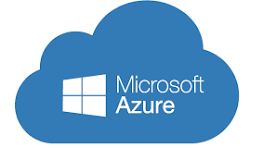
**AZURE & DEVOPS**

Microsoft Azure and Azure DevOps are closely integrated to enhance software development and project management. Here’s an overview of their relationship and key features:

**Microsoft Azure**

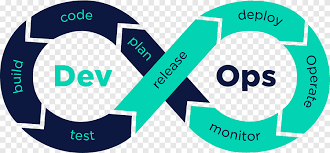
Microsoft Azure is a comprehensive cloud computing platform that provides a wide range of services, including computing power, storage, networking, and analytics. It enables organizations to build, deploy, and manage applications through Microsoft-managed data centers.



**Azure DevOps**

Azure DevOps is a set of development tools offered by Microsoft that supports the entire software development lifecycle. It includes features for planning, developing, delivering, and monitoring applications. Key components of Azure DevOps include:

* Azure Boards: For agile project management with Kanban boards and backlogs.
* Azure Repos: For source control using Git repositories.
* Azure Pipelines: For continuous integration and continuous delivery (CI/CD) to automate building, testing, and deploying applications.
* Azure Test Plans: For managing test cases and executing manual tests.
* Azure Artifacts: For hosting and sharing packages.



Integration with Microsoft Azure

Azure DevOps integrates seamlessly with Microsoft Azure to provide a unified development experience. Some key integration features include:

* CI/CD Pipelines: Azure Pipelines can deploy applications directly to Azure services, ensuring streamlined deployment processes.
* Resource Management: Developers can manage Azure resources directly from Azure DevOps, allowing for efficient resource allocation and monitoring.
* Viva Goals Integration: Azure DevOps integrates with Microsoft Viva Goals to automatically track progress on key results and initiatives based on work items in Azure DevOps. This integration helps align team efforts with organizational objectives[1](https://learn.microsoft.com/en-us/viva/goals/azure-devops-integration).
* Custom Integrations: Developers can build custom applications or services that integrate with Azure DevOps using REST APIs or .NET Client Libraries. This allows for enhanced collaboration with third-party tools like Slack or Jenkins[2](https://learn.microsoft.com/en-us/azure/devops/integrate/?view=azure-devops).

**Cloud service models** refer to the various frameworks through which cloud computing services are delivered to users and organizations. Understanding these models is essential for selecting the right solution based on specific needs and capabilities. The primary cloud service models include:

**1. Infrastructure as a Service (IaaS)**

IaaS provides virtualized computing resources over the internet, allowing businesses to avoid the complexities and costs associated with managing physical hardware and data centers. Users can access essential infrastructure components like virtual machines, storage, and networking.

**Key Features:**

* **Scalability**: Resources can be scaled up or down based on demand.
* **Control**: Users have control over operating systems and applications, allowing for customization.

****

**Benefits:**

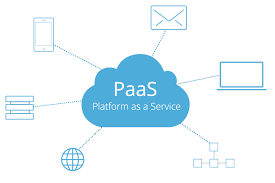
* Reduces upfront hardware investments.
* Facilitates quick deployment of applications and services.

**2. Platform as a Service (PaaS)**

PaaS offers a platform that enables developers to build, deploy, and manage applications without worrying about the underlying infrastructure. This model includes development tools, middleware, and database management systems.

**Key Features:**

* **Managed Environment**: The cloud provider manages the infrastructure, allowing developers to focus on coding.
* **Collaboration Tools**: Supports team collaboration during application development.

****

**Benefits:**

* Accelerates application development with integrated tools.
* Simplifies the deployment process by abstracting infrastructure management.

**3. Software as a Service (SaaS)**

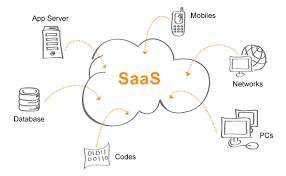
SaaS delivers software applications over the internet on a subscription basis. Users can access these applications via web browsers without needing to install or maintain them locally.

