JETSTACK

WHITE PAPER

Right-Shifting Your Kubernetes Maturity: A Blueprint for Levelling Up

Adoption of Kubernetes in a positive and impactful way is challenging; this white paper aids organisations in understanding their current level of Kubernetes maturity across six key aspects and provides them with direction on how to successfully advance their adoption efforts.

Executive Summary

Jetstack builds enterprise cloud-native platforms using Kubernetes and OpenShift. We give modern cloud native infrastructure efficiency, integrity and security. Our flagship products, open source credentials and deep expertise as a Kubernetes services provider empowers development and security teams to build modern cloud native environments that scale with full machine identity protection and efficient workload management.

Software development is an integral component of virtually every industry and organisations are continuing to seek tools and processes that accelerate software delivery. Kubernetes has become the leading container orchestration platform, as it improves velocity of both development and operations. But there is still considerable room for growth in the breadth and depth of its usage, and organisations need guidance on how to optimise their Kubernetes adoption to ensure maximum performance, security, disaster recovery and overall cost efficiency.

Most organisations are in the early stages of Kubernetes deployment, though adoption is more advanced in sectors such as financial services or online entertainment, which take a cloud-first approach to service delivery and operation. Yet regardless of industry sector, going forward clusters are undoubtedly accumulating. According to Gartner, Inc., a research and advisory firm, "By 2025, more than 85% of global organizations will be running containerized applications in production, which is a significant increase from fewer than 35% in 2019." [1] This growth will be driven primarily by cloud native Kubernetes deployments. In response to this need for scale, organisations are now exploring all aspects of automation and resource optimisation.

To improve their utilisation of Kubernetes and build effectively, organisations must be able to recognise their level of execution across six core aspects:

- Adoption
- Security
- Observability
- Cluster Operations
- Developer Experience
- Development Process

This paper provides a Kubernetes Maturity Matrix to help organisations self-assess their current level of Kubernetes use. This highly practical guide is intended for organisations that have initiated Kubernetes at any stage, from early planning and conceptualisation to initial adoption using Proof of Concepts to advanced optimisation. Each aspect is structured into five levels of technical maturity with clear descriptions of what is achieved at each level. This structure allows organisations to easily identify their maturity level and define a strategic and targeted plan to expand their Kubernetes environments using proven technical approaches to meet the companies' goals and better realise the benefits of Kubernetes.

Introduction

When Kubernetes is introduced into an organisation, it is most often pioneered by a single team. Generally, this is either operations or development, as both teams recognise significant benefits from Kubernetes. The journey usually starts from one of two drivers: 1) operations is looking for more efficient resource usage with better application hosting; automated, pain-free scaling; and easier, modern deployment mechanisms, or 2) a development team wants to move to distributed, microservices architectures for better development velocity and flexibility. Regardless of where it starts, as the use of Kubernetes expands, more teams get involved and the number and types of stakeholders increase.

As organisations grow their Kubernetes environment, they want to ensure they make the best use of the platform. Six key aspects have been identified that enable organisations to successfully implement and capitalise on Kubernetes: Adoption, Security, Observability, Cluster Operations, Developer Experience and Deployment Process. However, if organisations enhance some aspects but neglect others, they will limit the overall benefits realised.

The following Kubernetes Maturity Matrix depicts five progressive maturity levels for each of the six aspects. Using the matrix, organisations can assess their Kubernetes implementation—highlighting which aspects need to be enhanced. Collectively, this assessment allows organisations to align their implementation efforts across the key aspects to maximise the benefits realised at each level.

This paper includes the Kubernetes Maturity Matrix followed by separate pages for each aspect that outline the aspects' five maturity levels. This enables the reader to step through each aspect and conduct a self-assessment for their organization. The reader can refer to the section guide on each page to make it easier for organisations to navigate the document and follow the self-assessment process. This is followed by a summary of the achievements and benefits realised when a level is attained across all aspects. The paper concludes with a look at next steps—once an organisation has assessed its Kubernetes maturity, how can it best advance its Kubernetes deployment?

How to Use the Matrix

Rows

The matrix shown below includes six (6) aspects used in Kubernetes execution: adoption, security, observability, cluster operations, developer experience and deployment process. In the remainder of this paper, each of these aspects is considered separately to enable organisations to assess their progress for each one independently.

