
KubeCon
CloudNativeCon
North America 2023


KubeCon | CloudNativeCon

North America 2023

Harnessing Argo & Flux: The Quest to Scale Add-Ons Beyond 10k Clusters

Joaquin Rodriguez, Senior Software Engineer, Microsoft

Priyanka Ravi, Developer Experience Engineer, Weaveworks

Hello!

KubeCon CloudNativeCon North America 2023

Joaquin A. Rodriguez
Senior Software Engineer
Microsoft
𝕏 @joaquinrdzv

Priyanka “Pinky” Ravi
Developer Experience Engineer
Weaveworks
𝕏 @PinkyyRavi

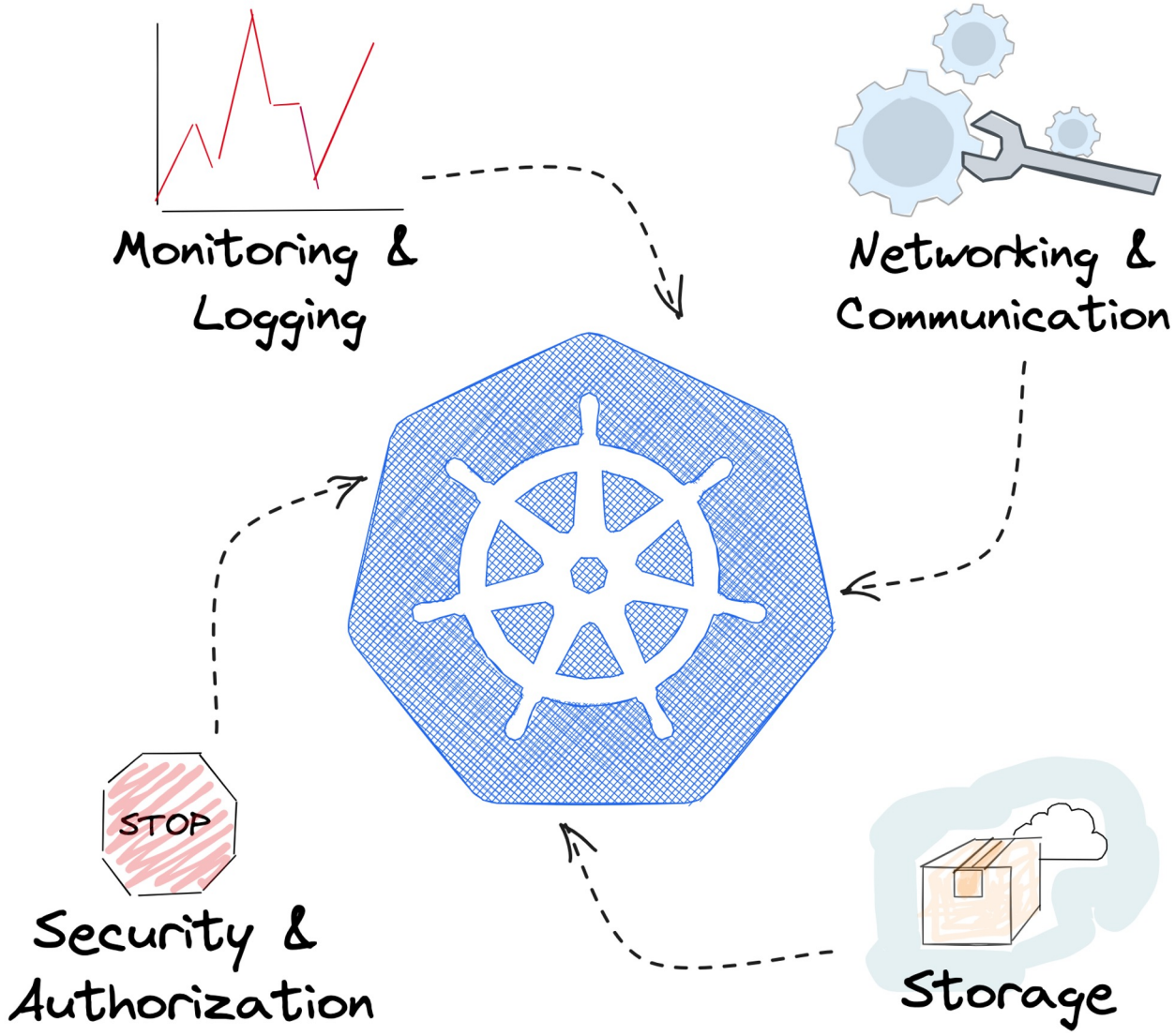
Agenda

- Overview and importance of Cluster Addons
- Challenges of Maintaining Cluster Addons
- GitOps: Argo, Flux, and Flamingo to the rescue
- Solution Diagram + Demo
- Scaling Options
- Key Takeaways

Cluster Addons

Cluster addons are tools, applications, or services that expand the functionality of Kubernetes clusters.

