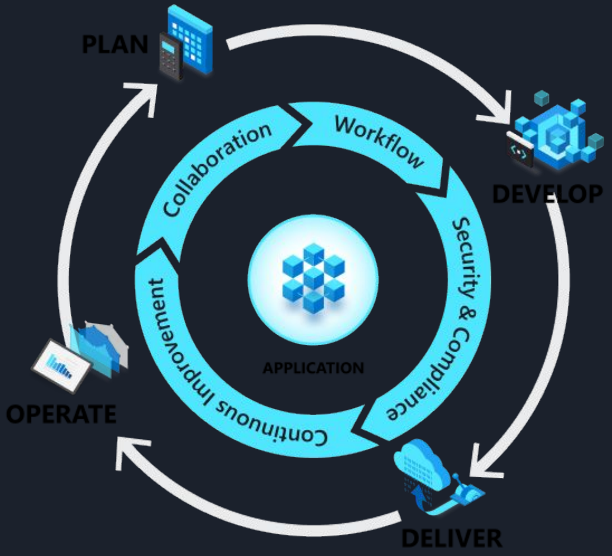
MLOPS

By: Ashish Pal

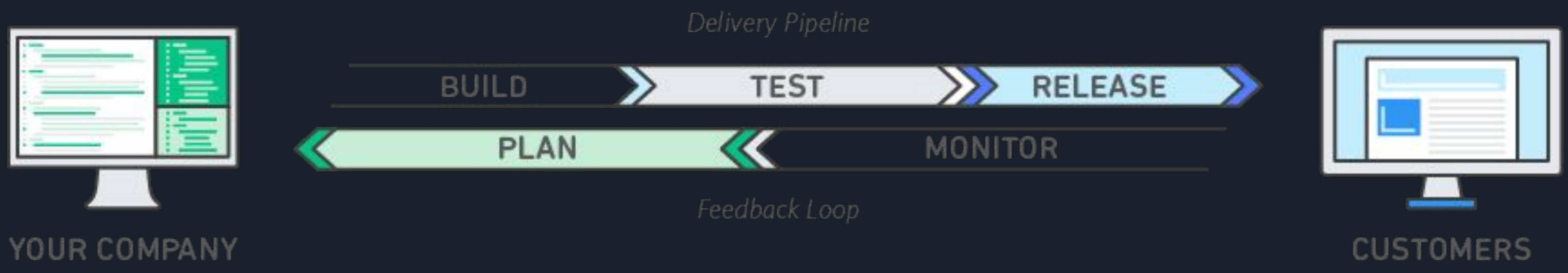
What is Devops?



DevOps combines development (Dev) and operations (Ops) to unite people, process, and technology in application planning, development, delivery, and operations. DevOps enables coordination and collaboration between formerly siloed roles like development, IT operations, quality engineering, and security.

What is 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.



DevOps goals and benefits

Accelerate time to market : Through increased efficiencies, improved team collaboration, automation tools, and continuous deployment--teams are able to rapidly reduce the time from product inception to market launch.

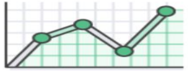
Adapt to the market and competition : A DevOps culture demands teams have a customer-first focus. By marrying agility, team collaboration, and focus on the customer experience, teams can continuously deliver value to their customers and increase their competitiveness in the marketplace.

Maintain system stability and reliability

By adopting continuous improvement practices, teams are able to build in increased stability and reliability of the products and services they deploy. These practices help reduce failures and risk.

ML + DevOps = 

Benefits of DevOps



Speed

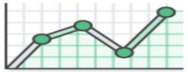
Move at high velocity so you can innovate for customers faster, adapt to changing markets better, and grow more efficient at driving business results. The DevOps model enables your developers and operations teams to achieve these results. For example, microservices and continuous delivery let teams take 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. Continuous integration and continuous delivery are practices that automate the software release process, from build to deploy.

Reliability

Ensure the quality of application updates and infrastructure changes so you can reliably deliver at a more rapid pace while maintaining a positive experience for end users. Use practices like continuous integration and continuous delivery to test that each change is functional and safe. Monitoring and logging practices help you stay informed of performance in real-time.



Scale

Operate and manage your infrastructure and development processes at scale. Automation and consistency help you manage complex or changing systems efficiently and with reduced risk. For example, infrastructure as code helps you manage your development, testing, and production environments in a repeatable and more efficient manner.

Improved Collaboration

Build more effective teams under a DevOps cultural model, which emphasizes values such as ownership and accountability. Developers and operations teams collaborate closely, share many responsibilities, and combine their workflows. This reduces inefficiencies and saves time (e.g. reduced handover periods between developers and operations, writing code that takes into account the environment in which it is run).

Security

Move quickly while retaining control and preserving compliance. You can adopt a DevOps model without sacrificing security by using automated compliance policies, fine-grained controls, and configuration management techniques. For example, using infrastructure as code and policy as code, you can define and then track compliance at scale.

Continuous integration and continuous delivery (CI/CD)

Continuous Integration (CI) is the practice used by development teams to automate, merge, and test code. CI helps to catch bugs early in the development cycle, which makes them less expensive to fix. Automated tests execute as part of the CI process to ensure quality.

Continuous Delivery (CD) is a process by which code is built, tested, and deployed to one or more test and production environments. Deploying and testing in multiple environments increases quality. CD systems produce deployable artifacts, including infrastructure and apps.

PATHTOAI

Continuous integration and continuous delivery (CI/CD) Tools

Continuous Integration (CI) -

Jenkins: Jenkins is one of the most widely used open-source CI/CD automation servers. It allows you to automate building, testing, and deployment processes and can integrate with a wide range of plugins and tools.

Travis CI: Travis CI is a cloud-based CI service that's popular for open-source projects. It provides easy integration with GitHub repositories and supports multiple programming languages.

CircleCI: CircleCI is a cloud-based CI/CD platform that offers powerful configuration options for building, testing, and deploying applications. It's known for its speed and ease of use.

GitLab CI/CD: GitLab offers built-in CI/CD capabilities as part of its DevOps platform. It provides a single interface for managing source code and CI/CD pipelines.

TeamCity: TeamCity is a CI/CD server developed by JetBrains. It's known for its user-friendly interface and powerful build configuration options.

Azure DevOps

Version Control

Version control is the practice of managing code in versions—tracking revisions and change history to make code easy to review and recover. This practice is usually implemented using version control systems such as Git, which allow multiple developers to collaborate in authoring code. These systems provide a clear process to merge code changes that happen in the same files, handle conflicts, and roll back changes to earlier states.

Version Control

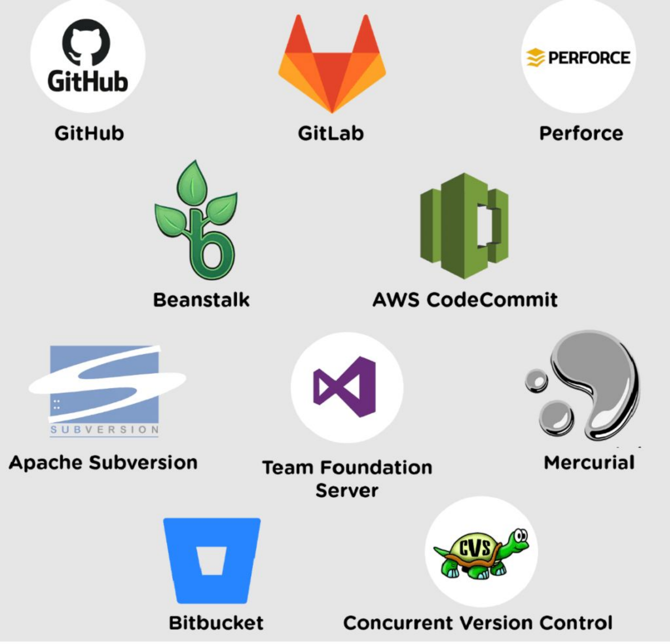
Git: Git is one of the most widely used distributed version control systems (DVCS). It's known for its speed, flexibility, and strong branching and merging capabilities. GitHub, GitLab, and Bitbucket are popular hosting platforms for Git repositories.

Subversion (SVN): Subversion is a centralized version control system that has been in use for many years. It's known for its simplicity and ease of use, especially in scenarios where distributed workflows are not a requirement.

Mercurial: Mercurial is another distributed version control system that is easy to learn and use. It's well-suited for projects of all sizes and offers a straightforward and consistent interface.

