Modern Apps and the Rise of the Cloud-Native Developer Platform

CHRIS MUNFORD
CEO/FOUNDER OF NETHOPPER

nethopper.io

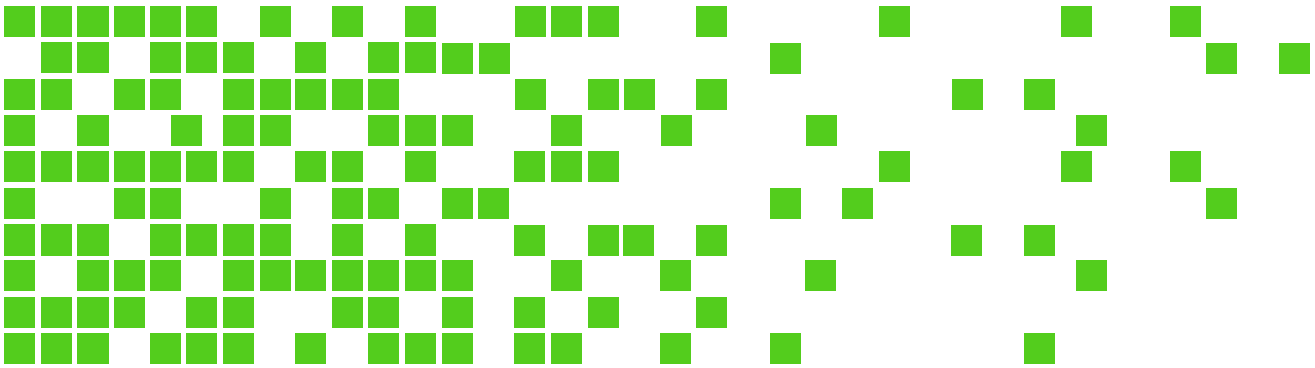
I'm the founder and CEO of Nethopper Inc, a pioneer in KAOPS, a cloud native Kubernetes application operations platform engineering/IDP framework, delivered as a SaaS.

I have an electrical engineer background and have held leadership roles in engineering, product management, solution architecture and sales at 5 Boston and Silicon Valley based technology startups over the last 25 years. Past Technology segments include datacom, telecom, optical transport, wireless backhaul, mobile OTT video, and software composable infrastructure. The companies I worked for were acquired by Alcatel, Ciena, Ericsson, and Twitter.

I started Nethopper in 2021 to make it easier to manage and operate Kubernetes across clouds and clusters. We've evolved KAOPS into an easy to use GitOps-based integrated platform framework with a neutral approach to Kubernetes and cloud providers. Powered by Nethopper, KAOPS helps Managed Services Providers (MSPs) and enterprises reduce the time it takes to build a production-ready Internal Developer Platform (IDP) and accelerate their modern app delivery.

Chris Munford
CEO/Founder of Nethopper

Contents



Digital Transformation and Modern Software Applications

Digital Transformation is such a broad term. It implies that Enterprise businesses are naturally and effortlessly going to blossom into a well run digitally automated machine.

The harsh reality is that Digital Transformation is unnatural, uncomfortable, and most often unsuccessful.

Digital Transformation is undeniably important, as it will likely determine the success or failure of a business. Businesses that successfully transform have something in common; they become very efficient at producing modern software applications, often 1000's of them.

These applications can be used both internally and externally. Internal applications increase productivity and are the tools that make the company run. External applications attract, retain, support and charge customers.

Although they don't get the same press as Data and AI, modern software applications are arguably more important to business than Data and AI.

For example, a business may have all the right data to make great data-driven decisions, but a software application is required to retrieve, correlate, analyze, and display that data to the decision maker.

Think about one instance, such as Artificial Intelligence (especially GenAI and LLMs), which is often praised as being the future of work, and promises to make us all more productive. However, humans can't interface to graphics processors and LLMs. AI can not be experienced by humans without a software application.

For instance, GPT would be of little value without OpenAI.com and other websites that let you interact with it. Those websites and AI tools are modern software applications.

Neither Data nor AI are effective without modern software applications.

Modern Software Applications (vs. Traditional)

What is a Modern Software Application, and how does it differ from traditional software applications?

Modern software applications represent a paradigm shift from traditional counterparts, reflecting advancements in technology, development methodologies, and user expectations.

Unlike their predecessors, modern applications are often built with a focus on cloud computing, microservices architecture, and user-centric design.

