Dr. D. Y. Patil Pratishthan's Institute for Advanced Computing &Software Development IACSD

DevOps

INDEX

What is Agile?

Agile is an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments. Requirements, plans, and results are evaluated continuously so teams have a natural mechanism for responding to change quickly.

Agile methodologies:

The term Agile actually refers to a concept, not a specific methodology. There are many, and sometimes conflicting, methods that can be used under the Agile umbrella. These include;

- Agile Unified Process
- Behaviour Driven Development (BDD)
- Crystal Clear
- Dynamic Systems Development Method (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Kanban
- Lean Development
- Rapid Application Development (RAD)
- IBM - Rational Unified Process (RUP)
- Scrum
- Test Driven Development (TDD),

SCRUM OVERVIEW

Scrum is described as a ‘framework within which you can employ various processes and techniques’, rather than a process, or a technique,

for building products. The Scrum framework is primarily team based, and defines associated roles, events, artefacts and rules. The three primary roles within the Scrum framework are: 1. The product owner who represents the stakeholders, 2. The scrum master who manages the team and the Scrum process 3. The team, about 7 people, who develop the software. Each project is delivered in a highly flexible and iterative manner where at the end of every sprint of work there is a tangible deliverable to the business. This can be seen in the following diagram

SCRUM Framework

The requirements that form the basis of the project are collated into what is called a Project Backlog, and is updated regularly. The features that are associated with these requirements are termed User Stories. This relationship is illustrated in the following diagram:

SCRUM Project Structure

The work is time-boxed into a series of 1-to-4-week cycles where the business and project team estimate which User Stories in descending priority order are achievable each cycle, or Iteration. This subset of User Stories from the Project Backlog forms the basis of the Iteration Backlog planned for delivery over that two-week period. Under Scrum, there are 3 timeboxed (or fixed duration) meetings held during an Iteration plus a daily stand-up meeting for the team, scrum master and (ideally) the product owner. At the beginning of a sprint, features to be developed during the sprint are decided during the sprint planning meeting. At the end of the Iteration are another 2 meetings, the Iteration review and Iteration retrospective where the team reviews the product and demonstrates the use of the software, as well as reflect on, and improve, the Iteration process itself. After the sprint is complete, the next set of User Stories is selected from the Project Backlog and the process begins again. Burn rate is monitored to determine when funding will be exhausted.

TABLE: KEY SCRUM CONCEPTS

Concept	**Description**
Project	Discreet set of end user requirements that have been grouped, prioritised and funded.
Requirement	The end user statement that outlines their information need.
Sprint	A sprint is a 1-to-4-week time-boxed event focused on the delivery of a subset of User Stories taken from the Project Backlog.
Project Backlog	The Project Backlog is the current list of User Stories for the Project. User Stories can be added, modified or removed from the Backlog during the Project.
Sprint Backlog	Subset of User Stories from the Project Backlog that are planned to be delivered as part of a Sprint.
User Stories	The User Story is a one- or two-line description of the business need, usually described in terms of features.
Tasks	Tasks are the activities performed to deliver a User Story.
Technical Debt	This refers to items that were either: • missing from the Planning meeting; or • deferred in favor of early delivery.

Kanban

In Japanese, Kanban (看板) means signboard or billboard. Taiichi Ohno (February 29, 1912 - May 28, 1990), an industrial engineer at Toyota, developed Kanban to improve production effectiveness and decrease wastes. Kanban ended up being an efficient framework to support running a production system as a whole and an excellent way to promote improvement. Identification of the lead time and the cycle time of a given process and its associated sub-processes, and incompatibilities among them highlight problem areas.

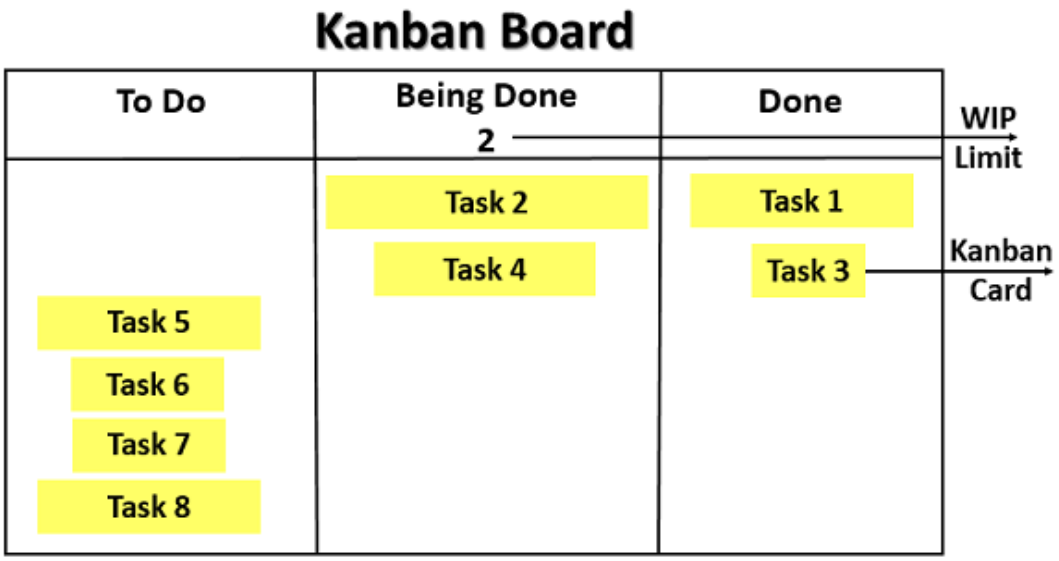
One of the main differences of Kanban compared to other processes is that it explicitly establishes an upper limit to work in progress inventory to prevent overcapacity. Less is more to get results (Remember how the Google landing page looks like). However, as human beings, we are tempted to get trapped with Complexity Bias. Kanban establishes maximum limits on the number of products waiting at supply points. Afterward, the Kanban team identifies and addresses any inefficiencies in their workflow. Whenever a limit is not honoured, this points to an inefficiency to be sorted out and a process improvement potential to be exploited. Therefore, it's safe to say that the primary goal of a Kanban system is to restrict the accumulation of excess inventory. The purpose of the Kanban team is to eliminate this excess inventory at any point in production. That will lead to better allocation of available resources (human, tools, financial) to increase business throughput and profitability, and to remove wastes, bottlenecks in the processes.

The core concept of Kanban includes −

- Visualize Workflow
    - Split the entire work into defined segments or states, visualized as named columns on a wall.
    - Write each item on a card and put in a column to indicate where the item is in the workflow.

- Limit WIP
    - Assign explicit limits to how many items can be in progress at each workflow segment / state. i.e., Work in Progress (WIP) is limited in each workflow state.
- Measure the Lead Time
    - Lead Time, also known as cycle time is the average time to complete one item. Measure the Lead Time and optimize the process to make the Lead Time as small and predictable as possible.

This concept of Kanban is a direct implementation of a Lean Pull Scheduling System. An item can move to the next segment / state only when it obtains a slot in there.



Kanban - Lean Practices

The implementation of Kanban, as well as other Lean Manufacturing Methods, such as Kaizen, can have significant benefits for almost any type of work. Kanban is more effective because it visually indicates when the production should start and stop. It is faster, more efficient, and saves significant money over most other production models. It is also far more directly responsive to customer demand.

Kanban - Benefits

- Bottlenecks become clearly visible in real-time. This leads people to collaborate to optimize the whole value chain rather than just their part.
- Useful for situations where operations and support teams have a high rate of uncertainty and variability.
- Tends to spread throughout the organization naturally, including sales and management. This increases visibility of everything that is going on at the company.
- Reduces inventory in the range of 25%-75%, thereby reducing company costs.
- Since all segments/states in the workflow are visually organized, the required items, reducing the wait times and ensuring speed, continually support all the tasks in the workflow.
- Overproduction of inventory is avoided, thereby saving resources and time as well. This is termed as eliminating waste.

