionals had separate (and often competing) objectives, department leadership, key performance indicators, and often worked on separate floors or even separate buildings.

Developers looked after building the system, IT Ops looked after supporting and monitoring the system.

This resulted in siloed teams concerned only with their processes and releases.

These silos often meant miscommunication, delayed deliveries and strains on morale and relationships.

What is DevOps?

A combination of **culture, practises** and **tools** that increases an organisation's ability to deliver services at high velocity.

More succinctly, it is a set of practices that combines software development (Dev) and IT operations (Ops).

Under a DevOps model, dev and ops are no longer siloed.

Both are merged into a single team where engineers work across the entire application lifecycle, from development and test to deployment to operations, and develop a range of skills not limited to a single function.

DevOps Culture

- **Increased collaboration** as a team as opposed to silos
- **Share responsibility** across the whole team so no one process is looked after by specific people, thus spreading the knowledge and pain
- **No silos** between development and operations
- Give power to **autonomous teams** to enable them to make their own decisions in order to collaborate effectively, and remove convoluted decision making processes
- **Build quality into the development process**
- **Value feedback** to continuously improve ways of working

- **Automate** as much as you can

DevOps Practices

- Continuous Integration
- Continuous Delivery
- Infrastructure as Code
- Microservices
- Monitoring and Logging
- Communication and Collaboration

In this module, we will be looking into Continuous Integration (CI), Continuous Delivery (CD) and Infrastructure as Code (IaC).

DevOps Tools

- Source Control (Git)
- Collaboration / communication tools (Slack, Jira, Trello)
- Issue tracking (Jira, Trello, ZenDesk)
- Configuration tools (Puppet, Chef)
- CI/CD tools (we will come onto these)
- And plenty more!

Benefits of DevOps

Speed: Microservices and continuous delivery lets teams take ownership of services and then release updates to them quicker.

Rapid delivery: Increase the frequency and pace of releases so you can innovate and improve your product faster.

The quicker you can release new features and fix bugs, the faster you can respond to your customers' needs and build competitive advantage.

Reliability: Quality of application updates and infrastructure changes so you can reliably deliver at a faster pace while maintaining a positive experience for end users.

Better internal culture: DevOps practises lead to better communication, increased productivity and agility.

Why does software need to change?

- Features
- Bug fixes
- Security patches
- Contract changes
- Performance optimisations

What does software need to be able to operate?

- Server
- Database
- Cache
- Storage
- APIs
- Networking

What is an environment?

An **environment** is the place where our software runs.

Environments are comprised of the resources and infrastructure that support the operation of a software system.

Who needs access to an environment?

- Developers
- Operations
- Testers
- Product Owner
- Client's support team
- End users

Software Lifecycle

The development environment is usually the first thing you set up when working on a software project

---

Types of environments

Development: maximise developer productivity, usually local.

Testing: hosted, similar to customer-facing environment, for more complex integration and E2E tests.

UAT/Staging: as stable and production-like as possible, used for customer demos and wider service integration.

Production: live, end-user-facing environment. *You must protect it at all costs*.

Different people scrutinising the product at various stages reduces the risk of severe defects.

Promoting software through our environments

Having multiple environments helps find bugs in our software before it's released to the public.

Changes to the software require deployments up the chain, all the way to production.

A recipe to deploy software reliably

- Make change
- Build code
- Run unit tests
- Run code quality metrics
- Install the required dependencies in the target environment
- Install our software in the target environment

Maintaining environments is hard work

- Repetition without automation = more manual work

- Manual work = time wasted
- Manual work = potential differences in environments
- Differences in environments = constant breakage
- Constant breakage = downtime + longer wait to go live

How can we best alleviate this?

Quiz Time!

Which statement best describes DevOps?

1. Developers taking over all operations tasks.
2. Automating the process of software delivery and infrastructure changes.
3. The collaboration and communication of both software developers and other IT professionals while automating the process of software delivery and infrastructure changes.
4. The collaboration and communication of just software developers and operations staff while automating the process software delivery and infrastructure changes.

Answer: `3`

Complete the following sentence with the best matching answer.

One goal of DevOps is to establish an environment where...

1. Change management does not control application releases.
2. Releasing more reliable applications faster and more frequently can occur.
3. Application development performs all operations tasks.
4. Releasing applications is valued over the quality of the released application.

Answer: `2`

Agile and DevOps are similar but differ in a few important aspects. Which statement is correct?

