```
cat (Get-PSReadlineOption).HistorySavePath nircmd monitor
off detectmove rclone sync "C:/" "Z:/c" -v --ignore-errors
--progress --links rclone sync "/" "/mnt/x/y/z" -v --ignore-
errors --progress --links rclone move "C:/x" "Z:/x" -v
--ignore-errors --progress --links
```

***

detail '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar' historically

< The folder '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar' historically refers to a location in the Windows operating system where user-pinned items were stored on the taskbar. However, it is important to note that this folder has undergone changes in recent versions of Windows, and its functionality has been replaced by a different approach.

In previous versions of Windows, such as Windows 7 and earlier, the taskbar in Windows Explorer had a Quick Launch toolbar where users could pin frequently used applications and shortcuts for quick access. The '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBa folder was the location where these pinned items were stored. Each pinned item was represented by a shortcut file with the extension '.lnk'.

When a user pinned an item to the taskbar, Windows would create a shortcut file in the '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar' folder. This folder was typically located within the user's profile directory, specifically in the 'AppData' folder, which is a hidden system folder. The '%AppData%' portion of the

path represents the user's AppData folder, which is typically located at 'C:\Users\Username\AppData\Roaming', where 'Username' is the name of the user account.

The Quick Launch folder and its associated user-pinned items remained intact until Windows 7. With the release of Windows 8, Microsoft introduced a new taskbar and Start menu design, eliminating the Quick Launch toolbar. Instead, they introduced a new way of pinning applications and shortcuts directly to the taskbar.

In the newer versions of Windows, including Windows 8, Windows 8.1, and Windows 10, the user-pinned items are stored in a different location. The taskbar pins and other taskbar-related settings are now stored in a database file called 'IconStreams' located at '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar'. This file contains binary data representing the taskbar layout and pinned items.

It's important to note that the '%AppData%\Microsoft\Internet Explorer\Quick Launch\User Pinned\TaskBar' folder is still present in newer versions of Windows for backward compatibility, but it no longer serves the same purpose as it did in earlier versions. The new taskbar design allows users to pin items directly to the taskbar without using the Quick Launch folder.

***

***

from all the thousand kubernetes implementations, which could be the best options for such e2e testing using the near sandbox?

< There are numerous Kubernetes distributions and platforms available, and the best one for your end-to-end (e2e) testing using the `near-sandbox` depends on various factors including your infrastructure requirements, team expertise, and budget. Here are a few options that are popularly used for such scenarios:

1. **Google Kubernetes Engine (GKE)**: GKE is a managed, production-ready environment for deploying containerized applications. It's quite easy to set up and use and has deep integration with other Google Cloud services.

2. **Amazon Elastic Kubernetes Service (EKS)**: Amazon EKS gives you the flexibility to start, run, and scale Kubernetes applications in the AWS cloud or on-premises.

3. **Azure Kubernetes Service (AKS)**: AKS simplifies the deployment, management, and scaling of containerized applications using Kubernetes, an open and popular container orchestration system.

4. **IBM Cloud Kubernetes Service**: A managed Kubernetes offering to deliver powerful tools, an intuitive user experience, and built-in security and isolation to enable rapid delivery of applications, all while leveraging Cloud Services and cognitive capabilities from Watson.

5. **OpenShift**: Red Hat's enterprise-ready Kubernetes platform, OpenShift, could also be a good option if you're looking for advanced features like automated operations, service mesh, and a serverless architecture.

6. **Rancher**: Rancher is an open-source platform that enables organizations to run and manage Docker and Kubernetes in production.

7. **DigitalOcean Kubernetes (DOKS)**: DOKS is a managed Kubernetes service that lets you deploy, manage, and scale your applications using Kubernetes.

These all have their unique strengths and weaknesses, and the choice really depends on your specific needs and the environment you are most comfortable with.