Alignment with Agile

In agile, if values are combined with Kanban characteristics, the outcome would be Agile Kanban. This practice is gaining popularity in Software Development wherein the Agile iteration approach and Kanban value stream focus are combined.

Flexibility in Planning

Kanban provides improvements in the workflow. With visual representation of the workflow, speed of moving from one task to another is reduced. This is accomplished through the creation of clearly marked flow lanes, Kanban cards and clearly marked columns to indicate where each item is in the workflow. If a task needs longer duration, it is allowed to execute without hindrance, and at the same time, the tasks that are completed will flow to the next state.

This allows −

- Sufficient duration for longer tasks that cannot be broken down logically.
- Preservation of value of such longer tasks.
- Effort required by each role to be expended.
- Continuous flow of the tasks that are completed without wait time.

Hence, planning is flexible and not time-boxed.

Limits Work-In-Progress (WIP)

Explicit limits are assigned to number of items that can be in progress at each workflow state, indicated by a column.

This allows −

- Reducing wait time.
- Avoiding stress on resources at a workflow state.
- Identifying bottlenecks causing an item to be in a workflow state than the anticipated time (usually average cycle time) immediately.
- Resolving bottlenecks with collaboration of the entire team.
- Decreasing dependencies in completing a task by splitting it into sub-tasks, so that the sub-task is tracked independently.

Pull Approach

When you have two teams and the first one is performing better than the second one, it is likely that it pushes more work than the other can actually handle. This often creates friction between the teams. A solution to this is the Pull approach.

In Pull Approach, the next team pulls work only when it is ready for it. Pull Approach is implemented by adding a buffer with limited capacity between the two teams.

The benefits of Pull Approach are −

- Avoids piling-up of work.
- Reduces wait time.

- Facilitates a team to maintain constant pace and focus on quality.
- Provides resource balancing.

Minimize Cycle Time

The cycle time for each task is measured and the process is optimized to reduce the cycle times.

- The bottlenecks are identified immediately and resolved collaboratively by the entire team.
- The correction loops are considered to reduce rework.

Continuous Delivery

Benefits of continuous delivery are −

- Short release cycles result in continuous delivery of growing product at regular intervals.
- Continuous interactions with the customer.
    - To understand what customer wants.
    - Not to produce anything that the customer does not need.
    - Feedback on delivered modules.
- Limited requirements in each release cycle.
    - Developers are not overloaded with requests. This enables them to focus on the delivery.
    - There is no partially completed work.
- Focus is on finishing work than on starting work.
    - This enables focus on sustaining pace and quality of the product.
    - Deliver before the customer changes mind.
- Optimize flow of Work from beginning to end.
    - Helps in incremental process improvements.

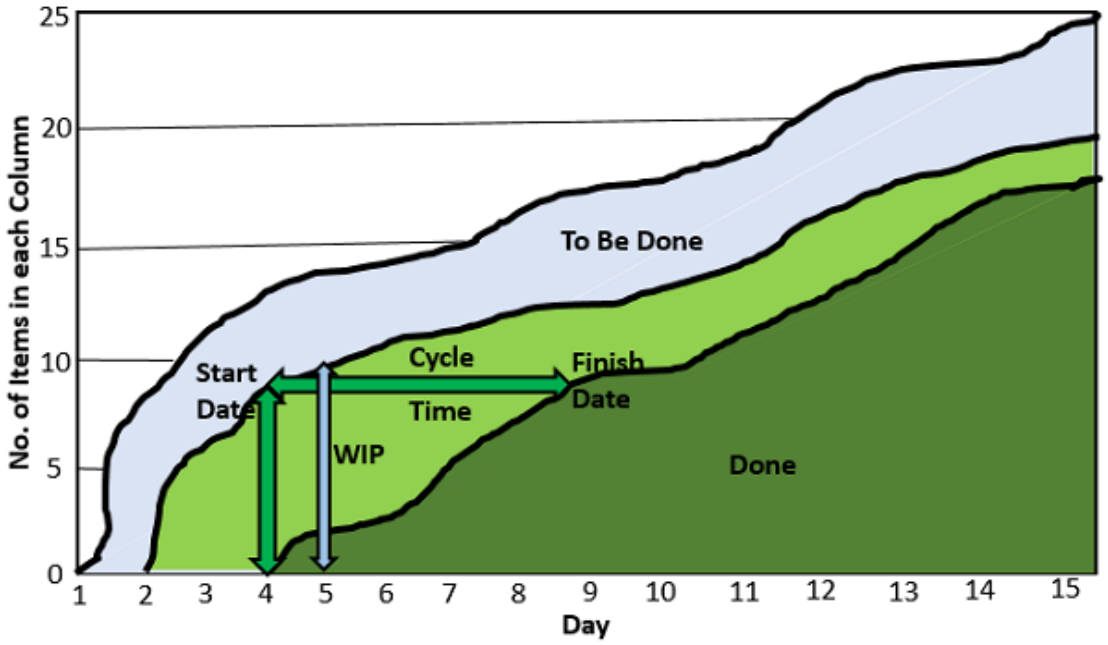
Visual Metrics

Visually organized workflows (on Kanban Boards) facilitate −

- Scheduling as per WIP limits on a workflow state.
- Tracking status and progress continually.
- Assigning resources dynamically based on the role requirements.

Advantages of Visual Metrics

Each day, for each column, mark how many tasks are in it, you will see a mountain-like chart. This chart shows the past performance and allows predicting future results.



You can gather the following information from the chart −

- Measure cycle time for each feature (or story) by marking a start date when the feature is scheduled and an end date when the feature finishes.
- Evaluate the quality of the growing product from technical, functional and user perspectives at regular time-boxes.
- Evaluate the pace of development by looking at the number of development items completed and looking at the average cycle time per development item.

- Adjust the pace of development by calculating the ratio of developer days per completed development item. You can use this ratio to estimate the completion time for the yet-to-develop items and adjust the development plan as necessary.
- Evaluate and adjust the process by using a collaborative session to identify changes that can be made to improve the quality of the product, or to improve the pace of development.
- Identify and resolve un-validated decisions by looking at the cycle time of validated decisions and focusing on the correction loops that are usually the invisible backed-up queues.

Efficiency Through Focus

By focusing on what a customer wants, the scope becomes clear. The focus is on delivering value to the customer.

Efficiency can be achieved in the following ways −

- A customer's expectations can be made realistic and focused with continuous interactions with the customer.
- Focus on the tasks is ensured with a limit on work-in-progress (WIP).
- The Pull approach enables resources to complete the tasks at hand before a new task is taken up.
- Optimizing lead-time (cycle time) results in faster delivery.
- Visualization of the workflow with Kanban board draws immediate attention to any bottlenecks that can be resolved immediately.
- Empowerment of the team makes the team accountable for the success.

Kanban is adapted to software development as a project management approach. Kanban in software development supports a continuous workflow, termed as Value Stream.

Value Stream

The Value Stream consists of all actions required to bring a project from creation to completion.

The actions can −

- Add Value to the project
- Add no Value, but unavoidable
- Add no Value, avoidable (termed as waste)

Elimination of Waste

Anything that does not add any value to the project is known as Waste. Kanban facilitates elimination of waste.