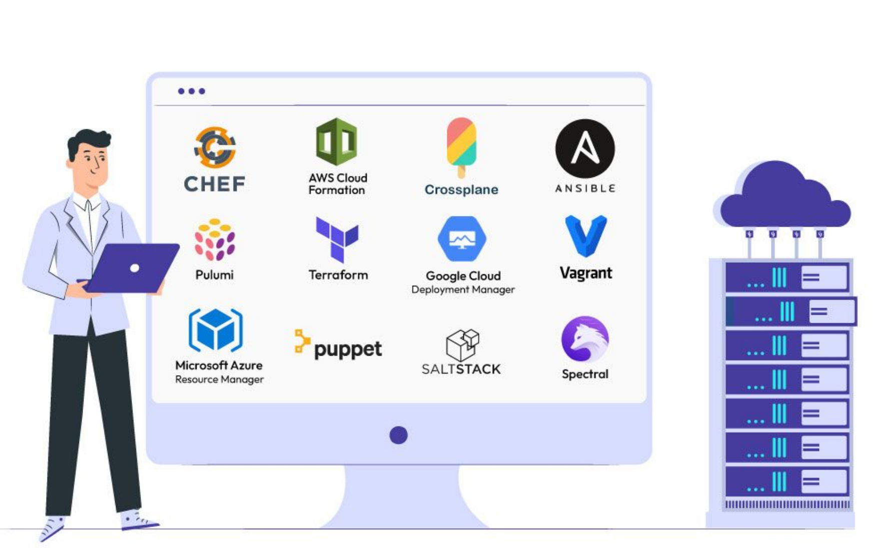
AWS CodeCommit: AWS CodeCommit is a managed Git service provided by Amazon Web Services (AWS). It integrates seamlessly with other AWS services and is suitable for cloud-native development.

Microsoft Team Foundation Version Control (TFVC): TFVC is a centralized version control system that is part of Microsoft's Azure DevOps Services (formerly known as Visual Studio Team Services). It integrates well with the Microsoft development ecosystem.



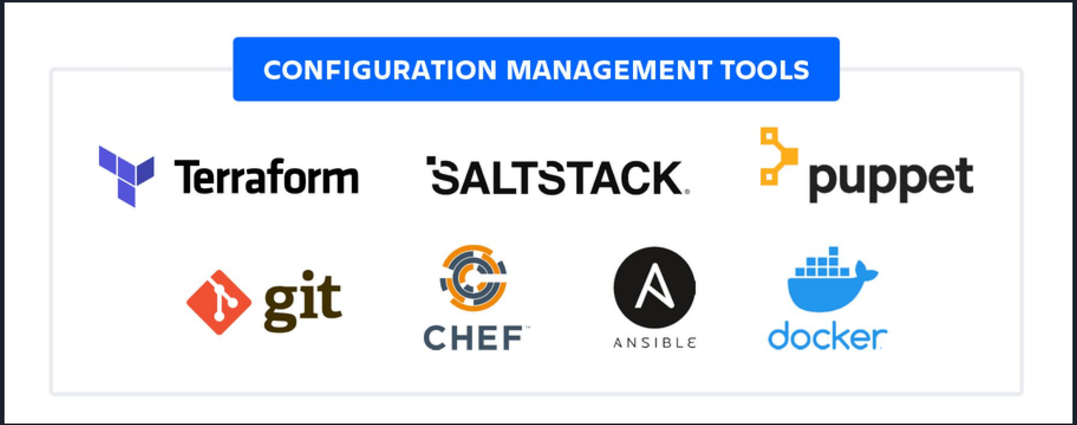
Infrastructure as code

Infrastructure as code defines system resources and topologies in a descriptive manner that allows teams to manage those resources as they would code. Those definitions can also be stored and versioned in version control systems, where they can be reviewed and reverted—again like code.



Configuration management

Configuration management refers to managing the state of resources in a system including servers, virtual machines, and databases. Using configuration management tools, teams can roll out changes in a controlled, systematic way, reducing the risks of modifying system configuration. Teams use configuration management tools to track system state and help avoid configuration drift, which is how a system resource's configuration deviates over time from the desired state defined for it.



Infrastructure as code

Infrastructure as code (IAC) is the managing and provisioning of infrastructure through code instead of using a manual process to configure devices or systems.

HashiCorp Terraform

CloudFormation

Azure ARM Templates

Google Cloud Deployment Manager

pulumi

Configuration Management as code

Configuration as code is the practice of managing configuration files in a repository. Config files establish the parameters and settings for applications, server processing, operating systems.

puppet

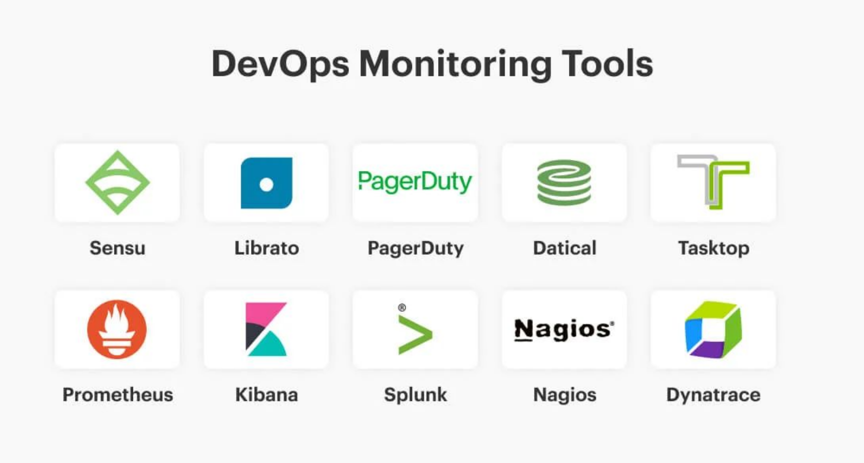
ANSIBLE

SALTSTACK

CHEF

Continuous monitoring

Continuous monitoring means having full, real-time visibility into the performance and health of the entire application stack. This visibility ranges from the underlying infrastructure running the application to higher-level software components. Visibility is accomplished through the collection of telemetry and metadata and setting of alerts for predefined conditions that warrant attention from an operator.



Planning

In the planning phase, DevOps teams ideate, define, and describe the features and capabilities of the applications and systems they plan to build. Teams track task progress at low and high levels of granularity, from single products to multiple product portfolios. Teams use the following DevOps practices to plan with agility and visibility:

- Create backlogs.
- Track bugs.
- Manage Agile software development with Scrum.
- Use Kanban boards.
- Visualize progress with dashboards.

Development

The development phase includes all aspects of developing software code. In this phase, DevOps teams do the following tasks:

Select a development environment.

- Write, test, review, and integrate the code.
- Build the code into artifacts to deploy into various environments.
- Use version control, usually Git, to collaborate on code and work in parallel.
- To innovate rapidly without sacrificing quality, stability, and productivity, DevOps teams

Use highly productive tools.

- Automate mundane and manual steps.
- Iterate in small increments through automated testing and continuous integration (CI).

Deliver

PATHTOAI

Delivery is the process of consistently and reliably deploying applications into production environments, ideally via continuous delivery (CD).

In the delivery phase, DevOps teams:

- Define a release management process with clear manual approval stages.
- Set automated gates to move applications between stages until final release to customers.
- Automate delivery processes to make them scalable, repeatable, controlled, and well-tested.
- Delivery also includes deploying and configuring the delivery environment's foundational infrastructure.

DevOps teams use technologies like **infrastructure as code (IaC), containers, and microservices** to deliver fully governed infrastructure environments.