They are not part of the core Kubernetes system but are often required to provide essential capabilities.



Cluster Addons: Considerations & Lifecycle

Importance:

- **Extend Kubernetes:** Fill gaps left by core Kubernetes
- **Adaptability:** Adapt to variety of use-cases and deployments
- **Security:** Security-related add-ons can reinforce cluster security, ensuring adherence to best practices and compliance requirements.
- **Insights:** Monitoring and logging add-ons provide insights into cluster performance

Considerations:

- **Compatibility:** Match Kubernetes version & components
- **Maintenance:** Regular updates and oversight
- **Resource Overhead:** Balance benefits vs. CPU/memory costs

Lifecycle:

Install → Upgrade → Scale → Decommission

Cluster Addons: Challenges

Cluster Addons are awesome! however there are some challenges that follow:

- Maintain a fleet of clusters to ensure that they are operating efficiently
- Ensure a consistent configuration across a fleet when each cluster may have specific needs
- Manage add-on dependencies efficiently
- Visualize/interact with your operations/deployments
- Scaling is hard!

GitOps to the rescue!

- Operating model for cloud native applications such as Kubernetes
- Utilizes a version controlled system (Commonly Git) as the “single source of truth”
- Enables continuous delivery through automated deployment, monitoring, and management by a version controlled system
- Managing your infrastructure and applications declaratively

Argo

Application Dashboard

- Powerful, real-time UI dashboard provides a holistic view of your application and resources

Health Monitoring & Configuration Drift Detection

- Detect and get notified when applications become unhealthy or are “OutOfSync”

Multi-cluster / Multi-tenant

- Create sandboxes and establish guardrails across multiple clusters/namespaces using Projects

Advanced Deployment Patterns

- Support complex pipeline-like deploys using Pre/Post Sync hooks and Sync waves

Highly Extensibile

- Customize resource actions, integrate any config management tooling, and extend the UI

Integrates into your Environment

- REST/gRPC API and CLI enables seamless integration with tools

Flux

GitOps for apps & infrastructure

- Just push to Git and Flux does the rest! It's Declarative, Automated, and Auditable

Designed with security in mind

- Pull vs. Push, least amount of privileges, adherence to Kubernetes security policies and tight integration with security tools and best-practices. Read more about our security considerations.

Multi-cluster, Multi-tenancy…Multi-everything!!

- Flux can use one Kubernetes cluster to manage apps in either the same or other clusters, spin up additional clusters themselves, and manage clusters including lifecycle and fleets.