In software development, there are three types of waste −

- Waste in code development
- Waste in project management
- Waste in team potential

Waste in Code Development

Waste in code development is due to the following reasons −

- **Partially completed work** − The partially completed work can become outdated and unusable. It can be eliminated with iterative cycles and with modular code that completes within the iteration.
- **Defects** − In developing a code, correction and retesting requires time and resources. It can be eliminated with up-to-date test suite, completing testing within the iteration and continuous customer feedback.

Waste in Project Management

Waste in project management is due to the following reasons −

- **Extra Processes** − Unnecessary documentation that requires time and resources. It can be eliminated with −
    - Pre-planning of what processes are relevant and necessary.

- Documentation review, that ensures relevant and necessary processes are followed.

- **Code Handoffs** − means passing the work from one person or team to another, after the first person's work is complete. It may give rise to lack of knowledge. It can be eliminated by keeping the flowcharts and wireframes visible and clear.
- **Extra Functions** − These are features that are not required by the customer. Effort and time are wasted in developing the functions required to implement the features that the customer does not want. It can be eliminated with continuous interaction with customer and testers involving in the requirements gathering as they can better visualize the scenarios and expected behavior of the system.

Waste in Team Potential

Waste in team potential is due to the following reasons −

- **Task Switching** − It leads to the danger of multi-tasking, which is a waste. It can be eliminated with focus on a task with every release. Large process steps are segmented into tasks to −
  - Improve visibility
  - Reduce dependencies
  - Enable easy flow of work
  - Focus on the cycle-time of delivered work
  - Give a way to detect and resolve bottlenecks
- **Waiting** − Time for getting instructions or information – Team is subjected to sit idle if the decisions are not made by the team, or if the information provided to the team (developers, testers, etc.) are expensive resources. It can be eliminated by allowing the team members (developers, testers, etc.) to −
  - Take decisions so that they do not have to wait for instructions
  - Have access to information so that it can be used as and when required

Agile Kanban is Agile Software Development with Kanban approach. In Agile Kanban, the Kanban board is used to visualize the workflow. The Kanban board is normally put up on a wall in the project room. The status and progress of the story development tasks is tracked visually on the Kanban board with flowing Kanban cards.

Kanban Board

Kanban board is used to depict the flow of tasks across the value stream. The Kanban board −

- Provides easy access to everyone involved in the project.
- Facilitates communication as and when necessary.
- Progress of the tasks are visually displayed.
- Bottlenecks are visible as soon as they occur.

Advantages of Kanban board

The major advantages of using a Kanban board are −

- **Empowerment of Team** − This means −
    - Team is allowed to take decisions as and when required.
    - Team collaboratively resolves the bottlenecks.
    - Team has access to the relevant information.
    - Team continually communicates with customer.
- **Continuous Delivery** − This means −
    - Focus on work completion.
    - Limited requirements at any point of time.
    - Focus on delivering value to the customer.
    - Emphasis on whole project.

The tasks and stories are represented by Kanban cards. The current status of each task is known by displaying the cards in separate columns on the board. The columns are labeled as **To Do, Doing**, and **Done**. Each task moves from **To Do** to **Doing** and then to **Done**.

Kanban Board is updated on a daily basis as the team progresses through the development.

WIP Limit

The label in the Doing column also contains a number, which represents the maximum number of tasks that can be in that column at any point of time. i.e., the number associated with the **Doing** column is the WIP (Work-In-Progress) Limit.

Pull Approach

Pull approach is used as and when a task is completed in the Doing column. Another card is pulled from the To Do column.

Self-directing

In Agile Development, the team is responsible for planning, tracking, reporting and communicating in the project. Team is allowed to make decisions and is accountable for the completion of the development and product quality. This is aligned to the characteristic of empowerment of the team in Kanban.

Continuous Flow

In Agile development, there is no gate approach and the work flows across the different functions without wait-time. This contributes in minimizing the cycle time characteristic of Kanban.

Visual Metrics

In Agile Kanban, the metrics are tracked visually using −

- Kanban Board
- Burndown Chart

Uses of Kanban board

Kanban Board is used to −

- Measure the cycle times, that can be used to optimize average cycle time.
- Track WIP limit to eliminate waste.
- Track resource utilization to eliminate waste.

Uses of Burndown chart

Burndown chart is used to capture −

- The current status of the tasks and stories.
- The rate of progress of completing the remaining tasks.

As Kanban Board is updated daily, it contains all the information that is required by the Burndown charts.

In Agile Kanban, the user stories are broken into tasks and Kanban cards are used to track the tasks on the Kanban board. Agile Kanban has a concept of iteration that is not present in Kanban. Further, no processes are considered.

Kanban in Value Stream

Kanban is defined to be executed in value stream with focus on delivery of value. Kanban in software development can be visualized as the features flowing across the value stream. All the Kanban characteristics (Refer Chapter - Characteristics of Kanban in this Tutorial) are met in the Kanban approach for software development.

Feature Kanban Board

Feature Kanban Board is used to track the Feature Driven Development with Kanban Approach. Each Feature is assigned to a particular release. The columns in the Kanban board represent releases. Hence, each column contains all the features assigned to the release represented by it.

Each feature is broken into stories. Each release is broken into iterations. The iteration is executed in an Agile Development approach. This can be treated as a sub-stream in the value stream, with the stories to be completed within that iteration assigned to it.

Agile Kanban in Sub-stream

Agile Kanban approach is followed within each sub-stream that is implemented as an iteration. Each story is broken into tasks in the iteration. Task Kanban board is used to track the status and progress of the story development tasks. The current status of each task is known by displaying the cards in separate columns on the board. The columns are labeled as To Do, Doing, and Done. Each task moves from To Do to Doing and then to Done.

Continuous Delivery

Continuous delivery to the customer is ensured with features tracked on feature Kanban board and stories representing features tracked on task Kanban board.

Delivery through a release is accomplished by −

- Continuous tracking
- Constant communication with the customer
- Adjusting development plan as required
- Focusing on delivery of value to the customer

Agile development as well as Kanban maintain team collaboration. This, in turn helps in identifying and resolving Bottlenecks immediately as required by Kanban. This results in accomplishment of all the needed tasks within the iteration to deliver quality product, which meets customer expectations.

Continuous Process Improvement

Kanban supports process improvements to enhance the delivery approach continuously.

Consider a requirement that is a change or addition to the product. In such a case, Kanban cards can be used to visualize the requirement passing through the processes of analysis, design, development, product integration and testing. This is different from the Waterfall approach in the sense that it does not require completion of one process for all the requirements to flow to the next process in the sequence.

Such an implementation of Kanban in product maintenance allows maintainability, reliability and integrity of the product. The required process improvements are gathered at regular intervals and implemented on a continuous basis.

Kanban and Scrum - Similarities

Similarities between Kanban and Scrum are −

- Both are Agile.
- Both use pull scheduling.
- Both limit WIP, Kanban at task level and Scrum at sprint level.
- Both use transparency across the development.
- Both focus on delivering releasable software early.
- Both are based on self-organizing teams.
- Both require breaking the work into pieces.
- In both the methods, the release plan is continuously optimized based on empirical data (Scrum – Velocity, Kanban - Lead Time/Cycle Time).

Kanban and Scrum - Differences

The differences between Kanban and Scrum are as follows −

Sr. No	Scrum	Kanban
1	Scrum prescribes roles.	In Kanban, roles are optional.
2	Product backlog is to be prioritized.	Prioritization is optional.

