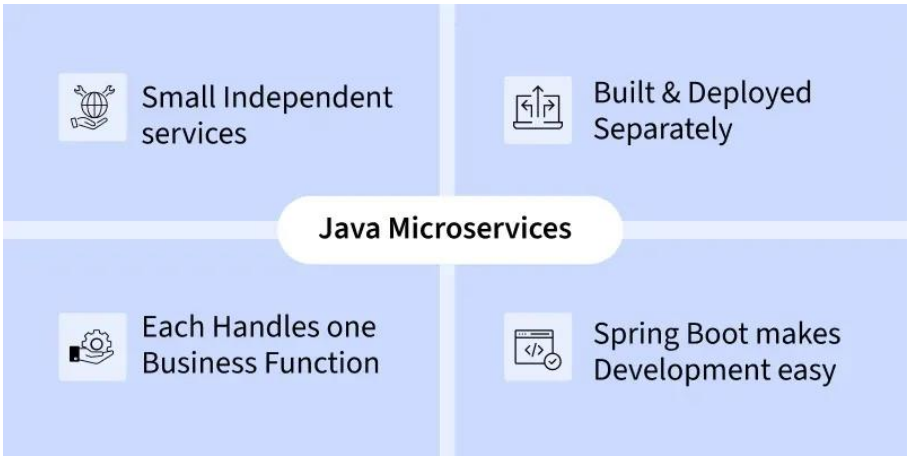
4.5 USE CASE

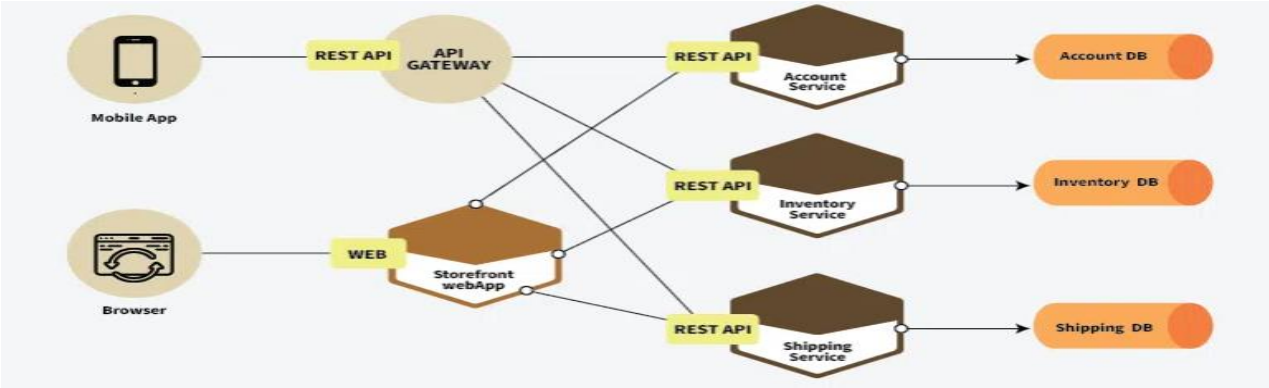
1. Microservices:

- Microservices are a way of developing software applications as a collection/set of small, independent services that communicate with each other over a network.
- Instead of building a monolithic application where all the functionality is tightly integrated into a single codebase, microservices split (break down) the application into smaller, loosely coupled services.
- Can be written in a variety of programming languages, and frameworks, and each service acts as a mini-application on its own.
- A small, loosely coupled service that is designed to perform a specific business function and each microservice can be developed, deployed, and scaled independently.



Working of Microservices:

- Each microservice handles a particular business function/feature, like user authentication, product orders, payments etc.,.
- Services interact with other services via APIs, facilitating standardized information exchange and integration.



- Different technologies can be used for each service, enabling teams to select the best tools for their needs.
- Updates can be made to one service **without affecting the others**.
- This makes the system more flexible and reliable.

Main components of Microservices Architecture:

Main components of microservices architecture include:

- **Microservices:** Small, loosely coupled services that handle specific task, each focusing on a distinct capability.
- **API Gateway:** Acts as a central entry point for external clients also they manage requests, authentication and route the requests to the appropriate microservice.
- **Service Registry and Discovery:** Keeps track of the locations and addresses of all microservices, enabling them to locate and communicate with each other dynamically.
- **Load Balancer:** Distributes incoming traffic across multiple service instances and to prevent any of the microservice being overload.
- **Containerization:** Docker encapsulate microservices and their dependencies and orchestration tools like Kubernetes manage their deployment and scaling.
- **Event Bus/Message Broker:** Facilitates communication between microservices, allowing pub/sub asynchronous interaction of events between components/microservices.
- **Database per Microservice:** Each microservice usually has its own database, promoting data autonomy and allowing for independent management and scaling.
- **Caching:** Cache stores frequently accessed data close to the microservice which improved performance by reducing the repetitive queries.
- **Fault Tolerance and Resilience Components:** Components like circuit breakers and retry mechanisms ensure that the system can handle failures smoothly, maintaining overall functionality.

Design Patterns for Microservices Architecture:

Below are the main design pattern of microservices:

1. API Gateway Pattern:

- API Gateway pattern simplifies the client's experience by hiding the complexities of multiple services behind one interface.

- It can also handle tasks like authentication, logging, and rate limiting, making it a crucial part of microservices architecture.

2. Service Registry Pattern:

- It acts like a phone book for microservices.
- It maintains a list of all active services and their locations (network addresses).
- When a service starts, it registers itself with the registry.
- Other services can then look up the registry to find and communicate with it.
- This dynamic discovery enables flexibility and helps services interact without hardcoding their locations.

3. Circuit Breaker Pattern:

- If a service fails repeatedly, the circuit breaker trips, preventing further requests to that service.
- After a timeout period, it allows limited requests to test if the service is back online.
- This reduces the load on failing services and enhances system resilience.

4. Saga Pattern:

- Saga pattern is useful for managing complex business processes across multiple services.
- The single process is broken down into smaller steps, each handled by different services.
- If one step fails, compensating actions are taken to reverse the previous steps.
- This way data consistency is maintained across the system, even in the case of failures.

5. Event Sourcing Pattern:

- In Event Sourcing Pattern, Each event describes a change that occurred, allowing services to reconstruct the current state by replaying the event history.
- This provides a clear audit trail and simplifies data recovery in case of errors.

6. Strangler Pattern:

- Strangler pattern allows for a gradual transition from a monolithic application to microservices.

- New features are developed as microservices while the old system remains in use.
- Over time, as more functionality is moved to microservices, the old system is gradually "strangled" until it can be fully retired.
- This approach minimizes risk and allows for a smoother migration.

7. Bulkhead Pattern:

- Similar to compartments in a ship, the bulkhead pattern isolates different services to prevent failures from affecting the entire system.
- If one service encounters an issue, it won't compromise others.
- By creating boundaries, this pattern enhances the resilience of the system, ensuring that a failure in one area doesn't lead to a total system breakdown.

8. API Composition Pattern:

- When we need to gather data from multiple microservices, the API composition pattern helps us do so efficiently.
- A separate service (the composition service) collects responses from various services and combines them into a single response for the client.
- This reduces the need for clients to make multiple requests and simplifies their interaction with the system.

9. CQRS Design Pattern:

- CQRS Design Pattern divides the way data is handled into two parts: commands and queries.
- Commands are used to change data, like creating or updating records, while queries are used just to fetch data.
- This separation allows us to tailor each part for its specific purpose.