Operations

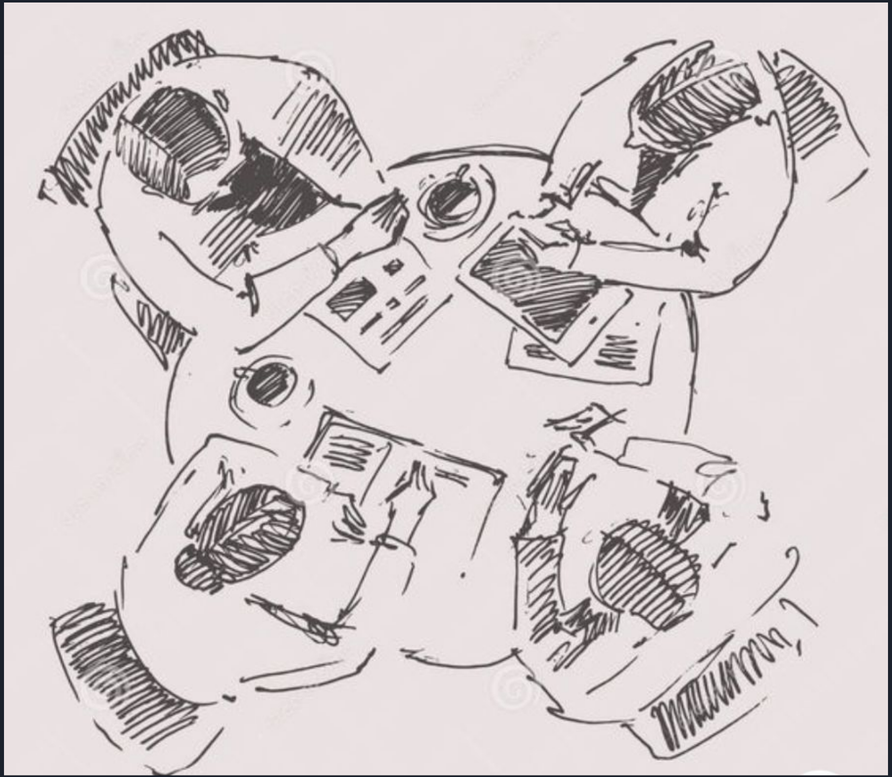
The operations phase involves maintaining, monitoring, and troubleshooting applications in production environments, including hybrid or public clouds like Azure.

DevOps teams aim for system reliability, high availability, strong security, and zero downtime.

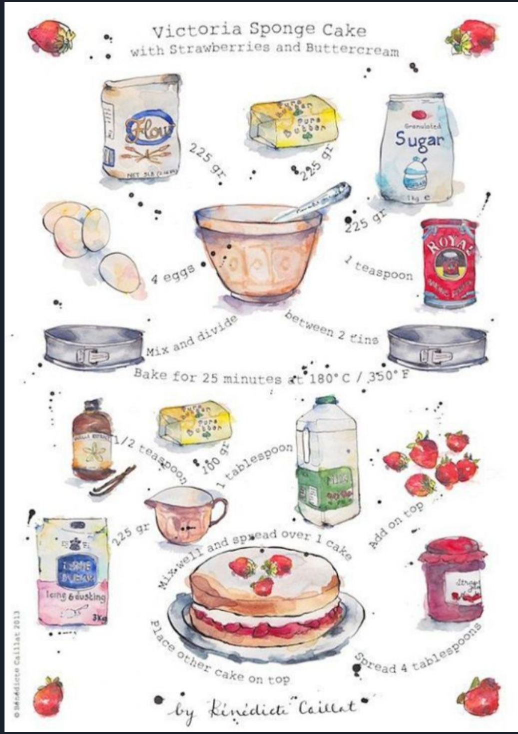
Automated delivery and safe deployment practices help teams identify and mitigate issues quickly when they occur. Maintaining vigilance requires rich telemetry, actionable alerting, and full visibility into applications and underlying systems.

