BEST PRACTICES FOR HYBRID CLOUD KUBERNETES WITH EKS AND WEAVE GITOPS

Executive Summary

The benefits of running containerized workloads on Kubernetes are now widely recognized across the industry. The wide reach of Kubernetes has also made it perhaps the first truly portable platform to deploy applications across public and private clouds as well as on premise and the edge.

As Kubernetes' reach in the enterprise extends, multiple challenges must be overcome to harvest the real value of a common, hybrid platform.

In this white paper, we will look at the various challenges teams will run into as their architecture grows and becomes hybrid – and look at the most relevant aspects of the design and operation of these hybrid Kubernetes environments. The goal is an easy-to-manage, easy-to-scale and truly portable platform, with which your development teams can deliver applications to where users need them most, while maintaining robust security posture, governance and compliance.

The reasons behind hybrid Kubernetes

Enterprises are faced with increasingly complex requirements related to where and how applications and workloads must be deployed.

The edge is rapidly enabling use cases where workloads must be deployed very close to users. Perpetually changing laws are exposing enterprises to regulations and policies around data and operational sovereignty that vary given the location of users and the nature of the data the application is operating on - all while looking to keep costs visible and under control.

The rise of 5G and the race to take advantage of the unique capabilities of this new mobile environment are also pushing organizations towards building software that relies on unique hardware capabilities that may not be available anywhere else but bare metal.

Whether you need to deploy software close to your users, need access to specialized hardware or are constrained by data or operational sovereignty regulations, picking a location and cloud provider is no longer a single choice question.

As the landscape of runtime environments for your various applications diversifies and grows, the need to provide development and operations teams with common workflows, tooling and development experience becomes ever more critical.

Containerization has been a key step in this direction. The immutable and self-contained nature of the technology did simplify the build and packaging of applications, with the promise of portability. But the container itself is not enough to guarantee portability across environments.

The next sore spot that had to be considered was the platform – and this is the gap that Kubernetes has come to fill. Kubernetes has become the de-facto platform for delivering and running containerized workloads, regardless of where you need to run them. Kubernetes distributions are available to run upstream, compliant Kubernetes basically anywhere you can think of, regardless of form factor, architecture or location. Indeed, it is the only managed container platform available across all public clouds.

Clearly, hybrid Kubernetes is something enterprises must rapidly become good at.

But hybrid Kubernetes is not without challenges

Kubernetes is certainly a solid choice for any cloud and environment-agnostic container platform. Yet although it provides a consistent API, there are challenges that must be well understood, in order to leverage Kubernetes as common infrastructure across hybrid environments for container orchestration and application portability.

Security and Compliance

Platform teams must ensure that the control planes as well as the workloads running across all clusters are secured and that they comply with policies, regulations and best practice.

Regulations will determine how and where data must be stored and transported, as well as constraints around access and operations of the infrastructure and workloads. Some industries such as finance, healthcare and real money gaming are more heavily regulated than others, however, the rapid increase in the number and complexity of attacks, as well as the growing volume of data organizations are storing for their users, is turning data and operational sovereignty into an issue that most IT teams will run into.

“Operational sovereignty: Assurance that all people, processes and systems are local to the jurisdiction of the application jurisdiction for a given service or application”

This is particularly challenging in hybrid environments due to the heterogeneity of the underlying infrastructure, as well as the physical location of your various Kubernetes clusters, whether in the cloud or on-premise.

Day- 2 Operations and Lifecycle Management

Clusters require ongoing management, from handling node failures to patching and upgrades, as well as scaling requirements as more teams deploy new services onto them.

Day-2 Operations is a common pain point for enterprises managing a growing fleet of clusters. Keeping proper visibility, managing costs and reducing the effort required in operating that growing complexity is critical for successful operations.

If not effectively designed, the number of tools and mechanisms needed for operating clusters across different locations and clouds will rapidly become unmanageable.

Application Delivery and Portability

This landscape becomes ever more challenging to manage given the growing number of teams that are looking to deliver applications to this diverse range of runtime environments.

Development teams are looking for autonomy and consistency across environments, without having to integrate complex processes into their development lifecycle. As mentioned earlier, containerization has solved part of this problem, but to achieve application portability there are other angles that must also be considered.