Real-World Example of Microservices

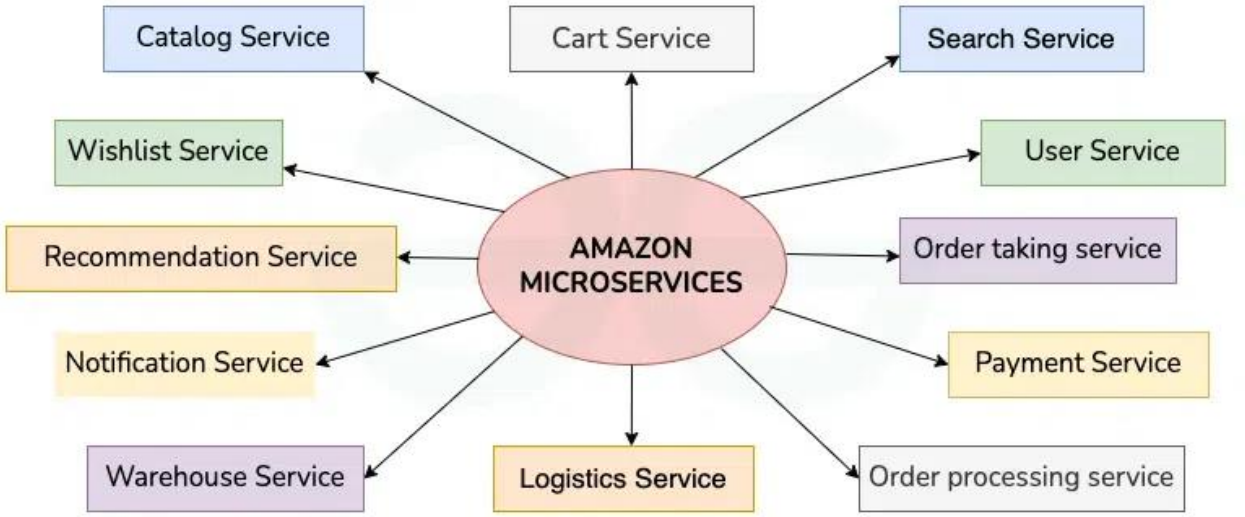
Let's understand the Miscroservices using the real-world example of Amazon E-Commerce Application:

- Amazon's online store runs on many small, specialized microservices, each handling a specific task.
- Working together, they create a smooth shopping experience.

Below are the microservices involved in Amazon E-commerce Application:

- **User Service**: Handles user accounts and preferences, making sure each person has a personalized experience.

- **Search Service**: Helps users find products quickly by organizing and indexing product information.

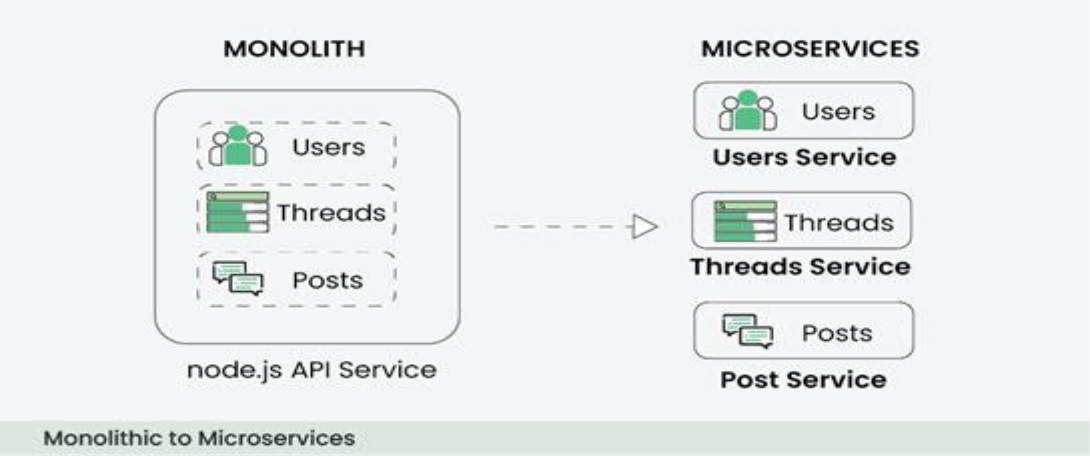


Amazon e-commerce Microservices

- **Catalog Service**: Manages the product listings, ensuring all details are accurate and easy to access.
- **Cart Service**: Lets users add, remove, or change items in their shopping cart before checking out.
- **Wishlist Service**: Allows users to save items for later, helping them keep track of products they want.
- **Order Taking Service**: Processes customer orders, checking availability and validating details.
- **Order Processing Service**: Oversees the entire fulfillment process, working with inventory and shipping to get orders delivered.
- **Payment Service**: Manages secure transactions and keeps track of payment details.
- **Logistics Service**: Coordinates everything related to delivery, including shipping costs and tracking.
- **Warehouse Service**: Keeps an eye on inventory levels and helps with restocking when needed.
- **Notification Service**: Sends updates to users about their orders and any special offers.
- **Recommendation Service**: Suggests products to users based on their browsing and purchase history

How to migrate from Monolithic to Microservices Architecture?

Below are the main the key steps to migrate from a monolithic to microservices architecture:



- **Step 1**: Begin by evaluating your current monolithic application. Identify its components and determine which parts can be shifted to microservices.
- **Step 2**: Break down the monolith into specific business functions. Make each microservice handle a specific business task.
- **Step 3**: Implement the Strangler Pattern to gradually replace parts of the monolithic application with microservices. This method allows for a smooth migration without a complete transition at once.
- **Step 4**: Make clear ways for services to interact/communicate to each other.
- **Step 5**: Create Continuous Integration and Continuous Deployment (CI/CD) pipelines. This automates testing and deployment, enabling faster and more reliable releases(updates).
- **Step 6**: Introduce mechanisms for service finds, can dynamically locate and communicate with each other, enhancing flexibility.
- **Step 7**: Set up centralized logging and monitoring tools. This provides insights into the performance of your microservices, helping to identify and resolve issues quickly.
- **Step 8**: Ensure consistent management of cross-cutting concerns, such as security and authentication, across all microservices to maintain system integrity.
- **Step 9**: Take an iterative approach to our microservices architecture. Continuously refine and expand our services based on feedback and changing requirements

Applications of Microservices:

- **E-commerce platforms:** Separate services for inventory, payments, orders, and user management.
- **Banking & FinTech:** Independent services for accounts, transactions, fraud detection, and customer support.
- **Ride-Hailing Apps** are mobile applications that allow users to **book rides on demand** from drivers using their personal or commercial vehicles. These apps connect passengers with nearby drivers through the internet, providing a convenient alternative to traditional taxis.
- An **OTT Platform (Over-The-Top Platform)** is a service that delivers video, audio, or other media content directly to users over the **internet**, bypassing traditional distribution methods like cable TV, satellite TV, or radio.

- **Streaming services (e.g. Netflix, Spotify):** Services for user profiles, recommendations, content delivery, and billing.
- **Travel & Booking systems:** Services for flights, hotels, payments, and notifications.
- **Healthcare systems:** Patient records, appointment scheduling, billing, and reporting as separate services.
- **Social media platforms:** Microservices for feed, chat, notifications, and user profiles.

Benefits of using Microservices Architecture:

- Teams can work on different microservices simultaneously.
- Issues in one service do not impact others, enhancing reliability.
- Each service can be scaled based on its specific needs.
- The system can quickly adapt to changing workloads.
- Teams can choose the best tech stack for each microservice.
- Small, cross-functional teams work independently.

Challenges of using Microservices Architecture:

- Managing service communication, network latency, and data consistency can be difficult.
- Decomposing an app into microservices adds complexity in development, testing and deployment.
- Network communication can lead to higher latency and complicates error handling.
- Maintaining consistent data across services is challenging, and distributed transactions can be complex.