- Cloud computing enables applications to leverage remote servers for storage, processing, and scalability. This promotes flexibility, accessibility, and cost-effectiveness.
- Microservices architecture breaks down applications into smaller, independent services, fostering agility, ease of maintenance, and scalability.
- User-centric design emphasizes intuitive interfaces, seamless user experiences, and responsiveness across various devices.

Additionally, modern software applications often embrace DevOps practices, promoting collaboration between development and operations teams, facilitating faster releases and continuous integration.

In contrast, traditional software applications were typically monolithic, residing on local servers, with development cycles characterized by lengthy release cycles and limited user interactivity. Upgrades and maintenance were cumbersome, and scalability was a constant challenge.

Modern apps are built with a focus on cloud computing, microservices architecture, and user-centric design.

The Challenge of Modern App Development

While the velocity and value of a modern app can be far greater than a traditional one, making a modern software application is not easy.

of attempts to modernize apps fail.

With so much riding on the success of these applications, why do these efforts fail so often?

Some of the common reasons for failure include:

- Lack of planning, requirements, and expectations
- Inadequate team communication and collaboration
- Complexity of new technologies and processes
- Insufficient skills

Instead of diving into each of these categories, let's discuss what they all have in common. **They all increase cognitive load on the developer.**

Cognitive load leads to decreased productivity, project delays, budget overruns, and eventually project failures.

Developers have always had a high cognitive load, having to be experts in software programming languages, project requirements, system design methodology, etc.

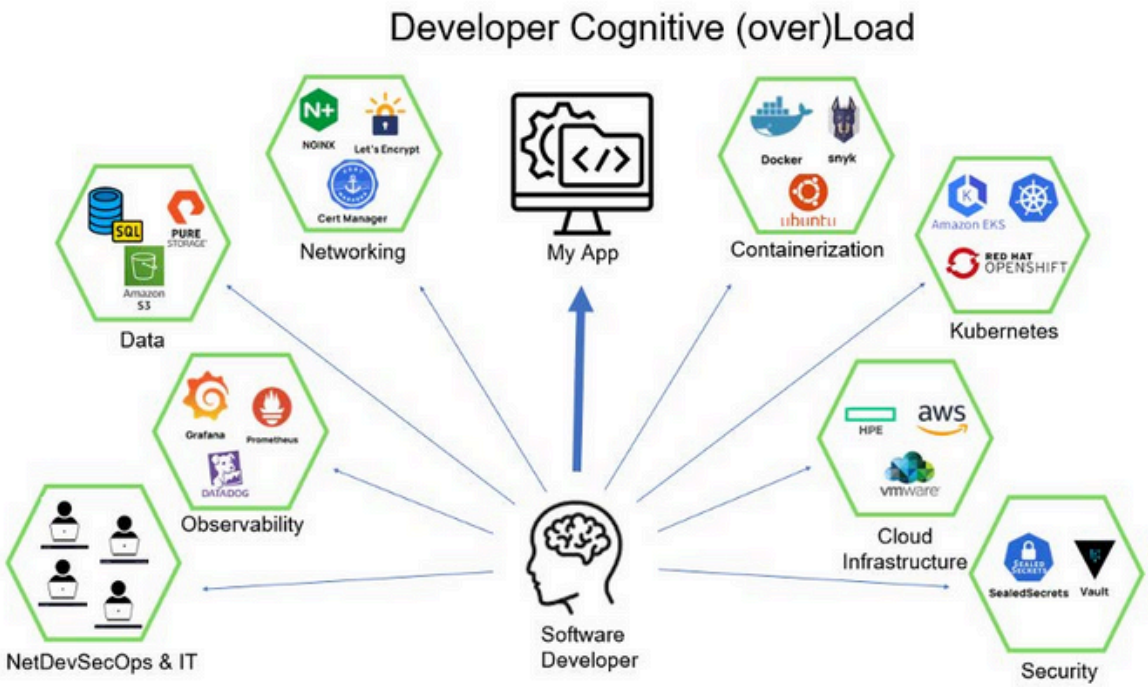
However, there are many new sources of cognitive load introduced by cloud and open-source software, causing developers to ***also*** learn and worry about:

- New and changing cloud infrastructure
- Increased cloud-induced security measures
- Microservices interactions with other developers
- Dependencies on cross-functional teams: DevOps, SecOps, SREs, IT, Cloud Engineers
- Creating and maintaining software containers (i.e., Docker)
- Deploying, observing, and upgrading software containers in Kubernetes
- Having to master 1000's of open-source tools.

1.Why App Modernization Projects Fail

Consider the diagram below.

The developer should only have to focus on their application (My App). Instead, all the other sources of cognitive load are distracting the developer from My App.