**Key Features:**

* **Accessibility**: Software can be accessed from any device with an internet connection.
* **Maintenance-Free**: The service provider handles updates, security, and maintenance.

**Benefits:**

* Reduces costs associated with software purchases and installations.
* Offers flexibility and scalability for businesses of all sizes.



**Cloud deployment services:** refer to the various methods and models used to deploy applications and services in a cloud environment. These models determine how resources are allocated, managed, and accessed. Here’s an overview of the main cloud deployment models:

**1. Public Cloud**

In a public cloud deployment model, services and infrastructure are hosted off-site by third-party providers and made available to multiple users over the internet. This model is characterized by shared resources, which can lead to cost savings.

**Benefits:**

* **Cost-Effective**: Users pay only for the resources they consume without needing to invest in physical hardware.
* **Scalability**: Resources can be scaled up or down based on demand.
* **Accessibility**: Services can be accessed from anywhere with an internet connection.

**Examples**: Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP).

**2. Private Cloud**

A private cloud deployment model provides dedicated resources for a single organization. This setup can be hosted on-premises or by a third-party provider but is not shared with other organizations.

**Benefits:**

* **Enhanced Security**: Offers greater control over data and security measures tailored to specific needs.
* **Customization**: Organizations can customize their infrastructure to meet unique requirements.
* **Compliance**: Easier to comply with regulatory requirements due to dedicated resources.

**Examples**: VMware vCloud Suite and Microsoft Azure Stack.

**3. Hybrid Cloud**

Hybrid cloud combines both public and private cloud environments, allowing data and applications to be shared between them. This model offers flexibility, enabling organizations to take advantage of both environments.

**Benefits:**

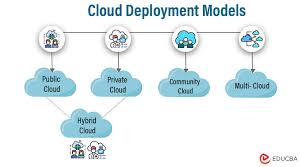
* **Flexibility**: Organizations can choose where to run their applications based on cost, performance, and compliance needs.
* **Cost Efficiency**: Sensitive data can be kept in a private cloud while utilizing the public cloud for less critical operations.
* **Scalability**: Organizations can scale their resources dynamically based on workload demands.

4. Community Cloud

A community cloud is shared among several organizations with similar interests or requirements, such as security, compliance, or performance needs. This model allows for collaborative resource sharing while still maintaining some level of privacy.

**Benefits:**

* **Shared Costs**: Costs are distributed among the organizations using the community cloud.
* **Collaboration**: Facilitates collaboration between organizations with similar goals or regulatory requirements.



Microsoft Azure operates a vast network of regions and availability zones to provide reliable and scalable cloud services. Here's an overview of Azure regions and availability zones:

**Azure Regions**

An Azure region is a set of data centers deployed within a specific geographic area, connected through a low-latency network. Azure has over 60 regions globally, allowing organizations to deploy applications close to their users for improved performance and compliance with local regulations.

**Key Features:**

* **Geographic Distribution**: Azure regions are strategically located around the world to provide redundancy and low-latency access to services.
* **Region Pairs**: Many Azure regions are paired for disaster recovery purposes, ensuring that if one region goes down, the other can take over.

**Availability Zones**

Availability zones are physically separate locations within an Azure region. Each zone is designed to be independent from others in the same region, providing high availability and resilience against failures.

**Key Features:**

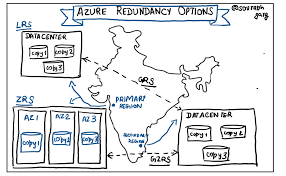
* **Isolation**: Each availability zone has its own power, cooling, and networking to protect applications from localized failures.
* **High Availability**: By deploying applications across multiple availability zones, organizations can ensure that their services remain available even if one zone experiences issues.