Microservices vs. Monolithic Architecture:

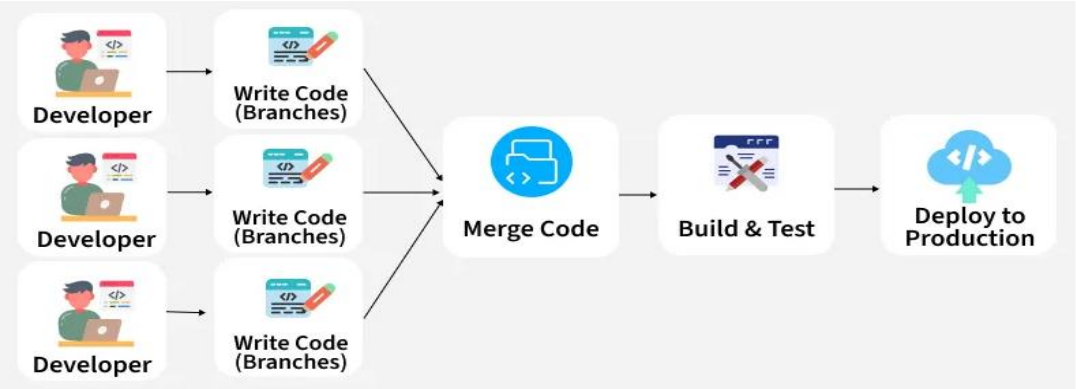
Aspect	Microservices Architecture	Monolithic Architecture
Architecture Style	Decomposed into small, independent services.	Single, tightly integrated codebase.
Development Team Structure	Small, cross-functional teams for each microservice.	Larger, centralized development team.
Scalability	Independent scaling of individual services.	Scaling involves replicating the entire application.
Deployment	Independent deployment of services.	Whole application is deployed as a single unit.
Resource Utilization	Efficient use of resources as services can scale independently.	Resources allocated based on the overall application's needs.
Development Speed	Faster development and deployment cycles.	Slower development and deployment due to the entire codebase.
Flexibility	Easier to adopt new technologies for specific services.	Limited flexibility due to a common technology stack.
Maintenance	Easier maintenance of smaller, focused codebases.	Maintenance can be complex for a large, monolithic codebase.

2.CI/CD (Continuous Integration and Continuous Delivery/Deployment):

- CI/CD stands for Continuous Integration and Continuous Delivery/Deployment.
- With CI/CD, we automate the integration of code changes from multiple developers into a single codebase.
- It is a software development practice where the developers commit their work frequently to the central code repository (GitHub or Stash).

Before CI/CD:

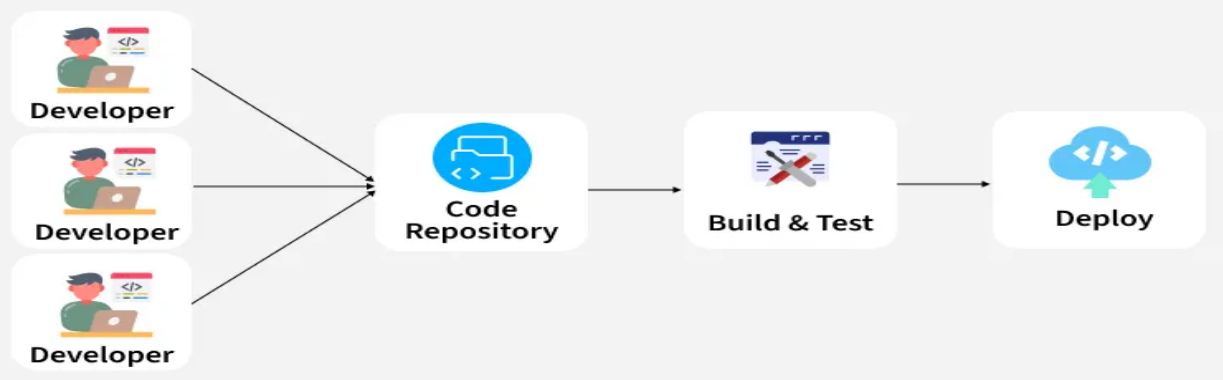
- In the past, software delivery was slow and full of manual work.
- Developers used to create separate feature branches and worked on them for weeks or even months.



- All these branches were merged at the end, which often caused big conflicts and broken builds.
- Testing and building the code were also manual, usually done at the final stage, so bugs were discovered very late and were costly to fix.
- Deployment was a long and risky process, taking days or sometimes weeks, because everything was released together in one big update.
- Teams were also divided: developers wrote code, testers tested it, and operations managed servers.

After CI/CD:

- With CI/CD, the whole process became faster, automated, and more reliable.
- Developers now commit their code frequently to a shared main branch, which prevents conflicts and ensures smooth integration.
- Automated pipelines run tests, builds, and checks as soon as code is pushed, so errors are caught early. Continuous Delivery makes sure that tested code is automatically packaged and ready for deployment



- While Continuous Deployment takes it a step further by directly releasing to production without manual steps. This means smaller, frequent updates instead of one big risky release.
- **Tools like Git, Jenkins, Docker, and Kubernetes automate** everything from building to deployment, making the workflow transparent and collaborative.
- Bugs are fixed quickly, rollbacks are easier, and overall software delivery happens in hours instead of days

Continuous Integration:

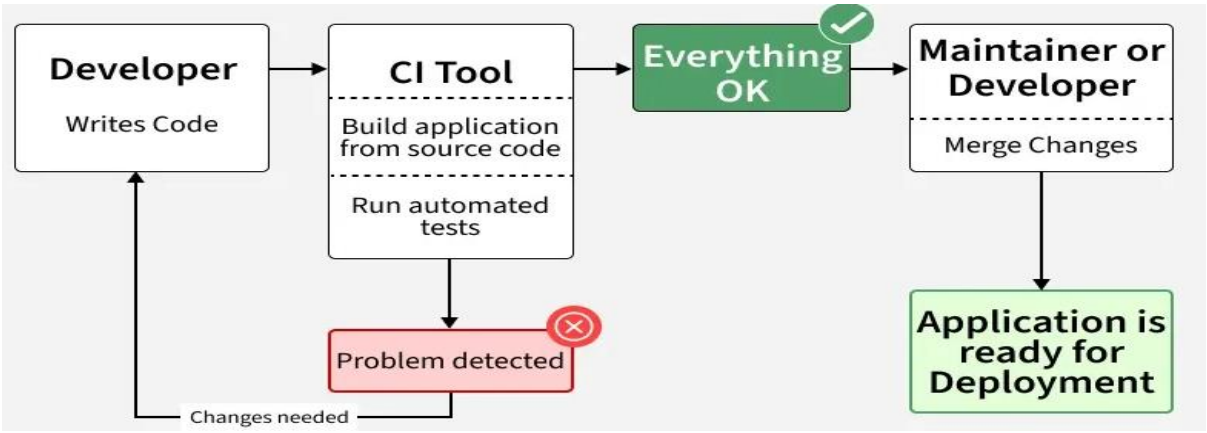
- Continuous Integration is about how developers integrate code using a shared repository multiple times a day with the help of automation.
- Examples of popular CI tools include Jenkins, CircleCI, and Travis CI.

Continuous Delivery:

- Continuous Delivery is about automatically releasing software to the test or production environment.
- Examples of popular CD tools include GitLab CI/CD, Bamboo, and CodeShip.

CI Workflow:

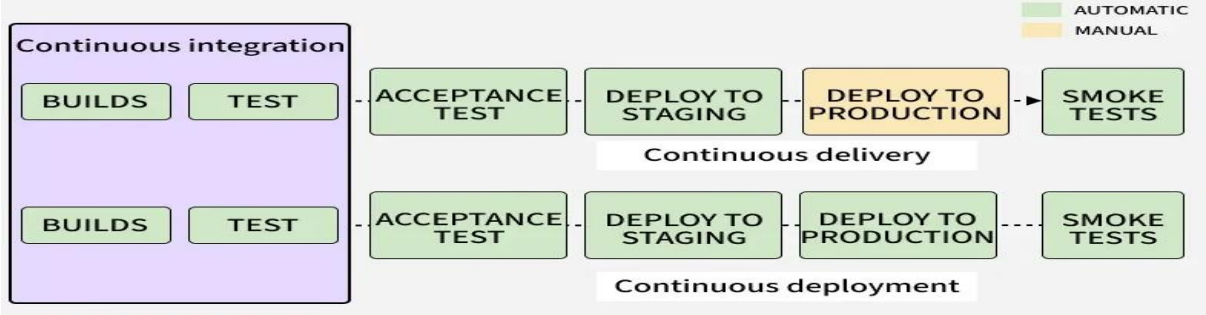
- Below is a pictorial representation of a **CI pipeline**- the workflow from developers checking in their code to its automated build, test, and final notification of the build status.