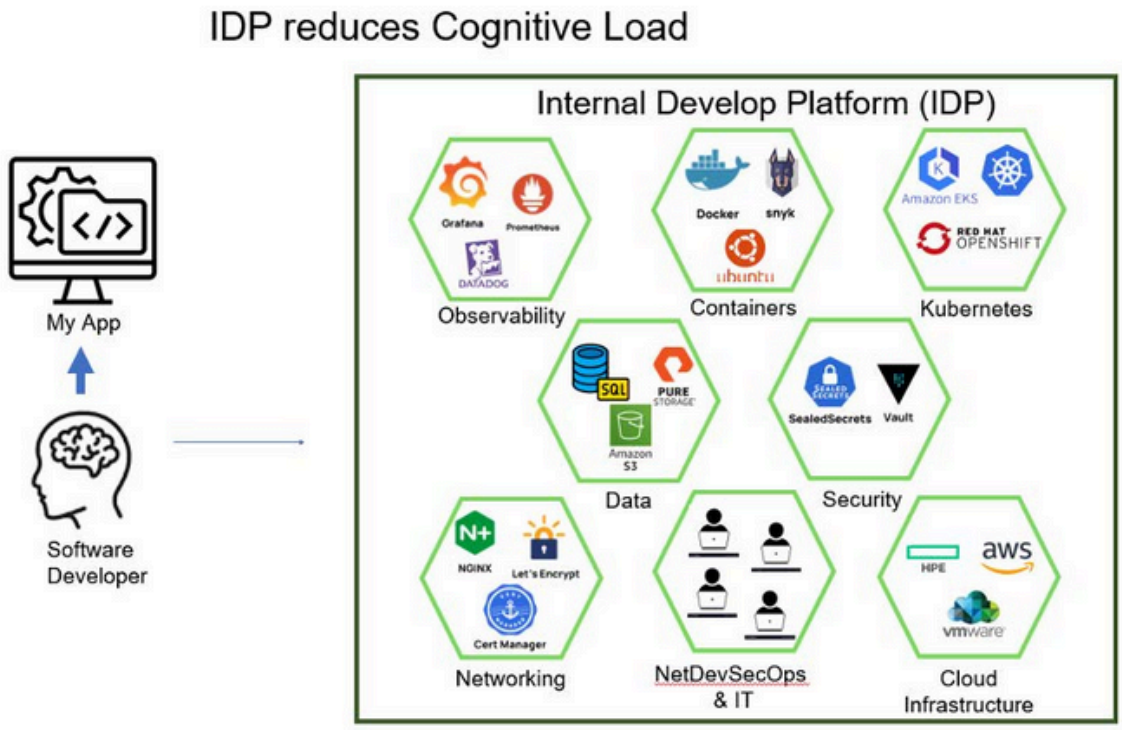
These new sources of people, process, and tools create cognitive overload for developers. For enterprises that can afford it, the natural thing to do is to hire more staff (developers, operators, cloud and IT folks). However, this often has the opposite of the desired effect. It increases the number and type of interfaces a developer has to maintain and the processes they have to follow, further increasing cognitive load on the developer.

New sources of people, process, and tools create cognitive overload for developers.

The Rise of the Internal Developer Platform (IDP)

Many enterprises are reducing developer cognitive load and increasing productivity with the introduction of an Internal Developer Platform.

Consider the diagram below showing the developer reduced cognitive load using an Internal Developer Platform:



"An Internal Developer Platform (IDP) is the sum of all the tech and tools that a platform engineering team binds together to pave golden paths for developers. IDPs lower cognitive load across the engineering organization and enable developer self-service, without abstracting away context from developers or making the underlying tech inaccessible."

Kaspar von Grünberg, CEO, Humanitec

Platform engineers designed the IDP so the developer only has to maintain one simple interface to the IDP, allowing the developer to focus on what matters, developing new application features, faster. This accelerates the return on investment to the business, which increases the value of the application, and the chances for successful business outcomes and digital transformation.

Well designed IDPs follow a Platform as a Product approach, where a platform team builds, maintains and continuously improves the IDP, following product management principles and best practices.

An IDP reduces developer cognitive load and increases productivity.

What is a Cloud Native IDP?

Platform engineers must develop their IDPs quickly, because developers can't deliver applications until the IDP is ready.

Building an IDP from scratch takes a team of platform engineers several years, so almost all platform engineers leverage open-source software (OSS) tools.

Many OSS tools have been proven to successfully work at large software organizations, like Google, Intuit, Netflix and others. Therefore, it is a reasonable assumption that the OSS tools will meet the platform engineers needs.

