DevOps – The Context (Total # of Exam Questions: 9)

WALL OF CONFUSION - THE PRIMARY CHALLENGE WITH TRADITIONAL IT ORGANIZATIONS

IT organizations face a variety of challenges due to the wall of confusion between the different teams. Tearing down the wall of confusion is, therefore, difficult due to the various challenges.

Factors Leading to Wall of Confusion	Problems Due to Wall of Confusion
■ Organizational Silos ■ Different Mindsets ■ Different or Incohesive processes ■ Different Tools ■ Different Environments ■ Absence of Procedural Transition ■ Lack of Communication ■ Lack of Short Feedback Loops ■ Ineffective Knowledge Articles ■ Lack of End-to-End Responsibility ■ Biased Feedback	■ Blame Game ■ Build Rollback to Maintain Data Accuracy ■ Low creativity ■ Manual Releases ■ Low Quality ■ Product Backlog ■ Sporadic Releases ■ Communication Problems ■ Delayed Solutions to Customers ■ High Cost ■ Low Level of Employee Satisfaction (Less Happiness)

DIGITAL ORGANIZATIONS AND BUSINESS BENEFITS OF DEVOPS

Common Success Factors of Digital Enterprises (or DevOps Focus)	Characteristics of Digital Enterprises (or Most Compelling Business Benefits of DevOps)	Impacts of DevOps Practices in IT Organizations
■ **IT** – The Strategic Differentiator ■ **Speed** – Automation, Continuous Improvement, and Simplified Operating Models ■ **Oneness** – No Confusion, Teams in Sync ■ **Antifragility** – Getting Better as a Result of Stress and Tension	■ Faster Delivery ■ Improved Continuity / Stability ■ Focus on Innovation ■ Reduced Cost	■ Faster and Smarter Deployment ■ Agility is the Key ■ Reduced Risk ■ Increased Value Delivery ■ Increased Stability ■ Faster Time-to-Market ■ A Penny Saved is a Penny Earned

DASA DEVOPS PRINCIPLES

Customer-centric Action	Create with the End in Mind	End-to-End Responsibility	Cross-Functional Autonomous Teams	Continuous Improvement	Automate Everything You Can
■ Feedback Loops ■ No Bureaucracy ■ Courage ■ Risk Takers ■ Innovation	■ Product and Service Thinking ■ Collaboration ■ Engineering Mindset ■ Trust ■ Early Feedback	■ Vertically-Oriented Teams ■ Decision Making (Control) ■ Learning ■ Performance ■ Simplicity	■ Cross-Functional ■ Autonomous ■ Self-Directed ■ T-Shaped Profiles ■ Complementary Skills	■ Structured Problem-Solving ■ Experimentation ■ Measurement ■ Minimum Waste ■ Optimization	■ Increased Productivity ■ Improved Quality ■ Consistency ■ Increased Availability ■ Scalability

The principles are connected with each other and somehow shares the knowledge and skills areas (or characteristics) of each other. The principles do not follow necessarily follow any sequencing.

DevOps – The Context (Total # of Exam Questions: 9)

GOALS AND MEASUREMENTS

Aspects of Measurement

Value

- Value denition and measurement
- Value driven prioritization
- Optimized ow of value

Goals and Objectives

- Shared goals
- Cascading indicators
- Transparency
- Ownership

Indicators and Metrics

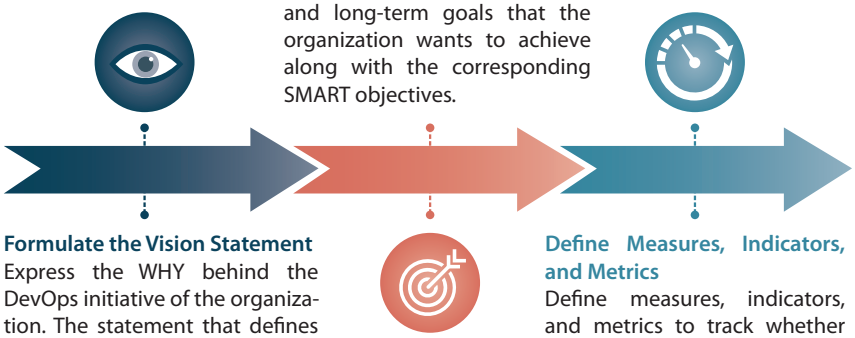
- Business-driven, end-to-end metrics
- Rigorous automated measurement

Compliancy

- Continuous compliance
- Built-in controls and audit trails
- Risk-aware and autonomous teams

The Business Case for DevOps is Clear But You Need Goals

Organizations often fail to drive the full adoption of DevOps primarily due to people's resistance or lack of knowledge. The following figure lists the steps that can help organizations to be successful in their DevOps initiative.



Identify Goals and Objectives
Recognize the short-, medium-, and long-term goals that the organization wants to achieve along with the corresponding SMART objectives.

Formulate the Vision Statement
Express the WHY behind the DevOps initiative of the organization. The statement that defines the need for applying DevOps principles.

Define Measures, Indicators, and Metrics
Define measures, indicators, and metrics to track whether you are achieving the set goals.

CHOOSING THE RIGHT METRICS

Survivorship Bias

> "Beware of advice from the successful."
> — *Barnaby James*

Survivorship Bias is the phenomenon of considering success stories along with the accompanying failures.

Performance Metrics vs. Performance Predictors

Performance Metrics

- Performance metrics are typically output-oriented, easy to measure, but hard to improve or influence.
- An indicator of past performance that measures how we performed.
- **Examples**: Weight, Customer satisfaction

Performance Predictors

- Performance predictors are typically input-oriented, hard to measure, and easy to influence.
- An indicator that helps predict future performance.
- **Examples**: Calories intake per day, User guide usage

Top Practices (Leading Indicators) Correlated to Performance Metrics

Leading Indicators for Deployment Frequency:

- Continuous Delivery
- Version Control

Leading Indicators for Lead Time for Changes:

- Version Control
- Automated Testing

Leading Indicators for MTTR:

- Version Control
- Monitoring System and Application Health

DevOps for Individuals (Total # of Exam Questions: 1)

ROLE OF DEVOPS ENGINEER

The engineer act as a mediator between various departments to communicate and apply the DevOps methodology while driving cultural differences to facilitate cross-team collaboration. Their primarily focus on creating awareness and encouraging responsibility.

1. **Awareness**: Make people aware of the consequences of their actions.
2. **Responsibility**: Make people responsible for the consequences of their actions.

DIFFERENT TYPES OF PROFILES

I-Shape (Specialist)

- **Vertical Bar**: Functional Disciplinary Skill – Jack of no trade, and master of (at least) one.

- **Profile**: Poor
- **Examples**:
  - Microsoft CRM Expert
  - Mobile Analytics

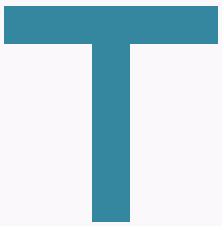
Dash-Shape (Generalist)

- **Horizontal Bar**: Cross-Discipline Skills - Jack of many trades, and master of none.

- **Profile**: Poor
- **Examples**:
  - Utility Player in Baseball
  - Project Coordinator

T-Shape (Generalizing Specialist)

- **Horizontal and Vertical Bars**: Classic Agile Team Member - Jack of many trades, and master of (at least) one.