Columns

The columns break down each aspect into five (5) levels of progressive implementation maturity: initial, exploratory, fundamental, repeatable and optimisation. The matrix indicates the achievements and/or limitations for each aspect at each of the five levels. In the following pages that discuss each aspect, the levels are delineated in more detail.

Evaluation

Likely, organisations will be at different levels for different aspects.
To optimise Kubernetes usage, the six aspects should be grouped together at roughly the same level. Otherwise, an aspect at a lower level will hinder the progression of the other aspects. The outcome of this Kubernetes maturity assessment will indicate where an organisation needs to invest more effort to improve their Kubernetes utilisation and continue to move to the right on the Maturity Matrix.

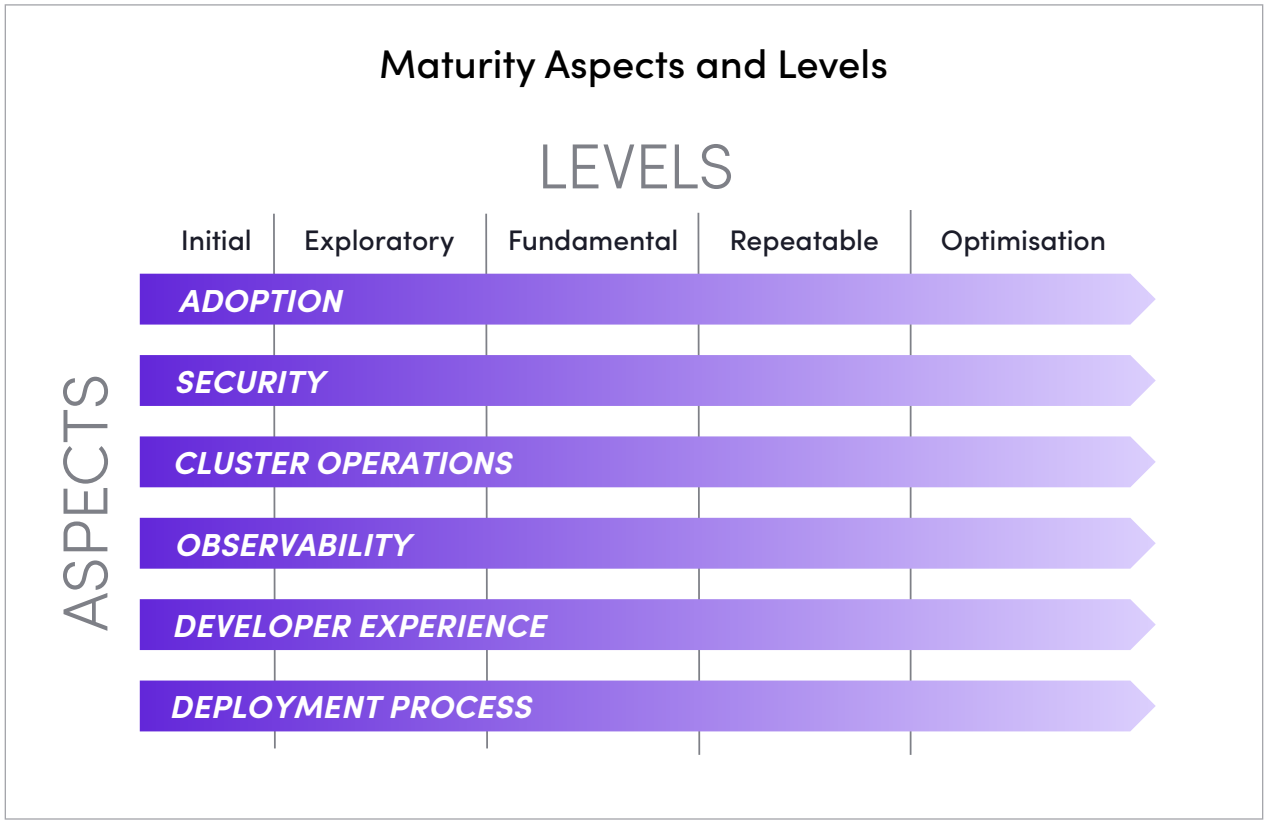
Maturity Matrix

	Level 1: Initial	Level 2: Exploratory	Level 3: Fundamental	Level 4: Repeatable	Level 5: Optimisation
Adoption	Introduction of Kubernetes through proof of concepts (PoCs)	Hosting low-risk, non-mission-critical, small footprint applications	Hosting new, cloud native applications with distributed architectures	Hosting critical applications and migrating cloud-optimised legacy applications	Use of custom operators and Kubernetes design patterns
Security	Use of only default role-based access control (RBAC) with no custom roles	Expansion to custom RBAC, image scanning, use of custom service accounts and TLS ingress	Introduction of application isolation via network and RBAC policies; use of container runtime scanning	Employment of a service mesh solution to enable machine identity via in-cluster mTLS	Utilization of admission webhooks and binary authorisation
Observability	Observability limited to centralised logging	Expansion to centralised metrics collection and basic cluster health monitoring	Implementation of full application and cluster monitoring, metrics collection and alerting	Addition of tracing and event capture	Observability leveraged for advanced scaling and APM
Cluster Operations	Cluster management on a manual, ad-hoc basis	Development of scripts for common operational tasks with manual script execution	Clusters still managed individually but with HA clusters and either burstable or guaranteed quality of service	Introduction of automated and consistent cluster management; advanced resource utilisation	Centralised visibility and management with Kubernetes CLI access to production clusters no longer required
Developer Experience	Medium-level Kubernetes knowledge required; usage of Kubernetes CLI is essential	Extensive Kubernetes knowledge required to configure resource manifests and toubleshoot issues	Kubernetes interaction abstracted via UI-based tools; adoption of Kubernetes-specific development tools	Widespread adoption and integration of selected Kubernetes-specific development tools and processes	Incorporation of chaos engineering; availability of developer self-service options
Deployment Process	Limited to manual deployments using Kubernetes CLI	Use of templating tools such as Helm and Kustomize to introduce more automation	Extensive use of custom Helm charts and/ or Kustomize configuration	Adoption of GitOps workflows and patterns	Exploration and adoption of Kubernetes operators for specific applications