3	Sprints are to be time-boxed. You can choose the length of the sprint, but once chosen, the same length is to be maintained for all the sprints.	Time-boxed iterations are optional.
4	Scrum team needs to commit to a particular amount of work for the sprint.	Commitment is optional.
5	Cross-functional teams are prescribed.	Cross-functional teams are optional. Specialist teams are allowed.
6	Uses velocity as default metric for planning and process improvement.	Uses lead time (cycle time) as default metric for planning and process improvement.
7	Items such as stories, tests must be broken down so that they can be completed within one sprint.	No particular item size is prescribed.
8	Sprint backlog shows what tasks are to be executed during the current sprint. These tasks are displayed on Scrum board. Scope of the sprint is fixed. WIP is limited per unit of time (WIP limit is the velocity).	Tasks are defined at workflow level. WIP is limited per workflow state.
9	Additions/Changes cannot be done within a sprint.	Additions /changes can be done if WIP limit is not crossed.
10	New Scrum board is set at the beginning of every sprint.	Kanban board is persistent.

11	Daily meetings need to be conducted.	Daily meetings are optional.
12	Burn-down charts are prescribed.	No particular chart is prescribed.

Kanban vs. Scrum

The following advantages can help you choose between Kanban and Scrum −

- You need to choose Kanban if you already have working processes and you want to improve without disturbing the whole system whereas you need to choose Scrum if you want to introduce a new process in the organization.
- You can use Kanban in the product development with Feature Driven Development to track the workflows in the value stream whereas you can use Scrum for the development in each iteration.
- You need to define the WIP Limits in Kanban explicitly whereas you need to define the sprint length in scrum that imposes WIP limits implicitly.
- Both Kanban and Scrum are adaptive but Scrum is more prescriptive than Kanban.
- Kanban imposes only two Rules: Visualize workflow and limit WIP whereas Scrum imposes more constraints such as time-boxed Sprints.
- Kanban leads to organizational process improvements, both in management and development. Kanban also supports maintenance activities. Scrum leads to high throughput in small development teams. It does not contribute to product development and maintenance workflows that are longer in duration with unpredictability on the size of work units and changes. Scrum does not emphasize on optimizing management activities.
- In Kanban, you can choose when to do planning, process improvement, and release. You can choose to do these activities on a regular basis or on-demand. Scrum iteration is one single

time-boxed Sprint combining three different activities: planning, process improvement, and release (if required).

Thus, Kanban and Scrum are effective tools in their specific contexts. You can combine Kanban and Scrum to derive maximum benefits from both.

Adapting Kanban and Scrum Together

You can use Kanban and Scrum together by implementing those characteristics that will suit your needs. The constraints of both need to be considered before adapting them. For instance, Scrum requires Time-boxed Sprints and if you do away with those, you cannot say that you have implemented Scrum. Both give you a basic set of constraints to drive your own process improvement.

What is lean?

Lean has proven to be an effective model for managing teams in some of the most demanding industries, like software development, manufacturing, construction, and many others. A huge role in this has the fact that the methodology is simple to understand and quick to make an impact when implemented properly.

Implementation of Lean

Step 1: Specify Value

Define value from the perspective of the final customer. Express value in terms of a specific product, which meets the customer's needs at a specific price and at a specific time.

Step 2: Map

Identify the value stream, the set of all specific actions required to bring a specific product through the three critical management tasks of any business: the problem-solving task, the information management task, and the physical transformation task. Create a map of the Current State and the Future State of the value stream. Identify and categorize waste in the Current State, and eliminate it!

Step 3: Flow

Make the remaining steps in the value stream flow. Eliminate functional barriers and develop a product-focussed organization that dramatically improves lead-time.

Step 4: Pull

Let the customer pull products as needed, eliminating the need for a sales forecast.

Step 5: Perfection

There is no end to the process of reducing effort, time, space, cost, and mistakes. Return to the first step and begin the next lean transformation, offering a product which is ever more nearly what the customer wants.

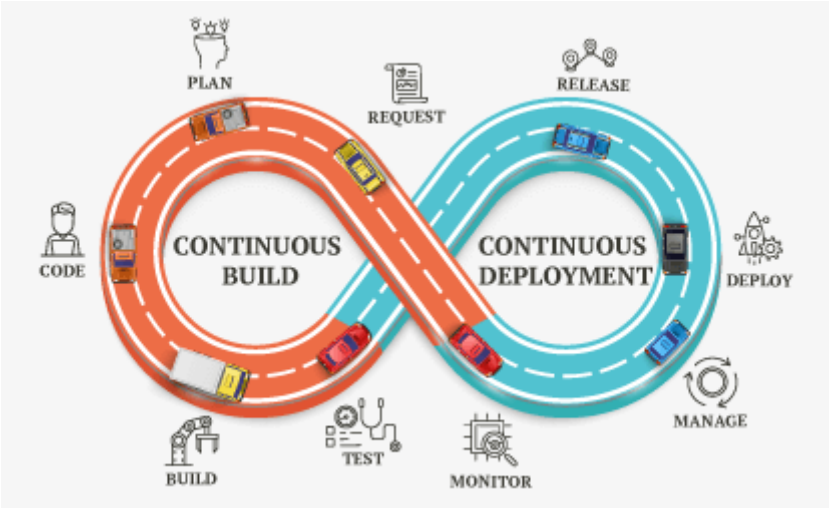
Introduction to DevOps

DevOps is the acronym given to the combination of Development and Operations. It refers to a collaborative approach to make the Application Development team and the IT Operations team of an organization to seamlessly work with better communication. It is a philosophy that encourages adopting iterative software development, automation, and programmable infrastructure deployment and maintenance.

DevOps emphasizes building trust and better lesioning between developers and system administrators. This helps the organization in aligning technological projects to business requirements. Changes rolled

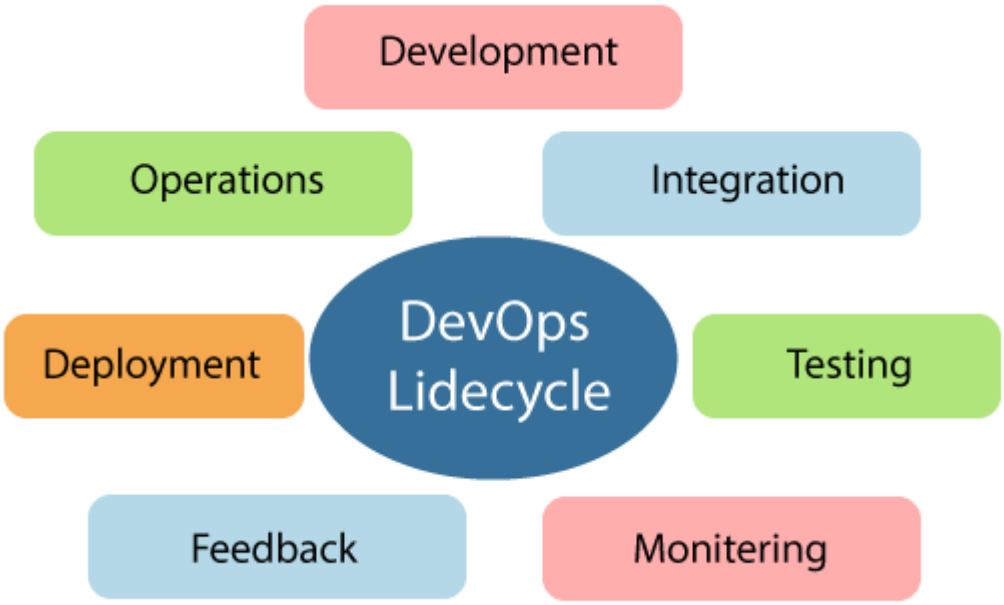
out are usually small and reversible, which the entire team begins to comprehend.

DevOps is visualized as an infinite loop comprising the steps: plan, code, build, test, release, deploy, operate, monitor, then back to plan, and so on.