If your delivery is not portable, your application is not portable either.

A great first step is to have containerized artifacts that are self-contained and use tools that have now become intuitive and integrated into most workflows for development teams. But if your delivery of those artifacts to environments is not portable, then you have greatly reduced your ability to deliver your workloads simply and reliably to a diverse range of target environments.

There are two basic sources for this challenge:

1. Imperative definition of deployment pipelines on a variable set of environment characteristics.
2. Inconsistent service availability across the various environments used by development teams.

We will discuss how declarative configuration and autonomous deployment solve the former, and how reusable, extensible declarations simplify the latter.

Autonomy with compliance

The solution to these challenges must consider a very important element: development teams need autonomy. As organizations continue to adopt and mature in their DevOps journey, it becomes an increasingly important requirement to give development teams the tools and visibility they need to succeed in owning their services from code to production.

Ownership and autonomy are key requirements to achieve many of the goals in a DevOps focused organization, from reduction of handoffs between teams (therefore accelerating the stream of value) to decreased lead time for changes.

“Lead time for changes: the time it takes for a new feature to go from code to production”

But this must be accomplished while guaranteeing compliance and security, concerns that are growing in complexity and are particularly relevant in hybrid environments, due to government regulations.

Building a manageable, portable hybrid architecture

We will now examine some key areas that are critical in overcoming and ideally avoiding these challenges altogether.

We will see how GitOps abstracts away the complexity of application delivery and cluster lifecycle management, while dramatically simplifying security and governance. We will also dive into the relevance of a consistent underlying Kubernetes distribution, and how EKS together with Weave GitOps provide a unique combination of technologies that build up on one another, to offer an easy-to-manage and high-performing hybrid infrastructure and development workflow.

It all starts with a Kubernetes distribution

Whether you are running a managed Kubernetes service offered by a cloud provider or Kubernetes clusters on premise, a Kubernetes distribution will be used under the hood.

There are five key aspects to evaluate when picking a managed Kubernetes service or choosing a Kubernetes distribution to provision clusters across hybrid locations:

1. Is the distribution tracking upstream Kubernetes and maintaining API compatibility?
2. Is it possible to run the same distribution across the various target environments in your architecture?
3. What is the security posture of the team maintaining the distribution?
4. How is the distribution supported?
5. What opinions are built into the distribution? Are they consistent with your current and future architecture?

In a hybrid environment where clusters may be operating across public and private clouds as well as on-premise and on the edge, choosing a Kubernetes distribution that is common across all environments will greatly reduce complexity down the road.

A common distribution will make sure new versions as well as patches are available across all your environments, and that the various services that comprise the cluster's control plane are compatible and can be commonly supported.

EKS as a common foundation for your hybrid Kubernetes Architecture

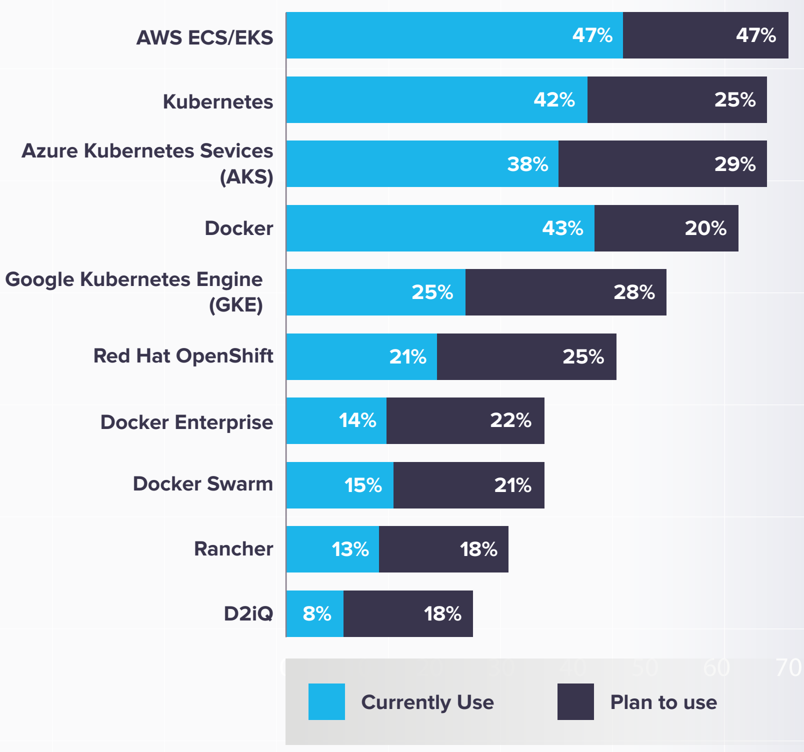
According to Flexera's 2022 State of the Cloud Report, AWS' EKS service remains the most widely used managed Kubernetes service, with 47% of respondents currently using it.