- When a developer saves (commits) their code to a version control system like Git, the CI pipeline starts.
- It takes the new changes, builds the project, and runs tests automatically.
- After that, the server tells the developer if adding the new code to the existing code worked or failed.
- This helps find and fix bugs faster, saves developers from doing the same tasks again and again, and allows the team to release updates to customers more often.
- Studies show that using CI can save developers about 25–30% of their time.

CI and CD Workflow:

- The below image describes how Continuous Integration combined with Continuous Delivery helps quicken the software delivery process with lower risks and improved quality.



- We have seen how Continuous Integration automates the process of building, testing, and packaging the source code as soon as it is committed to the code repository by the developers.
- After the **Continuous Integration (CI) step**, the code is deployed to a staging environment.
- In the staging environment, it undergoes further automated testing such as: Acceptance Testing & Regression Testing
- Once tests are successful, the code is deployed to the production environment for the final release.
- If the production deployment is manual, the process is called Continuous Delivery.
- If the production deployment is automated, the process is called Continuous Deployment.

Common CI/CD Tools:

- CI and CD tools can help team in the development, deployment, and testing, some of the tools are highly recommended for the integration part and some are for the development and management of the testing and related functionality.
- Most of the famous tools for the CI and CD is **Jenkins**. It is open source and it will help to handle all types of work and design a simple CI server to complete the CD hub.

Jenkins Workflow in CI/CD:

- **Developer Commit:** Code is pushed to GitHub/GitLab/Bitbucket.
- **Trigger:** Jenkins detects the commit (via webhook or polling).
- **Build:** Jenkins builds the code using tools like Maven, Gradle, or Docker.(Code is compiled into executable/Docker image)
- **Test:** Automated tests are run to verify quality.
- **Deploy:** Code is deployed to staging (like practice ground where the code runs in a safe place before the real users see it) or production environments (customer's system) automatically.
- **Feedback:** Developers get instant notifications of success or failure (via email, Slack, etc.).

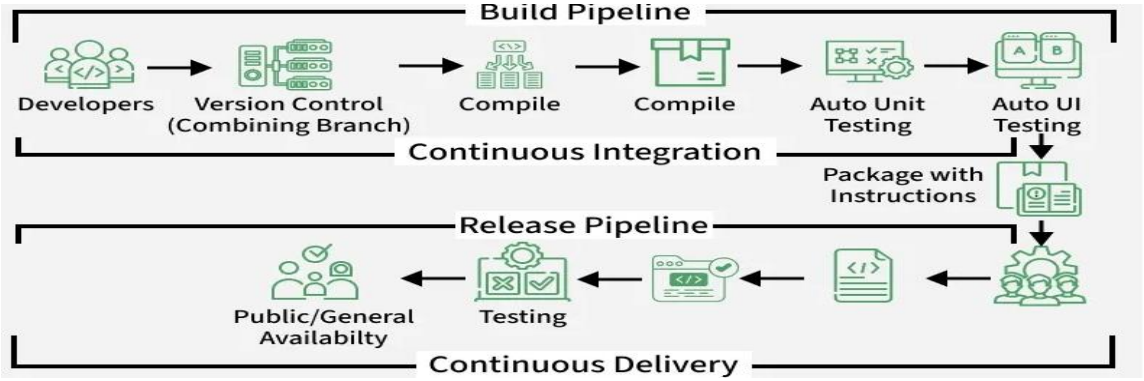
Apart from the Jenkins, many more sources are available for the proper way of managing CI and CD which are listed below:

- Concourse: It is an open-source tool to build the mechanics of CI and CD.
- GoCD: it's used for the modeling and visualization.
- Screwdriver is a building platform for CD.
- Spinnaker: it's a CD platform used to build a multi-cloud environment.

CI/CD Pipeline:

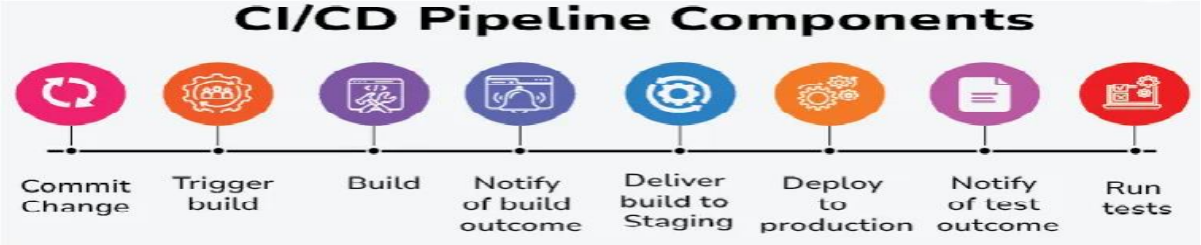
- **CI/CD (Continuous Integration and Continuous Delivery):** This is a way of working where developers can make many code changes and deliver updates quickly.
- **CI/CD Pipeline:** This is the actual step-by-step process that takes the code and automatically moves it from development to live production.

In simple words, a CI/CD pipeline is a series of automatic steps that help developers build, test, and deliver code safely and quickly.



Components of a CI/CD Pipeline:

These components create a **CI/CD pipeline workflow**



Components of CI/CD Pipeline:

1. Commit Change:

- Developers make code changes and commit these changes to a version control system (e.g., Git).
- This step initiates the CI/CD pipeline. Committing code changes ensures that they are tracked and versioned properly.

2. Trigger Build:

- The version control system detects the new commit and triggers the build process automatically.
- Automated triggering of the build process ensures that new changes are continuously integrated and tested.

3. Build:

- The **code is compiled and packaged into a deployable** artifact, such as a binary executable or a Docker image.
- Common tools include Maven, Gradle, Ant for Java projects, and Docker for containerized applications.
- The build step verifies that the code compiles correctly and that all dependencies are resolved.

4. Notify of Build Outcome:

- The CI/CD system **notifies** the team of the build results, whether it **passed or failed**.

- Immediate feedback on the build status helps developers quickly identify and resolve any build issues.
- Notification systems can include emails, chat integrations (like Slack or Microsoft Teams), or dashboards in tools like Jenkins or GitLab CI.

5. Run Tests:

- Automated tests are executed(run) on the build artifact.
- These can include unit tests, integration tests, end-to-end tests, and more.
- Testing frameworks like JUnit, Selenium, TestNG, pytest, etc.
- Running tests ensures that the new code does not introduce any bugs or regressions and that it meets the required quality standards.

6. Notify of Test Outcome:

- The results of the test are reported to the development team.
- Quick feedback on test outcomes allows developers to fix the failures quickly.
- Similar to build notifications, results can be sent via email, chat, or dashboards.

7. Deliver Build to Staging:

- If tests pass, the build is deployed to a staging environment.
- The staging environment simulates the production environment, allowing for final validation before production deployment. Deployment tools like Ansible, Chef, Puppet, Kubernetes, or cloud-specific services like AWS CodeDeploy.

8. Deploy to Production:

- After successful validation in staging, the build is automatically or manually promoted to the production environment.
- This step makes the new features and fixes available to end-users. Deployment strategies might include blue-green deployment, canary releases, or rolling updates to minimize downtime and risk.

CI/CD Security

- **Code & Dependencies:** Avoid unsafe/outdated libraries.
- **Pipeline Access:** Use strong passwords & role-based access.
- **Container Security:** Only use verified images.
- **Infrastructure Security:** Correctly configure cloud and servers.

3. VIRTUALBOX

Introduction :

When using a traditional you need to install the operating system on a physical machine for evaluating software that cannot be installed on your current operating system. Oracle VirtualBox is what you need in this case, instead of reinstalling software on your physical machine. VirtualBox is designed to run virtual machines on your physical machine without reinstalling your OS that is running on a physical machine. One more VirtualBox advantage is that this product can be installed for free.