DevOps phases

DevOps defines an agile relationship between operations and Development. It is a process that is practiced by the development team and operational engineers together from beginning to the final stage of the product.



Learning DevOps is not complete without understanding the DevOps lifecycle phases. The DevOps lifecycle includes seven phases as given below:

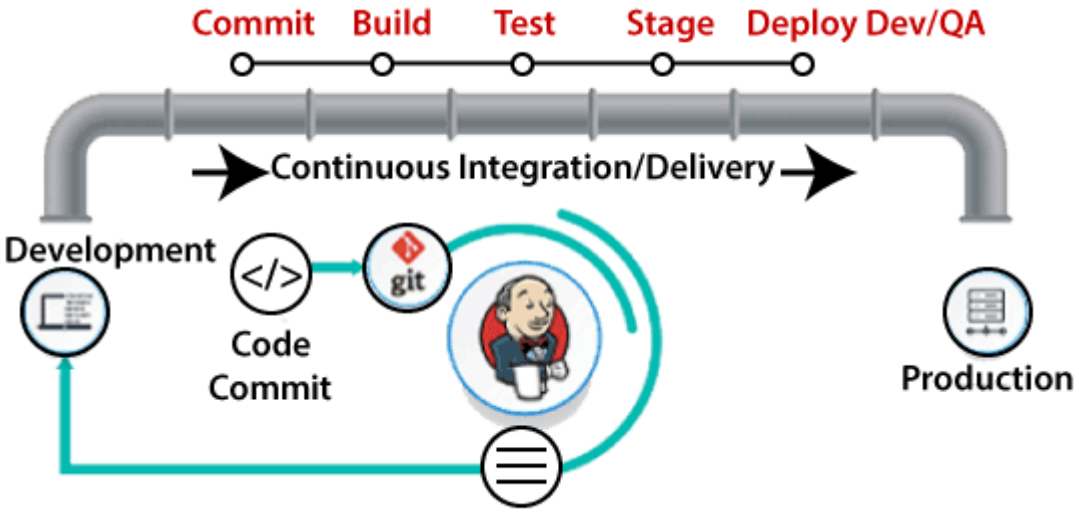
1) Continuous Development

This phase involves the planning and coding of the software. The vision of the project is decided during the planning phase. And the developers begin developing the code for the application. There are no DevOps tools that are required for planning, but there are several tools for maintaining the code.

2) Continuous Integration

This stage is the heart of the entire DevOps lifecycle. It is a software development practice in which the developers require to commit changes to the source code more frequently. This may be on a daily or weekly basis. Then every commit is built, and this allows early detection of problems if they are present. Building code is not only involved compilation, but it also includes unit testing, integration testing, code review, and packaging.

The code supporting new functionality is continuously integrated with the existing code. Therefore, there is continuous development of software. The updated code needs to be integrated continuously and smoothly with the systems to reflect changes to the end-users.



Jenkins is a popular tool used in this phase. Whenever there is a change in the Git repository, then Jenkins fetches the updated code and prepares a

build of that code, which is an executable file in the form of war or jar. Then this build is forwarded to the test server or the production server.

3) Continuous Testing

This phase, where the developed software is continuously testing for bugs. For constant testing, automation testing tools such as TestNG, JUnit, Selenium, etc are used. These tools allow QAs to test multiple code-bases thoroughly in parallel to ensure that there is no flaw in the functionality. In this phase, Docker Containers can be used for simulating the test environment.

Selenium does the automation testing, and TestNG generates the reports. This entire testing phase can automate with the help of a Continuous Integration tool called Jenkins.

Automation testing saves a lot of time and effort for executing the tests instead of doing this manually. Apart from that, report generation is a big plus. The task of evaluating the test cases that failed in a test suite gets simpler. Also, we can schedule the execution of the test cases at predefined times. After testing, the code is continuously integrated with the existing code.

4) Continuous Monitoring

Monitoring is a phase that involves all the operational factors of the entire DevOps process, where important information about the use of the software is recorded and carefully processed to find out trends and identify problem areas. Usually, the monitoring is integrated within the operational capabilities of the software application.

It may occur in the form of documentation files or maybe produce large-scale data about the application parameters when it is in a continuous use

position. The system errors such as server not reachable, low memory, etc are resolved in this phase. It maintains the security and availability of the service.

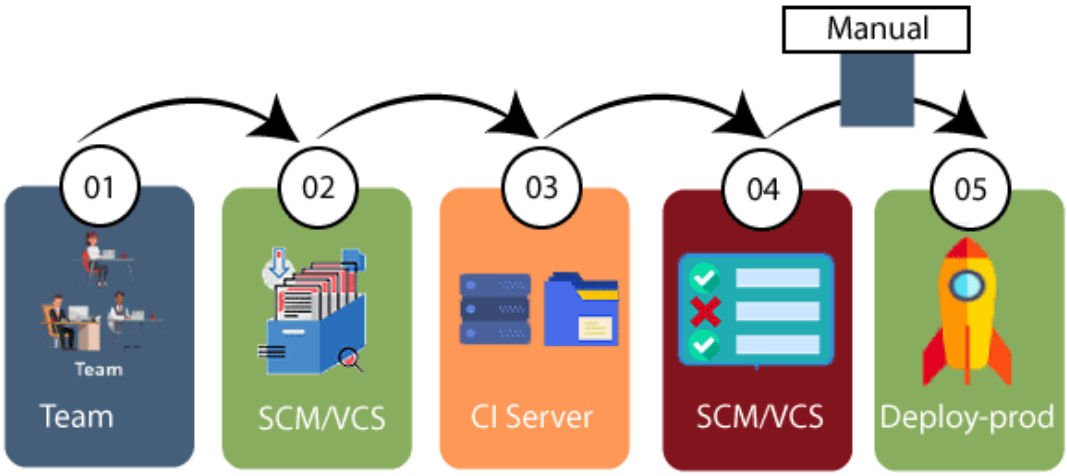
5) Continuous Feedback

The application development is consistently improved by analyzing the results from the operations of the software. This is carried out by placing the critical phase of constant feedback between the operations and the development of the next version of the current software application.

The continuity is the essential factor in the DevOps as it removes the unnecessary steps which are required to take a software application from development, using it to find out its issues and then producing a better version. It kills the efficiency that may be possible with the app and reduce the number of interested customers.

6) Continuous Deployment

In this phase, the code is deployed to the production servers. Also, it is essential to ensure that the code is correctly used on all the servers.



The new code is deployed continuously, and configuration management tools play an essential role in executing tasks frequently and quickly. Here are some popular tools which are used in this phase, such as Chef, Puppet, Ansible, and SaltStack.

Containerization tools are also playing an essential role in the deployment phase. Vagrant and Docker are popular tools that are used for this purpose. These tools help to produce consistency across development,

staging, testing, and production environment. They also help in scaling up and scaling down instances softly.

Containerization tools help to maintain consistency across the environments where the application is tested, developed, and deployed. There is no chance of errors or failure in the production environment as they package and replicate the same dependencies and packages used in the testing, development, and staging environment. It makes the application easy to run on different computers.

7) Continuous Operations

All DevOps operations are based on the continuity with complete automation of the release process and allow the organization to accelerate the overall time to market continuingly.

It is clear from the discussion that continuity is the critical factor in the DevOps in removing steps that often distract the development, take it longer to detect issues and produce a better version of the product after several months. With DevOps, we can make any software product more efficient and increase the overall count of interested customers in your product.

CAMS model

The CAMS Model was created by Damon Edwards and John Willis. CAMS stands for Culture, Automation, Measurement, and Sharing. The CAMS model is a set of values used by many DevOps engineers.