Cake Making

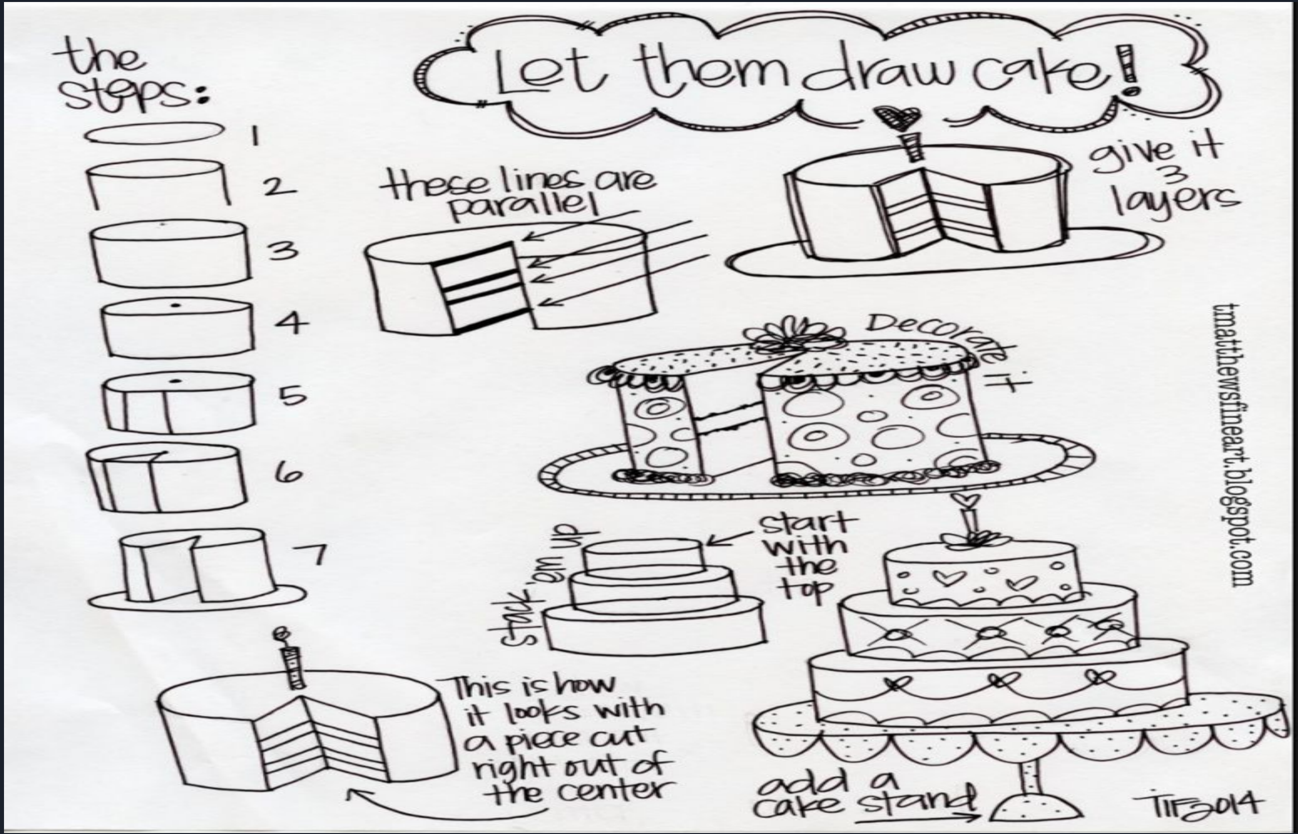
Planning



Raw Materials

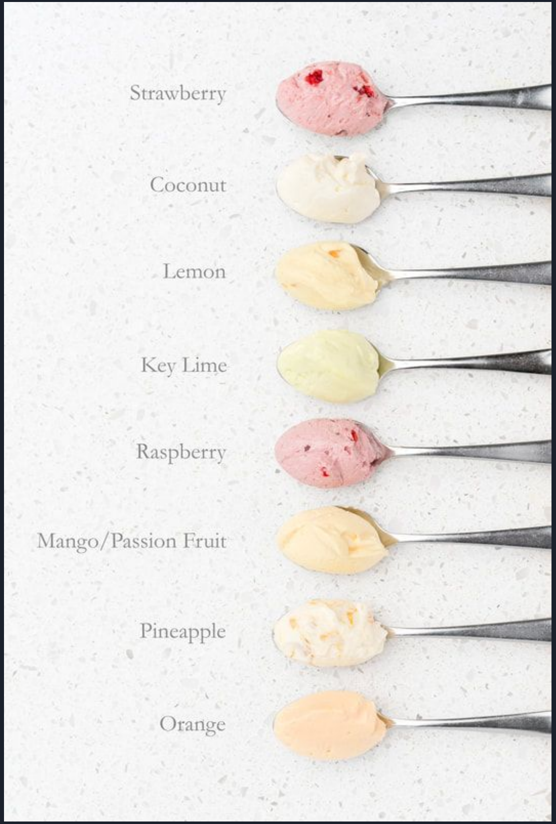


Design

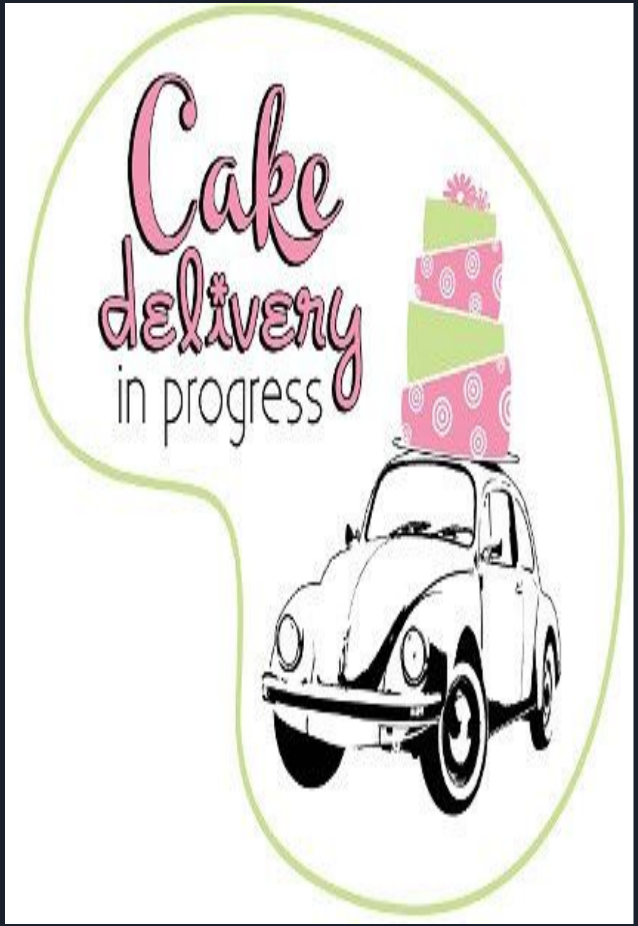

the steps:
1
2
3
4
5
6
7
Let them draw cake!
these lines are parallel
give it 3 layers
Decorate it
tmatthewsfineart.blogspot.com
stack 'em up
start with the top
This is how it looks with a piece cut right out of the center
add a cake stand
TIF2014

Development / Making

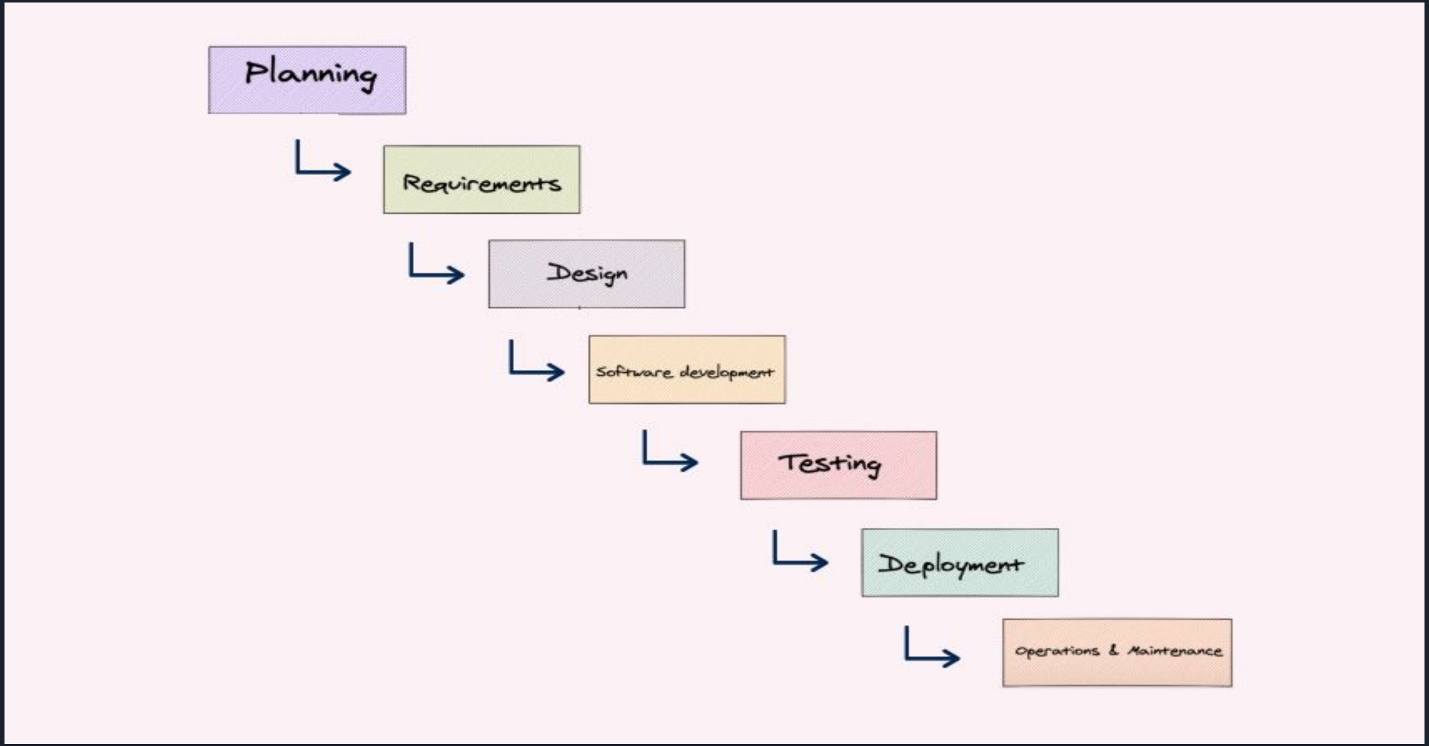
Testing


Strawberry
Coconut
Lemon
Key Lime
Raspberry
Mango/Passion Fruit
Pineapple
Orange

Delivery



SDLC

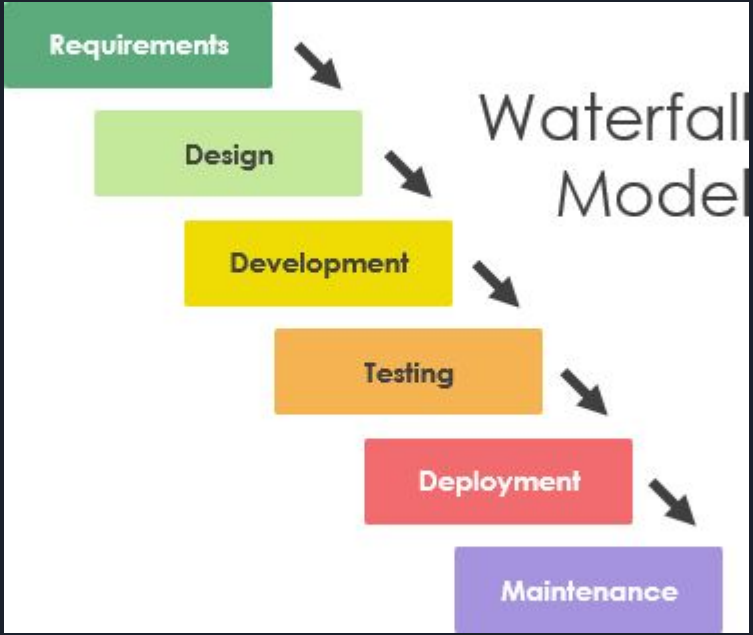

Planning
Requirements
Design
Software development
Testing
Deployment
Operations & Maintenance

Waterfall Model

The waterfall model arranges all the phases sequentially so that each new phase depends on the outcome of the previous phase. Conceptually, the design flows from one phase down to the next, like that of a waterfall.

Pros and cons

The waterfall model provides discipline to project management and gives a tangible output at the end of each phase. However, there is little room for change once a phase is considered complete, as changes can affect the software's delivery time, cost, and quality. Therefore, the model is most suitable for small software development projects, where tasks are easy to arrange and manage and requirements can be pre-defined accurately.