A virtual machine (VM) works much like a physical one. An OS and applications installed inside a VM "think" that they are running on a regular physical machine since emulated hardware is used for running VMs on VirtualBox. Virtual machines are isolated from each other and from the host operating system. Thus, you can perform your tests in isolated virtual machines without any concerns of damaging your host operating system or other virtual machines.

Operating Systems Supported by VirtualBox

VirtualBox supports a long list of host and guest operating systems. A host OS is the operating system installed on a physical machine, on which VirtualBox is installed. A guest OS is an operating system installed on a virtual machine running inside VirtualBox. VirtualBox can be installed on Windows, Linux, macOS, Solaris, and FreeBSD. On VirtualBox you can run VMs with Windows, Linux, macOS, Solaris, FreeBSD, Novell Netware, and other operating systems.

How to Set Up VirtualBox?

Let's explore how to set up VirtualBox on Windows. The process of VirtualBox installation is not difficult and is similar for all supported operating systems.

Enable CPU virtualization features

First, you need to enable hardware virtualization features on your CPU (Central Processor Unit) such as Intel VT-X or AMD-V in UEFI/BIOS of your physical computer. Otherwise, if you run 64-bit guest operating systems, you can get the error: VT-x is not available. If Hyper-V is installed on your Windows machine, uninstall Hyper-V before installing VirtualBox (otherwise Hyper-V will block

hardware virtualization extensions needed by VirtualBox to run VMs). The majority of modern processors support hardware virtualization.

Download the VirtualBox installer

Go to the official web site to download the VirtualBox installer for your operating system (Windows in this case). If you are looking for how to set up VirtualBox on mac, download the OS X installer. At the moment of writing this blog post, the latest VirtualBox version is 6.0.8. You can also download older versions, for example, version 5.2. VirtualBox 5.2 supports 32-bit hosts while VirtualBox 6.0 doesn't. In the current example, you can see how to set up VirtualBox on an example of VirtualBox 5.2. In the next blog post, the VirtualBox upgrade process will be explained.

Run the installer and define the installation options

1. Run the VirtualBox installer. The installation wizard that has a GUI (graphical user interface) should appear.
2. Select the manner in which you want features to be installed, clicking on the installation directory and installed components—you can leave the default values. Then tick the checkboxes near shortcut options and file associations.
3. Confirm installation of VirtualBox network interfaces (click Yes).
4. On the *Ready to Install* Screen, hit Install to start the installation process.
5. After finishing installation, you can tick the checkbox for starting VirtualBox after installation.

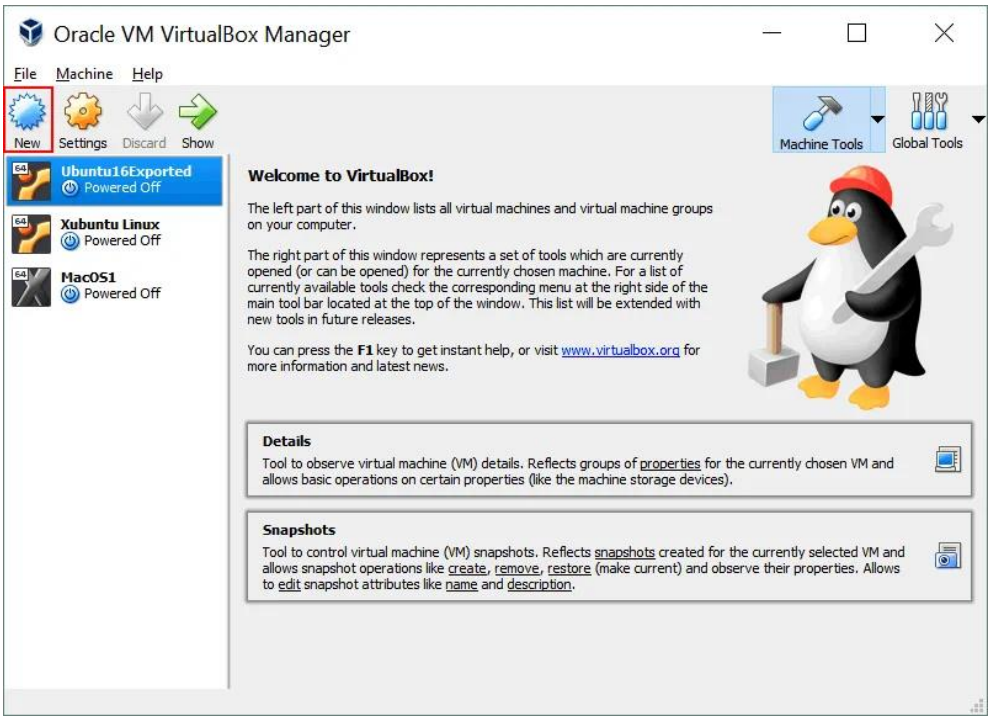
Deploying a New VM

Once you have installed VirtualBox, open the application. You can see the graphical user interface of VirtualBox which is unified for all supported host operating systems. You can also use the command line interface and VBoxManage if needed. In the current example VirtualBox is set up on Windows 10. Imagine that you need to evaluate Windows Server 2019 before making a decision – to buy or not to buy. Let's look at how to use VirtualBox for running Windows Server 2019 on a virtual machine.

Download the ISO image of Windows Server 2019 installer from the Microsoft's site. Rename the downloaded ISO file to *WinServer2019.iso* for more convenience.

Creating a Virtual Machine

Click **Machine > New** or hit the icon with the blue star to create a new virtual machine in VirtualBox GUI.



Define the new VM options.

Name: WinServer2019.

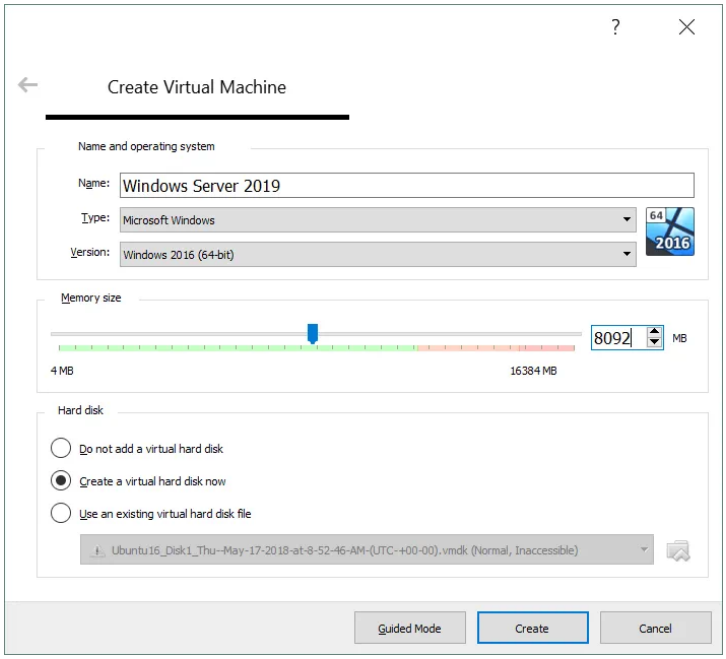
Type: Microsoft Windows.

Version: Windows 2016 (64-bit). This parameter defines the reasonable default amount of virtual memory, virtual disk size; a set of emulated hardware (devices that are supported by the selected OS version, drivers for which are included); as well as a set of system features such as EFI, PAE (physical address extension), I/O APIC (input/output advanced programmable interrupt controllers). If you are using the latest version of VirtualBox, Windows 2019 is available in the list of OS versions.

Memory size: Set memory for the VM. 8 GB of RAM should be enough for Windows Server 2019 for the beginning. You can add more memory later, after installing the guest OS (a VM must be powered off to change the amount of memory).

Hard Disk: Create a virtual hard disk now.

Click the **Create** button.



Creating a Virtual Hard Disk

Set the following parameters:

The name and file location for the virtual disk. Try not to use a system partition for storing virtual disks if possible.

The file size of the virtual disk. Select 50 GB for Windows Server 2019.