1. Agile is a change of thinking, DevOps is a cultural change in an organisation.
2. Agile is cultural change in an organisation, DevOps is a change of thinking.
3. Agile is process driven, DevOps is role driven.
4. Agile is role driven, DevOps is process driven.

Answer: `1`

Continuous Integration

What is CI?

> Continuous Integration (CI) is a development practice that requires developers to integrate code into a shared repository several times a day. Each check-in is then verified by an automated build, allowing teams to detect problems early. By integrating regularly, you can detect errors quickly, and locate them more easily.
>
> -- *ThoughtWorks*

CI has three stages of workflow.

Bugs and regressions are minimised because they're detected early on and automatically.

---

1. Integrating changes to the main branch (every day)

- All about ensuring code gets into the main branch, as that branch is used for releasing software
- Easiest to do technically, hardest to do culturally
- Also known as trunk-based development
- Integrating code continually means branches are short-lived, which means code is going into main faster

Feature flags can be used for pushing changes to production that aren't necessary ready.

---

2. Rely on automated tests

- Run an automated suite of tests on each integration to the main branch
- Instantly gives you information about failing tests, therefore failing software
- Can help spot bugs or errors
- Only run tests that are important, keep build time low

---

3. Prioritise broken builds

- If a build fails (code didn't compile, tests failed etc.), make fixing it the priority before doing any other tasks
- Everyone can see the same broken build which means better visibility of when something goes wrong

---

The enemies of CI

- Knowledge silos
- Manual work
- Inconsistent environments
- Big PRs

1. Knowledge silos means teams aren't communicating effectively 2. Manual work means more human errors 3. Inconsistency means it's hard to verify things work or don't work as intended 4. Big PRs means more time spent figuring out what the PR is doing, and easier for bugs to be introduced

---

Continuous Deployment

What is CD?

- A way of getting your changes (features, bug fixes etc.) into production in a quick and sustainable way
- Code is checked to be in a deployable state by the CI stage

These methodologies allow you to catch bugs and errors early in the development cycle, ensuring that all the code deployed to production complies with the code standards you established for your app. Big-bang releases which introduce risk and instability and make rollbacks harder.

Principles of CD

Frequent, small deployments: Deploying smaller changes rather than an accumulation of changes means less chance of something going wrong.

Automation: Computers are infinitely better at repetitive tasks. Deployment steps can be automated so that we reduce human errors. Things like Infrastructure as Code (IaC) can help.

Keep improving: Always look to improve the deployment process. Are you manually doing anything that can be automated?

Shared responsibility: Everyone in the team is responsible for creating safe, fast and deterministic delivery of the deployed product, it shouldn't be siloed between devs and ops.

CI/CD Pipelines

What is a pipeline?

A route, channel, or process along which something passes or is provided at a steady rate.

Build Pipeline

- Provides a steady supply of changes to our end-users
- Automates the integration and delivery flow of your software

---

Gated Releases

- A stage in the build pipeline that must be triggered manually to deploy to the next stage
- Allows you to ensure everything is how it should be in the stage before deploying to production

---

CI/CD Terms

Pipeline: automated stream of work that verifies, integrates and deploys software.

Build: a full cycle through the pipeline, triggered when a change is merged into the repo's stable branch.

Job: an individual step in the pipeline that carries out one or more tasks.

Task: a script run in a job.

CI/CD Software

- GitHub Actions
- CircleCI
- Jenkins
- Concourse

- And plenty more!

YAML

YAML **A**in't a **M**arkup **L**anguage.

A markup language designed primarily for humans rather than computers.

Structurally similar to JSON, except indentation defines the structure, like in Python.

YAML example

```yaml
# Strings don't need quotes (this is a comment BTW)
key1: hello
key2:
  # You can nest keys
  subkey1: 1
  subkey2: hello
  # You can also make lists
  subkey3:
    - listitem1
    - listitem2
    - listitem3
```

GitHub Actions Example

```
 # Name your job/action
name: Python application
# When to run this job
on: [push]
jobs:
build:
  # Which operating system to test on - e.g. "macos-latest, ubuntu-latest, windows-latest"
  runs-on: [ operating system ]

  # The steps to run in the job
  steps:
    # Checkout code from github
    - uses: actions/checkout@v2

    # Install python
    - name: Set up Python
      uses: actions/setup-python@v1
      with:
        python-version: [ python version ]

    # Run pip to install all packages/dependencies
    - name: Install dependencies
      run: pip install -r requirements.txt

    # Further tasks
    - name: Another task
      run : Command to run

    - name: Another task
      run: Another command to run
```

Demo: GitHub Actions

Creating a new GitHub Actions workflow from scratch

Demonstrate to the class adding a GitHub actions workflow to a repository using the provided `example-action.yml` Any repository is suitable Talk learners through the syntax of the file and what function each part of it performs Recommend that you: 1. Code the yml file step-by step rather than pasting it in whole 1. Create the workflow file in VSCode in the `.github/workflows/` directory and push it 1. This will demonstrate how workflows are discovered by convention if placed in the appropriate location (as opposed to if creating it via the GitHub web interface)

Infrastructure as Code

What is IaC?