- **Profile**: Good
- **Examples**:
  - Social Manager
  - Content Marketer

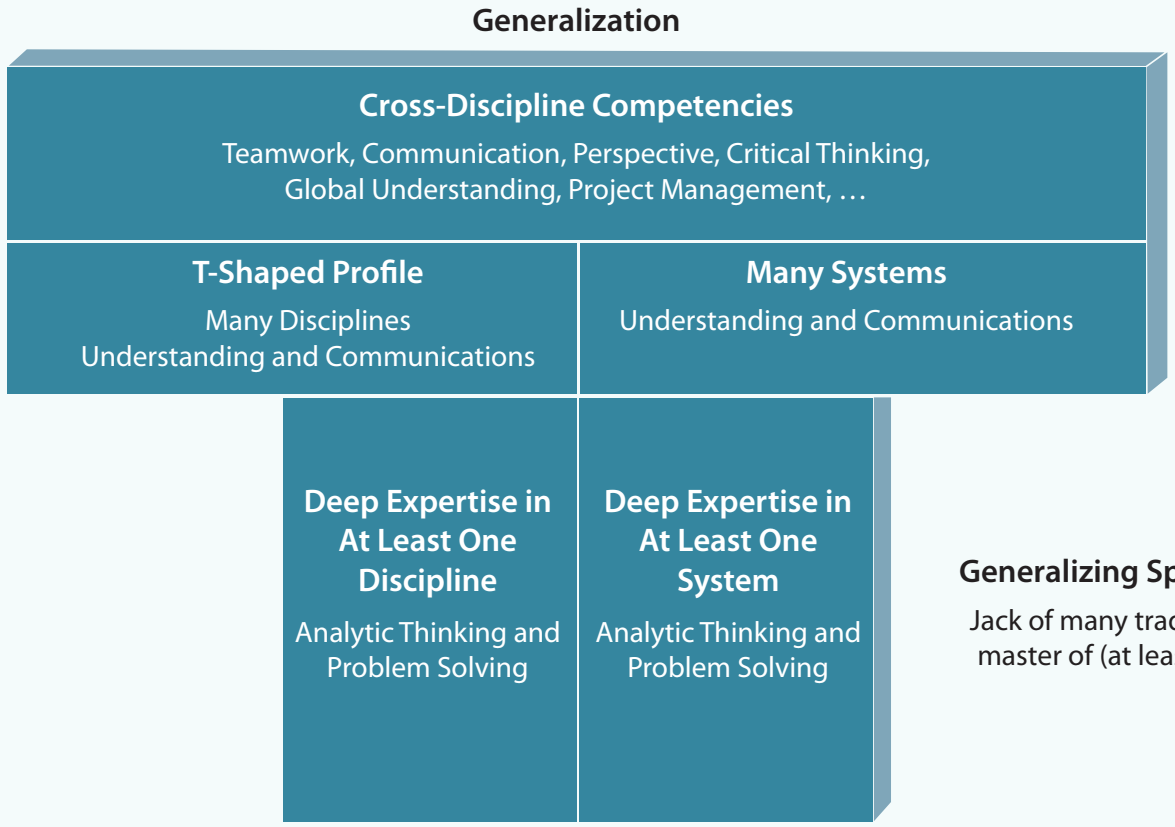
BENEFITS OF HAVING T-SHAPED PROFILES

- Effective Cross-Discipline Collaboration
- Diverse Experience with Satisfaction of Depth
- More Focus on Innovation
- Eye-Catchy Profiles for Employers

DevOps for Individuals (Total # of Exam Questions: 1)

DEVOPS ENCOURAGES T-SHAPE PROFILES

Characteristics of T-Shaped People

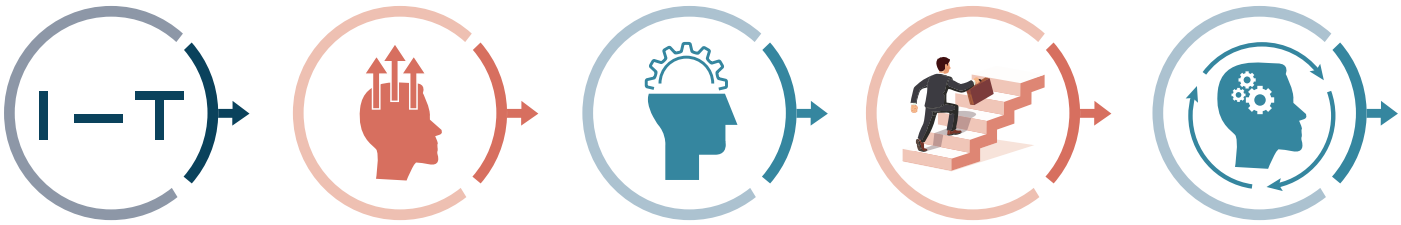


Generalizing Specialist

Jack of many trades, and master of (at least) one.

BECOMING A T-SHAPED PERSON

Steps to Becoming a T-Shaped Person



1
Identify your type of profile.

2
Identify the required skills and learning.

3
Assess your current skills.

4
Take the necessary actions.

5
Continue learning and assessing yourself.

DevOps for Teams and Organizations (Total # of Exam Questions: 11)

DEVOPS CULTURE AND BEHAVIOR

Core of DevOps Culture

DevOps culture emphasizes small, multidisciplinary teams, who work autonomously and take collective accountability for how actual users experience their software. In other words, it focuses on service mindset to deliver a high-quality product.

Ron Westrum Typologies of Organizational Culture

Pathological (power-oriented)
- Low cooperation
- Messengers shot
- Responsibilities shirked
- Bridging discouraged
- Failure leads to scapegoating
- Novelty crushed

Bureaucratic (rule-oriented)
- Modest cooperation
- Messengers neglected
- Narrow responsibilities
- Bridging tolerated
- Failure leads to justice
- Novelty leads to problems

Generative (performance-oriented)
- High cooperation
- Messengers trained
- Risks are shared
- Bridging encouraged
- Failure leads to enquiry
- Novelty implemented

Typical Cultural Aspects of a DevOps Team

Team cultural aspect	Description	Tips to Facilitate
Continuous Learning & Continuous Improvement	The desire to explore and learn should be there in teams. Working together, transparency, and sharing knowledge is necessary.	Organize by teams, facilitate knowledge sessions, conduct hackathons, make people responsible. Use concepts of MVP. Set a clear DoD and coach teams in Agile way of working.
Experimentation & Risk Taking	Always conduct experiments using solid methodologies to capture highly accurate data.	Provide time, use instant sandbox environment, remove hurdles, applaud ideas, fail safely. Award failure!
Build Quality in	Always show full transparency in teams for the work the team is doing as it ensures building quality products.	Advocate end-to-end responsibility; Encourage the team to utilize their skills as they know the best; avoid cutting corners; practice automation (test, deploy, and provision), continuous improvement, and transparency (monitors).
An Engineering Culture	Utilize knowledge, skills, and creativity to solve problems, implement product features, and optimize delivery process.	Hire people with matching ambitions, move away from function-title model, support experimentation, support automating manual tasks, do not focus only on utilization.
A Culture of Effectiveness	Measure effectiveness, culture, and efficiency to optimize DevOps transformation.	Begin with end in mind, in retrospectives: ask people why? people are responsible to choose activities in teams, avoid or eliminate handover moments, move away from rigid processes, achieve results in small batches, generate awareness by Lean.
A Culture of Product Thinking	Ensure that applications must deliver value when they run in production.	Encourage customers to attend demos, implement user feedback into a storyboard, allow customers to write about product and respond, organize people around the product, encourage the team to write blogs about product (you build it; you run it).
A Culture of Taking Responsibility	All members of the DevOps team are responsible for the complete product lifecycle.	Do not explain 'how' to do, ask 'what' is required, address derailment openly, let people figure out how to do things, reward responsibility, reward failure, bring transparency in what everyone is doing.
Inspirational and Fun Environment	An environment in which people perform at best, where they feel inspired, where they want to be, feel welcomed and are encouraged to think out of the box.	Provide good coffee, create nice and open environments, invest in additional facilities like games, conduct contests or arrange drinks at the end of the week, allow teams to modify/tailor the office to their needs.

DevOps for Teams and Organizations (Total # of Exam Questions: 11)

DEVOPS IMPACT ON THE ORGANIZATIONAL MODEL

Traditional vs DevOps Environment

Traditional Environment	DevOps Environment (Achieving high-performance IT through effective collaboration)
People work in technically-oriented silos or teams. For example, there is often one department for running the applications (Operations) and one department for developing the applications (Development).	DevOps focuses on one team doing the work, hence, reducing communication and coordination issues.
The multiple teams share different priorities, goals, time frames, and cultures, making successful communication and cooperation difficult.	The team share the same goal of providing valuable customer experience.
Many traditional organizations are not aligned with the way the IT services are structured and organized, with multiple or distributed teams working on producing and delivering the same IT service.	DevOps focuses on operational model that facilitates collaboration within and between the teams to provide the valuable customer experience.