Hard disk file type. VirtualBox supports a lot of virtual disk formats:

- VDI (VirtualBox Disk Image) is a native VirtualBox format. Select this virtual disk type if you don't plan to migrate a VM to other platforms such as VMware of Hyper-V.
- VHD (Virtual Hard Disk) is a Hyper-V format.
- VMDK (Virtual Machine Disk) is the VMware virtual disk format.
- HDD is the Parallels Hard Disk.
- QCOW (QEMU Copy-On-Write).
- QED (QEMU enhanced disk).

Let's select VDI in the current example.

Storage on physical hard disk: Dynamically allocated or fixed size (the analog of thin and thick provisioning in VMware). Select the dynamically allocated option if you want to save disk space, as in this case, the size of your virtual disk

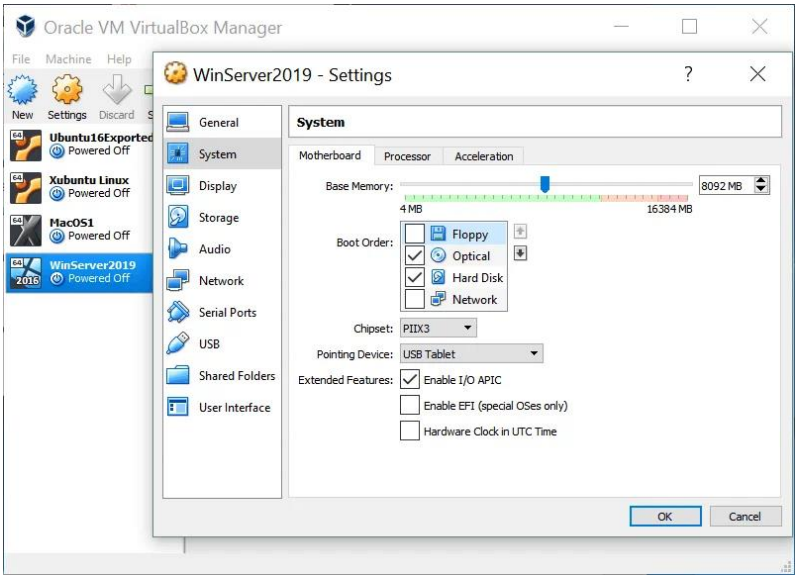
is near zero right after creation, and grows after writing data to the virtual disk before reaching the maximum allocated size.

Click **Create** to finish VM creation.

Virtual machine Tuning. Tune your virtual machine by going to **Machine > Settings**. The following sections are in this window.

General. You can edit the VM name, enable shared clipboard, drag & drop, write the VM description, and enable virtual disk encryption.

System. Disable a virtual floppy drive in the *Motherboard* tab. In the System tab, you can add more processors for the VM, configure acceleration, and select one of the two emulated chipsets.



Display. In the *Screen* tab, it is possible to set the video memory, monitor count, and scale factor as well as enabling 2D and 3D acceleration. The *Remote Display* tab allows you to manage your guest OS by connecting to the VirtualBox VM desktop remotely with RDP (Remote Desktop Protocol). Capturing video is configured in the *Video Capture* tab.

Storage. Add and remove virtual hard disks, virtual DVD drives, and disk controllers if needed. Select your DVD drive (it should be empty until this moment) and insert the virtual ISO DVD disc into the drive. Click the disc icon and select "Choose virtual optical disc file." Browse your *WinServer2019.iso* file and open it. Now your ISO disc is displayed in the list of storage devices.

Audio. Audio can be enabled or disabled; the host audio driver, audio controller and extended features can be selected.

Network. Virtual network adapters are configured in this section. The maximum number of virtual network adapters per VM is four. A virtual network adapter can

use a variety of different network modes: Not attached, NAT, NAT Network, Bridged Adapter, Internal Network, Host-only Adapter, Generic Driver, among which the NAT network mode is used by default.

When the *NAT* mode is selected, your VM is connected to the virtual router and can access the host, the network to which the host is connected and external networks that can be accessed by the host. If you deploy a single VM, which is not needed to be seen as a regular machine in your physical network, you can select the *NAT* mode for VM networking. If you want your VM to be fully representative in your physical network, use the *Bridged* mode.

The model of the emulated virtual network adapter is selected from the drop-down menu. You may set the MAC address manually if necessary. Configure *Port Forwarding* for accessing VMs that use the *NAT* network mode from the physical network your host machine is connected to (if necessary). If you use the bridged networking, you don't need to configure port forwarding.

Serial Ports. Enable Serial ports if for some particular reason you need COM ports to be enabled on a VM.

USB. USB options for a VM are configured in this section.

Shared Folders. Shared folders are used for file exchange between host OS and guest OS.

User interface. Customize the elements of GUI if you wish.

Click **OK** to apply the edited VM configuration.

Installing a guest OS

Now you can start the VM. Hit **Machine > Start > Normal Start**.

Normal Start. Opens a VM window and displays the video output of the VM in that window similarly as the output of the physical machine is displayed on a monitor. When you close the VM window, VirtualBox asks you what to do:

- Save the machine state. The VM is hibernated (a VM is on pause). Start the VM to continue VM operation from the saved state.
- Send the shutdown signal. The VM is shutting down correctly, similarly as you would shut down the machine from the operating system.
- Power Off the machine. This option is the equivalent of unplugging the power cable from the physical computer.

Headless Start. A VM is started, but the Window with the video output of the VM is not appeared. You can connect to the VM for managing it by using a remote

desktop protocol (including VRDP – VirtualBox Remote Display Protocol that is backward compatible with Microsoft RDP), SSH etc.

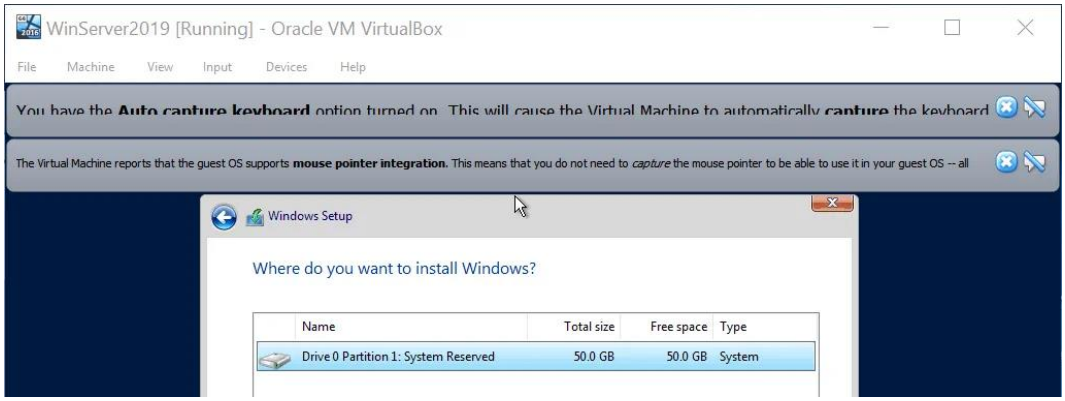
Detachable start. This is the combination of the normal start and headless start. When you close the VirtualBox VM window, one more option is available – *Continue running in the background*. You can close the VirtualBox VM window without interrupting the VM.

The OS installer is now booting from the ISO image inserted to a virtual DVD drive. This process is displayed in the new VirtualBox VM window. If you know how to set up Windows 10 on VirtualBox, or on a physical machine, the installation process of Windows Server 2019 in a GUI mode shouldn't cause any problems.

Select suitable options for the Windows installation wizard:

- Windows Server 2019 Datacenter Evaluation (Desktop experience).
- Custom: Install Windows only (advanced).