**Examples of Regions with Availability Zones:**

* **East US**: 3 availability zones
* **West US**: 3 availability zones
* **Canada Central**: 3 availability zones
* **Australia East**: 3 availability zones

Types of Availability Zone Support

1. **Zone-Redundant Services**
   * **Description**: These services automatically replicate or distribute resources across multiple availability zones. If one zone experiences an outage, the services remain available through the other zones.
   * **Benefits**: This approach ensures high availability and data durability, as the system can seamlessly failover to another zone without user intervention.
   * **Examples**: Services like Azure Storage (with zone-redundant storage), Azure SQL Database, and Azure Kubernetes Service can be configured for zone redundancy.
2. **Zonal Services**
   * **Description**: Zonal services are deployed to a specific, self-selected availability zone. While this configuration does not inherently provide redundancy, it allows users to meet specific latency or performance requirements.
   * **Benefits**: Users can optimize their applications for performance by deploying resources in a particular zone, but they must implement their own redundancy strategies across zones if needed.
   * **Examples**: Virtual machines and managed disks can be deployed in a specific zone.



1. **Non-Zonal or Regional Services**
   * **Description**: These services do not explicitly utilize availability zones unless configured to do so. Resources may be placed in any availability zone within the region and can be moved based on Azure's management.
   * **Risks**: If an outage occurs in the zone where these resources are located, they may experience downtime.
   * **Examples**: Some Azure services operate on a regional basis without guaranteed placement in specific zones unless configured for zonal or zone-redundant deployment.

Azure Virtual Network (VNet) is a fundamental service in Microsoft Azure that enables users to create a logically isolated network in the cloud. This allows various Azure resources, such as virtual machines (VMs), to communicate securely with each other, the internet, and on-premises networks. Here’s an overview of Azure VNets and their key functionalities:

Key Features of Azure Virtual Network

**1. Isolation and Segmentation**

* **Private IP Address Space**: Users can define a private IP address range for their VNet using CIDR notation, ensuring that resources can communicate privately.
* **Subnets**: VNets can be divided into subnets, allowing for better organization and management of resources. Each subnet can have its own address range within the VNet.

**2. Secure Communication**

* **Inter-Resource Communication**: Resources within the same VNet can communicate directly and securely without requiring public IP addresses.
* **Hybrid Connectivity**: VNets can connect to on-premises networks via VPN gateways or ExpressRoute, enabling hybrid cloud scenarios.

**3. Traffic Filtering and Control**

* **Network Security Groups (NSGs)**: NSGs allow users to define rules to control inbound and outbound traffic to resources in a VNet based on IP address, port, and protocol.
* **Route Tables**: Users can create custom route tables to manage traffic routing between subnets and external networks.

**4. Integration with Azure Services**

* VNets can integrate with various Azure services, allowing for enhanced functionality. For example, you can extend your VNet to Azure services like Azure Storage or Azure SQL Database using service endpoints.

**5. Virtual Network Peering**

* This feature allows VNets in the same or different regions to connect seamlessly, enabling resources in peered VNets to communicate as if they were part of the same network.

Use Cases for Azure Virtual Network

* **Cloud-Only Applications**: Create dedicated cloud environments where applications run entirely within the VNet without needing on-premises connectivity.
* **Hybrid Cloud Solutions**: Extend your on-premises datacenter into the cloud securely, allowing for scalable resource management.
* **Secure Multi-Tenant Environments**: Isolate different applications or departments within the same Azure subscription by deploying them in separate VNets.

**Azure Virtual Network (VNet)** is a fundamental service in Microsoft Azure that enables users to create a logically isolated network in the cloud. This allows various Azure resources, such as virtual machines (VMs), to communicate securely with each other, the internet, and on-premises networks. Here’s an overview of Azure VNets and their key functionalities:

Key Features of Azure Virtual Network

**1. Isolation and Segmentation**

* **Private IP Address Space**: Users can define a private IP address range for their VNet using CIDR notation, ensuring that resources can communicate privately.
* **Subnets**: VNets can be divided into subnets, allowing for better organization and management of resources. Each subnet can have its own address range within the VNet.