The #1 source of open-source tools for the cloud is the CNCF (Cloud Native Computing Foundation) with over 150 OSS projects. However, it is estimated that there are over 50 million OSS projects globally, so the CNCF is certainly not the only source.

Why are there so many OSS tools? Isn't there one OSS tool that a platform engineer can just download and use as an IDP?

The answer is no, an IDP requires a collection of integrated OSS tools. The reason is a time-tested software development principle called 'modularity.'

Modularity is the concept of building a small program that does just one thing very well, and can be leveraged, like a building block, in many other projects.

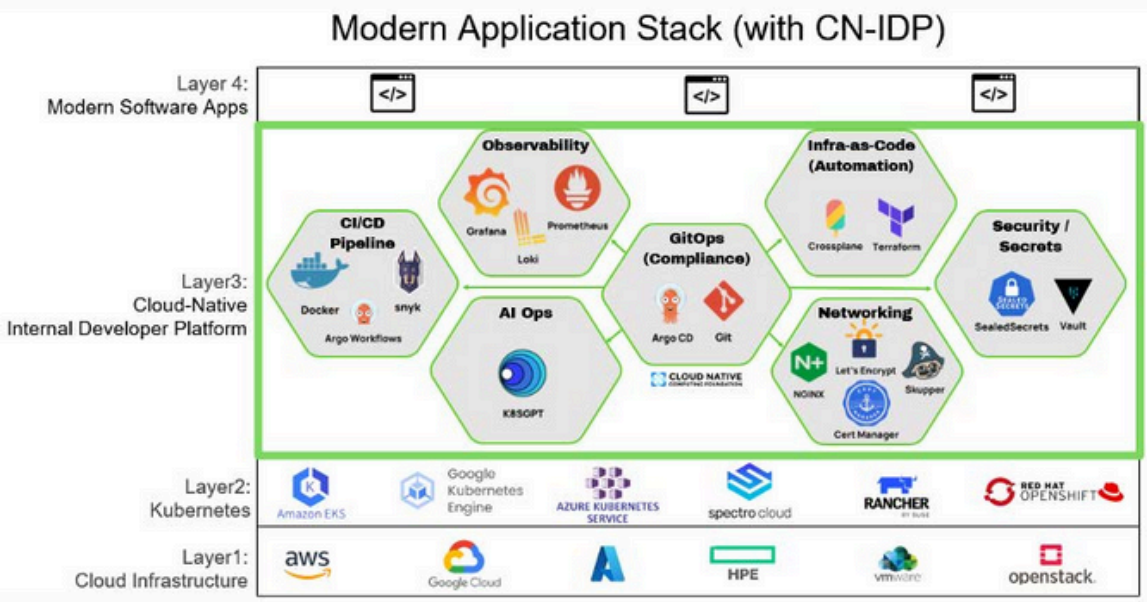
Almost all OSS tools follow this principle, and therefore one single tool cannot stand alone as a complete IDP. As most IDPs are made of 10 or more OSS tools, OSS integration becomes an area of focus for platform engineers.

What is a Cloud Native IDP?

A cloud native IDP is built using cloud native open-source tools. Because all the best OSS tools are containerized, the IDP is best run and integrated in a Kubernetes environment.

Since Kubernetes is purpose built for container communication and API integration, it is an ideal home for creating an IDP control plane.

Consider the diagram in the next page, which shows how a cloud native IDP fits into a modern application software stack.



A modern application software stack is comprised of:

- **Layer1: Infrastructure.** The foundation layer is cloud infrastructure. compute, storage, and networking. IDP users should be able to choose any public or private cloud they like, or multiple.

- **Layer2: Kubernetes.** It is often easiest to choose the Kubernetes type that your cloud provider offers or recommends. However, IDP users should be able to use any Kubernetes or multiple.

- **Layer3: IDP (Internal Developer Platform)**. Because cloud native tools are containerized, they run on top of Kubernetes. Platform engineers can decide on which tools they choose, based on the needs of the application team. Good IDPs are multi-faceted, and incorporate tools for Deployment, GitOps, Observability, Networking, Security, CI/CD software releases, Upgrades, and even AI Operations.

- **Layer4: The modern business applications (and microservices) that Developers are creating.** Once Layer 1, 2, and 3 are in place, these applications are ready to run. App developers can simply deploy and upgrade their applications, without any knowledge (or cognitive load) of the bottom 3 layers of the stack. Just as importantly, operators should be able to manage those applications without the help of the developers.

A cloud native IDP is an IDP built using cloud native open-source tools.