AWS' EKS service the most used

47%

of repondents

CONTAINERS TOOLS USED BY ALL RESPONDENTS



The experience acquired by servicing this very large user base has enabled AWS to build a uniquely reliable, secure and scalable Kubernetes distribution to operate the clusters provisioned for its managed service.

One of the remarkable advantages of EKS is the free and open source availability of EKS Distro, the Kubernetes Distribution built and used by AWS to power its EKS service.

This is a unique capability, providing a consistent Kubernetes distribution you can deploy to any environment. The distribution includes extended maintenance of critical security patches, even for versions no longer supported by the open-source Kubernetes community.

Cluster Declarations and Cluster API

Having picked EKS Distro for your Kubernetes distribution, you now have to identify how you will provision your clusters.

There are many tools and mechanisms available to provision a Kubernetes cluster, and most of them will require different types of actions, configurations and know-how, depending on the target infrastructure. This is a problem for hybrid environments.

If each cluster must be provisioned using different tooling and the steps necessary to achieve consistency must be managed by human cluster operators, you will run into huge operational overhead and complexity as the diversity of your environments grows.

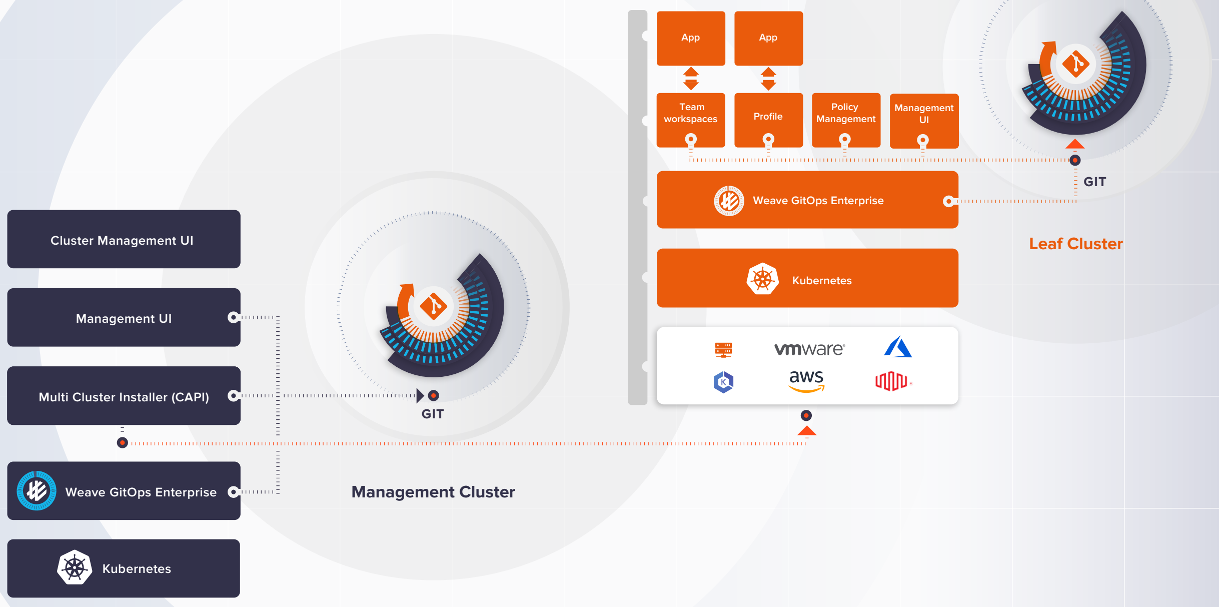
Cluster API (CAPI) aims to solve this problem with declarative definitions of clusters, and by abstracting the provider-specific tasks required to create a Kubernetes environment.