- The management of infrastructure (servers, databases, storage, networking etc.)
- Generates the exact same environment every time through a code file
- Used in conjunction with CD
- Without IaC, teams must maintain the settings of all environments individually
- Enables teams to test applications in production-like environments early on

---

Why is IaC important?

- Repeatable process
- Self documenting system architecture
- Idempotent (creates the same environment every time)
- Can tear down and recreate easily
- Dynamic infrastructure
- As configuration is in a code file, it can be tracked with version control

---

CloudFormation

- AWS proprietary IaC
- Provisions AWS resources automatically (lambda, S3, EC2 etc.)
- Written in YAML (or JSON)

Key terms:

Template: the file used to describe your resources and properties.

Stack: collections of AWS resources created from the templates.

Change Set: the difference between the current state and the updated state of the infrastructure (for instance, when you update the template to add/edit/delete resources).

CloudFormation

CloudFormation Example - Lambda

```
AWSTemplateFormatVersion: '2010-09-09'
Description: Test function
Resources:
  Function:
    Type: AWS::Lambda::Function
    Properties:
      Handler: index.handler
      Role: arn:aws:iam::123456789012:role/lambda-role
      Code:
        S3Bucket: my-bucket
        S3Key: function.zip
      Runtime: python3.8
      Timeout: 5
```

Describe what this snippet is doing.

CloudFormation Example - S3 Bucket

```
Resources:
S3Bucket:
  Type: 'AWS::S3::Bucket'
  DeletionPolicy: Retain
  Properties:
    BucketName: DOC-EXAMPLE-BUCKET
```

Further Cloudformation

We will dive further into CloudFormation in a seperate module.

Quiz Time!

What is meant by Continuous Integration?

1. A way of getting code changes into production in a quick and sustainable way.
2. A way of releasing new changes to your customers quickly in a sustainable way.
3. A way of requiring developers to push code into a shared repository several times a day. Each check-in is then verified by an automated build.
4. An architectural style that integrates an application as a collection of small autonomous services.

Answer: `3`

Steer clear of saying that 1 is Continuous Delivery as that's the next question.

What is meant by Continuous Deployment?

1. A way of getting code changes into production in a quick and sustainable way.

2. A way of releasing new changes to your a user acceptance environment quickly in a sustainable way.
3. A way of releasing new changes into a development environment in a quick and sustainable way.
4. An architectural style that integrates an application as a collection of small autonomous services.

Answer: `1`

What is meant by Infrastructure as Code?

1. Managing your IT infrastructure using configuration files.
2. Managing your IT cloud infrastructure using configuration files.
3. Managing your codebase from within IT infrastructure.
4. An AWS-specific resource that allows you to manage your IT infrastructure using configuration files.

Answer: `1`

Exercise: Getting started with GitHub Actions

See exercise and handout

Distribute to the cohort the exercise, and also the completed `example-action.yml` to be used as a starting point.

Learning Objectives Revisited

- Explain the role DevOps plays in modern software
- Identify the role CI/CD plays in modern software
- Create a Continuous Integration workflow with GitHub Actions

---

Terms and Definitions Recap

DevOps: A set of practices that combines software development (Dev) and IT operations (Ops).

Continuous Integration: The practice of merging all developers' working copies to a shared mainline several times a day.

Continuous Delivery: A software engineering approach in which teams produce software in short cycles, ensuring that the software can be reliably released at any time and, when releasing the software, doing so automatically.

Terms and Definitions Recap

Environment: A computer system in which a computer program or software component is deployed and executed.

Build Pipeline: The process of taking code from version control and making it readily available to users of your application in an automated fashion.

Infrastructure as Code: The process of managing and provisioning computer data centers through machine-readable definition files, rather than physical hardware configuration or interactive configuration tools.

Further Reading

- DevOps Culture
- State of DevOps Report 2020
- The Phoenix Project (DevOps Novel)

- Accelerate, The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations (Book)