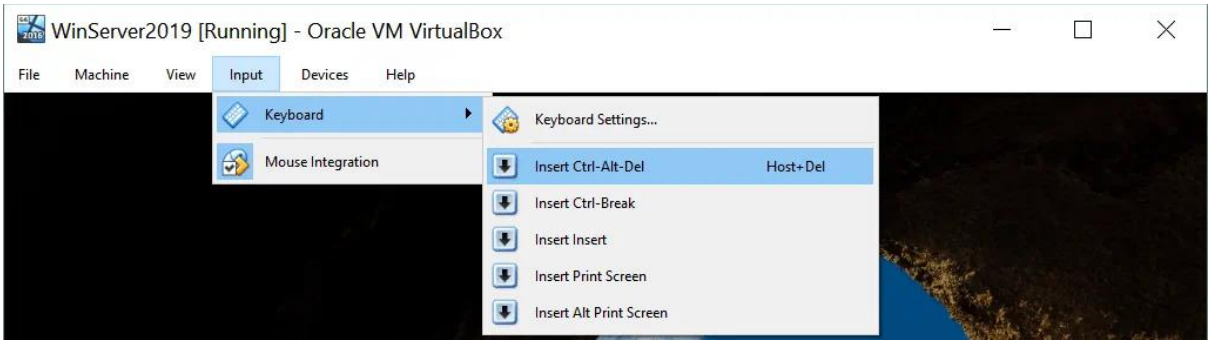
  Create a new partition (or partitions) on your 50-GB virtual hard disk.



The VM automatically restarts a few times during Windows Server 2019 installation.

Set the Windows administrator password to finish Windows Server 2019 setup on VirtualBox.

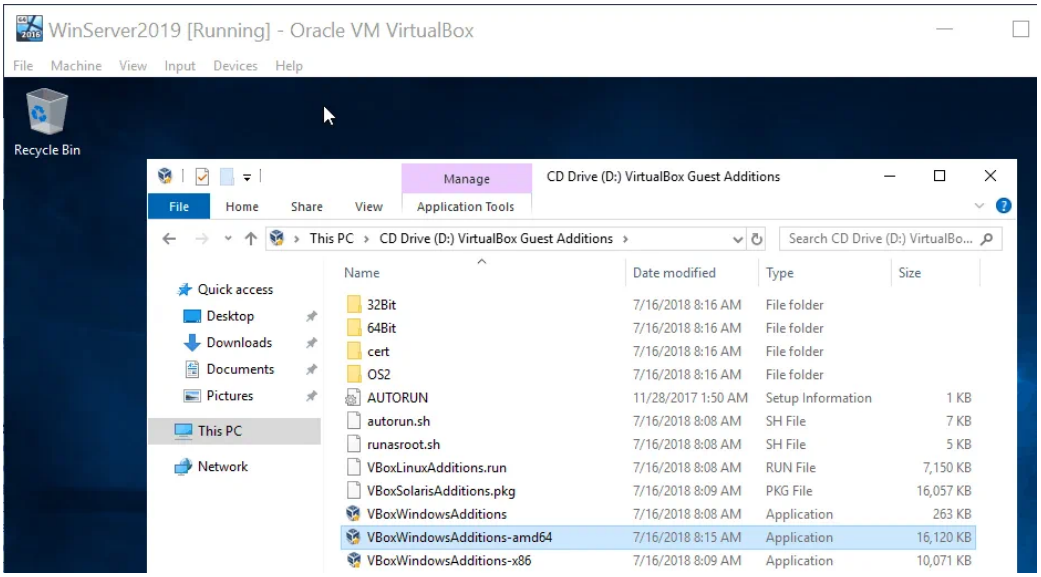
After loading, Windows asks you to press **Ctrl+Alt+Delete** to unlock. Click **Input > Keyboard > Insert Ctrl+Alt+Del** in the VirtualBox VM window.



Once you have logged in your Windows system, install VirtualBox Guest Additions. Guest Additions are a set of drivers and system utilities for supported guest operating systems that optimize the guest OS performance and usability, in addition to providing closer interaction between host and guest operating systems. Features such as mouse pointer integration, enhanced video mode support, shared folders, shared clipboard, and time synchronization can be enabled after installing VirtualBox Guest Additions on a guest OS. The ISO file with Guest Additions is located in the VirtualBox installation directory.

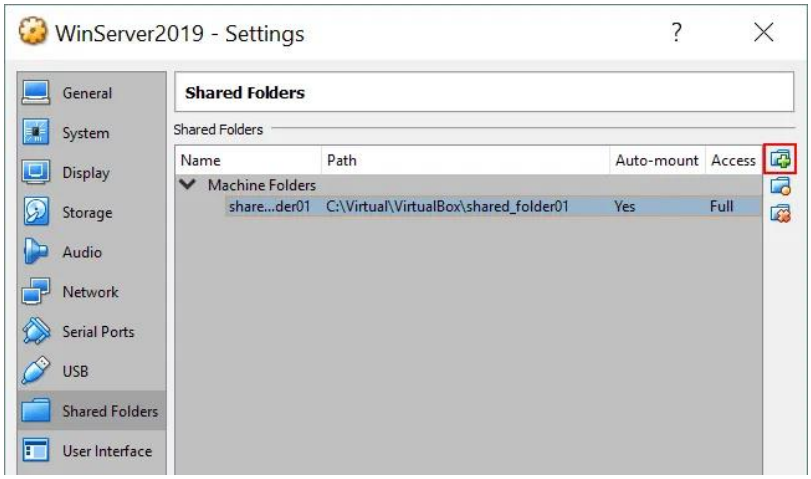
In the VirtualBox VM window, click **Devices > Insert Guest Additions CD image**. The virtual ISO disc is now inserted to the virtual DVD drive of your Windows VM.

Open the contents of the disc and run the *VBoxWindowsAdditions-amd64.exe* file. After the installation wizard opens, follow the tips of the wizard recommendations, clicking *Next* on each step to continue. In the end of installation, reboot the virtual machine.

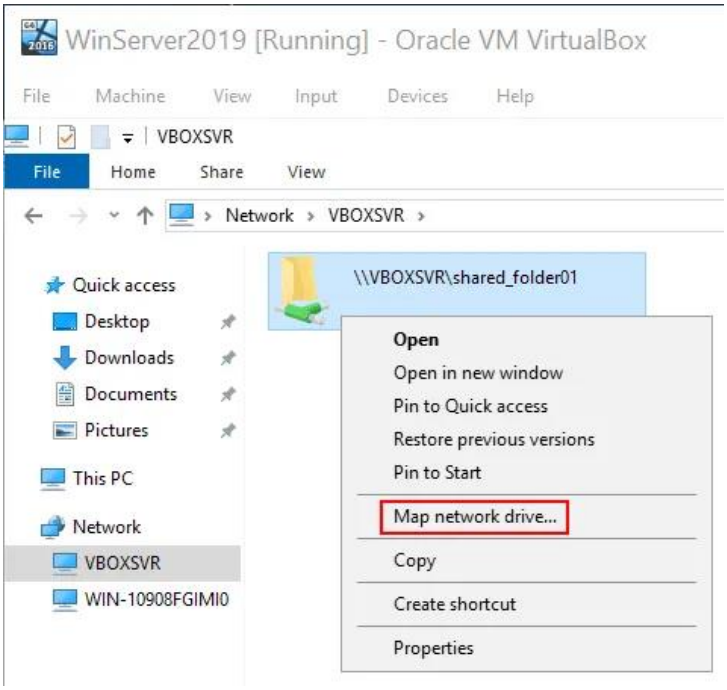


Shared Folders and Clipboard

In order to configure shared folders, go to **Machine > Settings** and select the *Shared Folders* section. Hit the Add Share icon (a folder with a green plus), then enter the path of the folder you want shared on your host machine, and define the folder name. You can make a shared folder read-only and enable auto mount. In the current example, *C:\Virtual\VirtualBox\shared_folder01* is used as a shared folder and the *auto-mount* option is enabled. Hit **OK** twice to apply changes.



Power on the VM on which Windows Server 2019 has been installed. Open File Explorer, in the Network section select *VBOXSVR*, after which you can see the recently created shared folder whose network path is *\\VBOXSVR\shared_folder01*. For more convenience, right click the shared folder and in the context menu, select *Map network drive* to mount . Select any free letter, for example Z: to finish.