Determining Maturity

As organisations conduct this assessment of their Kubernetes maturity, they should select the highest level in which they have completed all the achievements listed. If they have partially completed a level, they should flag the tasks that still need to be accomplished for that level to help identify and prioritise next steps in advancing their Kubernetes maturity.

This whitepaper can be used as a blueprint to help plan effective Kubernetes adoption by helping organisations identify where they should build their technical capability and shift right along each of the aspects to reach the higher maturity levels. Having considered the full matrix on page 5, it may be useful to use the high-level diagram below to mark where the organisation currently sits for each of the aspects. This position can then be reviewed with colleagues to collectively identify the organisation's position and discuss how to advance the overall maturity level to realise the benefits outlined in the following sections.



Adoption

This Maturity Matrix assesses the technical implementation of Kubernetes. Hence, the levels of adoption do not include the initial planning and strategy prior to implementation. Instead, it starts with the actual deployment of Kubernetes within the organisation.

Of enterprises that have deployed Kubernetes, 59% are running Kubernetes in production.[2] However, the footprint of the applications supported and whether they are non-critical or critical applications are a significant indicator of an organisation's Kubernetes maturity.

Level 1: Initial - Introduction of Kubernetes through proof of concepts (PoCs)

When Kubernetes is first introduced into an organisation, it is likely pioneered by a specific team. This team will often create small clusters for experimentation and "tyre kicking." This initial introduction of Kubernetes may also be used as an opportunity to understand the basics of Docker images and containers.

These clusters are probably provisioned using a cloud provider managed service, e.g., Google Kubernetes Engine (GKE) or Amazon Elastic Kubernetes Service (EKS). However, many often start by deploying Kubernetes on premises, perhaps using existing virtual environments.

Level 2: Exploratory - Hosting low-risk, non-mission-critical, small footprint applications

The use of Kubernetes has now gone beyond experimentation and clusters are hosting "real" workloads. For example, these workloads could be non-critical components of an application or transitory workers used by the continuous integration (CI) service. At this point, the Kubernetes userbase has widened with different types of users employing clusters.

Level 3: Fundamental - Hosting new, cloud native applications with distributed architectures

With sufficient confidence in their ability to operate and manage Kubernetes clusters, teams are now comfortable with hosting distributed cloud native applications. In the new applications, developers are starting to use distributed architectures and design patterns optimised for Kubernetes hosting.