Weave GitOps provides native support to the numerous providers built for CAPI. It is compatible with a wide array of public and private clouds as well as for deploying EKS Distro on top of bare metal.

Weave GitOps: lightning fast, reliable and scalable

Weave GitOps is Weaveworks' continuous deployment and operations product that makes it easy to deploy and manage Kubernetes clusters and applications in any environment. A single management console lets you operate clusters running anywhere: in the public cloud, on the edge or in any hybrid scenario. Strong multi-tenancy can accelerate app delivery by providing developers with self-serve isolated workload namespaces across environments.

WEAVE GITOPS IS INFRASTRUCTURE AGNOSTIC



Declarative Operations - Abstracting environment specific complexity

Cluster API brings us the ability to declare our clusters and hides away the environment-specific complexity in creating full Kubernetes environments. But before we go any further, let's define a clear foundation around the concept of declarative configuration – and what agent-based automation is all about.

Imperative vs Declarative

With traditional imperative mechanisms for automation, pipelines and workflows are used to define the various actions required for a target system or environment to reach the state we desire. This becomes a big problem as the number and diversity of target environments increases, thus requiring an increasing number of uniquely configured automations with environment-specific idioms.

A declarative model removes that complexity by replacing a set of actions with a description of the desired state one wishes to achieve. It eliminates the need to specify *how* to reach a given state, and delegates that responsibility to agents that will interpret the description and take any necessary actions to reach the desired state.

This model allows teams to define and operate target infrastructure with minimal effort, and without worrying about target environment specifics. This simplification allows small teams to operate very large infrastructures, and reduces overhead, tooling sprawl and operational know-how, as your hybrid architecture diversifies.

But then the question becomes, how do we get those declarations over to our target systems?

Pull based and autonomous deployment

Authentication and network access against a growing number of diverse target environments can quickly get out of hand. Given varying network constraints, as well as the need to have secured control planes, traditional push-based mechanisms do not scale well in hybrid environments. A pull-based approach is critical for effective scale handling, and works hand-in-hand with declarative configuration. But how does it work?

Instead of having an external system pushing descriptions into the cluster, agents running within the cluster continuously pull from a single source of truth. This simplifies access control and network connectivity.

On the one hand, your Git repository becomes the central and consolidated location to control access by your clusters. On the other, there is no need to enable external access to the control plane, since the cluster is establishing an outgoing connection to the repository. The outgoing network connectivity from the cluster dramatically reduces the need for network and access control configurations at the cluster's location, and therefore reinforces a solid security posture. Once the agents in the cluster have pulled the latest desired state declarative configuration, they will autonomously perform all actions necessary to reach that desired state.

So we understand why GitOps, with its declarative configuration, pull-based and agent-based automation abstraction, is key to simplifying the deployment and operation of cluster fleets across hybrid locations. Now, let's dig deeper into managing configuration for both infrastructure as well as application delivery.

Configuration Management and Application Delivery

The benefits of a single source of truth

Having a clear understanding of the runtime configuration of your infrastructure, as well as the workloads running on it, is indispensable. Fast and simple access to this information will be necessary as you choose how and where to locate your various workloads. Especially when things don't go as planned, you quickly want to understand sizing demands and cluster tenancy.

A single source of truth, however, doesn't mean a single repository for everything. It means a single and authoritative source of configuration for every element (whether cluster, application or team) in your architecture – one that can be traced back to a common origin. This will become clearer when we discuss infrastructure and team repositories, in the next section.

Structuring your GitOps Repositories

In hybrid architectures you will have multiple clusters, each running multiple shared services, as well as workloads for multiple teams. Managing this growing body of configuration can become complicated if not effectively structured.

You will find multiple approaches to organizing your GitOps repositories. Some enterprises have individual repos per environment, others use branches. But these models can result in problems down the road, such as lack of traceability and difficulties managing conflicting codebases.