**2. Secure Communication**

* **Inter-Resource Communication**: Resources within the same VNet can communicate directly and securely without requiring public IP addresses.
* **Hybrid Connectivity**: VNets can connect to on-premises networks via VPN gateways or ExpressRoute, enabling hybrid cloud scenarios.

**3. Traffic Filtering and Control**

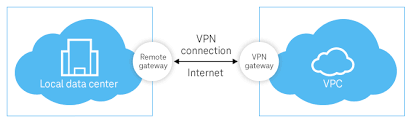
* **Network Security Groups (NSGs)**: NSGs allow users to define rules to control inbound and outbound traffic to resources in a VNet based on IP address, port, and protocol.
* **Route Tables**: Users can create custom route tables to manage traffic routing between subnets and external networks.

**4. Integration with Azure Services**

* VNets can integrate with various Azure services, allowing for enhanced functionality. For example, you can extend your VNet to Azure services like Azure Storage or Azure SQL Database using service endpoints.

**5. Virtual Network Peering**

* This feature allows VNets in the same or different regions to connect seamlessly, enabling resources in peered VNets to communicate as if they were part of the same network.



**Internet Protocol (IP)** is a fundamental protocol used for communication over the internet and within networks. It defines how data packets are addressed, transmitted, and routed between devices. Here’s an overview of key aspects of the Internet Protocol:

Types of Internet Protocol

1. **IPv4 (Internet Protocol version 4)**
   * **Description**: IPv4 is the fourth version of the Internet Protocol and is widely used for identifying devices on a network through an addressing system.
   * **Address Format**: It uses a 32-bit address format, allowing for approximately 4.3 billion unique addresses, represented in decimal as four octets (e.g., 192.168.1.1).
   * **Limitations**: The rapid growth of the internet has led to the exhaustion of available IPv4 addresses, prompting the need for a new version.
2. **IPv6 (Internet Protocol version 6)**
   * **Description**: IPv6 was developed to replace IPv4 and address its limitations, particularly the shortage of IP addresses.
   * **Address Format**: It uses a 128-bit address format, allowing for a vastly larger number of unique addresses (approximately 340 undecillion addresses), represented in hexadecimal (e.g., 2001:0db8:85a3:0000:0000:8a2e:0370:7334).
   * **Features**: IPv6 includes improvements such as simplified address configuration and enhanced security features.

Key Functions of Internet Protocol

* **Addressing**: Each device on a network is assigned a unique IP address that identifies it for communication purposes.
* **Routing**: IP determines how data packets are routed across networks to reach their destination based on the destination IP address.
* **Fragmentation and Reassembly**: Large data packets can be broken down into smaller fragments for transmission and reassembled at the destination.

Communication Protocols

IP operates alongside other protocols to facilitate communication:

* **Transmission Control Protocol (TCP)**: Ensures reliable, ordered delivery of data packets between applications.
* **User Datagram Protocol (UDP)**: Provides a connectionless communication method that is faster but does not guarantee delivery or order.
* **Internet Control Message Protocol (ICMP)**: Used for diagnostic purposes and error reporting in network communications.

**Public and private IP addresses** serve distinct roles in networking, each with specific address ranges defined by the Internet Assigned Numbers Authority (IANA). Here’s a detailed overview of both types, including their ranges:

Public IP Addresses

Public IP addresses are unique identifiers assigned to devices that connect directly to the internet. They allow devices to communicate with other devices across the global internet.

**Public IP Address Ranges:**

* **1.0.0.0 – 9.255.255.255**
* **11.0.0.0 – 126.255.255.255**
* **129.0.0.0 – 169.253.255.255**
* **169.255.0.0 – 172.15.255.255**
* **172.32.0.0 – 191.0.1.255**
* **192.0.3.0 – 192.88.98.255**
* **192.88.100.0 – 192.167.255.255**
* **192.169.0.0 – 198.17.255.255**
* **198.20.0.0 – 223.255.255.255**

Any IP address that falls outside of these public ranges is considered a private IP address.