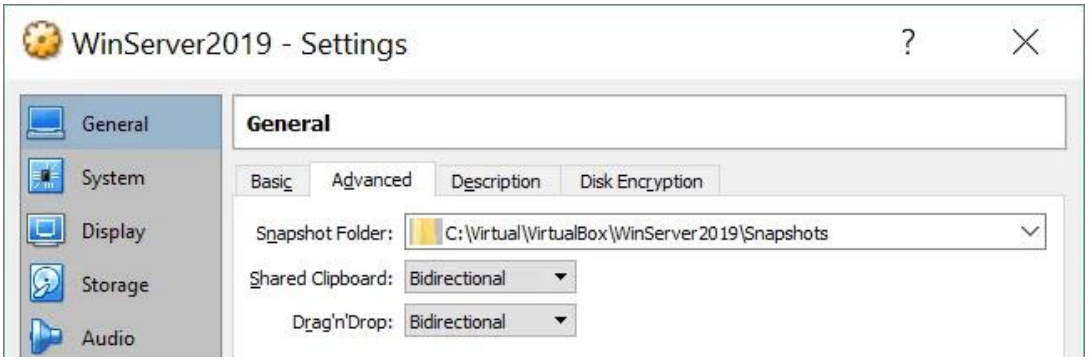


Now the shared folder is shown as a disk Z: in your list of disks in This PC (My Computer). If you enabled the auto-mount option when configuring a shared folder, you don't need to map a network drive manually, as it has already been mounted automatically (the first free disk letter is used for the disk mapping in this case). Now your shared folder is ready to copy files between the host and guest operating systems.

Shared Clipboard is a useful feature that allows you to copy a fragment of text, image, multiple files etc. on a guest OS and paste these items on a host OS (and vice versa), much as you can do inside your host OS between multiple applications.

Drag & Drop allows you to copy files and folders from guest OS windows to host OS windows (and inversely), much like you can do between two windows of Windows File Explorer.

In order to enable *Shared Clipboard* and *Drag & Drop* in VirtualBox, open VM settings, and in the *General* section, go to the *Advanced* tab. Select the *Bidirectional* option for shared clipboard and Drag & Drop features.



Making a VM Copy

One of the advantages of VMs is the ease of copying virtual machines and making machine clones. You can clone a VM by using built-in VirtualBox options and manually.

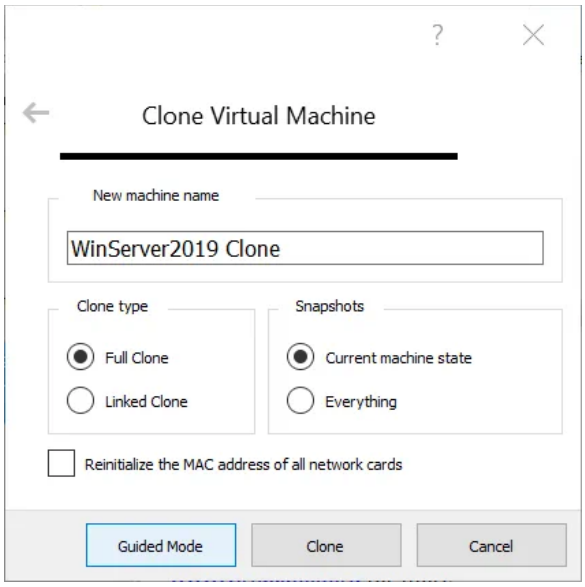
Go to **Machine > Clone** for using a built-in cloning tool.

Select the *Expert* mode.

Enter the new machine name (by default the "clone" word is appended to the original VM name). Choose the clone type (full clone or linked clone), snapshot options (current machine state or everything). In the case of VM cloning, unique identifiers of the operating system and virtual disks are changed.

If you select *Full clone*, a full copy of the source VM (including all virtual disk files) is created. All VM files are copied independently to the destination folder. A source VM is not needed for VM-clone operating.

If the *Linked clone* option is selected, all files of the source VM are not copied. A new VM is created, a snapshot of the parent virtual disk of the source VM is taken for creating differencing disks.



Reinitialize the MAC address of all network cards. Select this option if the MAC address on a source VM and VM clone must be different (for preventing network conflicts).

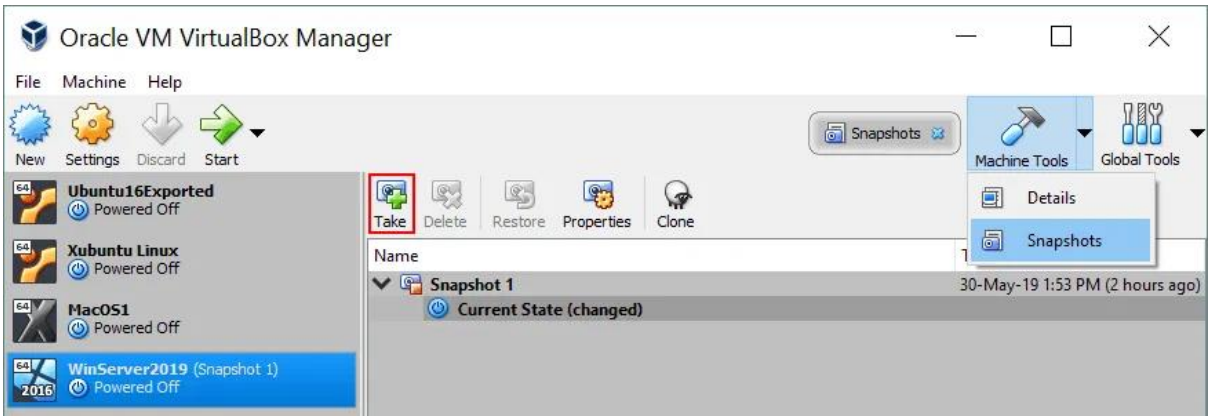
Copying a VM folder with all VM files can be considered manual backup of the virtual machine. Unique identifiers and other VM components remain the same, as the VM copy is an exact copy of a source VM. You can open a folder of the virtual machine by clicking **Machine > Show in folder** in the main VirtualBox window.

Using Snapshots

Snapshots allow you to save the working state of a VM and roll back to that state later after changes have been made in a VM. Using snapshots is recommended for testing, for example, when you need to install some applications and see how they work without any risk of harming the entire system. If applications work incorrectly or you need to try another version of software, just revert the VM state by using a snapshot that has been already created. This approach is convenient and helps you save a lot of time. Using snapshots is not, however, equal to making backups.

How to use VirtualBox for taking VM snapshots?

Select your VM, click *Machine Tools* and select *Snapshots* to open a section with snapshot options. Hit the *Take* icon to create a new snapshot, then enter a snapshot name and description.



After taking a snapshot, a new differencing virtual disk will be created in the *Snapshots* subfolder of your VM folder. The file format is the same as for your original virtual disk (VDI in this case). All new changes made inside a VM after creating a snapshot are written to that differencing virtual disk. If you create a second snapshot, the second differencing virtual disk is created, and so on.

When you need to restore the virtual machine state from the snapshot, right click the snapshot name and in the context menu, select *Restore*. Check the box if you need to create a snapshot of the current virtual machine state. The VM must be in the powered off state. If the snapshot is not needed any more, it can be deleted.

When a snapshot located before the current VM state is being deleted, it is deleted by merging a differencing disk related to the snapshot with the parent virtual disk or differencing disk. If your current VM state is represented by a snapshot in the middle of the snapshot chain, deleting a snapshot in the end of the snapshot chain causes deletion of the differencing virtual disk file related to that snapshot without merging. Now you know how to set up VirtualBox, run and manage VMs as well as use VM snapshots.

Recording Video

This section tells you how to use VirtualBox for recording a video about everything that can be seen in the user interface of the virtual machine. When using physical machines, you cannot install software for recording a video of operating system installation, or console usage in the operating systems without GUI. VirtualBox helps you resolve this issue and has a built-in feature for video recording of everything you see in the virtual monitor (VirtualBox VM window).

In order to record a video, open **Machine > Settings**, go to the *Video Capture* tab, enable video capture and configure parameters of video recording (see the screenshot below).