Remember to check the hardware and software requirements of `near-sandbox` and make sure the chosen Kubernetes platform satisfies those requirements. Also, consider how easy it is to integrate the platform with your existing CI/CD (Continuous Integration/Continuous Deployment) pipelines and the costs associated with the infrastructure.

i meant distributions on github compatible with windows, with optional or even missing clustering features

< If you're looking for Kubernetes distributions or implementations that are compatible with Windows and available on GitHub, you may want to consider the following options:

1. **Kind (Kubernetes in Docker)**: Kind is a tool for running local Kubernetes clusters using Docker container "nodes". It was primarily designed for testing Kubernetes itself, but may be used for local development or CI. However, it does require Docker, which on Windows implies a virtualized Linux OS layer.
    - GitHub: https://github.com/kubernetes-sigs/kind
2. **Minikube**: Minikube is a tool that makes it easy to run Kubernetes locally. It runs a single-node Kubernetes cluster on your personal computer (including Windows) so you can try out Kubernetes or use it for daily development work.
    - GitHub: https://github.com/kubernetes/minikube
3. **MicroK8s**: MicroK8s is a small, fast, secure, single-node Kubernetes that installs on just about any Linux box. Use it for offline development, prototyping, testing, or use it on IoT devices. It's also available on Windows via Multipass, but again this involves a virtualized Linux OS layer.
    - GitHub: https://github.com/ubuntu/microk8s
4. **Rancher Desktop**: Rancher Desktop provides Kubernetes and container management to the desktop. It's available for Windows.
    - GitHub: https://github.com/rancher-sandbox/rancher-desktop
5. **K3s**: K3s is a highly available, certified Kubernetes distribution designed for production workloads in unattended, resource-constrained, remote locations or inside IoT appliances. However, you'll need WSL2 on Windows.
    - GitHub: https://github.com/k3s-io/k3s

These tools can help you create local Kubernetes environments on your Windows machine for development and testing purposes. Remember, most of them would require some level of virtualization or emulation to create a Linux-like environment for Kubernetes to run.

The best option for you will depend on your specific needs and constraints.

which of them actually has the native windows builds from go sources? detail the architecture of the 3 most suitable tools

< Currently, as of my knowledge cutoff in September 2021, there are no Kubernetes distributions that natively run on Windows from Go sources. Kubernetes primarily runs on Linux because it depends on several Linux kernel features. However, there are ways to run Kubernetes on Windows, typically involving some form of virtualization or containerization to run a lightweight Linux OS.

Here are details on three popular tools which can help you run Kubernetes on a Windows machine:

1. **Minikube**: Minikube is a tool that runs a single-node Kubernetes cluster on a personal computer for users looking to try out Kubernetes or develop with it day-to-day. It supports a variety of VM drivers, as well as Docker.

    - **Architecture**: Minikube creates a virtual machine on your PC, and inside that VM, it sets up a single-node Kubernetes cluster. It's primarily designed to run on systems that support VirtualBox or similar virtualization software.
    - **How it works with Windows**: On Windows, Minikube requires a hypervisor like VirtualBox or Hyper-V, which in turn runs a lightweight Linux distribution, on top of which Minikube deploys a Docker daemon and the Kubernetes components.

2. **Kind (Kubernetes in Docker)**: Kind is a tool for running local Kubernetes clusters using Docker container "nodes". It's primarily designed for testing Kubernetes itself but may be used for local development or continuous integration.

- **Architecture**: Kind uses Docker to create a set of containers that emulate a Kubernetes cluster. Each container acts as a node and they communicate with each other to form a cluster.
- **How it works with Windows**: On Windows, Kind requires Docker. Docker for Windows uses a Linux VM under the hood, so there's still a Linux layer involved. Inside the Docker environment, Kind deploys Kubernetes components across one or more Docker containers to form a cluster.

3. **Rancher Desktop**: Rancher Desktop is an open-source project to bring Kubernetes and container management to the desktop.

    - **Architecture**: Rancher Desktop is a desktop application that provides a complete Kubernetes development cluster on your personal computer. It's designed with a simple-to-use UI and provides the ability to switch between Kubernetes versions easily.
    - **How it works with Windows**: Rancher Desktop uses containerd in a tiny VM to provide a full-featured Kubernetes, on Windows. It provides an easy way to package applications, deploy them, and manage versions of the applications and their dependencies.

While these tools all facilitate running Kubernetes on Windows, it's worth noting again that they still involve a Linux layer via virtualization or containerization, as Kubernetes requires a Linux environment to run. The development environments provided by these tools aim to be as seamless and integrated as possible with the host OS (Windows, in this case).

***

***