Organizational Model and Conway's Law

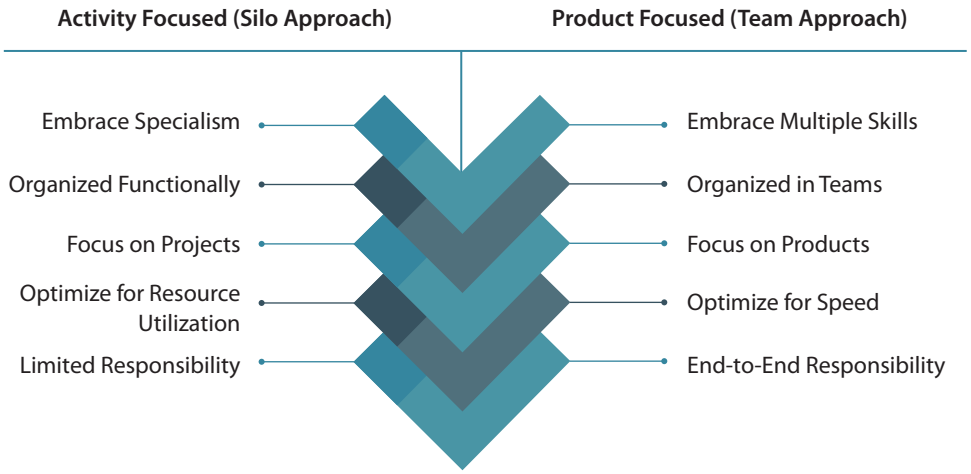
Any piece of software reflects the organizational structure.

> Organizations which design systems are constrained to produce designs which are copies of the communication structures of these organizations.
>
> *Source: Conway, Melvin E. (April 1968)*

The DevOps Approach to Organizational Model

In a DevOps environment, Business Systems teams are structured around a self-service platform. The self-service platform is a product in itself and is maintained by the Platform team. This platform (or the Platform team) offers a comprehensive set of standardized self-service capabilities to Business System teams, such as monitoring and security. Such a setup enable Business System teams to operate autonomously by placing their application and infrastructure code on the self-service platform and completely manage their products. On the other hand, the Platform team manages and monitors the system qualities of their products independently from the associated application products. Such an organizational model enables teams to take end-to-end responsibility of their products or services. Therefore, both the teams, Business System teams and Platform teams, are responsible for the qualities of their products, such as availability and performance.

Product-Focused Approach of DevOps - The Benefits



Business System vs Platform Teams

Business System Teams	Platform Teams
Manage end users, services, and products.	Manage platform products used by Business System teams.
Take end-to-end responsibility of the products/services with the required support from the Platform teams.	Support Business System teams to work more effectively by providing platform (or Infrastructure) services through self service and automation.

DevOps for Teams and Organizations (Total # of Exam Questions: 11)

CORE ELEMENTS OF A DEVOPS CULTURE

Teambuilding and Collaboration

Teamwork : People joining a DevOps team often come from different technical teams but have similar skills. However, they do not always have the same purpose or goals. They might be working together with other people (outside their group) to achieve the enterprise's goals. Therefore, when transitioning from a technical team to a DevOps team, it is essential to give proper attention to teambuilding.

Collaboration: The most important behavior of teamwork is collaboration.

Benefits	Pitfalls
Combines different perspectives	Irrational decision-making due to group thinking
Encourages creativity	Ambiguity in roles and responsibility
Takes advantage of synergies	High cost of collaboration
Brings balance to decision making	Longer decision times
Improves delivery times	Conflict within the group

Visual Management: Visual Management, such as Kanban, is one of the strongest tools to communicate. In its most basic form, Visual Management includes three boards.

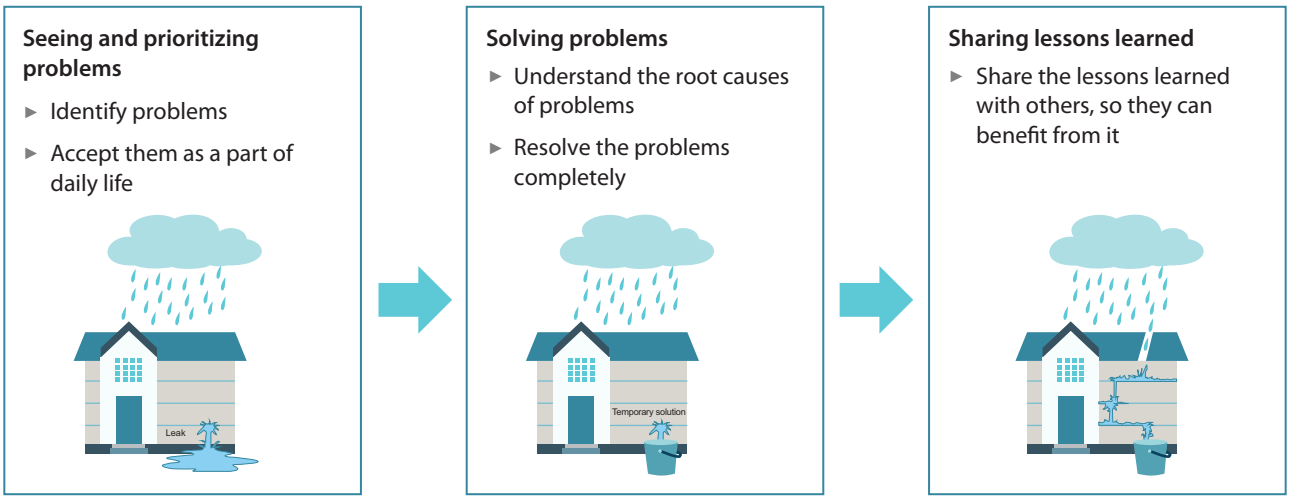
- **Day Board (Daily Progress)**: Managing the daily progress of work by using a simple (To Do-Doing-Done) or a complicated approach to track the progress.
- **Week Board (Feedback Repository)**: Ensuring the work is planned and agreed by discussing KPIs and comments from customers and employees.
- **Improvement Board (Progress of Problem-Solving)**: Gathering all the problems or improvements initiatives to be solved or carried out by communicating and sharing solutions and learnings.

Continuous Improvement and Problem-Solving

Engineering Culture: Continuous improvement is integral to an engineering culture focused on solving problems. A well-known structured way to facilitate continuous improvement is DMAIC.

1. **Define**: Define the problem.
2. **Measure**: Collect data and facts about the problem.
3. **Analyze**: Analyze the cause of the problem.
4. **Improve**: Define alternative solutions and implement improvements.
5. **Control**: Maintain the solution.

Kaizen Mindset: The Kaizen (means improvement) mindset means integrating the following three behaviors into the DevOps team.



Quality at the Source: It means doing things first time right using automated IT processes and exhibiting the required behaviors. Some of the characteristics of having quality at the source are:

- Problems are identified early in the process, leading to more security.
- Solving problems is easier and cheaper.
- Customers like a good product and quick solutions.

DevOps for Teams and Organizations (Total # of Exam Questions: 11)

Courage and Experimentation

Courage to Act: Businesses need innovation that requires removing away apathy, fear, and complacency form the environment. Therefore, a DevOps team needs to first focus on ensuring a fail-to-safe environment and then embed courage by performing the following steps:

STEP 1	STEP 2	STEP 3	STEP 4
Ensure customer buy-in	Define and deliver a Minimum Viable Product (MVP)	Focus on the goal "customer value is delivered first time, right in flow"	Take small steps and do not carry out large experiments

Experimentation Comes with Complications: Experimenting with low risk requires hypotheses that you want to test.

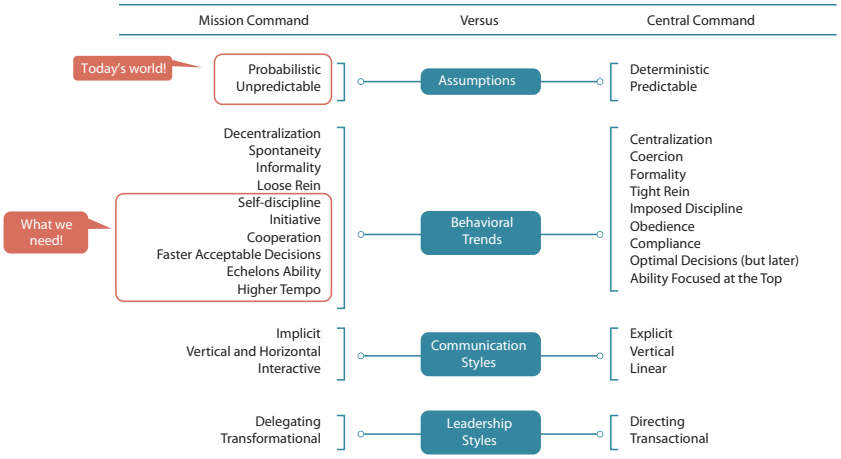
Hypotheses are tested in a running business when:	Hypotheses are tested in a changing IT environment when:
■ Business requirements tend to evolve over time. ■ Gaining consensus on requirements might be difficult.	■ Parts of the IT service are changed or the service itself is changed. The baseline changes thereby create new possibilities or impossibilities. ■ New tools and capabilities of IT make actions that were previously difficult.