Level 4: Repeatable - Hosting critical applications and migrating cloud-optimised legacy applications

At this level, teams are confident and experienced in hosting critical applications in production Kubernetes clusters. In addition, they are well underway in migrating legacy applications to Kubernetes. The challenges of migrating legacy applications are well understood, and some migrations are already in place.

Level 5: Optimisation - Use of custom operators and Kubernetes design patterns

Instead of using Kubernetes primitives directly, teams are using custom operator supersets, such as Knative Services and Argo Workflow. Applications are being developed which take advantage of these operators and use design patterns which lend themselves to the Kubernetes controller-reconciler paradigm.

Security

Security is an essential component of Kubernetes deployments and should not trail behind other Kubernetes maturity aspects. In a survey by StackRox, 90% of respondents experienced a security issue in their Kubernetes or container environments in the last 12 months as of Autumn 2020, and 44% said they delayed or slowed an application's deployment into production because of a Kubernetes or container security issue. [3]

The teams that play a role in Kubernetes security greatly vary across organisations, including IT security, developers, operations and platform. Kubernetes can provide significant security advantages, however, execution will require coordination across teams and security expertise, as well as implementation of tools and processes.

Level 1: Initial - Use of only default role-based access control (RBAC) with no custom roles

When first introduced, organisations tend to rely on the security inherent within Kubernetes. By default, RBAC is enabled in Kubernetes clusters, however, it is likely that most initial users will have broad privileges (probably cluster administrator). This results in a lack of protection from user error during initial use. Even at this first level, external access to clusters should be controlled, but it might not be.

Level 2: Exploratory - Expansion to custom RBAC, image scanning, use of custom service accounts and TLS ingress

At this second level, organisations go beyond the security defaults provided with Kubernetes. As organisations advance past level one in the other technical aspects, more sophisticated RBAC definitions are necessary to reflect different user types and their access requirements.

Also, as the clusters are now hosting real workloads (and possibly integrating with organisation data sources), a greater defence-in-depth security posture is necessary. For example, this includes image scanning and use of custom service accounts to add finer-grained permissions to pods running in the cluster. Plus, TLS ingress is employed to secure and manage external access to services in the clusters.

Level 3: Fundamental - Introduction of application isolation via network and RBAC policies; use of container runtime scanning

For additional layers of security, teams introduce network policies and custom RBAC to isolate applications from each other and system components. At this level, organisations restrict which image registries can be used (ideally only private ones which host only approved images). Runtime scanning of containers is also used at this level.

Level 4: Repeatable - Employment of a service mesh solution to enable machine identity via in-cluster mTLS

To provide security in a distributed microservice architecture that supports repeatability, a service mesh solution, such as Istio or Linkerd, is in place to at least cover the production clusters. Often the primary motivation for installation of a service mesh solution is to provide mutual TLS in-cluster and across clusters.

With Kubernetes, TLS is used both for cluster ingress and to establish machine identities. However, introducing TLS for these purposes involves extensive certificate management. Therefore, organisations usually rely on a certificate management solution, such as cert-manager, at this level.

Level 5: Optimisation - Utilisation of admission webhooks and binary authorisation

To better manage and control what is being hosted in production clusters, platform teams implement binary authorisation and other types of authorisation controllers which take advantage of the Kubernetes admission webhook architecture.

Observability

Kubernetes is designed to efficiently host distributed applications which, by their very nature, typically consist of many separate components and moving parts—all of which require monitoring and measuring. As the breadth of Kubernetes deployments increase, so must the sophistication of monitoring the health and capacity of the hosted components and resources. This is especially true for teams and organisations who use service level objectives (SLOs) and indicators (SLIs) to track and measure system health and performance. As organisations progress through these levels, they need to continue to adapt their monitoring strategies to cover their increasingly complex Kubernetes environments to ensure appropriate levels of metrics and troubleshooting data are captured and analysed.

Level 1: Initial - Observability limited to centralised logging

At the very least, containers should be configured to write their logs only to standard output (stdout) with a separate log collector, such as Fluentd or Logstash. The log collector gathers the log entries and transports them either to a tool like Elasticsearch to assist with data search and analytics or to a managed logging service.

Level 2: Exploratory - Expansion to centralised metrics collection and basic cluster health monitoring

At this level, it is important to be able to monitor the health of workloads and clusters. Hence, organizations are now collecting, reporting and alerting on elementary metrics and health indicators.