At Weaveworks we have identified that organizations dealing with GitOps at scale benefit from a repository structure that allows for team-specific autonomy, while using extensible and reusable code by defining a flexible directory structure within a single branch. This setup can be used and minimally customized as new clusters, workloads or services are added to a growing number of hybrid environments.

The defined structure will allow for consistency across repositories and provide a simplified mechanism for teams to deploy applications and consume shared services on any target environment, without the need to make requests to platform operators.

The platform repository

The platform repository is the root of your architecture, the one that holds the definition of every cluster in the organization. It holds configurations to enable teams to deploy applications to one or more clusters, as well as the shared services deployed and managed by the platform operators across various hybrid environments. This repository will give you a consolidated view of what clusters you are currently operating across all locations and environments, as well as the specific configuration for each cluster. It will also hold policy and security definitions that will be applied throughout your hybrid architecture.

Team/Application specific repositories

Each team may own one or more applications. They will likely use multiple separate repos and will want to have autonomy in defining and deploying their applications. What you are looking to define is a common repository structure that all teams can use on their own, but that will map effectively to the various environments where you want to deliver those applications.

Creating reusable, extensible code

Using an extensible construct is very important, as it adapts to the changing requirements of a growing number of target environments and applications. Simultaneously, it ensures that there is little to no duplication of configuration, while maintaining portability of configuration.

The simplest and most portable solution is Kustomize, which natively supports the concept of bases and patches. These two fundamental concepts allow you to create reusable bases that can be customized through patches, reducing code duplication and simplifying configuration changes at scale across a range of target environments, while allowing for specific modifications where applicable.

Keeping infrastructure consistent and secure

Declaring and reusing services

Using reusable and extensible bases is ideal for including shared services across your various environments. This is a common pain point among development teams, who are looking for commonality of services across any environment they're deploying into, without having to worry about the specifics of each instance of the service. Declaring your services in your infrastructure repo and applying them across your hybrid environments becomes a relatively effortless process, simply by reusing bases and customizing as necessary for each given environment.

Wrapping teams in policy and security

We continue to see the value of this concept as we look into adding teams and deploying applications to clusters.

Every team requires a set of common resources. Platform operation teams want to enable development teams to deliver software autonomously to any environment while maintaining compliance with policy and security.

Creating a base that is applicable across all teams allows us to reach this very objective. The goal is to have a common "wrapper" that the platform team has full control over, while allowing for customization of team specific attributes, such as repository, namespace names and access control specifics.

This "wrap and extend" concept will enable teams to feel safe in delivering code to production quickly, while remaining able to make operational and runtime decisions for the applications they own. Platform teams will sleep easily knowing that artifacts must be signed or else they will be rejected from the clusters, and that all network and security best practices will be continuously validated.

Key takeaways

We hope you find these guidelines and recommendations valuable as you design, deploy and operate your hybrid platform. Here are the top five key takeaways:

1. A declarative and pull-based approach to cluster and application operation is critically important, as the number and diversity of clusters grows in hybrid architectures.
2. True application portability depends on delivery portability.
3. Using a common Kubernetes distribution across all your different environments greatly reduces operational complexity, by ensuring version and component compatibility. EKS Distro is uniquely positioned to offer a common distribution for AWS users using managed EKS.
4. Repository and directory structure are critical for extensible and manageable codebases. Using bases and patches to structure reusable constructs dramatically simplifies the management of large fleets of clusters with little effort.
5. These extensible constructs can be leveraged towards shared service deployment, and to wrap teams in security and compliance.

Next Steps: Demo Weave GitOps and EKS

Weave GitOps allows enterprises to deploy and operate EKS clusters across any target location, offering native GitOps support and visibility, regardless of where your cluster is located.

Weave GitOps will instantly provide you with Cluster API capabilities including cluster templates and service profiles. These capabilities will allow your platform and operation teams to provision, configure and manage numerous clusters following a common set of requirements with extreme simplicity.

Weave GitOps will also enable a common workflow and provide you with a platform to visualize and operate your infrastructure and applications as a single pane of glass, no matter where your cluster is running.

Learn more about Weaveworks on the AWS Marketplace

www.weave.works