A DevOps team gathers its courage largely from the fact that experimentation should occur in a live business environment with varying requirements. However, these days the IT environment is regularly changing; therefore, it is difficult to run longer term experiments. Considering the changing requirements, the aim of a DevOps team should be to take calculated risks, fail fast in a small way, and learn which solutions will work the best.

Experimentation Meetups: These are meetings where new actions or initiatives are tested. Experimentation meetings, such as hackathon, is a great way to simulate creative actions.

Leadership and Feedback

Facilitating Leadership through Empowerment:



Mission Command: Teams know the mission and goals and are empowered to make necessary decisions.

Central Command: Teams follow the detailed instructions (micromanagement) of their leaders to achieve the mission.

Leadership Requires Removing Barriers to Effective Collaboration:

1. **Lack of Trust**: Lack of trust between the team members is like a team of individuals working together.
2. **Fear of Conflict**: Fear of conflict means the team is afraid of discussing the differences of opinion.
3. **Avoidance of Accountability**: Avoidance of accountability means that the teams do not take accountability for each other's work.
4. **Lack of Commitment**: Lack of commitment means the team members do not support each other in committing decisions.
5. **Inattention to Results**: Inattention to results means the team is lacking focus on the goals of the team.

Barriers of Effective Collaboration By Lencioni

Leadership Style:

Go See	Ask Why	Show Respect
"Senior management must spend time on the plant floor."	"Use the "Why?" technique daily."	"Respect your people."

Feedback: Feedback is a key leadership tool to facilitate teams efficiently. However, it requires leaders to practice some habits while receiving and giving feedback.

DevOps for Teams and Organizations (Total # of Exam Questions: 11)

DEVOPS FOCUS ON TEAM STRUCTURING AND AUTONOMY

Team Autonomy

Autonomy is the independence that results in self-directed actions.

End-to-End Responsibility

The power and freedom brings with it more responsibility. Being self-directed means more accountability and responsibility of the products or services.

Customer-Centric Action

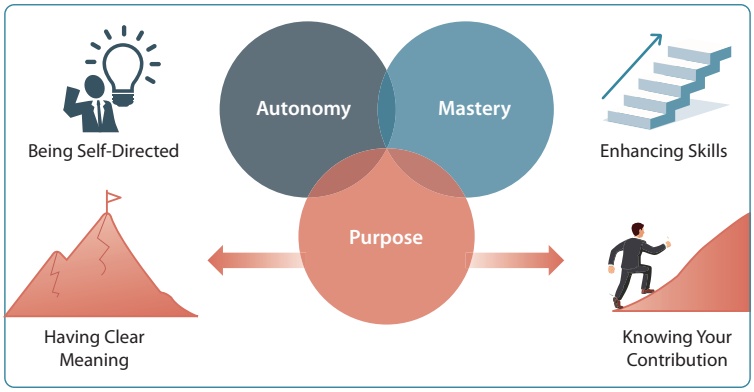
Autonomy allows you to consider the customer's needs and deliver the maximum value.

Cross-Functional Autonomous Teams

These teams are problem solvers, and they cannot work without having the required freedom.

Autonomy Makes Teams Intrinsically Motivated

DevOps teams are intrinsically motivated and care about what they do. According to Daniel Pink, three primary aspects of intrinsic motivation are Autonomy, Mastery, and Purpose.



Aspects of Intrinsic Motivation

Achieving Autonomy: In a DevOps environment, autonomy is primarily achieved by structuring the teams around distinct services and products and making them independent.

How does team structure enable autonomy?

In a DevOps environment, the Business System teams are structured around automated self-services, which are maintained by the Platform team. These services enable Business System teams to manage their products or services almost autonomously. In the same way, the Platform teams monitor and manage their products independently from the availability of the application products, which use the platform products. Both, the Platform teams and the Business System teams are responsible for the qualities of their own products.

How do independent teams help achieve autonomy?

The focus on Continuous Delivery helps independent teams to become autonomous. With this ultimate aim, they work almost independently from other teams to deliver continuous value to the customers. It is possible only when they have are free to make their own decisions and can solve the problems. Being independent, these teams are end-to-end responsible for the quality of the product or service throughout the entire product lifecycle, thus removing the need for any methodical handover.

Determining the Autonomy of Teams

The three-step design criteria helps organizations to determine the autonomy of teams.

CUSTOMER

- Uses the product/service.
- Is responsible for the existence of the business process.

Aspects to Consider:

- Who is your customer?
- What are their requirements?
- What services are they looking for?

TECHNOLOGY STACK

- Defines the team's area of responsibility.
- Determines the technology required to deliver the product/service.

Aspects to Consider:

- What type of application does the team need to build?
- What is the development cost and time to market?
- Is the application secure and scalable?

KNOWLEDGE

- Ensures the team's have the required
- knowledge and skills.
- Identifies the team's strength to deliver value.

Aspects to Consider:

- What skills do the team require to deliver the product/service?
- What are the units-of-work, the team will be processing?
- What is the Definition of Done (DoD) for the units-of-work?

DEVOPS AT SCALE

Dealing with Complexity

The first rule of scaling is "Don't Scale." However, it becomes inevitable or unavoidable for organizations when they need to deal with complex dynamics.

Examples of Complex Dynamics

- Turbulent markets and changing customer demands
- Complex products and dependencies between products, systems, and teams
- Formal and informal social structures and communication lines
- Multi-vendor or cloud landscapes
- Different combinations of skills, attitudes, and behaviors
- Compliance and regulatory influences

The popular ways (or strategies) or practices to deal with complexity include:

- Strategies: Experimentation; Constraints enablement; Continuous adoption of emergent practices
- Practices: Apply principles and practices from several scaling frameworks.

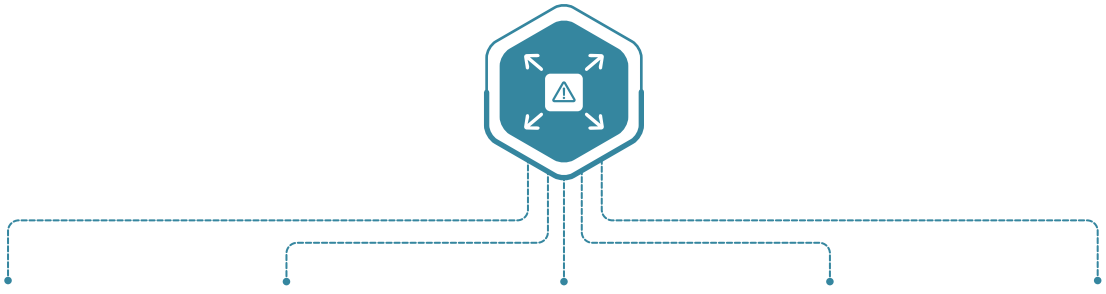
Dealing with Scaling - Aspects to Consider

- Create only one Product Backlog for one product regardless of the number of teams working on it.
- Segregate work between multiple self-directed, cross-functional teams.
- Ensure to keep the user stories independent in the Product Backlog to the maximum extent.
- Do not force different teams working on different products to have a global Definition of Done (DoD).
- Ensure compatibility between the various DoDs so that on combining their outputs, a potentially releasable Increment will be produced.

Principles of Mitigating Scale

You can mitigate scale utilizing any of the several scaling frameworks, models, or practices. These frameworks share some common characteristics or principles.

Principles of Mitigating Scale



- Ensure alignment between teams (joint events like planning or review, cadence, direct communication).
- Create common goals and practices (share practices and tools where desired).
- Organize for optimal value delivery (value streams, value measurement).
- Maximize transparency (visual management).
- Facilitate leadership. (be a servant leader)

MANAGEMENT PRACTICE - ITSM

ITSM

ITIL 4: It guides modern organizations, planning to adopt the latest technologies, operational processes, and concepts, in a service context using the key elements, **Service Value System** (SVS) and **Four Dimensions**.