Culture

Even though we use some pretty advanced technology and tools in DevOps, at the core of what we're trying to solve are problems related to people and business. Culture is defined by the interaction of people and groups and is driven by behavior. Substantial communication improvement can result when there is a mutual understanding of others and their goals and responsibilities.
Traditional information technology business models split developers and operations into two distinct groups. They essentially spoke different languages as developers were tasked with innovating, creating and

"moving fast and breaking things" whereas operations staff were in charge of maintaining stable environments and infrastructure. These competing goals often caused friction as each group accused the other of preventing them from doing their jobs. Changing the business culture by sharing responsibility and getting teams on the same page is a major goal of DevOps.

Automation

Automation can save time, effort, and money, just like culture, it truly focuses on people and processes and not just tools. The impact of implementing infrastructure as code as well as using continuous integration and continuous delivery pipelines can be magnified after understanding an organization's culture and goals. It helps to think of automation as an accelerator that enhance the benefits of DevOps as a whole.

Measurement

Using measurements will help in determining if progress is being made in the intended direction. There are two major bumps that may occur when using metrics. First, make sure the metrics you are the correct ones. Secondly, incentivize the right metrics. DevOps practices encourage you to "see the forest from the trees" by taking a look at the entire operation and assessing it as a whole and not just focusing on small parts. Major metrics include (but are certainly not limited to) income, costs, revenue, mean time to recovery, mean time between failures, and employee satisfaction.

Sharing

DevOps processes, similar to agile and scrum, place a very high premium on transparency and openness. Spreading knowledge helps to tighten feedback loops and enables the organization to continuously improve. This collective intelligence makes the team a more efficient unit and allows it to become greater than just the sum of its parts.

Kaizen

Kaizen (Continuous Improvement) is a strategy where employees at all levels of a company work together proactively to achieve regular, incremental improvements to the manufacturing process. In a sense, it combines the collective talents within a company to create a powerful engine for improvement.

The Dual Nature of The Kaizen System

Kaizen is part action plan and part philosophy.

- As an action plan, Kaizen is about organizing events focused on improving specific areas within the company. These events involve teams of employees at all levels, with an especially strong emphasis on involving plant floor employees.
- As a philosophy, Kaizen is about building a culture where all employees are actively engaged in suggesting and implementing improvements to the company. In truly lean companies, it becomes a natural way of thinking for both managers and plant floor employees.

Kaizen works hand-in-hand with Standardized Work. Standardized Work captures the current best practices for a process, and Kaizen aims to find improvements for those processes. Note the emphasis on current; Standardized Work is living documentation (it continually evolves through Kaizen).

Kaizen is part action plan and part philosophy. Consistent application of Kaizen as an action plan develops Kaizen as a philosophy.

KAIZEN EVENTS

A typical Kaizen event has a process that goes something like this:

1. Set goals and provide any necessary background.
2. Review the current state and develop a plan for improvements.
3. Implement improvements.
4. Review and fix what doesn't work.
5. Report results and determine any follow-up items.

This type of Kaizen process cycle is frequently referred to as PDCA (Plan, Do, Check, and Act). PDCA brings a scientific approach to making improvements:

- Plan: develop a hypothesis
- Do: run experiment
- Check: evaluate results
- Act: refine your experiment; then start a new cycle

THE KAIZEN PHILOSOPHY

Interestingly, Kaizen as an action plan is exactly what develops Kaizen as a philosophy. When Kaizen is applied as an action plan through a consistent and sustained program of successful Kaizen events, it teaches employees to think differently about their work. In other words, consistent application of Kaizen as an action plan creates tremendous long-term value by developing the culture that is needed for truly effective continuous improvement.

CI/CD Pipelines

A CI/CD pipeline is a series of steps that must be performed in order to deliver a new version of software. Continuous integration/continuous

delivery (CI/CD) pipelines are a practice focused on improving software delivery using either a DevOps or site reliability engineering (SRE) approach.

A CI/CD pipeline introduces monitoring and automation to improve the process of application development, particularly at the integration and testing phases, as well as during delivery and deployment. Although it is possible to manually execute each of the steps of a CI/CD pipeline, the true value of CI/CD pipelines is realized through automation.

Elements of a CI/CD pipeline

The steps that form a CI/CD pipeline are distinct subsets of tasks grouped into what is known as a *pipeline stage*. Typical pipeline stages include:

- **Build** - The stage where the application is compiled.
- **Test** - The stage where code is tested. Automation here can save both time and effort.
- **Release** - The stage where the application is delivered to the repository.
- **Deploy** - In this stage code is deployed to production.
- **Validation and compliance** - The steps to validate a build are determined by the needs of your organization. Image security scanning tools, like Clair, can ensure the quality of images by comparing them to known vulnerabilities (CVEs).

This is by no means a comprehensive list of pipeline stages. This list is just an example of common stages you may find. Your pipeline will be unique to the requirements of your organization.

IAM

Identity and access management (IAM) is a framework of business processes, policies and technologies that facilitates the management of electronic or digital identities. With an IAM framework in place, information technology (IT) managers can control user access to critical information within their organizations. Systems used for IAM include single sign-on systems, two-factor authentication, multifactor authentication and privileged access management. These technologies also provide the ability to securely store identity and profile data as well as data governance functions to ensure that only data that is necessary and relevant is shared.

IAM systems can be deployed on premises, provided by a third-party vendor through a cloud-based subscription model or deployed in a hybrid model.

On a fundamental level, IAM encompasses the following components:

- how individuals are identified in a system (understand the difference between identity management and authentication);
- how roles are identified in a system and how they are assigned to individuals;
- adding, removing and updating individuals and their roles in a system;
- assigning levels of access to individuals or groups of individuals; and
- protecting the sensitive data within the system and securing the system itself.

Why is IAM important?

Businesses leaders and IT departments are under increased regulatory and organizational pressure to protect access to corporate resources. As a result, they can no longer rely on manual and error-prone processes to assign and track user privileges. IAM automates these tasks and enables granular access control and auditing of all corporate assets on premises and in the cloud.

IAM, which has an ever-increasing list of features -- including biometrics, behavior analytics and AI -- is well suited to the rigors of the new security landscape. For example, IAM's tight control of resource access in highly distributed and dynamic environments aligns with the industry's transition from firewalls to zero-trust models and with the security requirements of IoT. For more information on the future of IoT security, check out this video.

While IT professionals might think IAM is for larger organizations with bigger budgets, in reality, the technology is accessible for companies of all sizes.

Read more about the importance of IAM.

Basic components of IAM

An IAM framework enables IT to control user access to critical information within their organizations. IAM products offer role-based access control, which lets system administrators regulate access to systems or networks based on the roles of individual users within the enterprise.

In this context, access is the ability of an individual user to perform a specific task, such as view, create or modify a file. Roles are defined according to job, authority and responsibility within the enterprise.

IAM systems should do the following: capture and record user login information, manage the enterprise database of user identities, and orchestrate the assignment and removal of access privileges.

That means systems used for IAM should provide a centralized directory service with oversight and visibility into all aspects of the company user base.

Digital identities are not just for humans; IAM can manage the digital identities of devices and applications to help establish trust.

In the cloud, IAM can be handled by authentication as a service or identity as a service (IDaaS). In both cases, a third-party service provider takes on the burden of authenticating and registering users, as well as managing their information. Read more about these cloud-based IAM options.