Level 3: Fundamental - Implementation of full application and cluster monitoring, metrics collection and alerting

With organisations now leveraging distributed cloud native applications, monitoring requires the collection of a broad set of metrics from system and application components, including application performance metrics (APM). In addition, metrics and log-based alerting is implemented.

Level 4: Repeatable - Addition of tracing and event capture

At this level, organisations address some of the challenges of running distributed applications by introducing tracing, which enables them to better understand component interactions and bottlenecks. Organisations also introduce event capture to flag significant events that occur in their Kubernetes clusters.

Level 5: Optimisation - Observability leveraged for advanced scaling and APM

The ability to scale pods and nodes both inwards and outwards is essential for running cost-effective Kubernetes clusters and this is facilitated by using APM metrics to help drive the scaling decision making. Organisations running complex and critical applications benefit from tooling, such as Honeycomb and New Relic, that specialise in analysing and surfacing insights gleaned from all the observability data being collected.

Cluster Operations

Kubernetes is a complex and feature-rich platform, and ongoing management of the clusters will always be a critical activity, irrespective of whether a managed service or more vanilla implementation is used. Cluster upgrades, installation of additional middleware components, user administration and numerous operational tasks are required. As the number of clusters and their importance to the business grows, so does the significance of undertaking these tasks efficiently and consistently.

Level 1: Initial - Cluster management on a manual, ad-hoc basis

When Kubernetes is first introduced, cluster operations are executed manually, often using kubectl, the Kubernetes command line tool. At this small initial scale, a manual approach is not a problem, but it is not sustainable when the use of Kubernetes grows within an organisation.

Level 2: Exploratory - Development of scripts for common operational tasks with manual script execution

At this level, organisations start the transition from manual cluster management to a more automated approach to ensure effective cluster operations as the use of clusters grows. Organisations develop and use scripts to carry out common cluster management tasks, such as cluster upgrades and manual scaling of pods and nodes. However, at this level, scripts are still executed manually by authorised administrators.

Level 3: Fundamental - Clusters still managed individually but with HA clusters and either burstable or guaranteed quality of service

By the time Kubernetes is hosting new, cloud-native applications with distributed architectures, high availability (HA) clusters should be the norm in production. Though, at this level, each cluster is still likely to be managed independently and often manually.

In addition, cluster operations at this stage include CPU and memory values (request and limit) in pod specifications to ensure a quality of service (QoS) class of burstable or guaranteed.

Level 4: Repeatable - Introduction of automated and consistent cluster management; advanced resource utilisation

To achieve repeatability, cluster operations move away from manual configuration of clusters to an automated approach that provides better consistency and traceability. Tools can help with this automation, such as Anthos Config Management. Plus, teams are now able to more accurately set pod resource quotas due to the collection and analysis of metrics and tracing.

Level 5: Optimisation - Centralised visibility and management with Kubernetes CLI access to production clusters no longer required

Cluster management is now almost fully automated and driven via a Gitops approach to configuration management. With automated processes in place, direct kubectl access to production clusters is no longer permitted except under emergency "break glass" circumstances.

To optimise cluster operations, management and visibility are centralised and conducted through a "single pane of glass" management console. Plus, cluster self-healing is enabled, monitoring clusters and automatically redeploying to the desired state if a service or node fails.

Developer Experience

A lack of experience and expertise was named as the top challenge for Kubernetes adoption, deployment and management.[4] Introducing Kubernetes into an organisation requires significant Kubernetes knowledge, as initial deployments are very manual and command line driven. Without sufficient developer experience and careful execution, Kubernetes deployments can be at risk; over two-thirds (67%) of organisations have experienced a misconfiguration in their Kubernetes or container environment within the last 12 months as of Autumn 2020.[3]

However, as Kubernetes maturity increases within an organisation, more automation and often the addition of deployment tools, increase ease-of-use. This helps to facilitate an organisation's reliance on Kubernetes, making it easier to grow both the number of users and clusters leveraged and feel safe relying on Kubernetes for more critical and legacy applications.

Level 1: Initial - Medium-level Kubernetes knowledge required; usage of Kubernetes CLI is essential

A developer's initial exposure to Kubernetes can be daunting: it requires new tools and different patterns, and much of it is command line driven. Trying to locally develop a reasonably-sized microservices application is not practical, which means developers must start using Kubernetes clusters as actual development platforms. For developers to acquire sufficient knowledge to achieve this (deploying, connecting and troubleshooting), they must scale a steep learning curve.