Service Value System (SVS): The central element of the SVS is the service value chain, an operating model which outlines the key activities required to respond to demand and facilitate value creation through the creation and management of products and services.

Key Components of SVS

Service Value Chain: Refers to a set of activities (Plan, Improve, Engage, Design and Transition, Obtain/Build, and Deliver and Support) performed by an organization to deliver a valuable product or service to its consumers.

Practices: Refer to a set of organizational resources designed to perform work or accomplish an objective. ITIL 4 includes 14 general management practices, 17 service management practices, and 3 technical management practices.

Governance: Refers to the means by which an organization is directed and controlled.

Guiding Principles: Refer to recommendations that guide organizations in all circumstances, regardless of changes in its goals, strategies, type of work, or management structure. The seven guiding principles are Focus on value; Start where you are; Progress iteratively with feedback; Collaborate and promote visibility; Think and work holistically; Keep it simple and practical; and Optimize and automate.

Continual Improvement: Refers to a recurring activity performed at all levels to ensure that an organization's performance continually meets stakeholders' expectations. To support continual improvement at all levels, the ITIL SVS includes the ITIL continual improvement model, improve service value chain activity, and continual improvement practice.

Four Dimensions: To support a holistic approach to service management, ITIL defines the four dimensions that collectively are important for the effective and efficient facilitation of value. The four dimensions are Organizations & People; Information & Technology; Partners and Suppliers; and Value Streams and Processes.

DevOps Practices (Total # of Exam Questions: 18)

DEVOPS AND ITSM

DevOps and ITSM Processes – The Relation

Traditional IT Organizations	Modern Agile or DevOps Organizations
▪ These consist of task-oriented silos, requiring specialized skills. ▪ The teams need formal processes for managing services, leading to waste. ▪ The teams require a Process Manager, a Process Owner, and KPIs for managing and (or) aligning the various silos, leading to a lot of overhead, and hence, waste. ▪ **Challenge**: Results in a complex matrix organization involving many handovers and transition moments, leading to a lot of waste.	▪ These consist of customer-centric, service-oriented teams directly and closely integrated with the customers, requiring business representation skills and End-to-End responsibility. ▪ The teams are cross-functional and autonomous, fully responsible for the lifecycle of products and services, leading to having Development and Operations in the same team. ▪ The focus is on optimizing the flow via value streams. ▪ **Challenge**: Scaling up teams is a challenge, as maintaining a standard strategy and governance is difficult.

ITSM in the DevOps World

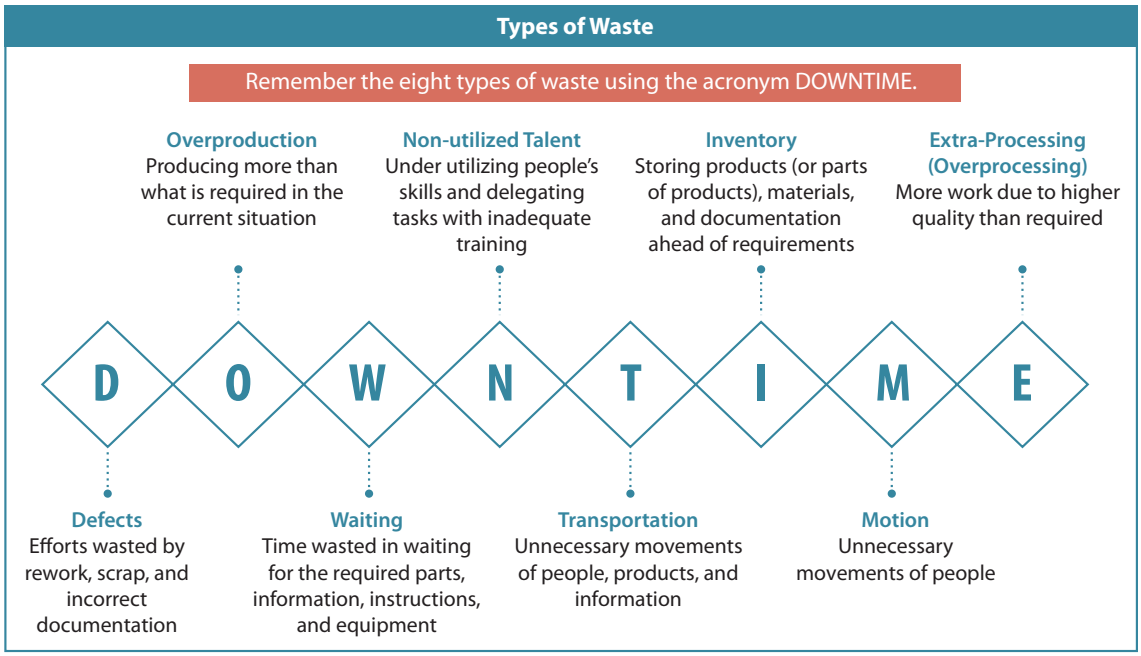
- The ITIL SVS and Service Value Chain support High-Velocity IT in optimizing speed and quality. These also enable teams to deliver the required outputs to the customer and realize the necessary outcomes.
- The ITIL 4 practices are supporting Value Streams that help optimize the flow and reduce the amount of waste.
- ITIL 4 Practices help optimize value delivery based on the organization's / team's needs by supporting the steps in the values streams.
- ITIL also helps in automating the required tasks, collaborating with other teams, and managing an end-to-end service.
- Some examples of novel practices utilizing ITSM are Swarming for Incident Management; Chatbots, Event Automation, Predictive Analytics; Peer Reviews for Change Approval; and eXperience Level Agreements (XLAs).

DevOps Practices (Total # of Exam Questions: 18)

MANAGEMENT PRACTICE - LEAN

Lean: The core idea of Lean is to maximize customer value by eliminating or minimizing waste (Muda), eliminating overburden (Muri), and lack of balance in workloads (Mura).

Types of Lean Wastes - DOWNTIME



Lean Principles

1. **Identify Customer Value**: Define the customer value considering the customer's needs, deadline, and cost.
2. **Map the Value Stream**: Create the process flow to eliminate the waste by identifying all the non-value steps.
3. **Ensure Flow**: Ensure the remaining steps will flow smoothly in production without any interruptions, delays, or bottlenecks.
4. **Establish Pull**: Deliver products to the customers as and when required using the Just-in-Time approach in production.
5. **Seek Perfection**: Start the process again for further enhancements and continue it until a perfect value is created with no waste.

Possible Ways to Improve the Flow of Work:	**Optimizing a Process Using Value Stream Mapping:**
■ Visualize the workflow. ■ Limit work in progress. ■ Create a pull and flow. ■ Make process policies explicit. ■ Improve collaboratively.	1. Define customer objectives and process actors. 2. Define activities. 3. Define work in progress. 4. Identify rework. 5. Assess activities. 6. Determine process cycle efficiency. 7. Determine value add for each activity.

Lean Startup

The Lean Startup is a scientific approach to creating and managing startups and getting the desired product to customers' hands faster. The methodology focuses on following the loop of three steps: Find, Execute, and Validate.

1. **Find the Business Idea**: Choose the business idea by answering, "Should this product be built?" compared to the traditional methodology of answering, "Can this product be built?"

2. **Execute the Idea**: Start building the Minimum Viable Product (MVP). An MVP is a version of a product with limited but enough features that allow gathering as maximum information as possible about the potential customers and their feedback on the product.

3. **Validate the Idea**: Experiment with the idea in the real world by allowing real customers (early adopters and others) to test the product in the marketplace (or the live environment).