Benefits of IAM

IAM technologies can be used to initiate, capture, record and manage user identities and their related access permissions in an automated manner. An organization gains the following IAM benefits:

- Access privileges are granted according to policy, and all individuals and services are properly authenticated, authorized and audited.
- Companies that properly manage identities have greater control of user access, which reduces the risk of internal and external data breaches.
- Automating IAM systems allows businesses to operate more efficiently by decreasing the effort, time and money that would be required to manually manage access to their networks.
- In terms of security, the use of an IAM framework can make it easier to enforce policies around user authentication, validation and privileges, and address issues regarding privilege creep.
- IAM systems help companies better comply with government regulations by allowing them to show corporate information is not being misused. Companies can also demonstrate that any data needed for auditing can be made available on demand.

Companies can gain competitive advantages by implementing IAM tools and following related best practices. For example, IAM technologies allow the business to give users outside the organization -- like customers, partners, contractors and suppliers -- access to its network across mobile applications, on-premises applications and SaaS without compromising security. This enables better collaboration, enhanced productivity, increased efficiency and reduced operating costs.

IAM technologies and tools

IAM technologies are designed to simplify the user provisioning and account setup process. These systems should reduce the time it takes to complete these processes with a controlled workflow that decreases errors and the potential for abuse while allowing automated account fulfilment.

An IAM system should also allow administrators to instantly view and change evolving access roles and rights.

These systems should balance the speed and automation of their processes with the control that administrators need to monitor and modify access rights. Consequently, to manage access requests, the central directory needs an access rights system that automatically matches employee job titles, business unit identifiers and locations to their relevant privilege levels.

Multiple review levels can be included as workflows to enable the proper checking of individual requests. This simplifies setting up appropriate review processes for higher-level access as well as easing reviews of existing rights to prevent privilege creep, which is the gradual accumulation of access rights beyond what users need to do their jobs.

IAM systems should be used to provide flexibility to establish groups with specific privileges for specific roles so that access rights based on employee job functions can be uniformly assigned. The system should also provide request and approval processes for modifying privileges because employees with the same title and job location may need customized, or slightly different, access.

Types of digital authentication

With IAM, enterprises can implement a range of digital authentication methods to prove digital identity and authorize access to corporate resources.

Unique passwords. The most common type of digital authentication is the unique password. To make passwords more secure, some organizations require longer or complex passwords that require a combination of letters, symbols and numbers. Unless users can automatically gather their collection of passwords behind a single sign-on entry point, they typically find remembering unique passwords onerous.

Pre-shared key (PSK). PSK is another type of digital authentication where the password is shared among users authorized to access the same resources -- think of a branch office Wi-Fi password. This type of authentication is less secure than individual passwords.

A concern with shared passwords like PSK is that frequently changing them can be cumbersome.

Behavioural authentication. When dealing with highly sensitive information and systems, organizations can use behavioural authentication to get far more granular and analyse keystroke dynamics or mouse-use characteristics. By applying artificial intelligence, a trend in IAM systems, organizations can quickly recognize if user or machine behaviour falls outside of the norm and can automatically lock down systems.

Biometrics. Modern IAM systems use biometrics for more precise authentication. For instance, they collect a range of biometric characteristics, including fingerprints, irises, faces, palms, gaits, voices and, in some cases, DNA. Biometrics and behaviour-based analytics have been found to be more effective than passwords.

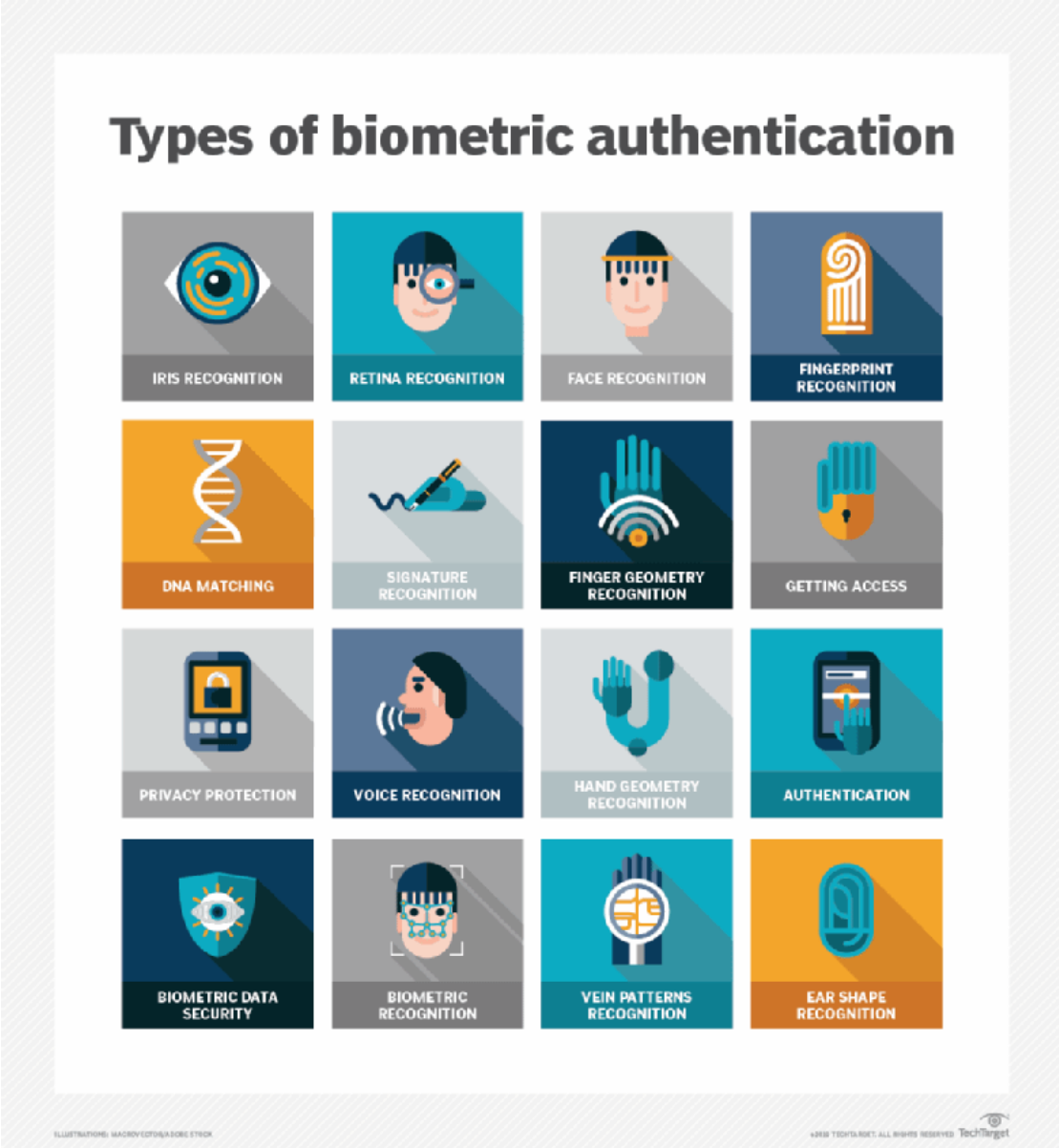
When collecting and using biometric characteristics, companies must consider the ethics in the following areas:

- data security (accessing, using and storing biometric data);
- transparency (implementing easy-to-understand disclosures);
- optionality (providing customers a choice to opt in or out); and
- biometric data privacy (understanding what constitutes private data and having rules around sharing with partners.

One danger in relying heavily on biometrics is if a company's biometric data is hacked, then recovery is difficult, as users can't swap out facial recognition or fingerprints like they can passwords or other non-biometric information.

Another critical technical challenge of biometrics is that it can be expensive to implement at scale, with software, hardware and training costs to consider.

Before getting attached to password less IAM, make sure you understand the pros and cons of biometric authentication.