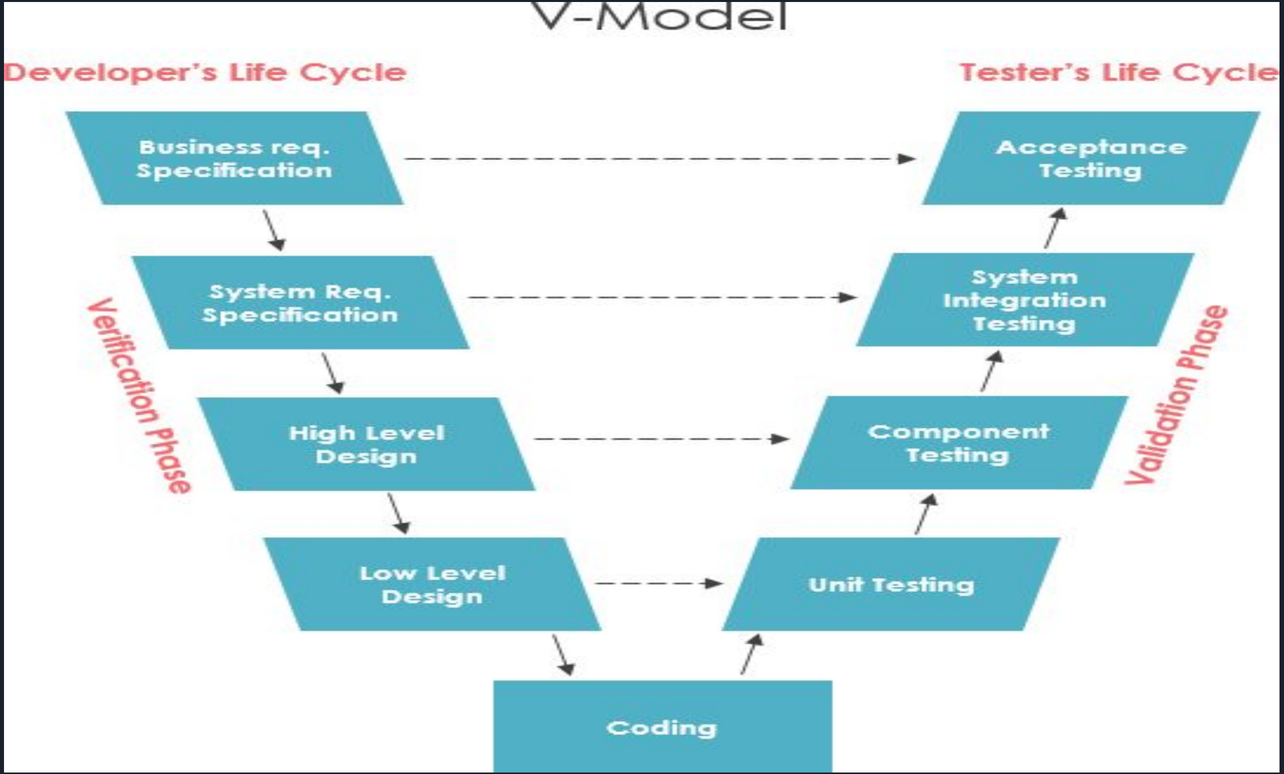
Spiral Model

The spiral model combines the iterative model's small repeated cycles with the waterfall model's linear sequential flow to prioritize risk analysis. You can use the spiral model to ensure software's gradual release and improvement by building prototypes at each phase.

Pros and cons

The spiral model is suitable for large and complex projects that require frequent changes. However, it can be expensive for smaller projects with a limited scope.

V-Model



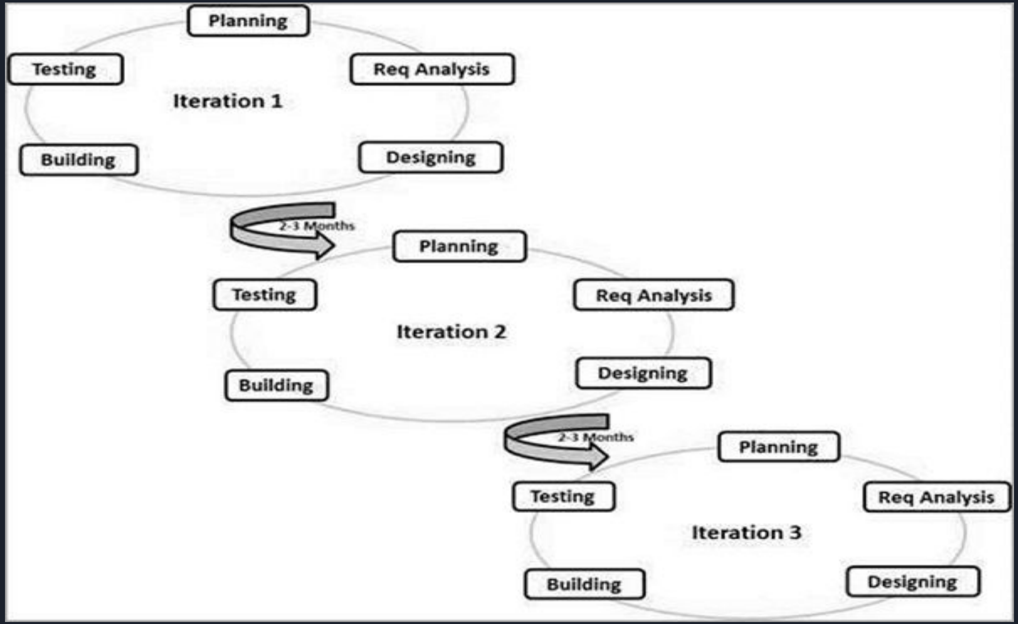
V-Model
Developer's Life Cycle
Tester's Life Cycle
Business req. Specification
Acceptance Testing
System Req. Specification
System Integration Testing
High Level Design
Component Testing
Low Level Design
Unit Testing
Coding
Verification Phase
Validation Phase

What is Agile?

The Agile methodology is a project management approach that involves breaking the project into phases and emphasizes continuous collaboration and improvement. Teams follow a cycle of planning, executing, and evaluating.

Software Agile

Agile is a software development approach that emphasizes team collaboration, customer and user feedback, and high adaptability to change through short release cycles. Teams that practice Agile provide continual changes and improvements to customers, collect their feedback, then learn and adjust based on customer wants and needs. Agile is substantially different from other more traditional frameworks such as waterfall, which includes long release cycles defined by sequential phases.



DEPLOYMENT STRATEGIES

When it comes to production, a ramped or blue/green deployment is usually a good fit, but proper testing of the new platform is necessary.

Blue/green and shadow strategies have more impact on the budget as it requires double resource capacity. If the application lacks in tests or if there is little confidence about the impact/stability of the software, then a canary, a/b testing or shadow release can be used.

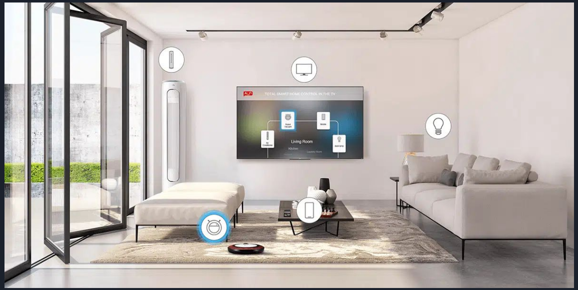
If your business requires testing of a new feature amongst a specific pool of users that can be filtered depending on some parameters like geolocation, language, operating system or browser features, then you may want to use the a/b testing technique.

Strategy	ZERO DOWNTIME	REAL TRAFFIC TESTING	TARGETED USERS	CLOUD COST	ROLLBACK DURATION	NEGATIVE IMPACT ON USER	COMPLEXITY OF SETUP
RECREATE version A is terminated then version B is rolled out	✗	✗	✗	■□□	■■■	■■■	□□□
RAMPED version B is slowly rolled out and replacing version A	✓	✗	✗	■□□	■■■	■□□	■□□
BLUE/GREEN version B is released alongside version A, then the traffic is switched to version B	✓	✗	✗	■■■	□□□	■■□	■■□
CANARY version B is released to a subset of users, then proceed to a full rollout	✓	✓	✗	■□□	■□□	■□□	■■□
A/B TESTING version B is released to a subset of users under specific condition	✓	✓	✓	■□□	■□□	■□□	■■■
SHADOW version B receives real world traffic alongside version A and doesn't impact the response	✓	✓	✗	■■■	□□□	□□□	■■■