Works with any Kubernetes and all common tooling

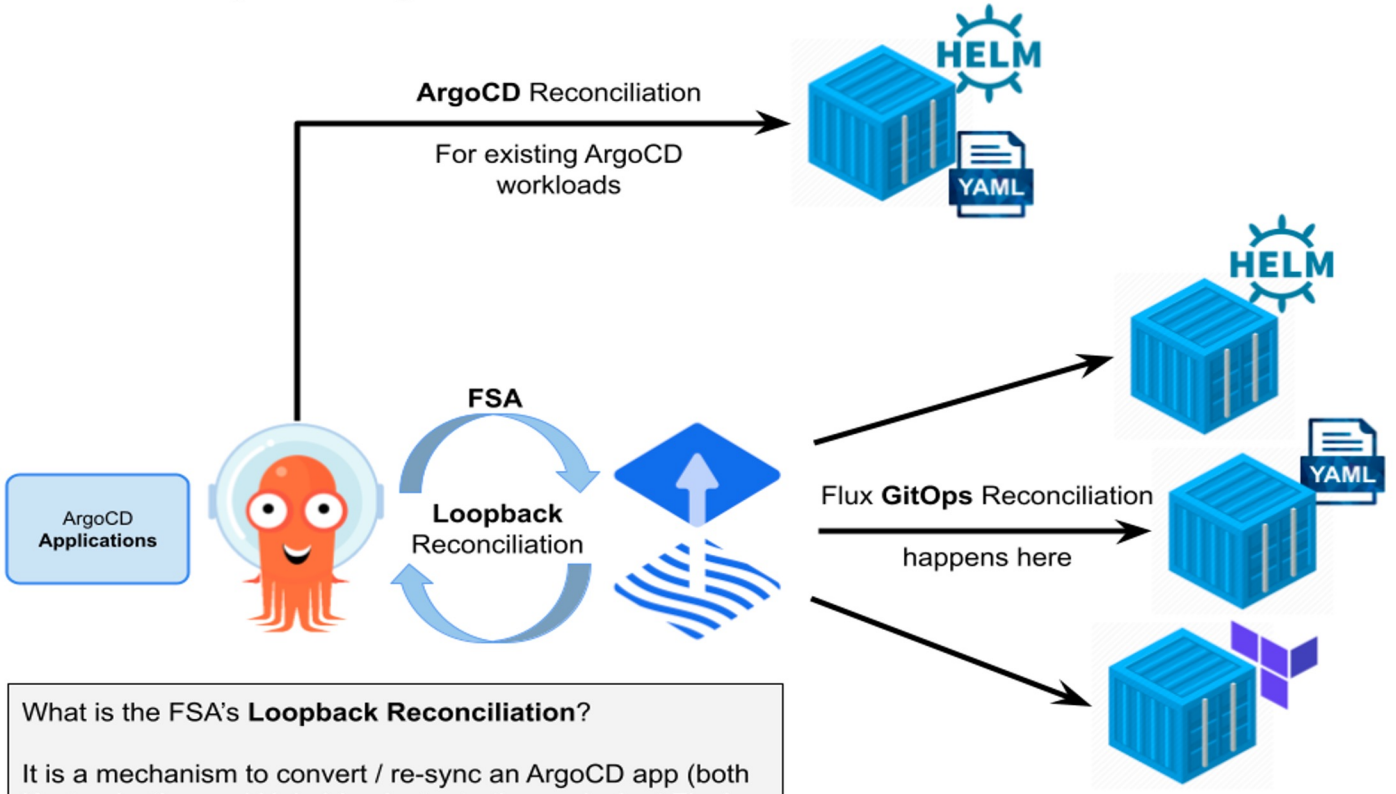
Dashboards love Flux

- No matter if you use one of the Flux UIs or a hosted cloud offering from your cloud vendor, Flux has a thriving ecosystem of integrations and products built on top of it and all have gr dashboards for you.

Flamingo

- Flamingo's container image can be used as a drop-in replacement for the equivalent ArgoCD version to visualize, and manage Flux workloads, alongside ArgoCD
- Flamingo on GitHub: **https://github.com/flux-subsystem-argo/flamingo**

FSA - the Flux System for Argo: How does it work?



What is the FSA's **Loopback Reconciliation**?

It is a mechanism to convert / re-sync an ArgoCD app (both Kustomization and Helm) in-cluster to the equivalent Flux's counterparts, so that ArgoCD does not conflict with Flux.

What are the use cases of **FSA**?

- You can use Flux as another GitOps backend for your system. The system won't break or need to stop while you're upgrading ArgoCD.
- When ArgoCD breaks your Helm, you can easily switch to use FSA without worrying about finding the workarounds. Flux has Helm SDK built-in.

Does FSA **stop** me from using normal ArgoCD applications?

No, you can use ArgoCD normally.

FSA is totally an optional.
You can use both ArgoCD and Flux together on a single system.

Why combine Argo and Flux?