DevOps Practices (Total # of Exam Questions: 18)

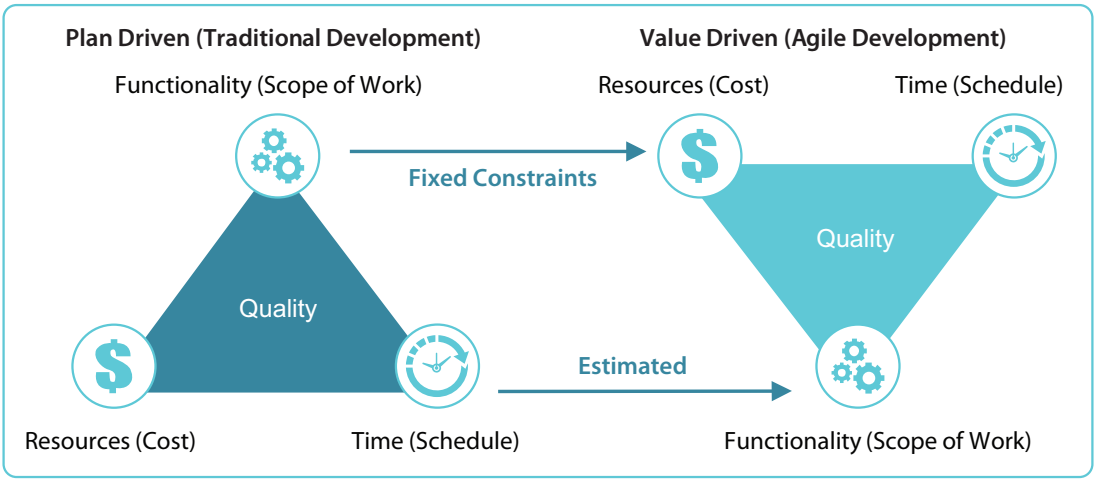
MANAGEMENT PRACTICE - AGILE

Agile: Agile is an iterative and incremental approach of software development. It breaks the development of new functionalities into smaller functional units according to user stories and prioritizes them to continuously deliver the software in short cycles known as iterations

Traditional vs Agile Product Development

Traditional Development	Agile Development
■ It starts with a complete product design. ■ Building the product is followed by testing the final product. ■ The final product is tested in the live environment. ■ It involves no feedback loops. ■ It is a systematic way to develop products. (**Plan-Driven**) ■ It is focused on activities. (**Activity-Focused**)	■ It is focused on delivering functional product every Sprint. ■ It is an iterative way to deliver products starting with the basic functionality and adding features with each iteration (Sprint). ■ It is focused on delivering value to the customer early through Continuous Delivery. (**Value-Driven**) ■ It is focused on the deliverable, which is the product. (**Product-Focused**)

Plan Driven vs Value Driven



Activity Focused vs Product Focused

Aspect	Activity-Focused	Product-Focused
Development Style	■ Traditional Way of Product Development	■ Agile Way of Product Development
Working Style	■ Work with Individuals (in silos)	■ Collaborative Work (as one team)
People	■ Functionally Organized	■ Team Organized
Team	■ Project Focused	■ Product Focused
Responsibility	■ Work-oriented Limited Responsibility	■ End-to-end Responsibility

Agile Teams: Agile Teams are multidisciplinary feature teams have multiple skills and are responsible for a fully working product.

The Agile Manifesto

Individuals & Interactions Over Processes & Tools	Working Software Over Comprehensive Documentation	Customer Collaboration Over Contract Negotiation	Responding to Change Over Following a Plan

TECHNICAL PRACTICE - ARCHITECTURE

Architecture

Aim of IT Architecture: Functional and Non-Functional Requirements: The aim of IT architecture is to provide a solution that meet the needs of the stakeholders and the business and deliver value.

Solution Requirements	
Functional Requirements (Functionality)	**Non-Functional Requirements (Quality)**
■ These are the primary requirements stated by the stakeholders that describe the features and the behavior of a product.	■ These are the secondary requirements that describe the quality characteristics/attributes of a system.
■ Describes how a product should function to meet the given requirements.	■ Some of the attributes include accessibility, compliance, fault tolerance, autonomy, portability, resiliency, maintainability, testability, scalability, and reliability.
■ Defines the tasks a system should do to deliver the required product/service.	■ If these requirements are ignored, the product/ service does not function as expected.

Quality is defined as the sum total of features (functionality) and the inherent characteristics of a product that enable it to meet the given requirements.

Building Qualities Through Smaller Services

To achieve quality through Continuous Delivery, organizations focus on delivering smaller services. These services should be independent that can carry out a business function, without impacting other business functions.

- **Drawback**: Due to a larger number of smaller services, the architecture can even become more complex. As a result, the quality will go down. (The number of services is directly proportional to complexity, which in turn is indirectly proportional to quality.)

Building Qualities Through Microservice Architecture (MSA)

An organization should use MSA to avoid the complexity associated with smaller services. It helps build an application as a suite of small services, each running in its own process and is independently deployable.

Qualities of MSA	Guidelines to Developing Faster, Cheaper, Better Microservices	Characteristics of MSA
■ Responsive ■ Resilient ■ Elastic ■ Loosely coupled (message-driven)	■ Autonomous Systems ■ Simplicity ■ Low Coupling/High Cohesion	■ Componentization via Services ■ Organized Around Business Capabilities ■ Products not Projects ■ Smart Endpoints and Dumb Pipes ■ Decentralized Governance ■ Decentralized Data Management ■ Infrastructure Automation ■ Design for Failure ■ Evolutionary Design

TECHNICAL PRACTICE - MODERN INFRASTRUCTURE

Provisioning - Pets vs. Cattle

The pets vs. cattle metaphor is a useful way to think about IT operations. Manual system provisioning has many similarities with keeping pets, and automated provisioning has similarities with managing cattle.

Pets	Manual Provisioning
Pets are given names, such as Pussy and Tiger.	Servers are given distinct names.
People raise them with proper care and love.	The servers are handled manually, which includes activities such as updating and configuring the software.
They are unique.	Most systems become unique over time as a result of the applied manual software and configuration changes.
When they are unwell, the owners take them to an animal health care for proper nursing.	When the server does not perform, its (unique) problems are analyzed and fixed. Everyone notices the unavailability of these servers.

Cattle	Automated Provisioning
Cattle are given numbers, such as A001 and A002.	Servers are assigned random numbers.
People manage them as livestock.	They are managed as a group.
They are identical.	They are standardized. Software and configuration are defined centrally and applied automatically to many systems.
When they are unwell, the owners replace them with the other one.	When there is a problem with any server (or any server fails), human intervention is not required. The servers can automatically route around failures by restarting or simply replacing itself. Therefore, no one notices the unavailability of these servers.

Automate Environments Through Desired State Configuration

DSC specifies the desired software and hardware configuration of systems using a declarative model. It uses a simple standard approach to specify "how you want to configure a server or workstation?" that is easy to maintain and understand. You do not need to get into the details of "how the configuration should actually happen?"

Types of Architecture with DSC

Pull	Push
Advantages: ■ Set up costs. It is not necessary to invest in a new server because the configurations are pushed from your workstation. ■ The simplicity of the architecture because all configurations are stocked on your workstation. ■ It is ideal for testing the functioning of the "Desired State Configuration."	**Advantages**: ■ Automation of the deployment enables sending the configurations to the compatible machines after a fixed interval, which is by default 15 minutes. ■ The management of numerous machines, connected or not. As soon as the machine is connected to the network, it asks to the "Pull Server" for its configuration.
Disadvantages: ■ The complexity required to manage the machines connected can fail the sending of the configuration as laptops are not always connected to the network.	**Disadvantages**: ■ You need to deploy one more server.

Automated Provisioning

Automated Provisioning with Mutable Infrastructure	Automated Provisioning with Immutable Infrastructure
In a traditional mutable server infrastructure, servers are continually updated and modified in place. In other words, these servers are mutable; they can be changed after they're created.	An immutable infrastructure is another infrastructure paradigm in which servers are never modified after they're deployed.