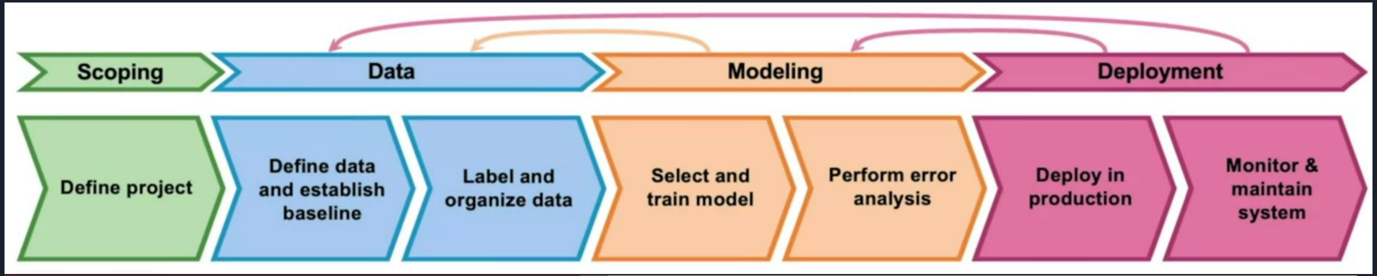
Container Solutions

PATHTOAI

AI is everywhere



Data Science Lifecycle



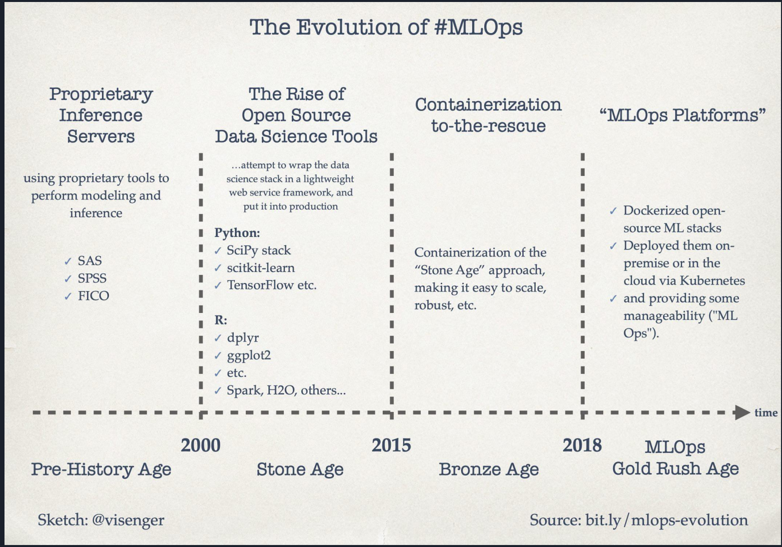
01	Data Extraction	• You select and integrate the relevant data from various data sources for the ML task.
02	Data Analysis	• You perform exploratory data analysis (EDA) to understand the available data for building the ML model.
03	Data preparation	• This preparation involves data cleaning, where you split the data into training, validation, and test sets.
04	Model Training	• The data scientist implements different algorithms with the prepared data to train various ML models and tune them.
05	Model evaluation	• The model is evaluated on a holdout test set to evaluate the model quality. The output of this step is a set of metrics to assess the quality of the model.
06	Model serving / Deployment	• The validated model is deployed to a target environment to serve predictions
07	Model Monitoring	• The model predictive performance is monitored to potentially invoke a new iteration in the ML process.

What is MLOPs?

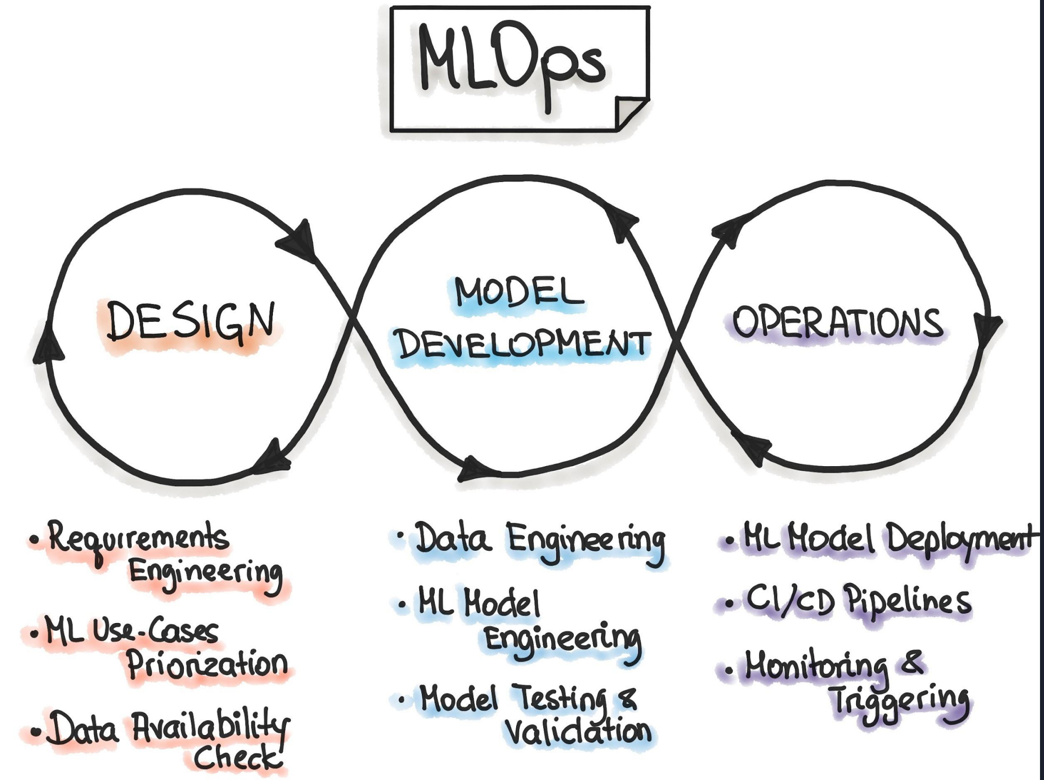
What is MLOPS?

The definition of MLOps (machine learning operations) includes the culmination of **people, processes, practices and underpinning technologies that automate the deployment, monitoring, and management of machine learning (ML) models** into production in a scalable and fully governed way to finally provide measurable business value from machine learning.

Laying an MLOps foundation allows data, development, and production teams to work collaboratively and leverage automation to deploy, monitor, and govern machine learning services and initiatives within an organization.



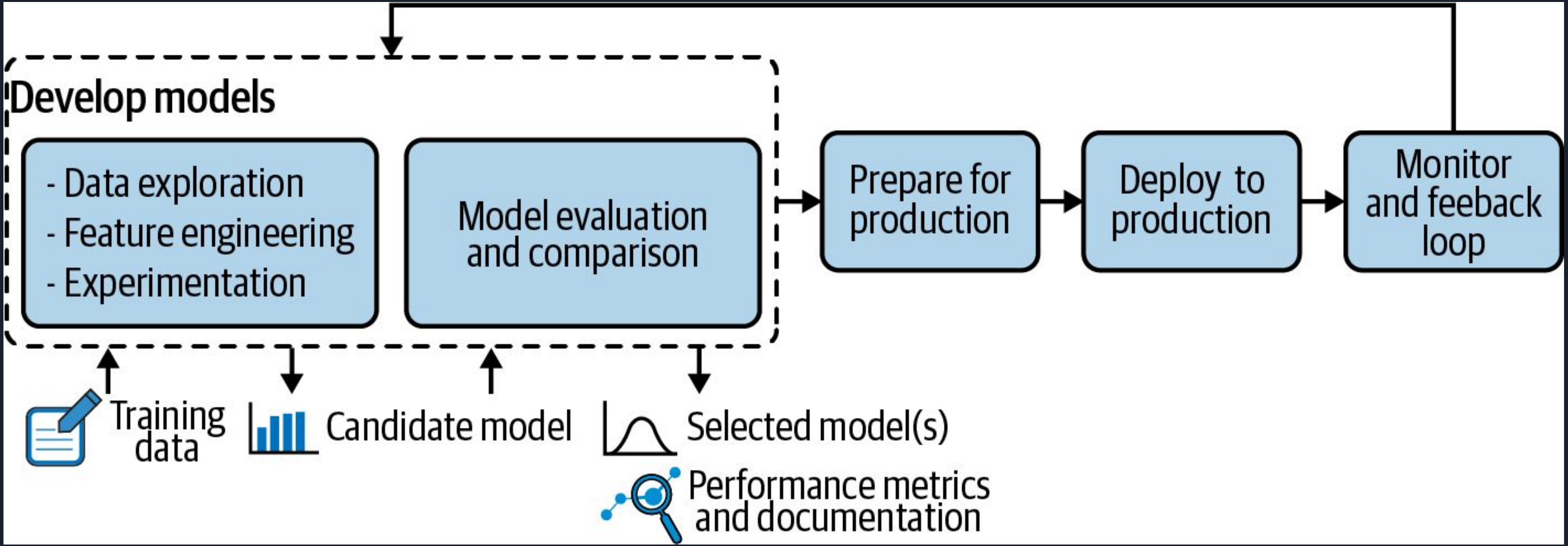
The term MLOps is defined as “the extension of the DevOps methodology to include Machine Learning and Data Science assets as first-class citizens within the DevOps ecology



Developing Models

PATHTOAI

- Anyone who wants to be serious about MLOps needs to have at least a cursory understanding of the **model development** process, which is presented in as an element of the larger ML project life cycle.
- Depending on the situation, the model development process can range from quite simple to extremely complex, and it dictates the constraints of subsequent usage, monitoring, and maintenance of models.