The Business Benefits of a Cloud Native IDP

A cloud native IDP reduces cognitive load on application developers and makes them more productive, resulting in the following business outcomes:

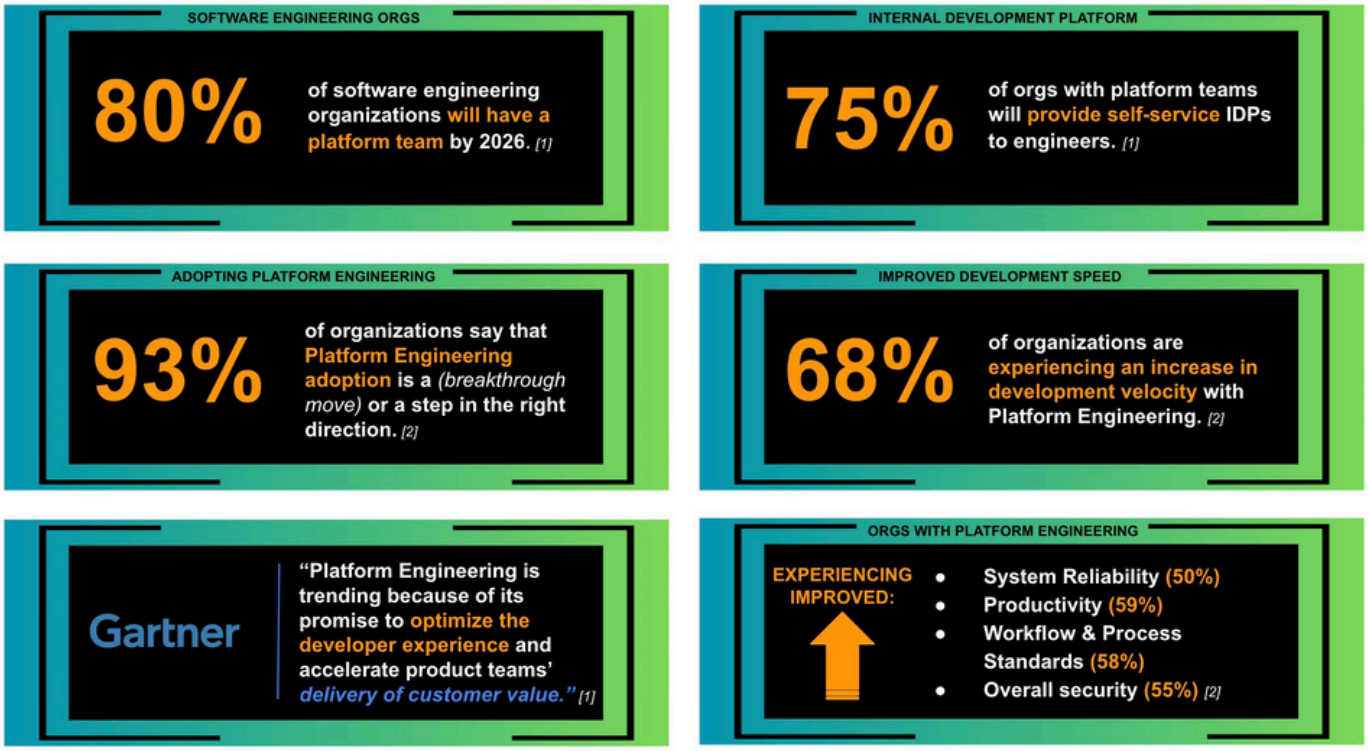
- Accelerated day-1 modern application delivery
- Simplified application operations, including day-2 upgrades
- Reduced operational burden/staff requirements
- Improved application availability and resilience
- Centralized management for multiple applications, clouds, and teams
- Increased application scale and performance
- Cloud provider portability and migration to avoid lock-ins
- Support for hybrid, multi, and edge cloud use cases
- Faster onboarding of new developers and operators
- Leverage lower skilled developers and operators
- Increased auditability and governance
- Improved security posture
- Reduced impact of key employee loss/churn
- Repeatable and automated production systems (Disaster Recovery)

A cloud native IDP can accelerate application delivery and simplify day-2 operations.

Conclusion

The Rise of the IDP

It's apparent why cloud native IDPs are so important to the success of modern applications. Business and technology leaders are taking notice. That is why so many of them are adopting an IDP-centric approach. Just take a look at the results of (1) Puppet's The State of Platform Engineering Report and (2) Gartner's study of enterprise IDP and platform engineering adoption plans:



www.nethopper.io

info@nethopper.io

+1 (617) 819-8009

nethopper.io