DevOps Practices (Total # of Exam Questions: 18)

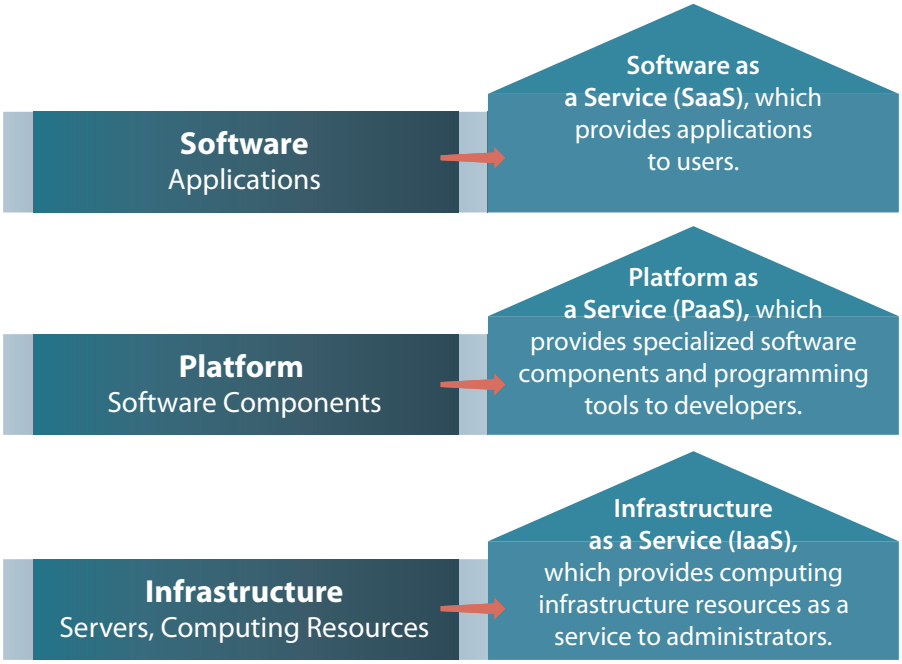
TECHNICAL PRACTICE - CLOUD

Cloud technology is the on-demand delivery of computing services over the Internet (the Cloud).

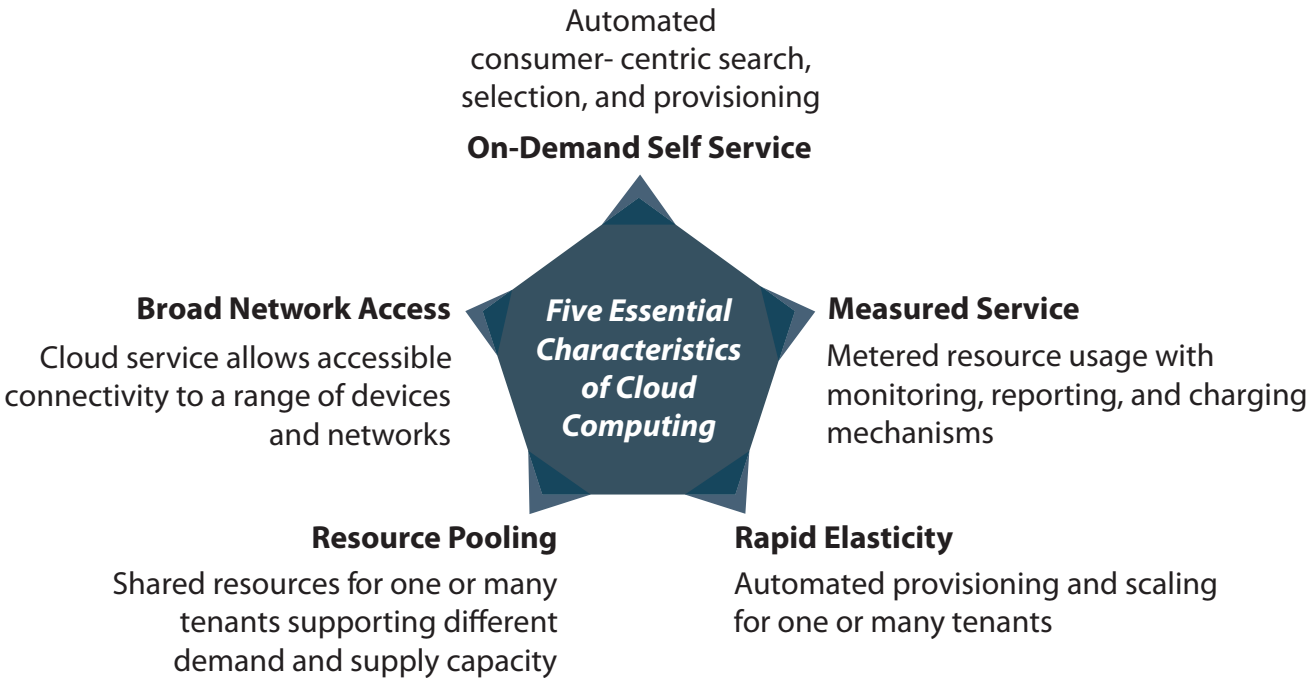
Computing Services

- Servers
- Storage
- Databases
- Networking
- Software
- Analytics
- Intelligence

Cloud Service Models



Cloud Characteristics



DevOps Practices (Total # of Exam Questions: 18)

Different Types of Clouds to Operate

Do DevOps organizations need Platform teams in the presence of cloud service products?

<table>
<tr><th colspan="3">Deciding Factors (Types of Cloud)</th></tr>
<tr><td colspan="2">Private Cloud</td><td>Public Cloud</td></tr>
<tr><td colspan="2">A private cloud gives a single Cloud consumer organization the exclusive access to and usage of the infrastructure and computational resources.</td><td>The cloud infrastructure is made available to the general public or a large industry group and is owned by an organization selling cloud services.</td></tr>
<tr><td>On-Site Private Cloud</td><td>Outsourced Private Cloud</td><td rowspan="2">No need of Platform teams If you are planning to use only public cloud services.</td></tr>
<tr><td>Build and manage your cloud end-to-end in-house.</td><td>Build and manage your cloud by a third-party provider.</td></tr>
</table>

DevOps Organizations Adopting Cloud

Some of the implications of adopting Cloud in DevOps organizations are:

- A clear separation of responsibilities and accountability across teams
- Self-service concepts required for optimized delivery of valuable software to customers (Continuous Delivery)
- Data center optimization using extensive automation
- Standardization and productization of infrastructure components

Services Required by DevOps Business System Teams

Services Required for Successful Execution of Applications and Tools	Services Required for Successful Delivery of Changes
■ Application management self-services, such as backup restore, server restart, and changing log levels ■ Logging self-services (aggregated) ■ Monitoring self-services (aggregated) ■ Billing services	■ Software source configuration/version management self-services ■ Test automation and reporting self-services ■ Continuous Integration and reporting self-services ■ Application deployment self-services ■ Artifact sharing, validation (among other security), and publication self-services ■ Test data syndication (or replication) self-services (ensure volatile, anonymized production data is available for certain types of tests) ■ Release orchestration self-services

DevOps Practices (Total # of Exam Questions: 18)

TECHNICAL PRACTICE – CONTINUOUS DELIVERY AND AUTOMATION

"Continuous Delivery is a software development discipline where you build software in such a way that the software can be released to production at any time."

"Continuous Delivery just means that you are able to do frequent deployments but may choose not to do it, usually due to businesses preferring a slower rate of deployment. In order to do Continuous Deployment, you must be doing Continuous Delivery."

By Martin Fowler, ThoughtWorks

"Continuous deployment is the practice of releasing every good build to users—a more accurate name might have been "continuous release."

By Jez Humble, ThoughtWorks

"One of the challenges of an automated build and test environment is you want your build to be fast, so that you can get fast feedback, but comprehensive tests take a long time to run. A deployment pipeline is a way to deal with this by breaking up your build into stages. Each stage provides increasing confidence, usually at the cost of extra time. Early stages can find most problems yielding faster feedback, while later stages provide slower and more through probing. Deployment pipelines are a central part of Continuous Delivery."

By Martin Fowler, ThoughtWorks

"Continuous Integration is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible. Many teams find that this approach leads to significantly reduced integration problems and allows a team to develop cohesive software more rapidly."

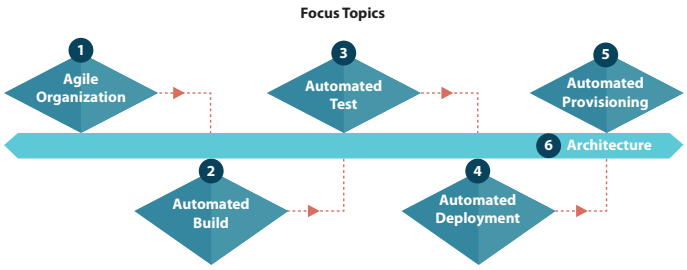
"Continuous Integration usually refers to integrating, building, and testing code within the development environment. Continuous Delivery builds on this, dealing with the final stages required for production deployment."

By Martin Fowler, ThoughtWorks

Principles of Continuous Delivery: According to the Jolt Award Winner book, Continuous Delivery by Jez Humble and Dave Farley, there are eight principles of Continuous Delivery.