Implementing IAM in the enterprise

Before any IAM system is rolled out into the enterprise, businesses need to identify who within the organization will play a lead role in developing, enacting and enforcing identity and access policies. IAM impacts every department and every type of user (employee, contractor, partner, supplier, customer, etc.), so it's essential the IAM team comprises a mix of corporate functions.

IT professionals implementing an IAM system largely on-premises and largely for employees should become familiar with the OSA IAM design pattern for identity management, SP-010. The pattern lays out the architecture of how various roles interact with IAM components as well as the systems that rely on IAM. Policy enforcement and policy decisions

are separated from one another, as they are dealt with by different elements within the IAM framework.

Organizations that want to integrate non-employee users and make use of IAM in the cloud in their architecture should follow these steps for building an effective IAM architecture, as explained by expert Ed Moyle:

1. Make a list of usage, including applications, services, components and other elements users will interact with. This list will help validate that usage assumptions are correct and will be instrumental in selecting the features needed from an IAM product or service.
2. Understand how the organization's environments, such as cloud-based applications and on-premises applications, link together. These systems might need a specific type of federation (Security Assertion Markup Language OpenID Connect, for instance).
3. Know the specific areas of IAM most important to the business. Answering the following questions will help:
    a. Is multifactor authentication needed?
    b. Do customers and employees need to be supported in the same system?
    c. Are automated provisioning and deprovisioning required?
    d. What standards need to be supported?

Implementations should be carried out with IAM best practices in mind, including documenting expectations and responsibilities for IAM success. Businesses also should make sure to centralize security and critical systems around identity. Perhaps most important, organizations should create a process they can use to evaluate the efficacy of current IAM controls.

Linux Virtualization: Linux Containers (LXC)

LXC is a user space interface for the Linux kernel containment features. Through a powerful API and simple tools, it lets Linux users easily create and manage system or application containers.

Features

Current LXC uses the following kernel features to contain processes:

- Kernel namespaces (ipc, uts, mount, pid, network and user)
- Apparmor and SELinux profiles
- Seccomp policies
- Chroots (using pivot_root)
- Kernel capabilities
- CGroups (control groups)

LXC containers are often considered as something in the middle between a chroot and a full-fledged virtual machine. The goal of LXC is to create an environment as close as possible to a standard Linux installation but without the need for a separate kernel.

Docker

Docker is an open source containerization platform. It enables developers to package applications into containers—standardized executable components combining application source code with the operating system (OS) libraries and dependencies required to run that code in any environment. Containers simplify delivery of distributed applications, and have become increasingly popular as organizations shift to cloud-native development and hybrid multi cloud environments.

Developers can create containers without Docker, but the platform makes it easier, simpler, and safer to build, deploy and manage containers. Docker is essentially a toolkit that enables developers to build, deploy, run, update, and stop containers using simple commands and work-saving automation through a single API.

Why use Docker?

Docker is so popular today that "Docker" and "containers" are used interchangeably. But the first container-related technologies were available for years — even decades (link resides outside IBM) — before Docker was released to the public in 2013.

Most notably, in 2008, **L**inu**x** Containers (LXC) was implemented in the Linux kernel, fully enabling virtualization for a single instance of Linux. While LXC is still used today, newer technologies using the Linux kernel are available. Ubuntu, a modern, open-source Linux operating system, also provides this capability.

Docker enhanced the native Linux containerization capabilities with technologies that enable:

- **Improved—and seamless—portability:** While LXC containers often reference machine-specific configurations, Docker containers run without modification across any desktop, data center and cloud environment.
- **Even lighter weight and more granular updates:** With LXC, multiple processes can be combined within a single container. With Docker containers, only one process can run in each container. This makes it possible to build an application that can continue running while one of its parts is taken down for an update or repair.
- **Automated container creation:** Docker can automatically build a container based on application source code.
- **Container versioning:** Docker can track versions of a container image, roll back to previous versions, and trace who built a version and how. It can even upload only the deltas between an existing version and a new one.
- **Container reuse:** Existing containers can be used as *base images*—essentially like templates for building new containers.
- **Shared container libraries:** Developers can access an open-source registry containing thousands of user-contributed containers.

Today Docker containerization also works with Microsoft Windows server. And most cloud providers offer specific

services to help developers build, ship and run applications containerized with Docker.

Docker tools and terms

Some of the tools and terminology you'll encounter when using Docker include:

Docker File

Every Docker container starts with a simple text file containing instructions for how to build the Docker container image. *Docker File* automates the process of Docker image creation. It's essentially a list of command-line interface (CLI) instructions that Docker Engine will run in order to assemble the image.

Docker images

Docker images contain executable application source code as well as all the tools, libraries, and dependencies that the application code needs to run as a container. When you run the Docker image, it becomes one instance (or multiple instances) of the container.

It's possible to build a Docker image from scratch, but most developers pull them down from common repositories. Multiple Docker images can be created from a single base image, and they'll share the commonalities of their stack.

Docker images are made up of layers, and each layer corresponds to a version of the image. Whenever a developer makes changes to the image, a new top layer is created, and this top layer replaces the previous top layer as the current version of the image. Previous layers are saved for rollbacks or to be re-used in other projects.

Each time a container is created from a Docker image, yet another new layer called the container layer is created. Changes made to the container—such as the addition or deletion of files—are saved to the container layer only and exist only while the container is running. This iterative image-creation process enables increased overall efficiency since multiple live container instances can run from just a single base image, and when they do so, they leverage a common stack.

Docker containers

Docker containers are the live, running instances of Docker images. While Docker images are read-only files, containers are live, ephemeral,

executable content. Users can interact with them, and administrators can adjust their settings and conditions using docker commands.

Docker Hub

Docker Hub (link resides outside IBM) is the public repository of Docker images that calls itself the "world's largest library and community for container images." It holds over 100,000 container images sourced from commercial software vendors, open-source projects, and individual developers. It includes images that have been produced by Docker, Inc., certified images belonging to the Docker Trusted Registry, and many thousands of other images.

All Docker Hub users can share their images at will. They can also download predefined base images from the Docker filesystem to use as a starting point for any containerization project.

Docker daemon

Docker daemon is a service running on your operating system, such as Microsoft Windows or Apple MacOS or iOS. This service creates and manages your Docker images for you using the commands from the client, acting as the control centre of your Docker implementation.

Docker registry

A Docker registry is a scalable open-source storage and distribution system for docker images. The registry enables you to track image versions in repositories, using tagging for identification. This is accomplished using git, a version control tool.

Docker deployment and orchestration

If you're running only a few containers, it's fairly simple to manage your application within Docker Engine, the industry de facto runtime. But if your deployment comprises thousands of containers and hundreds of services, it's nearly impossible to manage that workflow without the help of these purpose-built tools.

Docker Compose

If you're building an application out of processes in multiple containers that all reside on the same host, you can use *Docker Compose* to manage the application's architecture. Docker Compose creates a YAML file that specifies which services are included in the application and can deploy and run containers with a single command. Using Docker Compose, you

can also define persistent volumes for storage, specify base nodes, and document and configure service dependencies.

Kubernetes

To monitor and manage container lifecycles in more complex environments, you'll need to turn to a container orchestration tool. While Docker includes its own orchestration tool (called Docker Swarm), most developers choose Kubernetes instead.

Kubernetes is an open-source container orchestration platform descended from a project developed for internal use at Google. Kubernetes schedules and automates tasks integral to the management of container-based architectures, including container deployment, updates, service discovery, storage provisioning, load balancing, health monitoring, and more. In addition, the open source ecosystem of tools for Kubernetes—including Istio and Knative—enables organizations to deploy a high-productivity Platform-as-a-Service (PaaS) for containerized applications and a faster on-ramp to serverless computing.