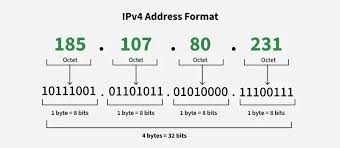
**Private IP Addresses**

Private IP addresses are used within local networks and are not routable on the internet, meaning they cannot be accessed directly from outside the local network.

**Private IP Address Ranges:**

* **Class A**: 10.0.0.0 to 10.255.255.255
* **Class B**: 172.16.0.0 to 172.31.255.255
* **Class C**: 192.168.0.0 to 192.168.255.255

These private ranges can be used by any organization without the need for coordination with IANA, allowing for reuse across different private networks.



**Storage accounts:**

Azure offers various types of storage accounts to manage different data storage needs, including Blob Storage, Queue Storage, and File Share Storage. Here’s an overview of each type:

1. Azure Blob Storage

Azure Blob Storage is designed for storing large amounts of unstructured data, such as images, videos, audio files, and documents. It provides a scalable and cost-effective solution for managing data in the cloud.

**Key Features:**

* **Scalability**: Can store vast amounts of data and scale up or down based on needs.
* **Durability**: Data is stored with high durability, often replicated across multiple data centers.
* **Security**: Offers encryption at rest and in transit, role-based access control, and shared access signatures.
* **Access Tiers**: Supports hot, cool, and archive tiers to optimize costs based on data access frequency.

**Use Cases:**

* Backup and disaster recovery
* Media storage and streaming
* Data analysis and big data processing
* Hosting static websites

**Types of Blobs:**

* **Block Blobs**: Best for storing text and binary data; ideal for media files.
* **Append Blobs**: Optimized for append operations; suitable for logging.
* **Page Blobs**: Used for random access files; primarily for virtual hard drives (VHDs).

2. Azure Queue Storage

Azure Queue Storage is a service that provides reliable messaging between application components. It allows applications to communicate asynchronously by sending messages to a queue.

**Key Features:**

* **Decoupling**: Helps separate application components, enabling them to operate independently.
* **Scalability**: Can handle large volumes of messages efficiently.
* **Durability**: Messages are stored reliably until they are processed.

**Use Cases:**

* Managing tasks in background processing
* Implementing workflows in distributed applications
* Decoupling microservices communication

3. Azure File Share Storage

Azure File Share provides fully managed file shares in the cloud that can be accessed via the SMB (Server Message Block) protocol. This service allows users to create file shares that can be mounted by multiple virtual machines.

**Key Features:**

* **File Sharing**: Enables multiple users or applications to share files easily.
* **Accessibility**: Can be accessed from anywhere using standard file system protocols.
* **Integration with Azure Services**: Works seamlessly with other Azure services for enhanced functionality.

**Use Cases:**

* Shared storage for applications running on Azure VMs
* Replacing on-premises file servers with cloud-based solutions
* Storing application data that needs to be accessed by multiple instances



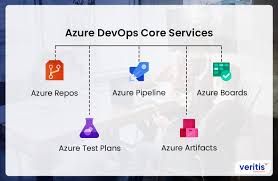
Azure DevOps is a comprehensive suite of development tools and services designed to support the entire software development lifecycle. It enables teams to collaborate effectively, automate processes, and deliver high-quality software efficiently. Here’s an overview of the core services offered by Azure DevOps:

Core Services of Azure DevOps

1. **Azure Boards**
   * **Description**: Provides a set of Agile tools for planning, tracking, and managing work items. It supports methodologies like Kanban and Scrum.
   * **Features**:
     + Work item tracking (user stories, bugs, tasks)
     + Backlogs and sprint planning
     + Dashboards and reporting tools
   * **Use Case**: Ideal for managing project workflows and enhancing team collaboration.
2. **Azure Repos**
   * **Description**: Offers cloud-hosted Git repositories or Team Foundation Version Control (TFVC) for source control.
   * **Features**:
     + Pull requests and code reviews
     + Branch policies to enforce code quality
     + Integration with CI/CD pipelines
   * **Use Case**: Facilitates collaborative coding efforts and version control management.
3. **Azure Pipelines**
   * **Description**: Provides continuous integration and continuous delivery (CI/CD) services for automating the build, test, and deployment processes.
   * **Features**:
     + Support for multiple languages and platforms
     + Integration with various CI/CD tools
     + Deployment to multiple environments (cloud, on-premises)
   * **Use Case**: Streamlines the application delivery process, ensuring faster releases.
4. **Azure Test Plans**
   * **Description**: A suite of testing tools for managing manual and exploratory testing.
   * **Features**:
     + Test case management
     + Integration with Azure Boards for tracking bugs
     + Continuous testing capabilities
   * **Use Case**: Ensures software quality through rigorous testing practices.
5. **Azure Artifacts**
   * **Description**: Allows teams to create, host, and share packages (e.g., Maven, npm, NuGet) from public and private sources.
   * **Features**:
     + Package versioning and retention policies
     + Integration with CI/CD pipelines for package deployment
   * **Use Case**: Facilitates dependency management in application development.

Additional Features

* **Integration with Other Tools**: Azure DevOps can integrate with popular services like Slack, Trello, GitHub, and more. This flexibility allows teams to enhance their workflows by connecting existing tools.
* **Custom Extensions**: Users can develop custom extensions to tailor Azure DevOps to their specific needs.
* **Hybrid Deployment Options**: Azure DevOps can be used in the cloud or deployed on-premises through Azure DevOps Server, providing flexibility based on organizational requirements.



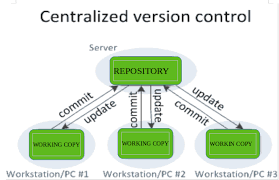
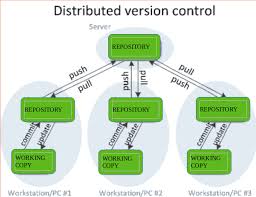
**Version control** also known as source control or revision control, refers to the processes and tools used to track and manage changes to files, particularly in software development. It allows developers to maintain a history of changes, collaborate effectively, and revert to previous versions if necessary. This practice is essential for ensuring code integrity and facilitating teamwork in complex projects.

Types of Version Control Systems

1. **Local Version Control**
   * **Description**: In this model, changes are tracked locally on a developer's machine. Each developer maintains their own version of the files.
   * **Advantages**: Simple to set up and use; ideal for individual projects.
   * **Disadvantages**: If the local version becomes corrupted or lost, recovery can be difficult.
2. **Centralized Version Control**
   * **Description**: This system uses a central server to store all versions of files. Developers check out files from the central repository, make changes, and then check them back in.
   * **Advantages**: Provides a single source of truth; easier to manage access and permissions.
   * **Disadvantages**: If the central server fails, access to all versions can be lost.
3. **Distributed Version Control**
   * **Description**: Each developer has a complete local copy of the repository, including its history. Changes are shared between repositories through patches or merges.
   * **Advantages**: Offers redundancy; even if the central server fails, local copies can restore the repository.
   * **Disadvantages**: More complex to manage due to multiple copies and potential for merge conflicts.

Key Features of Version Control Systems

* **Change Tracking**: Every modification made to files is logged with details about who made the change and when.
* **Branching and Merging**: Developers can create branches for new features or experiments without affecting the main codebase. Merging allows combining changes from different branches.
* **Reversion**: Easily revert files or entire projects back to previous versions in case of errors or issues.
* **Collaboration**: Multiple developers can work on the same project simultaneously without overwriting each other's changes.