DEVOPS Vs MLOPS

Aspect	DevOps	MLOps
Focus	General software development and operations	Machine learning and AI projects
Goals	Improve collaboration, automation, and efficiency in software development and operations	Automate and streamline machine learning model development, deployment, and management
Key Principles	Continuous integration, continuous delivery, automated testing, infrastructure as code, collaboration	Version control for data and models, model monitoring, model deployment automation, reproducibility
Expertise Required	Software development, system administration, automation tools	Machine learning, data engineering, model development, automation tools

Aspect	DevOps	MLOps
Lifecycle Management	Focuses on software development lifecycle from code commit to production deployment	Focuses on the end-to-end machine learning lifecycle, including data preparation, model training, deployment, and monitoring
Tools and Technologies	CI/CD tools (e.g., Jenkins, Travis CI), containerization (e.g., Docker), orchestration (e.g., Kubernetes), IaC (e.g., Terraform)	ML frameworks (e.g., TensorFlow, PyTorch), data processing tools (e.g., Apache Spark), model deployment platforms (e.g., Kubernetes with Kubeflow), ML monitoring tools (e.g., Prometheus, Grafana)
Key Challenges	Ensuring code stability, rapid release cycles, infrastructure management	Data versioning, model drift detection, managing diverse data sources, model explainability, hardware acceleration for model inference
	Faster development cycles, reduced errors, improved	Improved model reliability, faster model deployment, better model

DEVOPS vs MLOPS

DevOps is a set of practices in the traditional software development world that enables faster, more reliable software development into production. DevOps relies on automation, tools, and workflows to abstract accidental complexity away and allow developers to focus on more critical issues.

The reason that DevOps is not simply applied to ML is that ML is not merely code, but **code and data**. A data scientist creates an ML model that is eventually placed in production by applying an algorithm to training data. This huge amount of data affects the model's behavior in production. The behavior of the model also hinges on the input data that it receives at the time of prediction.

DEVOPS vs MLOPS

ML systems are similar to other software systems in many ways, but there are a few important differences:

CI includes testing and validating data, data schemas, and models, rather than being limited to testing and validating code and components.

CD refers to an ML training pipeline, a system that should deploy a model prediction service automatically, rather than a single software package or service.

In addition, ML systems feature **continuous testing (CT), a property that deals with retraining and serving models automatically** to ensure that the team can better assess business risks using immediate feedback throughout the process.

DEVOPS vs MLOPS

Teams: The team working on an ML project typically includes data scientists who focus on model development, exploratory data analysis, research, and experimentation. In contrast to team members on the DevOps side, these team members might not be capable of building production-class services as experienced software engineers are.

Development. To rapidly identify what best addresses the problem, ML is by necessity experimental. Team members test and tweak various algorithms, features, modeling techniques, and parameter configurations in this vein, but this creates challenges. Maximizing the reusability of code and maintaining reproducibility while tracking which changes worked and which failed are chief among them.

Testing. Compared to other software systems, testing an ML system is much more involved. Both kinds of systems require typical integration and unit tests, but ML systems also demand model validation, data validation, and quality evaluation of the trained model.

Deployment. ML models trained offline are not simply deployed in ML systems in real-world conditions for reliable predictions and results. Many situations demand a multi-step pipeline to automatically retrain the model before deployment. To create and deploy this kind of machine learning project pipeline, it is necessary to automate ML steps that data scientists complete manually before deployment as they validate and train the new models—and this is very complex.

Production. ML models are subject to more sources of decay than are conventional software systems, such as data profiles that are constantly changing and suboptimal coding, and it is essential to consider this degradation and reduced performance. Tracking summary data statistics and monitoring online model performance is critical, and the system should be set to catch values that deviate from expectations and either send notifications or roll back when they occur.

ML in research vs. in production

	Research	Production
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting

Data

Research	Production
● Clean ● Static ● Mostly historical data	● Messy ● Constantly shifting ● Historical + streaming data ● Biased, and you don’t know how biased ● Privacy + regulatory concerns

THE COGNITIVE CODER

By Armand Ruiz, Contributor, InfoWorld | SEP 26, 2017 7:22 AM PDT

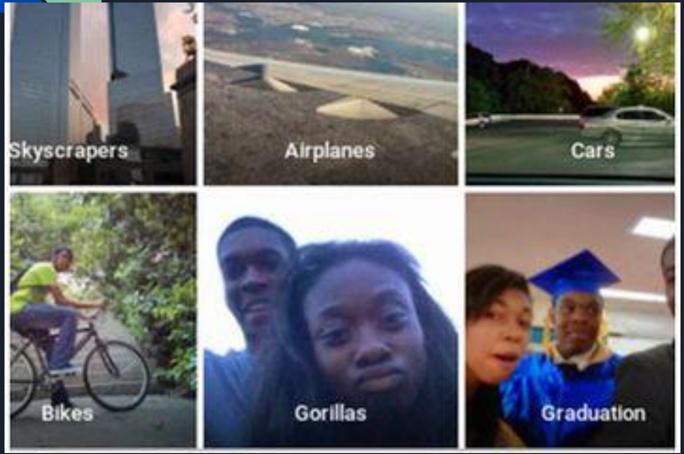
The 80/20 data science dilemma

Most data scientists spend only 20 percent of their time on actual data analysis and 80 percent of their time finding, cleaning, and reorganizing huge amounts of data, which is an inefficient data strategy

ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important

Fairness

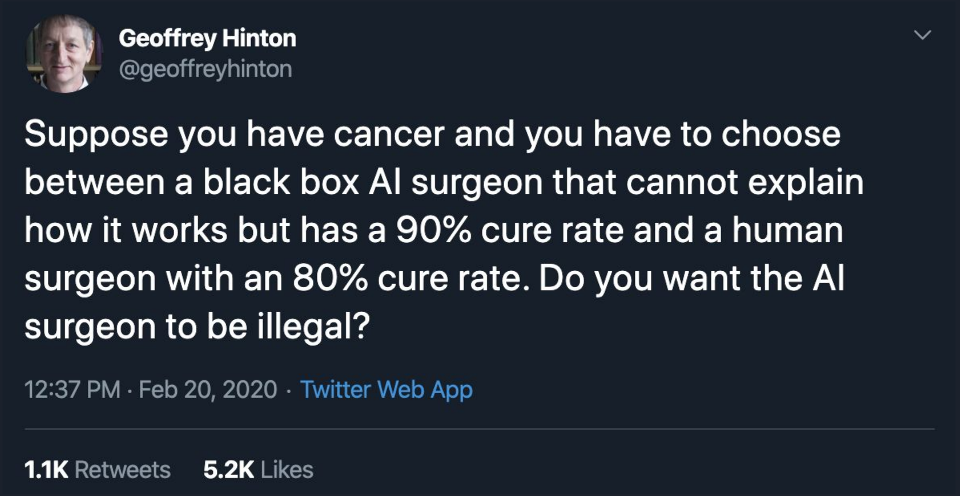


The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability*	Good to have	Important

Interpretability



Geoffrey Hinton
@geoffreyhinton

Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?

12:37 PM · Feb 20, 2020 · Twitter Web App

1.1K Retweets **5.2K** Likes

Zoom poll: which one would you want as your surgeon?

Interpretability

Geoffrey Hinton
@geoffreyhinton

Suppose you have cancer and you have to choose between a black box AI surgeon that cannot explain how it works but has a 90% cure rate and a human surgeon with an 80% cure rate. Do you want the AI surgeon to be illegal?