Level 2: Exploratory - Extensive Kubernetes knowledge required to configure resource manifests and troubleshoot issues

Developers expand their extensive Kubernetes knowledge even further in order to configure their applications' resource manifests as well as troubleshoot issues that arise.

Level 3: Fundamental - Kubernetes interaction abstracted via UI-based tools; adoption of Kubernetes-specific development tools

Developers continue to employ Kubernetes basics, but there is much less need for deep operational knowledge or to use the Kubernetes command line tool, kubectl. Instead, there is significant use of UI-based tools to view cluster and application state (for example, web dashboards, desktop UIs, such as Lens, and IDE plugins).

At this level, organisations also adopt Kubernetes-specific development tools, such as Skaffold, Telepresence, Tilt, etc. These tools streamline the process of rapid building, pushing and deploying images into clusters.

Level 4: Repeatable - Widespread adoption and integration of selected Kubernetes-specific development tools and processes

Once teams have reached a stage where they are relying on repeatable architecture, they have acquired a set of Kubernetes tools (or built their own) which meet their development and operations needs and fit within their processes and culture. This introduces a higher level of ease of use and automation with less friction in the development environment.

Level 5: Optimisation - Incorporation of chaos engineering; availability of developer self-service options

To optimise both developer resources and Kubernetes efficiency, chaos testing of applications and clusters is now incorporated into standard testing regimes. This automatically tests how an organisation's Kubernetes environment will react to randomly triggered infrastructure events such as system reboots or component failures. Further optimisation is achieved by introducing Kubernetes developer self-service options, reducing delays in service provisioning.

Deployment Process

Although initially command line driven, as organisations progress in their Kubernetes environments, there are many tools available to help in easing and automating the deployment process. The ability to efficiently and safely deploy, upgrade and rollback workloads into clusters has a positive impact on development velocity and reducing the time to value.

Level 1: Initial - Limited to manual deployments using Kubernetes CLI

When teams first begin using Kubernetes, they typically deploy workloads using the kubectl command line tool. However, relying strictly on kubectl limits teams to very manual deployment processes, which rapidly become problematic and will necessitate their advancing to the next level to better leverage Kubernetes capabilities.

Level 2: Exploratory - Use of templating tools such as Helm and Kustomize to introduce more automation

Interacting with Kubernetes requires extensive use of YAML, and manually producing it quickly becomes tedious and error prone. At this level, teams select tools to automate the generation of the YAML manifests and manage deployments into clusters. Examples of tools that introduce this type of automation include Helm and Kustomize.

Level 3: Fundamental - Extensive use of custom Helm charts and/or Kustomize configuration

At this level, organisations make increased and more sophisticated use of YAML templating and application deployment tools such as Helm and Kustomize, for example, making extensive use of custom Helm charts and/or Kustomize configuration. However, at this level teams will still mostly be performing deployments and upgrades in an imperative, rather than declarative, style.

Level 4: Repeatable - Adoption of GitOps workflows and patterns

Application deployment and configuration is much more declarative at this level. This is achieved by adopting a Gitops approach, using fully automated deployments of Gitops workflows and patterns and facilitated by the use of Kubernetes native tooling such as Flux, Tekton or Argo.

Level 5: Optimisation - Exploration and adoption of Kubernetes operators for specific applications

To further optimise the deployment of workloads into clusters, Helm charts should be replaced with Kubernetes operators. This use of Kubernetes operators is becoming increasingly popular, especially for middleware such as database services.

Levelling Up

In the sections above, Kubernetes maturity was assessed for each of the six aspects individually. In this section, each level is evaluated holistically across aspects. Once organisations know their maturity for individual aspects, they should determine the overall level of maturity they have accomplished collectively. Each level represents a unique stage of achievement.

Level 1: Initial

At the initial level, organisations and teams now have awareness of the complexities of Kubernetes as well as the benefits it can provide for their specific organisations. They are in a good position to determine whether committing to a strategy that includes Kubernetes as a hosting platform is a good investment for them.

Level 2: Exploratory

Transitioning to this second, exploratory level involves committing more time and effort to hosting meaningful workloads in a Kubernetes cluster. Organisations gain practical insight into what is involved in using and operating their own Kubernetes clusters.