**Azure DevOps Agents** are essential components that facilitate the execution of tasks, jobs, and pipelines within Azure DevOps. They play a crucial role in enabling Continuous Integration (CI) and Continuous Deployment (CD) processes. Here’s an overview of Azure DevOps Agents, their types, and how they operate:

Overview of Azure DevOps Agents

Azure DevOps Agents are responsible for running the jobs defined in your pipelines. When a pipeline is executed, one or more jobs are initiated, and these jobs require an agent to perform the tasks involved. Agents can be categorized based on their hosting and management.

Types of Azure DevOps Agents

1. **Microsoft-hosted Agents**
   * **Description**: These agents are hosted and managed by Microsoft. When a pipeline runs, a fresh virtual machine is created to execute the job, which is discarded after completion.
   * **Advantages**:
     + No need for maintenance or upgrades by the user.
     + Easy to set up and use for quick builds.
   * **Limitations**: Changes made during a job are not persistent since the virtual machine is discarded after execution.
2. **Self-hosted Agents**
   * **Description**: Self-hosted agents are installed and managed on your own infrastructure, such as physical or virtual machines, on-premises servers, or cloud VMs.
   * **Advantages**:
     + Complete control over the environment and installed software.
     + Ability to maintain state across jobs (e.g., caching dependencies).
   * **Use Cases**: Ideal for scenarios requiring specific tools or configurations that are not available in Microsoft-hosted agents.
3. **Managed DevOps Pools Agents**
   * **Description**: This type uses a fully managed service where the agents run in a Microsoft Azure subscription rather than your own.
   * **Advantages**: Simplifies management while still providing the benefits of self-hosted agents.
4. **Azure Virtual Machine Scale Set Agents**
   * **Description**: A form of self-hosted agent that utilizes Azure Virtual Machine Scale Sets, allowing for auto-scaling based on demand.
   * **Advantages**: Provides flexibility and scalability for handling varying workloads.

Agent Pools

* **Definition**: An agent pool is a collection of one or more agents that can be used to run builds and deployments. This allows teams to manage resources more efficiently by grouping agents based on their capabilities (e.g., operating system).
* **Usage**: Each pipeline targets an agent pool, and Azure DevOps determines which agent from the pool will execute a specific job.

Key Features of Azure DevOps Agents

* **Capabilities**: Agents can have specific capabilities (e.g., installed software) that match the demands of jobs they are assigned to run.

**Directories in devops**

In Azure DevOps, the directory structure created during pipeline execution is crucial for organizing and managing various types of files, including source code, binaries, artifacts, and test results. Here's an overview of the key directories involved:

Understanding the directory structure in Azure DevOps is essential for effectively managing builds and deployments. Each directory serves a specific purpose, enabling teams to organize their source code, binaries, artifacts, and test results systematically. Utilizing predefined variables allows for easy access to these directories during pipeline execution, enhancing automation and workflow efficiency.

**1. Source Directory (/s)**

* **Description**: This directory contains the source code that is checked out from the repository. If there are multiple checkout steps in your job, the source code is organized into subdirectories named after the repositories.
* **Predefined Variables**:
  + Build.SourcesDirectory
  + Build.RepositoryLocalPath
  + System.DefaultWorkingDirectory

**2. Binaries Directory (/b)**

* **Description**: This directory holds the build outputs generated during the pipeline execution. It contains compiled binaries and other output files from the build process.
* **Predefined Variable**:
  + Build.BinariesDirectory

**3. Artifacts Staging Directory (/a)**

* **Description**: This directory is used to stage build artifacts before they are published. It is cleaned between runs on self-hosted agents, ensuring that only relevant artifacts are retained.
* **Predefined Variables**:
  + Build.StagingDirectory
  + Build.ArtifactStagingDirectory
  + System.ArtifactsDirectory

**4. Test Results Directory**

* **Description**: This directory contains the results of tests executed during the pipeline run. It is also cleaned between runs on self-hosted agents.
* **Predefined Variable**:
  + Common.TestResultsDirectory