12:37 PM · Feb 20, 2020 · Twitter Web App

1.1K Retweets **5.2K** Likes

ML in research vs. in production

	Research	Production
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability	Good to have	Important

Myth #1: Deploying is hard

Myth #1: Deploying is hard

Deploying is easy. Deploying reliably is hard

Myth #2: You only deploy one or two ML models at a time

Myth #2: You only deploy one or two ML models at a time

Booking.com: 150+ models, Uber: thousands

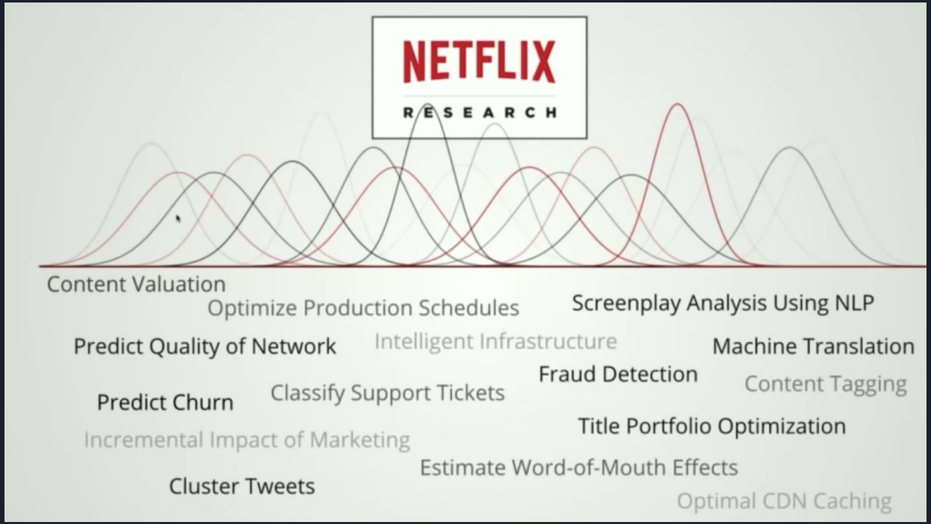


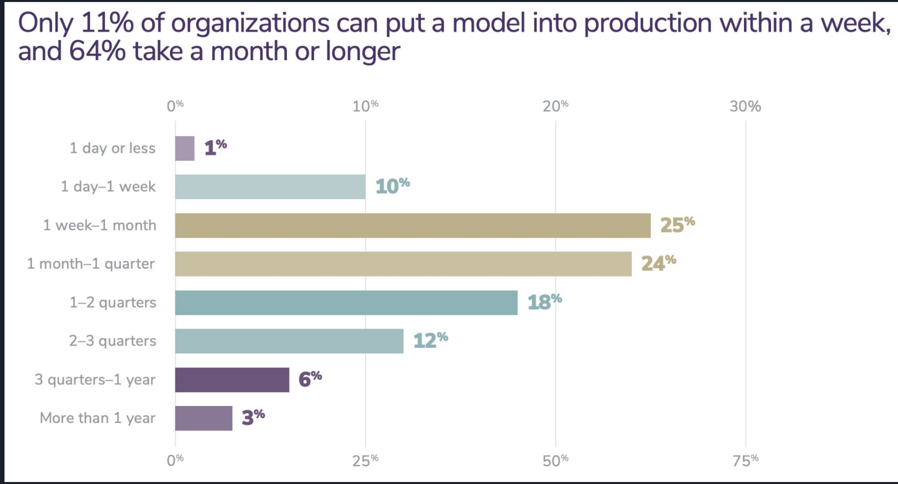
Image from Ville Tuulos (Netflix, Outerbounds)

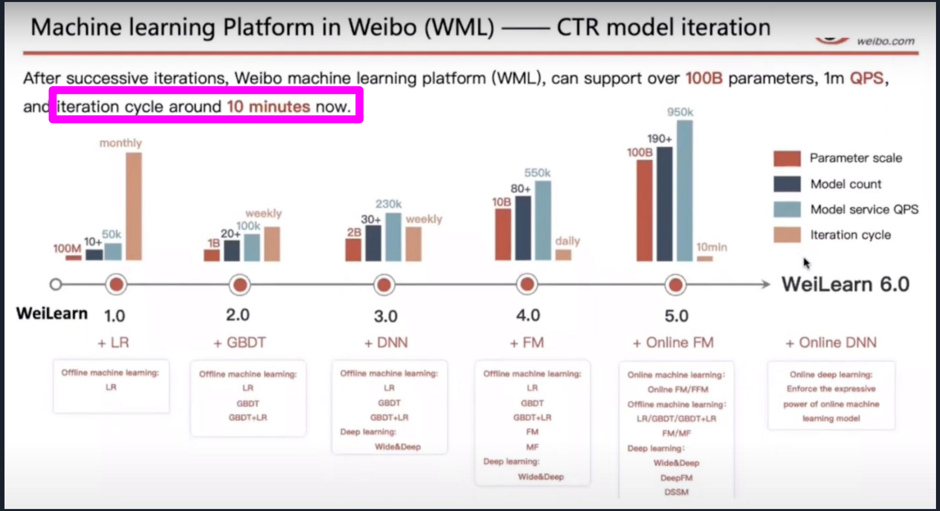
Myth #3: You won’t need to update your models as much

DevOps: Pace of software delivery is accelerating

- Elite performers deploy **973x** more frequently with **6570x** faster lead time to deploy (Google DevOps Report, 2021)
- DevOps standard (2015)
  - Etsy deployed 50 times/day
  - Netflix 1000s times/day
  - AWS every 11.7 seconds

DevOps to MLOps: Slow vs. Fast





Left image from Algorithmia | Right image: Machine learning with Flink in Weibo (Qian Yu, QCon 2019)

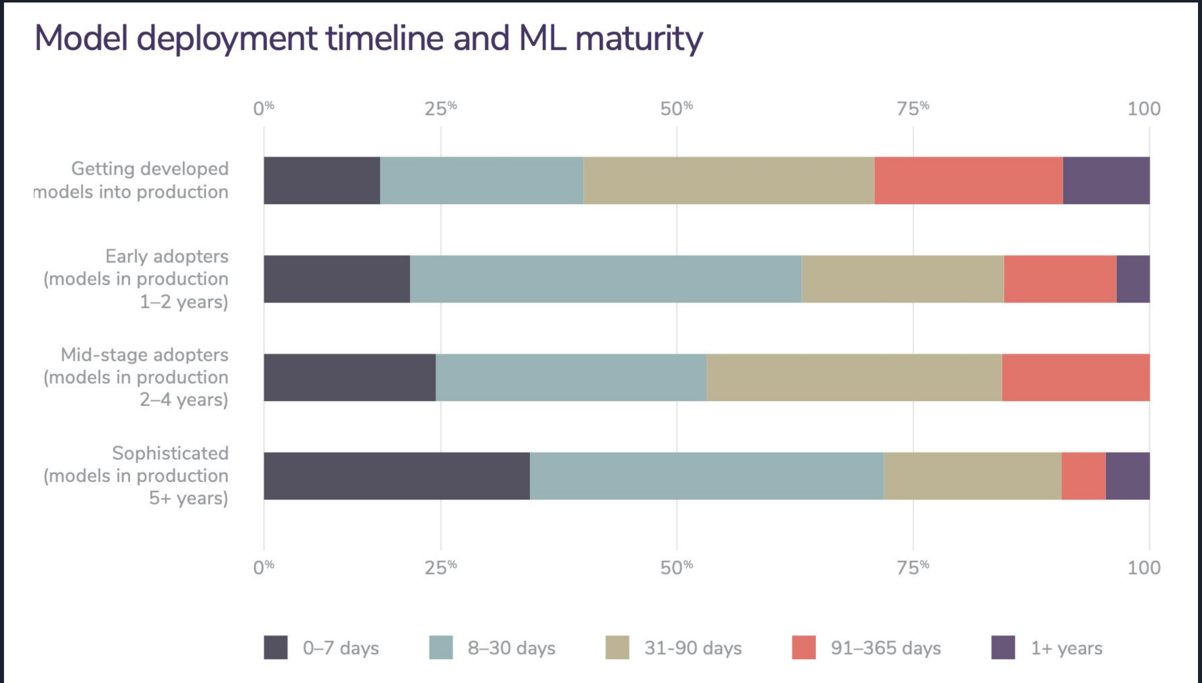
Accelerating ML Delivery

Myth #4: ML can magically transform your business overnight

Myth #4: ML can magically transform your business overnight

Magically: possible
Overnight: no

Efficiency improves with maturity



ML engineering is more engineering than ML

MLEs might spend most of their time:

- wrangling data
- understanding data
- setting up infrastructure
- deploying models

instead of training ML models

Hands On

https://github.com/ashishpal2702/ML_template

Thank You