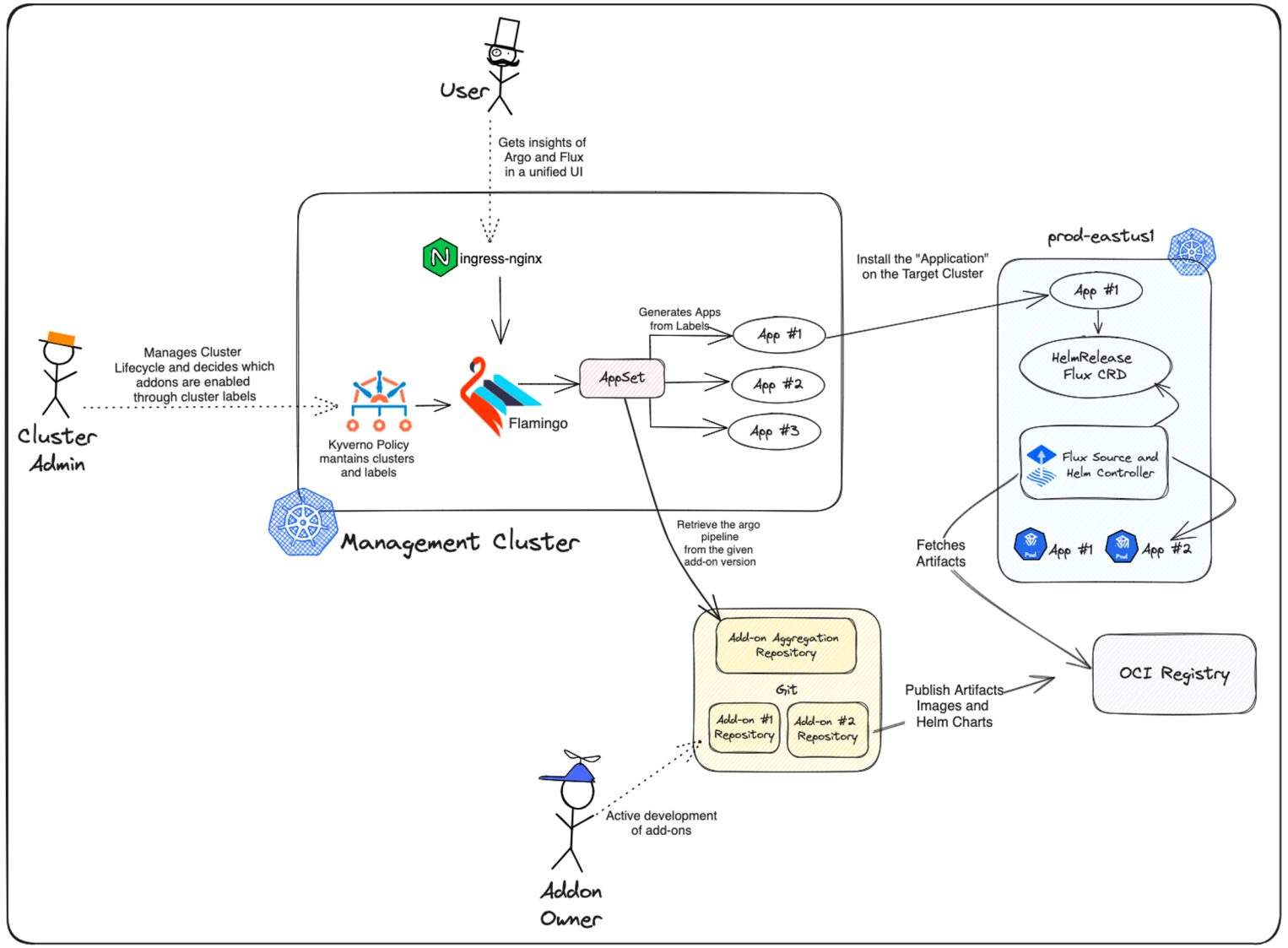
Argo

- UI, scalability, cluster management, centralized control plane, pre-sync and post-sync validation

Flux

- HelmRelease, dependsOn, rollback, upgrades, dynamic config helm values, helm hooks, retries