1. Create a Repeatable, Reliable Process for Releasing Software
2. Automate Almost Everything
3. Keep Everything in Version Control
4. If It Hurts, Do It More Frequently, and Bring the Pain Forward
5. Build Quality In
6. Done Means Released
7. Everybody Is Responsible for the Delivery Process
8. Continuous Improvement

Fully Automated Continuous Delivery – The Focus Topics: Organizations can successfully adopt a fully automated Continuous Delivery process using the six focus topics, as listed in the following figure.



Automated Continuous Delivery – The Benefits

- **Faster Speed to Market**: More frequent deployments to production, making the new features available to customers as soon as possible.
- **Faster Feedback**: Faster speed to market implies faster feedback from the customers.
- **Lower Level of Risk**: Identifying errors/bugs in a small piece of code is easier to detect or fix.
- **More Productivity**: Automation allows the software development team to delegate tedious and repetitive tasks to tools.

Continuous Monitoring

Continuous monitoring helps generate greater transparency. It helps to continuously ensure the health, performance, and reliability of your application and infrastructure as it moves from development to production.

Without complete transparency, it would be difficult to apply the "Plan, Do, Check, Act" or PDCA rules of continuous improvement.

Scope of Monitoring

- **Monitoring Infrastructure**: It includes the basic level of infrastructure monitoring, such as servers, storage devices, load balancers, and networking.
- **Monitoring Platform**: It includes monitoring the platforms provided and used by the Platform teams (or Business System teams) to run their applications.
- **Monitoring Application**: Having a healthy infrastructure and platform is not good enough for an application to function correctly.
- **Monitoring Business**: Monitoring should provide feedback to the business at the earliest to take corrective actions.
- **Monitoring the Monitoring**: With the growing emphasis on monitoring, the need to monitor the monitoring infrastructure is growing.
- **Log Aggregation Monitoring**: Tooling for log aggregation is capable of using all the data to detect issues that might have gone unnoticed.

Monitoring Optimized for DevOps

- **Monitoring Tool Agents**: Monitoring should not be an afterthought, but a part of the design of products and services.
- **Standardization and Niche Tools**: Experimentation, MVP, and courage are stimulated when niche tools are allowed.
- **Alerts/Incidents Directly to Teams**: DevOps teams are end-to-end responsible and need the direct feedback on the quality of their work.
- **Incorporate Service Management Data**: Such a system and knowledge provides a great source of insight into the behavior of your users and customers.
- **Real-Time Dashboards for All**: Visualize everything, including the real-time status of services.
- **Track Social Media**: Social media is one of the quickest ways to find out what customers are doing with our services/products in the market.
- **Single Point of Truth**: A single source speeds up solving problems and prevents unnecessary discussion.

Modern Operations

- **Machine Learning (ML)**: Chatbots and self-service, recommendation engines, anomaly detection, ticket completion, intelligent routing, predictive analytics, intelligent and contextual search, and smart notifications
- **Site Reliability Engineering (SRE)**: Using software engineering techniques to achieve reliability, implementing Service Level Objectives (SLOs), eliminating toil, and defining error budgets
- **Intelligent Swarming**: Cross-functional, rotating teams swarming to create more value, networks over hierarchy, collaboration over escalation, and suitable not only for major incidents

Logging is the process of creating an on-going record of application events in the form of log files, also known as logs.

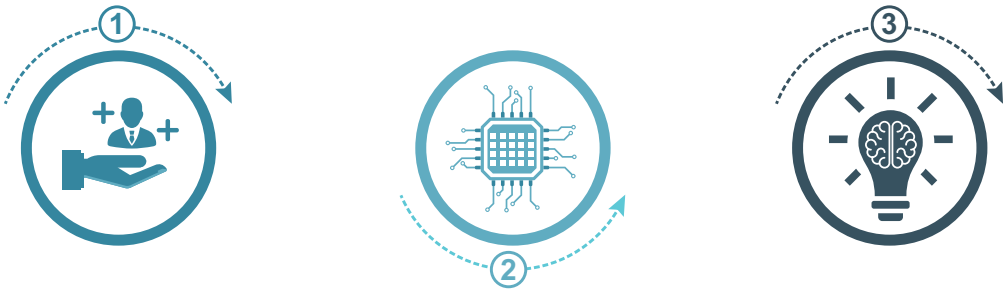
Log files provide a tracking mechanism to review any event within a system, or the usage of IT services by users and customers, including failures and state transformations. Logs can be the source of important data for everything from forensic analysis to audit records.

DevOps – The Next Steps (Total # of Exam Questions: 1)

UNDERGOING A DEVOPS TRANSFORMATION - HOW AND WHERE TO START?

Step 1: Analyze the Current Situation

1. Map the context using the basic design criteria for an autonomous team.



Customer

Define the customer and the product or service to be delivered to the customer.

Technology Stack

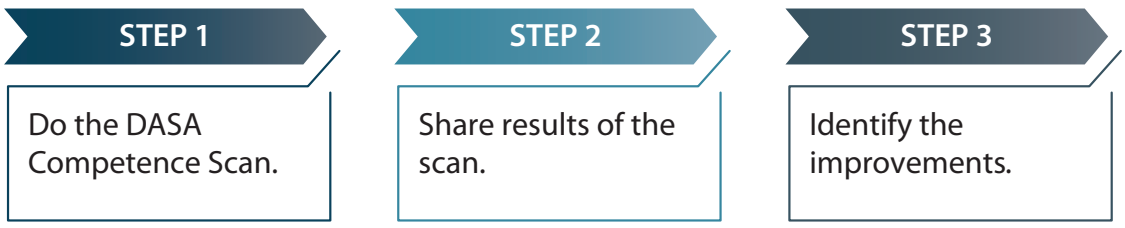
Map the technology stack related to the product or service.

Knowledge

Recognize the knowledge or skills required to deliver the product or service.

2. Assemble the cross-functional, autonomous business system team based on current involvement with the product/service/customer.

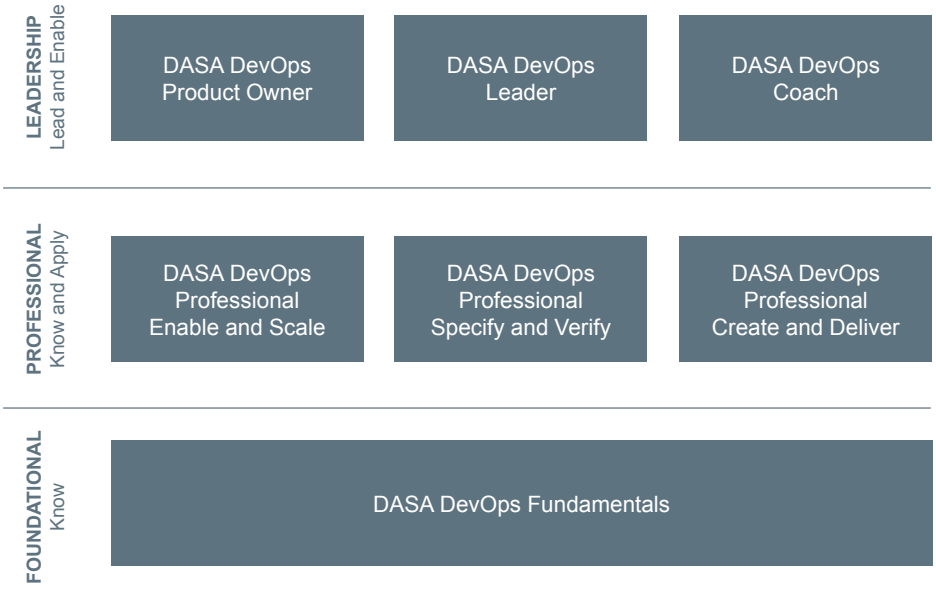
3. Do the DASA Competence Scan and identify the improvements areas for the team.



DevOps – The Next Steps (Total # of Exam Questions: 1)

Step 2: Improve Incrementally

1. Train the team through a set of training to enable them to work and lead effectively in a DevOps team.



2. Focus on the cultural elements to develop a healthy, transparent, and safe-to fail environment.

Core Beliefs of a DevOps Team

Teambuilding and Collaboration

Continuous Improvement and Problem-Solving

Courage and Experimentation

Leadership and Feedback

3. Improve the flow of work to remove the non-value add activities by.
   a. Automating manual and repeated work
   b. Identifying and solving problems
   c. Removing waste
   d. Increasing release frequency
   e. Shortening processes