The first production workloads are now being hosted in Kubernetes, most likely in a single production cluster with one or more other clusters for development and testing. With this increased complexity, teams start to understand the need for automation and scripted configuration. The security team works to ensure the Kubernetes environment is meeting the organisation's security and compliance standards.

Level 3: Fundamental

Teams are past the exploratory stage and have the fundamental aspects of Kubernetes in place to enable the hosting and support of cloud native applications in their clusters, often across multiple production clusters.

Transitioning to Level 3 involves a heavy focus on these improvements:

- Increasing security measures—cluster security is much more fine-grained with different privileges for developers and cluster operators
- Automating as much cluster management and application deployment as possible
- Improving cluster resilience through high availability, better monitoring and alerting
- Understanding and implementing Kubernetes best practices

Overall, there is much greater awareness of the health and performance of the clusters and hosted applications. When using Kubernetes as a hosting platform, this is the minimum level that organisations should aim to achieve across all six aspects.

Level 4: Repeatable

To achieve a repeatable architecture in a Kubernetes environment, a service mesh for clusters and TLS certificate management is implemented. Further automation ensures the management of clusters and deployment of applications to multiple clusters is consistent and reliable.

At this level, there is a focus on using clusters more efficiently. This includes ensuring accurate pod resource request values to enable the Kubernetes scheduler to run more containers on a node.

Increased automation reduces developer burden and implementation errors as well as increases resource efficiency. Organisations are now comfortable expanding their use of Kubernetes to include critical and legacy applications.

Level 5: Optimisation

At level five, organisations further optimise their Kubernetes deployment. This includes efforts for additional automation and centralisation. Goals include simplified management, further resource optimisation, developer self-service options, sophisticated scaling and APM analytics.

To set Kubernetes implementation goals, organisations need to determine the level at which they will extract the most value from Kubernetes with minimal risk. At a minimum, organisations should aim to achieve Level 3 across all aspects to enable the use of Kubernetes for distributed cloud native environments. However, many will want to achieve Level 4 to be able to safely expand Kubernetes usage to critical and legacy applications. Benefits achieved at Level 5 will be most realised by organisations with multiple clusters in production that can best leverage optimisation.

Conclusion

Kubernetes has a momentum—not just with industry-wide adoption as a cluster orchestration platform, but also within individual organisations. A single team may initiate Kubernetes and begin to explore its usage and benefits. However, Kubernetes tends to quickly expand to other teams and stakeholders who broaden its use across the organisation. Then Kubernetes quickly evolves with the addition of security, automation and tools.

However, organizations need to take control and strategically deploy Kubernetes to optimise results. The Kubernetes Maturity Matrix and assessment help organisations analyse the current state of their Kubernetes environment. By identifying their level of maturity within the six main aspects, organisations can recognise the milestones they have achieved as well as any gaps they need to fill to accomplish a particular level and better realise the benefits of Kubernetes.

Kubernetes is still relatively young, but small and large organisations alike are benefiting from its use. However, its youth means teams often lack the experience needed to optimise their Kubernetes deployments. To compensate for this lack of experience, organizations can turn to partners to help identify needed advancements:

- The Kubernetes maturity level to target to maximise the benefits and minimise the risk
- The best Kubernetes tools to support organisation's goals, processes and culture
- A strategic roadmap for further Kubernetes implementation that realistically leverages resources and meets desired goals

Using Kubernetes in cloud native infrastructure can greatly increase developer velocity and shrink the software time-to-market, increasing productivity and, ultimately, revenue. But only if used correctly. Kubernetes must be deployed to optimise cluster security, performance, cost efficiency, recoverability, robustness and more. This can be achieved with informed planning and the right partner support.

Want to accelerate your Kubernetes success? Partner with Jetstack to level up! To start a discussion on how we help you with your cloud native infrastructure, reach out directly by emailing info@jetstack.io or visit our website: www.jetstack.io.

Footnotes

1. Gartner, Best Practices for Running Containers and Kubernetes in Production, Arun Chandrasekaran, 4 August 2020.
2. VMware. The State of Kubernetes 2020.
3. StackRox. State of Container and Kubernetes Security. Fall 2020.
4. VMware. The State of Kubernetes 2020; StackRox. State of Container and Kubernetes Security. Fall 2020.