Solution Diagram

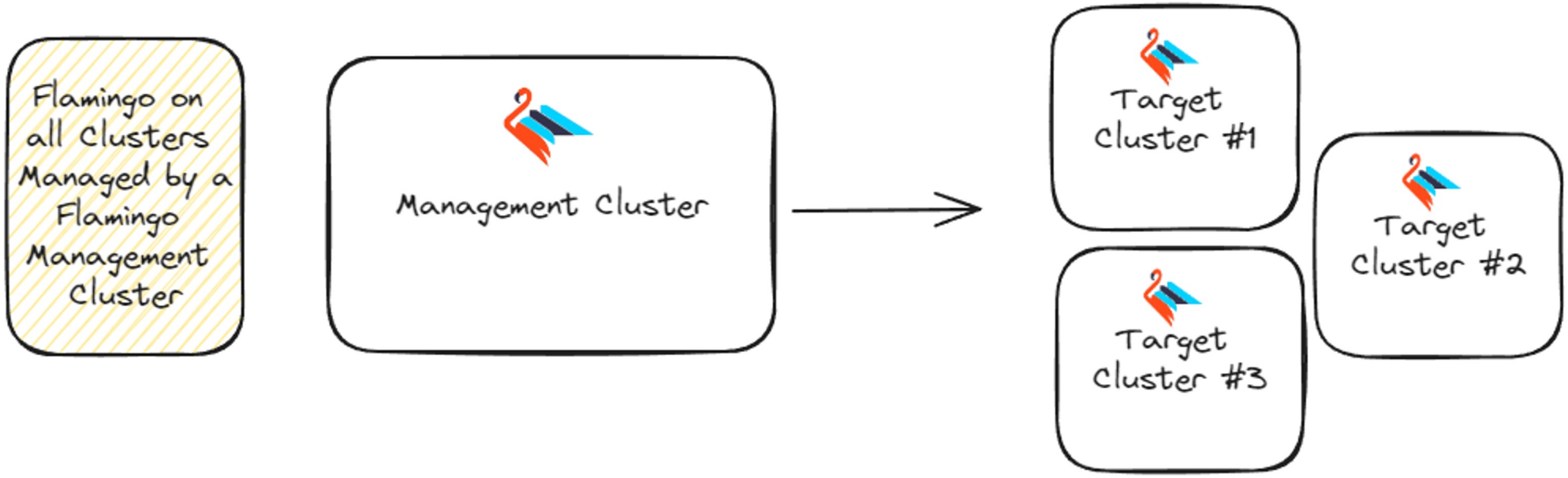


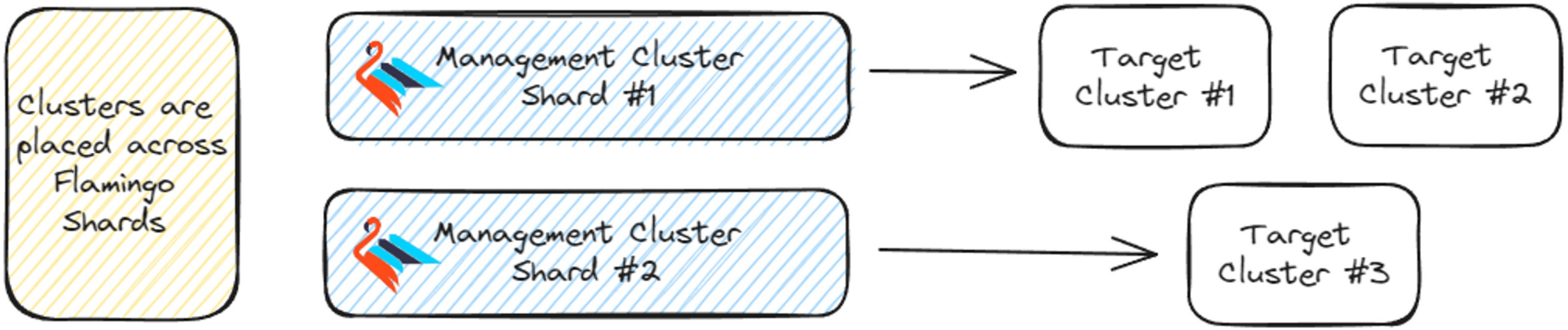

KubeCon | CloudNativeCon

North America 2023

Demo Time!

Scaling Options





Key Takeaways

- Cluster Addons are powerful components that enhance the Kubernetes Ecosystem
- Managing cluster addons at scale is hard
- With the right tools you can tame this complexity
- No “right” solution to solve all cases

Special Thanks

Thiebaud Ernstberger
Kuldip Madnani

Joaquin A. Rodriguez
Senior Software Engineer
Microsoft
@joaquinrdzv

Please scan the QR Code above to leave feedback on this session

Priyanka “Pinky” Ravi
Developer Experience Engineer
Weaveworks
@PinkyyRavi


KubeCon | CloudNativeCon

North America 2023

Thank You!